Sugar Developer Guide 12.0

Sugar Developer Guide 12.0

The Sugar Developer Guide describes how to configure and customize Sugar by making code- and database-level changes to the Sugar file system. You will need direct access to the server along with the proper file-system permissions to alter files in the Sugar instance directory.

Sugar customers that are hosted on Sugar's cloud service cannot directly access their database's file system but can work with a Certified Sugar Partner to customize their Sugar deployment. For a list of Certified Sugar Partners, please refer to the Partner Page to find a reselling partner to help with your development needs.

Introduction

Overview

The Sugar Developer Guide is an essential resource for developers who are new to Sugar or to CRM and web-based applications. It describes how to configure and customize the Sugar platform for a broad range of tasks applicable to any organization that has a need to manage business relationships with people.

Prerequisites

Using and understanding the documentation contained in the Sugar Developer Guide requires basic programming and software development knowledge. Specifically, you should be familiar with the PHP general-purpose scripting language and the SQL programming language for accessing databases.

Understanding Sugar's Framework

Designed as the most modern web-based CRM platform available today, Sugar has quickly become the business application standard for companies around the world. The Sugar application framework has a sophisticated extension model built into it, allowing developers to make significant customizations to the application in an upgrade-safe and modular manner. It is easy to modify the core files in the distribution; you should always check for an upgrade-safe way to make changes. Educating developers on how to make upgrade-safe customizations is one of the key goals of this Developer Guide. For more information on Sugar's structure, please review the architecture section.

Supported Platforms

Originally, Sugar® was written on the LAMP stack (i.e. Linux, Apache, MySQL, and PHP), but has since added support for every operating system on which the PHP programming language runs, for the Microsoft IIS web server, and for the Microsoft SQL Server, IBM® DB2®, and Oracle databases. For more information about supported software versions and recommended stacks, please refer to the main Supported Platforms page.

Sugar Products

Sugar has several CRM products available: Sugar Sell, Sugar Serve, Sugar Ultimate, Sugar Enterprise, and Sugar Professional, which are all sold under a commercial subscription agreement. These products are developed by the same development team using the same source tree with different modules and features available depending on the product. A comparison of each product's features is available in the License Types Matrix documentation of the Administration Guide.

Basic Development Rules for Sugar Products

Unless SugarCRM has given you express permission to do so, the following are what not to do when you are configuring, customizing or modifying this Sugar product:

- Do not remove or alter any SugarCRM or Sugar copyright, trademark or proprietary notices that appear in the Sugar products.
- Do not "fork" the Sugar software (e.g., take a copy of source code from this product and start independent development on it, creating a distinct and separate piece of software).
- Do not modify, remove or disable any portion of SugarCRM's "Critical Control Software."
- Do not combine or use the Sugar products with any code that is licensed under a prohibited license (e.g., AGPL, GPL v3, Creative Commons or another similar license that would "taint" the Sugar products and require you to share the source code for this product with a third party).
- Do not use any part of the Sugar products for the purpose of building a competitive product or service or copying its features or user interface.

Development Tools

Sugar has a set of built-in tools that you can use to your advantage when troubleshooting or developing.

Developer Mode

Developer Mode will allow for Sugar to recompile cached files when the page is reloaded. The following file types are rebuilt:

- Handlebar Templates (.hbt)
- Smarty Templates (.tpl)
- JavaScript Controllers (.js)

When Developer Mode is enabled, The Sidecar JavaScript library references the full JavaScript files located in ./sidecar/ rather than the concatenated and minified cached versions. You can turn on Developer Mode by navigating to Admin > System Settings. For more information, please refer to the System documentation.

Note: This setting should remain off unless developing because it will degrade system performance.

Diagnostic Tool

When troubleshooting issues, you may find the diagnostic tool to be helpful. This tool will export a zipped package containing the requested diagnostics and is available even if you are hosting your instance on Sugar's cloud service.

The diagnostic tool has the ability to export the following:

- SugarCRM config.php
- SugarCRM Custom directory
- phpinfo()
- MySQL - Configuration Table Dumps
- MySQL - All Tables Schema
- MySQL - General Information
- MD5 info
    - Copy files.md5
    - Copy MD5 Calculated array
- BeanList/BeanFiles files exist
- SugarCRM Log File
- Sugar schema output (VARDEFS)

You can use the diagnostic tool by navigating to Admin > Diagnostic Tool. For more information, please refer to the System documentation in the Administration Guide.

Composer

When building applications, some developers prefer to use Composer to manage their external dependencies and make them more intuitive. For more information, please refer to the Composer documentation.

Development Methodology

Overview

This page discusses standard practices that we recommend for improving the success rates of Sugar development projects.

Development Best Practices

When developing Sugar® customizations as part of an on-site CRM project implementation, we recommended placing the entire Sugar application filesystem under source code management. Sugar developers know that customizations made to Sugar are placed under the ./custom/ directory. But during the lifecycle of a CRM implementation, you will need to upgrade Sugar versions, which will change core files. Many projects will also need to track other related project files that may not all be Sugar platform code. For example, pre-flight SQL scripts, data migration scripts, Web server configuration settings, etc.

For SugarCloud projects and ISVs, if you are building a custom module package or integration designed to be installed into many Sugar instances (including SugarCloud instances), then tracking only ./custom/ directory files should be enough.

Using .gitignore Files

Today, many developers choose to use Git as their source control management. There are certain Sugar application files that you do not want to track; most of these are generated files that are created at runtime or are Sugar instance-specific configuration files.

Below is a sample .gitignore file that you can use or adapt to the source control management system of your choice.

```
  *.log
/.htaccess
/config.php
```

```
/config_override.php
/cache
/upgrades/module
/upload
/custom/blowfish
/custom/history
/custom/application/Ext
/custom/modules/*/Ext
/custom/Extension/**/*orderMapping.php
```

Recommendations for Development Teams

Code quality is important to maintain because Sugar customizations run in the same environment as the rest of the Sugar application. Here are a few best practices to help development teams uphold code quality.

- Adopt an appropriate Git workflow for development. For reference, see Atlassian's tutorial on Git workflow options.
- Develop within feature branches that are tested before being merged back into master to keep master stable.
- Avoid workflows that involve developers committing directly into the master branch to prevent code destabilization.
- Development teams should always perform code reviews before merging in new code.

Deploying Sugar Code

Where you plan to deploy Sugar code is the biggest factor in determining how Sugar code should be deployed and how your project should be managed.

Are you working on a Sugar project for an on-premise Sugar implementation? Are you working on a custom module that you plan on distributing through SugarExchange? Are you planning a solely REST API integration? The answers to these questions guide how you should develop and deploy Sugar code.

- **Sugar on-site projects** : Develop these customizations using the exact version and flavor of Sugar that you plan to use in production.
- **SugarCloud projects** : Develop these customizations on the latest available version of Sugar for the particular flavor the customer has purchased.
- **Custom modules or integrations** : If you plan on distributing your customization to many Sugar customers via SugarExchange or channel partners, design your customization with Sugar's cloud service in mind.
    - Sugar's cloud service is more restrictive than our on-site installs

regarding supported customizations.
    - A customization designed for Sugar's cloud service can be supported in Sugar on-site instances, but the inverse is not always true.

For more information on Sugar's cloud service restrictions, please refer to the SugarCloud Policy Guide.

Packaging

The packaging of customizations is not a concern for many Sugar projects. Many projects just use Git (or some other file version control) to manage the distribution and deployment of Sugar code customizations. However, there are situations where the packaging of Sugar customizations is necessary.

If you plan on distributing Sugar custom code, then you must package your customization as a Module Loadable Package (a .zip file that includes all custom code along with a manifest file). It is easy to write a script to build a module loadable package either from custom directory content or by extracting customizations out of a development environment. See examples on Github here and here.

Note: Sugar Sell Essentials customers do not have the ability to upload custom file packages to Sugar using Module Loader.

In some Enterprise environments, changes are tightly controlled, and ownership of various Sugar application components may be spread across multiple teams, requiring a coordinated deployment. For example, a Database Administrator (DBA) may be responsible for implementing database schema (DDL) changes and a System Administrator may be responsible for implementing file system changes.

For these situations:

File system changes can be accounted for using Git to determine the difference between current production state and the latest changes to be deployed.

DDL changes can be accounted for via deploying latest file system changes on a clone of Sugar production instance and running Quick Repair command. Sugar will prompt you with any DDL changes that need to be made that you can then capture and share with your DBA.

Data Manipulation Language (DML) changes, if necessary, should be managed within SQL or PHP scripts.

Deployment

If using the traditional Git-based deployment, then deploying new Sugar code is as easy as checking out the latest branch and then running Quick Repair and Rebuild. Run the quick repair from the Sugar user interface, or script it for fully automated deployment.

When deploying to an instance on Sugar's cloud service, it is necessary to install the package manually using Module Loader. It is not possible to automate the deployment of packages into instances on Sugar's cloud service.

In a coordinated deployment, database changes should be deployed first, followed by filesystem changes, followed by a Quick Repair (if permitted) to clear system caches. You can clear caches manually by deleting the contents of the ./cache/ directory and then truncate the metadata_cache table in the Sugar database. The Sugar application regenerates these caches as needed.

Using Sugar Studio to deploy changes into new environments (especially Production) is not recommended. Manually re-creating customizations using the Studio user interface can be error-prone. It also runs the risk of inadvertently overwriting other code customizations. It may be initially slower to create a custom field, etc. manually using extensions on filesystem but, in the long run, you benefit from better change management and automation.

Managing Multiple Environments

CRMs are business-critical applications so development should never be performed directly on your production environment. A typical Sugar project involves multiple staging environments as well as individual development environments that each Sugar developer uses for actual coding.

SugarCRM recommends 4 different environments:

- **Production environment** : used by real users
- **User Acceptance Test (UAT) environment** : used by business stakeholders to sign off on changes that go into production
- **Quality Assurance environment** : used for test and validation of new features and bug fixes
- **Development environments** : used by individual Sugar developers to create and test code (often running on a local laptop or PC)

Code changes that flow upstream from a development environment should not be allowed into production without going through quality assurance (QA) and user acceptance testing (UAT) first. The intermediate environments serve as gates between development and production that help intercept problems before they reach production.

User and CRM data that needs to flow downstream from production environment should pass through intermediate environments as well to ensure consistency and that it is cleaned or anonymized of any personally identifiable or sensitive information.

Consistency

Maintaining as much consistency as possible for each of these environments is essential. Inconsistent environments can create issues where bugs are reproducible in one environment and not others (for example, a bug that only appears in production). Many times, these bugs are traced to configuration parameters that are not directly related to Sugar or the features and customizations under development.

To replicate your production environment as accurately as possible, we recommend you use VM or container technology in your development and QA environments.

Your UAT environment should match the infrastructure of your production environment.

SugarCRM uses a variety of container technologies in developing the core Sugar application and for working on Sugar projects. In particular, we use Virtual Box, VMWare, Amazon AMIs, Docker containers, Vagrant, and Packer. SugarCRM Engineering also uses Puppet to centrally manage the provisioning and configuration of all these different environments.

Testing

SugarCRM recommends using a variety of testing methods to ensure the quality of your Sugar project. Perform unit testing, functional testing (either manual or GUI automation), system integration testing, user acceptance testing, and performance testing.

- **Unit testing** should be applied to ensure that even the smallest code units within your application are behaving as expected.
- **Functional testing** should be performed to ensure that each feature and function behaves the way it was designed.
- **System Integration testing** should be performed to ensure that Sugar co-exists with external systems and that data flows between all these systems successfully.
- **User acceptance testing** should be performed by key stakeholders to ensure that your project is meeting business requirements.
- **Performance testing** should be performed to ensure that the Sugar application's responsiveness meets user expectations and that the

application continues to scale.

In our experience, neglecting any of these tests can negatively impact a Sugar project in terms of maintainability, customer satisfaction, and business success.

In the next section, we introduce some tools and open-source resources that can help you start a QA practice for your project.

Sugar Test Tools

For Sugar customers and partners, SugarCRM provides Test Tools that can be used to verify and perform quality assurance on Sugar customizations. Use of Sugar test technology requires familiarity with PHPUnit for PHP unit testing, Jasmine for JavaScript unit testing, and Apache JMeter for performance load testing.

Sugar Unit Tests

The Sugar Unit Test suites are the automated unit tests developed and maintained by SugarCRM Engineering during the process of building and releasing each new Sugar release. As part of our development process, these tests are required to run "green" (100% passing) at all times on each master release branch. Essentially, these tests form a regression test suite for an uncustomized version of Sugar running in our controlled build environment.

With that understanding, here is a recommended approach to take advantage of these unit tests.

- Run test suite against an uncustomized copy of Sugar in your development environment to establish a baseline. Not all tests may immediately pass; some may fail due to configuration differences between your development environment and SugarCRM Engineering's controlled build environment.
- Correct any observed failures or skip/remove the failing tests to create a base test suite that is 100% passing.
- As you develop customizations on Sugar, ensure that your base test suite continues to pass.
- Create new tests for new code customizations that you create.

Sugar Performance Tests

Instead of sanitizing data from a production environment for purposes of load testing, SugarCRM provides an open source tool called Tidbit that can be used to generate pseudo-random data to populate a Sugar instance with representable data.

We also provide Apache JMeter scenarios to Sugar customers and partners who request access. These JMeter scenarios can be used to drive a simulated load against the Sugar REST API interface used by Sugar. They can validate that your customizations have not had an unexpected impact on the performance of a Sugar instance.

DevOps

In order to facilitate and streamline development processes, Sugar recommends implementing DevOps automation. Use a tool such as Jenkins to orchestrate test automation, stage changes in any environment (dev, QA, UAT, and prod) for manual verification, and manage the promotion of changes from one environment to the next.

- Perform automated tests (e.g. unit tests) on each commit.
- Stage development changes on at least a nightly basis for use by QA or demos.
- Automate the rollback of changes in any environment as needed.
- Automate notifications to affected and responsible parties whenever a test fails, or a build fails to deploy.

Co-Existing with Studio Customizations

Sugar Studio and Module Builder enable administrator users to implement quick changes to a Sugar environment. But Sugar Studio lacks the rigorous change control that larger CRM projects need. Sometimes, Studio changes can even break code-level Sugar customizations. For this reason, we often discourage using Studio on heavily customized Sugar instances.

However, if Sugar Studio is an important up-front requirement for a CRM project, then there are strategies you can adopt for your customizations to avoid conflicts.

Sugar Studio can be used to modify the layouts, fields, and relationships for the majority of modules in Sugar. Module Builder can be used to define new modules along with their layouts, fields, and relationships. In practice, this means that the fields and layouts for any module's record view, list view, mobile view, and subpanels can be customized using simple administration tools.

Studio users, therefore, could potentially break code customizations that are reliant on a particular field or having a field on a particular layout or location. Here are some practices to avoid conflicts between Studio customizations and custom code:

- Avoid custom record, list, and subpanel view controllers because Studio users can change fields and layouts that affect expectations within your

controller code.
- Avoid custom record validations because Studio users can change fields and layouts that affect expectations within your validation code.
- Avoid hard-coded integrations that rely on a particular field because Studio users can change fields that affect expectations within your integration.
- Avoid new field and relationship vardefs customizations because Studio users can create new fields and relationships.
- Avoid viewdefs customizations for record, list, and subpanel layouts because Studio users perform these customizations.

Many customizations have a smaller risk of side effects related to Studio customizations. Here is a list of some changes that administrator users cannot perform via Studio:

- Adding custom dashlets
- Adding custom layouts or additions to footer or header
- Adding custom actions in record view action dropdown
- Adding metadata-aware integrations that discover fields and modules on the system
- Adding logic hooks

Conversely, certain customizations can provide Studio users with additional tools:

- Adding custom Sugar Logic functions
- Adding custom Sugar field types

Composer

Overview

Composer dependency management is the de facto standard for managing PHP dependencies. Sugar platform ships with all required code bundled and optimied together for our developers. The installer will, however, deploy both composer.json and composer.lock in the web root directory as a reference of all dependencies which are controlled by Composer.

Autoloader

Sugar has a custom autoloader that is PSR-0 and PSR-4 compliant and integrates with Composer's mapping definitions.

Integrations

When updating the Composer configuration, Composer will generate different

mappings based on the settings of every dependency:

- Class map
- PSR-0 map
- PSR-4 map
- Include paths (deprecated)

SugarCRM's autoloader uses those generated maps directly to initialize itself and to figure out how to load different classes.

Internals

To prevent endless file stats, Sugar's autoloader maintains a full list of all files at its disposal. This list is only generated once and is maintained in ./cache/file_map.php. Most of the Sugar codebase uses the autoloader to determine whether a file is available rather than performing expensive file_exists calls.

A second file, stored in ./cache/class_map.php , maintains a flat list of class-name-to-file mappings. When you make any changes to the file system, clear both files before testing new code.

Optimization

When updating dependencies through Composer, Sugar team uses the optimize flag: composer update --optimize-autoloader --no-dev". By doing so, Composer will scan all files in the packages it manages and create a full list of PSR-0 and PSR-4 class name to file mappings, instead of performing this lookup on the fly by the autoloader itself on runtime.

Frequently Asked Questions

Is the composer package required to install a Sugar instance?

No. The Sugar installer ships with all required code bundled together. The installer process does not execute any Composer commands. The installer will deploy both composer.json and composer.lock in the web root directory as a reference of all dependencies which are controlled by Composer.

Why does Sugar ship ./composer.json and ./composer.lock if the installer doesn't rely on them?

Composer is used internally during development to manage Sugar's dependencies on third-party libraries and packages. The Composer files are shipped during development to manage Sugar's dependencies on third-party libraries and packages. The composer files are shipped with the product to give our customers

the ability to expand from them.

Is the Composer package required to upgrade a Sugar instance?

No. The Sugar upgrader ships, just like the installer, with all required code bundled together. The upgrade process will validate the present Composer configuration and verify if it is compatible with the upgrade. In the case of a custom configuration, the upgrade process will report any issues which need to be resolved by the system administrator before the upgrade can proceed.

Why are there no wildcards in the version constraints? Doesn't composer.lock keep track of exact version numbers?

The lock file is designed to lock the dependencies to a specific version, but to protect our customers from unintentionally pulling in newer versions of dependencies owned by SugarCRM, we have chosen to use explicit version numbers in the composer.json file too.

May I change the version number of a package?

You cannot change version numbers for packages added by Sugar. Sugar will always configure exact version numbers for all of its dependencies. Changing these version numbers will result in an unsupported platform. The version constraints of packages not owned by Sugar may be modified at the developer's discretion.

Can I use Composer's autoloader?

We strongly recommend against using Composer's autoloader. The Sugar autoloader is fully compatible with the PSR-0 and PSR-4 autoloading recommendations from PHP-FIG, which makes the registration of an additional autoloader like the one from Composer redundant. Sugar's autoloader consumes the different mappings which are generated by Composer directly.

How can I optimize the autoloader?

Sugar ships with an optimized class map out of the box, which is pre-generated through Composer. This class map contains all different class to file mappings known to the dependencies managed by Composer. When customizing the Composer configuration, it is sufficient to run composer update --optimize-autoloader which will refresh the class map. SugarCRM's autoloader takes full advantage of this optimization.

Can I load customizations in Sugar through Composer?

Although not yet available out of the box, we are investigating this as a future capability.

Can I move the vendor directory out of the web root?

You cannot currently move the vendor directory, but we are investigating this as a future capability.

The Composer integration in Sugar is not flexible enough for me. What can I do?

We continuously strive to make our platform better and to facilitate both end users and developers. Our goal is to deliver a state-of-the-art environment. Do not hesitate to reach out to Sugar Support if we have overlooked your use case or if there too many constraints in the current implementation to make this a useful feature.

Delivery and Deployment Guide for Enterprises

Overview

SugarCRM Professional Services has a set of best practices for managing instances, delivering upgraded customizations, and deploying those upgraded customizations into Sugar on-site for our Enterprise customers. The following is an example of deployment practices used by SugarCRM Professional Services team when engaged on Enterprise Sugar development projects. It does not list all possible customizations that can be made in the system, it is intended to be used as a guide for how to automate the deployment of certain types of customizations into an on-premise Sugar instance. The techniques below cannot be used with Sugar's cloud service.

Deploying Application Configuration and Metadata

Use Case: Deployment of System Settings

System settings are stored in various places. In this section, we will address each type of storage for settings, and how to migrate each.

Storage types:

- **config_override.php**
  - This is a file stored in the Sugar root directory that allows for overriding core config values (found in config.php). In the UI, the main place to make changes to this is via Admin -> System Settings
- **database 'config' table**
  - It is loaded, used, and accessible throughout the Sugar application through the Config API.

    - System Tab Settings
    - Forecasting Settings
    - Portal Settings
    - This is a simple key/value/category store. There aren't too many components that use this. Here are some:

config_override.php

System considerations:

- File System:
    - *config_override.php is placed in Sugar web root directory*
- Scripts required
    - PHP
        - You will need to write a script to read the current config_override.php and merge the existing array with the new values you'd like to change. This can be done one time, and re-used for all future config_override.php changes

Steps to migrate:

1. Assess the values to be changed, added, or removed
2. Write a script to read the existing config_override.php, make the changes to the array, and re-write the file back to the system.

database 'config' table

System considerations:

- Database:
    - *The 'config' table stores all these values. They are stored in a very simple table schema.*
- Scripts required
    - PHP
        - Write a simple PHP script to use Sugar object API. See Figure 1

Steps to migrate:

1. Write script to use our object API for config table changes (See Figure 1)
2. Copy script to Sugar root directory
3. Execute the script
4. Remove the script

Screenshots:

Figure 1:

```php
<?php

define('sugarEntry', true);
require_once('include/entryPoint.php');

$administration = BeanFactory::getBean('Administration');
$administration->saveSetting('MySettings', 'disable_useredit', 'no');
```

Use Case: Deployment of Reports

The customer creates a report in the Reports module. They would like to deploy that report so that end users can all access and run it.

System considerations:

- Database
    - *Row is inserted into the Reports module (saved_reports table)*
    - *(For new team combinations) Row is inserted into the team_sets table*
    - *(For new team combinations) Rows are inserted into the team_sets_teams table*
- Scripts required
    - SQL
        - *Retrieve the relevant rows (saved_reports, team_sets, team_sets_teams) and create a SQL script to insert them.*

Additional notes:

- *In the System considerations section, "new combination of teams" means that when creating the report, the end user created the Report with a set of teams that doesn't exist on any other record. This results in new entries in the team_sets and team_sets_teams tables.*

Steps to migrate:

1. Build a report in the dev instance
2. Select the database rows associated with that report (saved_reports, possibly team_sets and team_sets_teams tables)
    1. example: "SELECT * FROM saved_reports WHERE id = 'REPORT_ID'"
3. Export the row/rows into a SQL file
4. Execute on the next system

Use Case: Deployment of Dashboards

The customer wants to deploy the pre-built dashboards in an automated way. See Figure 1 below. This example has two dashboards, "Help Dashboard" and "My Dashboard". Each dashboard has zero or more dashlets.

System considerations:

- Database:
    - *Row is inserted into the **dashboards** table for each user dashboard. The metadata column stores all the dashlets associated with that dashboard, and the assigned_user_id column stores the user who will see this dashboard.*
- Scripts required
    - PHP
        - *After deploying the **dashboards** you will need to run the **Quick Repair and Rebuild** script found in the admin section.*
        - Custom scripts: **YES (if applying to multiple users)**
            - Because users and id are dynamic, if applying to multiple users, you will need a custom script to retrieve those user ids and set them for each sql insert.
    - SQL
        - *Script required to import the entry from the **dashboards** table.*

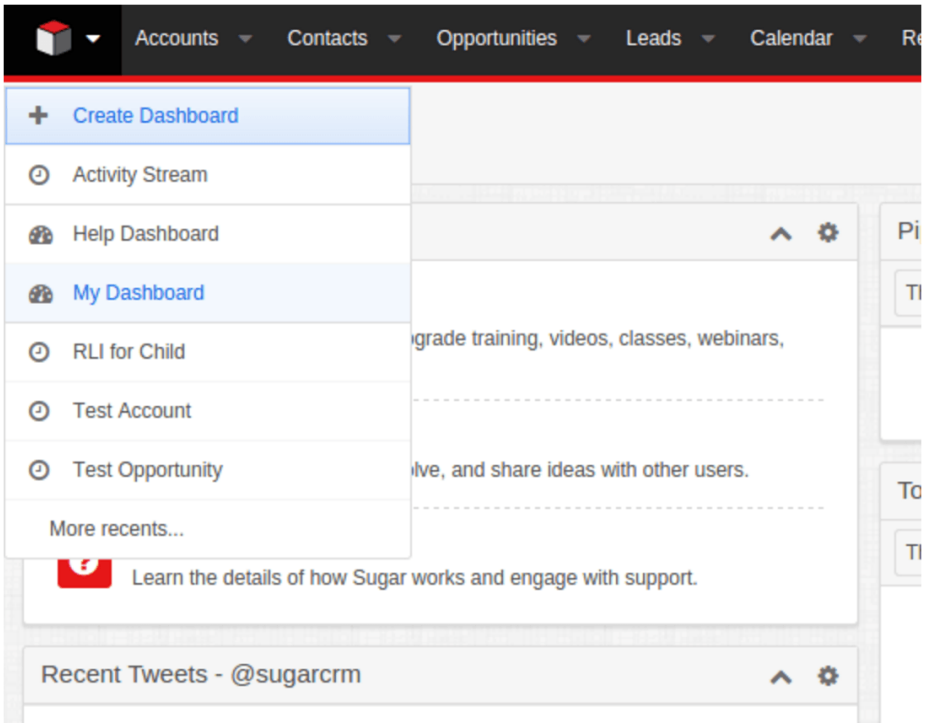
Steps to migrate:

1. Build a dashlet against a specific user
2. Select the database rows associated with that user
    1. example: "SELECT * FROM dashlets WHERE assigned_user_id = 'USER_ID'"
3. Pick the dashboard you'd like to apply to other users, and export it into a SQL file
4. Decide what set of users you need to create the dashboard for.

5. Write a script to pull that list of users, dynamically set the assigned_user_id and id (id must be unique) with the insert query you exported in step 3, and run for each one of those users.

Screenshots:

Figure 1



Use Case: Deployment of Roles

The customer wants to deploy the roles in an automated way. This includes creating new roles and updating previously existing roles.

System considerations:

- Database:
  - *Row is inserted into the **acl_roles** table for each role setting. Depending on how specific the role is, we might have **acl_fields** and **acl_actions** mapped to roles through **acl_roles_actions***
- Scripts required
  - PHP
    - Sugar has a SugarACL object API that can be used to create, read, and write roles and role definitions.

Steps to migrate:

1. Write a script using our object API to create or write roles
    1. Define the metadata for the changes or additions to be made
    2. Write logic to add/update based on metadata
2. Execute script on dev instance and confirm changes
3. Use script to promote to next instance
4. Note: See functional sample script below
    1. https://gist.github.com/sadekbaroudi/3191513e2bbce2170326

Use Case: Deployment of Teams

The customer wants to deploy the teams in an automated way. This should be done via the Sugar object API.

System considerations:

- Scripts required
    - PHP
        - *Follow steps in "Additional Notes" section below for details on building Team scripts.*
        - *TeamSets are cached per user by SugarCache. SugarCache should be cleared after installing new teams.*

Additional notes:

To build a script to do this, see the following:

- modules/Teams/Save.php
    - This file is called when a user posts data through the form in the UI. This code should be replicated (until the Teams module is refactored).
- modules/Teams/Team.php
    - function save() - this should be called as part of the save
    - function mark_deleted() - this should be called on the object when you want to delete a team, be sure to make sure there are no related users before doing so.

Steps to migrate:

1. Write a script using our Teams object API
    1. Create needed team object
    2. Set appropriate data on object and/or POST data
    3. After saving, potentially add users to the team

2. Execute script on dev instance and confirm changes
3. Use script to promote to next instance

Use Case: Deployment of User Settings

The customer wants to deploy the user settings in an automated way. There are a couple of places where we store User settings.

Storage types:

- User Preferences

This is a key value pair with a serialized and then base64 encoded value. We store many user preferences, all encoded. These are non-critical settings, and can be blown away. However, doing so will require the user to reconfigure their preferences. The data is stored in the user_preferences table. This includes data such as: Subpanel display order, Timezone preferences, etc.

- Users module settings

These are direct values on the Users module (users table). Here we track persistent User attributes such as: Address, Phone number, etc

User preferences

System considerations:

- Database:
  - *Row is inserted into the **user_preferences** table for each user setting change.*

- Scripts required
  - PHP
    - In order to update values with a user's preferences, you would need to write a custom script to read, update, and rewrite to the user_preferences table

Steps to Migrate:

1. Write a script using our User Preferences API
   1. Query the database to retrieve the row for a given user
   2. base64 decode the value
   3. unserialize the value
   4. update the data required

5. serialize the data
6. base64 encode
7. rewrite the row to the database
8. (repeat for all applicable users)

1. (See modules/UserPreferences/UserPreference.php or modules/Users/User.php, specifically getPreference() and setPreference())
2. Load the User object
3. Call getPreference for the specified value
4. Make changes
5. Call setPreference for the specified value

1. Better performance method (direct database queries and updates):
2. More robust, but slower performance method (API):

User table

System considerations:

- Database:
    - *Users table is updated*
- Scripts required:
    - SQL
        - You can directly update the Users table directly, provided the data is not encoded or encrypted (like password).

Steps to Migrate:

1. Write a SQL script to update values in the users table based on need
2. Execute script on dev instance and confirm changes
3. Use script to promote to next instance

Use Case: Deployment of custom fields

A user wants to deploy custom fields created in an automated way. This includes anything created through Studio.

System considerations:

- File System:
    - *Files are potentially created in the following directories:*
        - *Custom field vardef: custom/Extension/modules/<module_name>/Ext/Vardefs/sugarfield_<field_name>.php*
        - *Custom field label (and app_list_string if necessary): ./custom/Extension/modules/Accounts/Ext/Language/en_us.lang.php*

- Database:
    - *The <module_name>_cstm table is created, if it doesn't already exist.*
    - *The field <field_name>_c is created against that table*

- Scripts required
    - A Quick Repair and Rebuild is required after copying the files and fields_meta_data table values.
    - SQL
        - You will need to insert the relevant entries from the fields_meta_data table

Steps to migrate:

1. Export the fields_meta_data entries for the custom fields into a script
2. Copy the files for the custom fields
3. Apply #1 and #2 to another system, and execute a **Quick Repair and Rebuild**
    1. Execute the DDL generated by the QRR above

Use Case: Deployment of custom modules

A developer creates a custom module and wants to deploy it, this use case refers to a basic module, because each additional feature (logic hooks, relationships,dependencies, etc) has its own deployment scenario.

System considerations:

- File System:
    - *./custom/Extension/application/Ext/Include/<package_name>.php*
    - *./modules/<new_module>/**
    - *./custom/modules/<new_module>/**
    - *./custom/themes/default/images/*<new_module>*.(gif/png)*
    - *./custom/Extension/modules/<new_module>/**
- Database:

        - *new tables <new_module> and <new_module>_audit*
            - *Note: the DDL gets generated by the Quick Repair and Rebuild script, at which point you can execute manually or automatically*
        - *fields_meta_data table*
            - *Note: this stores all the custom fields built through Studio (not Module Builder) after the module is deployed. Make sure you retrieve all rows from this table that apply to this module and create a SQL script to insert into the next system*
- Scripts required
    - *After deploying the custom module, you will need to run the **Quick Repair and Rebuild** script found in the admin section.*
    - SQL
        - *Script required to import the entry from the **<new_module>** and **<new_module>**_audit table**.***
        - *Script required to import fields_meta_data table entries for this module (if there are any)*

Additional notes:

- For a full custom module deployment scenario, this deployment scenario should be ran **first** then all of the extended module features deployment scenarios should be run:

Steps to migrate:

1. Copy all files listed in file system section above
2. Export fields_meta_data table entries as apply to the custom module (if any)
3. Run Quick Repair and Rebuild
    1. Either manually or automatically run the DDL output from QRR
4. Test functionality

Use Case: Deployment of custom Relationships

A user wants to deploy custom relationships created in an automated way. This includes anything created through Studio.

Follow the same instructions as for Custom Fields, but:

- Ignore the fields_meta_data table
- Be sure to consider the following:
    - custom/Extension/modules/<side_1_of_relationship>/Ext/
    - custom/Extension/modules/<side_2_of_relationship>/Ext/
    - custom/Extension/modules/relationships/

Otherwise, the same process applies.

Use Case: Deployment of custom View or Layout metadata (Web)

The user creates custom layouts and views for web from studio and wants to deploy them.

System considerations:

- File System:
  - ***Layouts:***
    - *./custom/modules/<module>/clients/<platform>/layouts/<layout>/ <layout>.php*
  - ***Views***
    - *Creating a layout from Studio actually augments the Sidecar view metadata instead of Sidecar layout metadata*
    - *./custom/modules/<module>/clients/<platform>/views/<layout>/ <layout>.php*
- Scripts required
  - *After deploying you will need to run the **Quick Repair and Rebuild** script found in the admin section.*

Additional notes:

- *For layouts(record) you have the option to simply save the modified layout. In this case the metadata for the layout can be found in ./custom/working/modules/<module>/<platform>/views/<view>/<view>.php*

- *Layouts created from studio create views in .custom/modules/<module>/views*
  *The created views can be **record** | **list** | **selection-list***

- *For the two popup layouts(created from studio), extra metadata is provided in the ./custom/modules/<module>/metadata/popupdefs.php*

Use Case: Deployment of custom View or Layout metadata (mobile)

The user creates custom layouts and views for mobile from studio and wants to deploy them.

System considerations:

- File System: YES
    - ***Layouts:***
        - *./custom/modules/<module>/clients/mobile/layouts/<layout>/ <layout>.php*
    - ***Views***
        - *./custom/modules/<module>/clients/mobile/views/<view>/ <view>.php*
- Database: NO
- Scripts required
    - *After deploying you will need to run the **Quick Repair and Rebuild** script found in the admin section.*

Additional notes:

- *Layouts and views are handled the same as with web. From Studio, you can augment **detail**, **edit** and **list** views.*

Deploying Application Code and Integrations

Use Case: Deployment of custom CSS (LESS)

In order to update branding, developers can deploy customized CSS.

System considerations:

- File System: YES
    - *./custom/themes/custom.less*
- Scripts required
    - *You will need to run the **Quick Repair and Rebuild** script found in the admin section to rebuild the Sugar CSS bundles.*

Use Case: Deployment of Logic Hooks and Web Logic Hooks

The developer creates custom logic hooks, they want to deploy them.

System considerations:

Logic Hook:

- File System: YES
    - ***application hooks** :*
        - *./custom/Extension/application/Ext/LogicHooks/<file>.php*
    - ***module specific hooks:***

        - ./custom/Extension/modules/<module>/Ext/LogicHooks/<file>.php
- Scripts required
    - *You will need to run the **Quick Repair and Rebuild** script found in the admin section to rebuild the extensions.*

Web Logic Hook:

- Database: YES
    - ***Only for weblogic hooks :** row is inserted into the **weblogichooks** table**.***

- Scripts required
    - *After deploying the custom hooks and database entry in the **weblogichooks** table, you will need to run the **Quick Repair and Rebuild** script found in the admin section.*
    - SQL
        - *Script required to import the entry from the **weblogichooks** table**.***

Use Case: Deployment of custom API endpoints

The user creates a custom api endpoints, he wants to deploy them.

System considerations:

- File System
    - clients/<platform>/api/*
    - modules/:module/clients/<platform>/api/*
    - custom/clients/<platform>/api*
    - custom/modules/<module>/clients/<platform>/api/*
- Scripts required
    - *After deploying the custom api you will need to run the **Quick Repair and Rebuild** script found in the admin section. This will rebuild the ./cache/file_map.php and ./cache/include/api/ServiceDictionary.rest.php files to make your endpoint available.*

Additional notes:

- Logic for how api endpoints are loaded, can be found in ./include/api/ServiceDictionary.php (where api endpoints are loaded from, how they are built on Quick Build and Repair, etc.)

Use Case: Deployment of custom Administration Panels

The user creates custom administration panels, and wants to deploy them.

System considerations:

- File System:
  - *./custom/Extension/modules/Administration/Ext/Administration/<file>.php*
  - *./custom/Extension/modules/Administration/Ext/Language/<langtype.name>.php*
  - *./custom/themes/default/images/<icon_image_name>.<img_extension>*

and depending on the ***admin panel url, either*** *:*

  - *./custom/modules/<linkUrlModule>/**

or

  - *./custom/modules/<linkUrlModule>/clients/base/layouts/<route_name>/**
  - *./custom/modules/<linkUrlModule>/clients/base/views/<route_name>/**
- Scripts required
  - *After deploying the admin panels, you will need to run the* ***Quick Repair and Rebuild*** *script found in the admin section.*

Additional notes:

- The admin url can specify new sidecar routes, old bwc routes, edit view files, plain scripts, or just open a drawer. Determining the additional resources to be copied may be impossible without a standard in place.

Use Case: Deployment of custom Jobs / Schedulers

The user creates custom jobs and schedulers, and wants to deploy them.

System considerations:

- File System: **YES**:
  - *./custom/Extension/modules/Schedulers/Ext/ScheduledTasks/<jobname>.php*
  - *./custom/Extension/modules/Schedulers/Ext/Language/<langtype.jobname>.php*
- Database: **YES**
  - *Row is inserted into the* ***schedulers*** *table, if a job is added as a scheduled job using Administration > Scheduler.*

- Scripts required
  - *After deploying the custom jobs, you will need to run the **Quick Repair and Rebuild** script found in the admin section.*
  - **SQL**
    - *Script required to import the entry from the **schedulers** table.*

Sugar 11.3 to 12.0 Migration Guide

Overview

The purpose of this document is to provide insight to Sugar Developers for upgrading custom Sugar code, extensions, and integrations to the Sugar 12.0 (Q2 2022) release.

Cloud and On-Site Sugar Release

For those looking to upgrade from Sugar 11.0, you will be catching up with additional content released in the 11.1 (Q3 2021), 11.2 (Q4 2021), and 11.3 (Q1 2022) cloud-only Sugar releases. In addition to this guide, please review the Sugar 11.1, Sugar 11.2, and Sugar 11.3 migration guides and the release notes linked below to get a full view of changes since Sugar 11.0.

REST API Version Number

The current version for the Sugar REST API is 11_16.

Changes That May Affect Developers

The changes in Sugar 12.0 (Q2 2022) that could cause an immediate impact on customizations and integrations that were built for earlier versions of Sugar are highlighted in the following Developer Blog post in the SugarClub community:

- Sugar 12.0 (Q2 2022) Customization Guide

Related Documents

For more information about what changed in Sugar 12.0.x, please refer to the following related document(s):

- What to Expect When Upgrading to Sugar 12.0 (Q2 2022)
- 12.0.0 Upgrade Paths
- 12.0 Installation and Upgrade Guide

- 12.0.x Supported Platforms
- Release Notes
    - Sugar Sell 12.0.0 Release Notes
    - Sugar Serve 12.0.0 Release Notes
    - Sugar Enterprise 12.0.0 Release Notes

User Interface

Overview

Sugar's user interface is dependent on the client (i.e. base, mobile, or portal) being used to access the system. Clients are the various platforms that use Sugar's APIs to render the user interface. Each platform type will have a specific path for its components. While the Developer Guide mainly covers the base client type, the following sections will outline the various metadata locations.

Clients

Clients are the various platforms that access and use Sidecar to render content. Depending on the platform you are using, the layout, view, and metadata will be driven based on its client type. The following sections describe the client types.

base

The base client is the Sugar application that you use to access your data from a web browser. The framework's specific views, layouts, and fields are rendered using Sidecar. Files specific to this client type can be found in the following directories:

- ./clients/base/
- ./custom/clients/base/
- ./modules/<module>/clients/base/
- ./custom/modules/<module>/clients/base/

mobile

The mobile client is the SugarCRM mobile application that you use to access data from your mobile device. The framework-specific views, layouts, and fields for this application are found in the following directories:

- ./clients/mobile/
- ./custom/clients/mobile/
- ./modules/<module>/clients/mobile/

- ./custom/modules/<module>/clients/mobile/

portal

The portal client is the customer self-service portal application that comes with Sugar Enterprise and Sugar Ultimate. The framework-specific views, layouts, and fields for this application are found in the following directories:

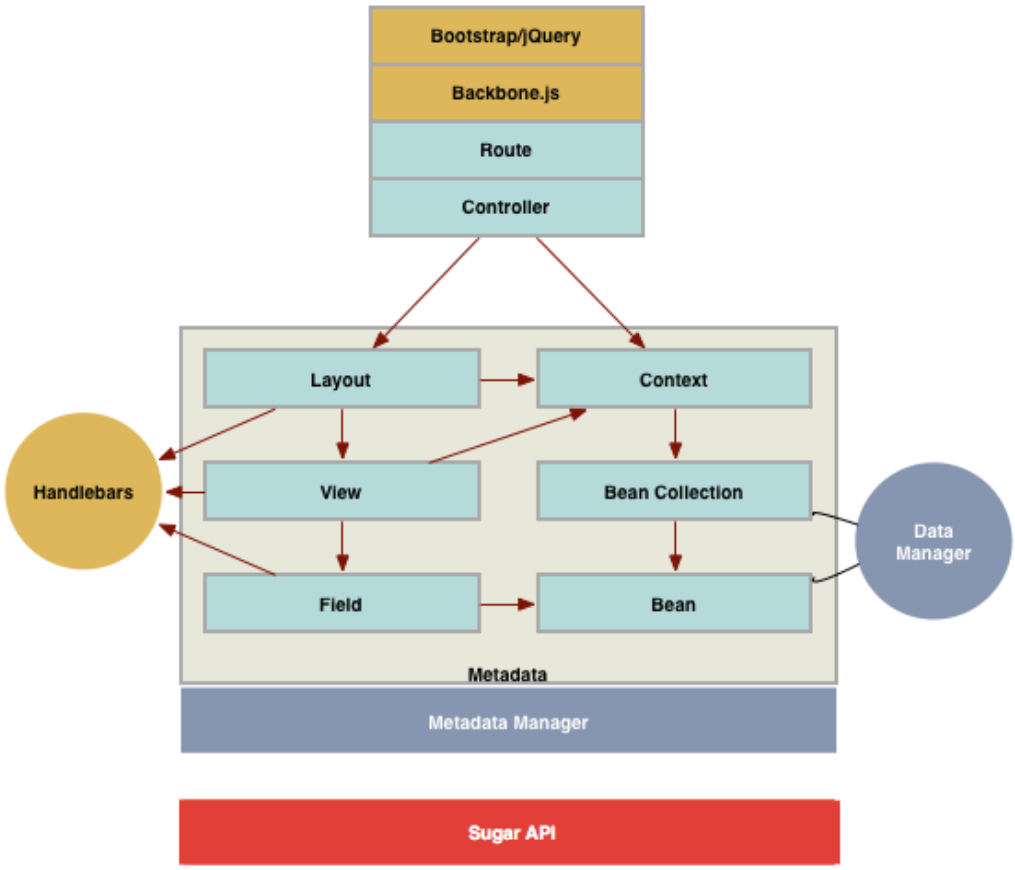
- ./clients/portal/
- ./custom/clients/portal/
- ./modules/<module>/clients/portal/
- ./custom/modules/<module>/clients/portal/

Sidecar

Overview

Sidecar is a platform that moves processing to the client side to render pages as single-page web apps. Sidecar contains a complete Model-View-Controller (MVC) framework based on the Backbone.js library.

By creating a single-page web app, server load drastically decreases while the client's performance increases because the application is sending pure JSON data in place of HTML. The JSON data, returned by the v10 API, defines the application's modules, records, and ACLs, allowing UI processing to happen on the client side and significantly reducing the amount of data to transfer.



Composition

Sidecar contains the following parts, which are briefly explained in the sections below:

- Backbone.js
- Components (Layouts, Views, and Fields)
- Context

Backbone.js

Backbone.js is a lightweight JavaScript framework based on MVP (model–view–presenter) application design. It allows developers to easily interact with a RESTful JSON API to fetch models and collections for use within their user interface.

For more information about Backbone.js, please refer to their documentation at Backbone.js.

Components

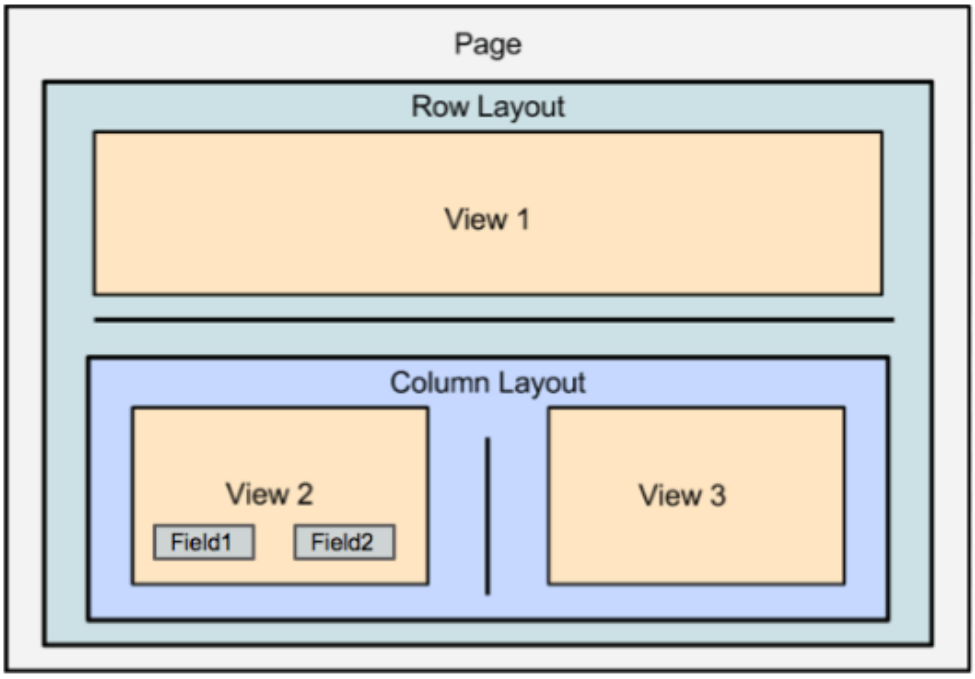
Everything that is renderable on the page is a component. A layout is a component that serves as a canvas for one or more views and other layouts. All pages will have at least one master layout, and that master layout can contain multiple nested layouts.

Layouts

Layouts are components that render the overall page. They define the rows, columns, and fluid layouts of content that gets delivered to the end user.

Example layouts include:

- Rows
- Columns
- Bootstrap fluid layouts
- Drawers and dropdowns



For more information about the various layouts, please refer to the Layouts page of this documentation.

Views

Views are components that render data from a context and may or may not include field components. Example views include not only record and list views but also widgets such as:

- Graphs or other data visualizations
- External data views such as Twitter, LinkedIn, or other web service integrations
- The global header

For more information about views, please refer to the Views page of this documentation.

Fields

Fields render widgets for individual values that have been pulled from the models and also handle formatting (or stripping the formatting of) field values. Like layouts and views, fields extend Backbone views.

For more information about the various layouts, please refer to the Fields page of this documentation.

Context

A Context is a container for the relevant data for a page, and it has three major attributes:

- **Module** : The name of the module this context is based on
- **Model** : The primary or selected model for this context
- **Collection** : The set of models currently loaded in this context

Contexts are used to retrieve related data and to paginate through lists of data.

Events

Overview

The Backbone events module is a lightweight pub-sub pattern that gets mixed into each Backbone class (Model, View, Collection, Router, etc.). This means that you can listen to or dispatch custom named events from any Backbone object.

Backbone events should not be confused with a jQuery events, which are used for working with DOM events in an API. Backbone supports an events hash on views that can be used to attach event handlers to DOM using jQuery. These are not Backbone events. This can be confusing because, among other similarities, both interfaces include an on() function and allow you to attach an event handler. The targets for jQuery events are DOM elements. The target for Backbone events are Backbone objects.

Sidecar classes extend these base Backbone classes. So each Sidecar object (Layouts, Views, Fields, Beans, Contexts, etc.) supports Backbone events.

Existing Backbone Event Catalog

The current catalog of Backbone events is supported and triggered by the Sugar application. For example, we can listen to built-in Backbone router events, such as the route event, that is triggered by Sidecar. Try running the following JavaScript code from your browser's console:

```
  SUGAR.App.router.on('route', function(arguments) {
    console.log(arguments);
});
```

As you click through the Sugar application, each time the router is called, you will see routing events appear in your browser console.

Sidecar Events

Global Application Events

Application events are all triggered on the app.events (SUGAR.App.events) object. Below is a list of application events with a description of when you can expect them to fire. However, please note that these events can be triggered in more than one place and some events, such as app:sync:error, can trigger events such as app:logout.

Name	Description
app:init	Triggered after the Sidecar application initializes **Note**: Initialization registers events, builds out Sidecar API objects, loads public metadata and config, and initializes modules.
app:start	Triggered after the Sidecar application starts
app:sync	Triggered when metadata is being synced with the user interface, for example, after login has occurred
app:sync:complete	Triggered after metadata has completely synced

app:sync:error	Triggered when metadata sync fails
app:sync:public:error	Triggered when public metadata sync fails during initialization
app:view:change	Triggered when a new view is loaded
app:locale:change	Triggered when the locale changes
lang:direction:change	Triggered when the locale changes and the direction of the language is different
app:login	Triggered when the "Login" route is called
app:login:success	Triggered after a successful login
app:logout	Triggered when the application is logging out
app:logout:success	Triggered after a successful logout

Bean Events

The following table lists bean object events.

Name	**Description**
acl:change	Triggered when the ACLs change for that module
acl:change:<fieldName>	Triggered when the ACLs change for a particular field in that module
validation:success	Triggered when bean validation is valid
validation:complete	Triggered when bean validation completes
error:validation	Triggered when bean validation has an error
error:validation:<fieldName>	Triggered when a particular field has a bean validation error
attributes:revert	Triggered when the bean reverts to the previous attributes

Context Events

The context object is used to facilitate communication between different Sidecar components on the page using events. For example, the button:save_button:click event triggers whenever a user clicks the Save button. The record view uses this event to run Save routines without being tightly coupled to a particular Save button. A list of these contextual events is not plausible because the user interface

is continuously changing between versions, and there are many more possibilities based on the views and layouts in each version.

Utilizing Events

Application events can be bound to in any custom JavaScript controller or any JavaScript file loaded into Sugar and included on the page (such as via JSGroupings framework). An example below shows how one could add custom JavaScript code to trigger after the application log out.

./custom/include/javascript/myAppLogoutSuccessEvent.js

```
  (function(app){
    app.events.on('app:logout:success', function(data) {
        //Add Logic Here
        console.log(data);
    });
})(SUGAR.App);
```

With the custom event JavaScript file written and in place, include it into the system using the JSGroupings extension.

./custom/Extension/application/Ext/JSGroupings/myAppLogoutSuccessEvent.php

```
  foreach ($js_groupings as $key => $groupings) {
    foreach  ($groupings as $file => $target) {
     //if the target grouping is found
        if ($target == 'include/javascript/sugar_grp7.min.js') {
            //append the custom JavaScript file
            $js_groupings[$key]['custom/include/javascript/myAppLogout
SuccessEvent.js'] = 'include/javascript/sugar_grp7.min.js';
        }
        break;
    }
}
```

Once in place, navigate to Admin > Repair > Rebuild JS Grouping Files. After the JSGroupings are rebuilt, clear your browser cache and the custom JavaScript will now trigger after a successful logout.

Routes

Overview

Routes determine where users are directed based on patterns in the URL.

Routes

Routes, defined in ./include/javascript/sugar7.js, are URL patterns signified by a hashtag ("#") in the URL. An example module URL pattern for the Sugar application is http://{site url}/#<module>. This route would direct a user to the list view for a given module. The following sections will outline routes and how they are defined.

Route Definitions

The router accepts route definitions in the following format:

```
routes = [
    {
        name: "My First Route",
        route: "pattern/to/match",
        callback: function()
        {
            //handling logic here.
        }
    },
    {
        name: "My Second Route",
        route: "pattern/:variable",
        callback: "<callback name>"
    }
]
```

A route takes in three properties: the name of the route, the route pattern to match, and the callback to be called when the route is matched. If a default callback is desired, you can specify the callback name as a string.

Route Patterns

Route patterns determine where to direct the user. An example of routing is done when navigating to an account record. When doing this you may notice that your

URL is:

```
http://{site url}/#Accounts/aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee
```

A stock route's definition is defined in ./include/javascript/sugar7.js as:

```
{
    name: "record",
    route: ":module/:id"
},
```

Variables in the route pattern are prefixed with a colon such as :variable. The route pattern above contains two variables:

- module
- id

Custom Routes

As of 7.8.x, you can add the routes during the initialization of the Router, so custom routes can be registered in the Sidecar router during both router:init and router:start events. It is recommended to register them in the Initialization event before any routing has occurred. There are two methods in the Sidecar Router, which allow for adding custom routes:

route()

Arguments

Name	Required	Type	Description
route	true	string	The Route pattern to be matched by the URL Fragment
name	true	string	The unique name of the Route
callback	true	function	The callback function for the Route

Example

The following example registers a custom Route during the router:init event, using the route() method.

./custom/javascript/customRoutes.js

```
(function(app){
    app.events.on("router:init", function(){
      //Register the route #test/123
      app.router.route("test/:id", "test123", function() {
          console.log(arguments);
          app.controller.loadView({
              layout: "custom_layout",
              create: true
          });
      });
    })
})(SUGAR.App);
```

addRoutes()

When you need to add multiple routes, you can define the routes in an array and pass the entire array to the addRoutes() method on the Sidecar Router to ensure they are registered. Please note, the addRoutes() method utilizes the above route() method to register the routes with the Backbone Router. The Backbone router redirects after the first matching route. Due to this, the order in which the routes are listed in the array is important as it will determine which route will be used by the application. It is recommended that the most specific routes be listed first in the array so that they are recognized first, instead of those routes which may contain a variable.

Arguments

Name	Required	Type	Description
routes	true	array	An array of route definition as defined above.

Example

The following example registers custom Route during the router:init event, using the addRoutes() method.

The route's JavaScript controller can exist anywhere you'd like. For our purposes, we created it in ./custom/javascript/customRoutes.js. This file will contain your

custom route definitions.

./custom/javascript/customRoutes.js

```
(function(app){
    app.events.on("router:init", function(){
        var routes = [
         {
                route: 'test/doSomething',
                name: 'testDoSomething',
                callback: function(){
                  alert("Doing something...");
                }
            },
            {
                route: 'test/:id',
                name: 'test123',
                callback: function(){
                  console.log(arguments);
                    app.controller.loadView({
                        layout: "custom_layout",
                        create: true
                    });
                }
            }
        ];
        app.router.addRoutes(routes);
    })
})(SUGAR.App);
```

Next, create the JSGrouping extension. This file will allow Sugar to recognize that you've added custom routes.

./custom/Extension/application/Ext/JSGroupings/myCustomRoutes.php

```
<?php

foreach ($js_groupings as $key => $groupings) {
    $target = current(array_values($groupings));

    //if the target grouping is found
    if ($target == 'include/javascript/sugar_grp7.min.js') {
        //append the custom JavaScript file
        $js_groupings[$key]['custom/javascript/customRoutes.js'] = 'in
```

```
clude/javascript/sugar_grp7.min.js';
    }
}
```

Once your files are in place, navigate to Admin > Repairs > Quick Repair & Rebuild. For additional information, please refer to the JSGroupings framework.

Handlebars

Overview

The Handlebars library, located in ./sidecar/lib/handlebars/, is a JavaScript library that lets Sugar create semantic templates. Handlebars help render content for layouts, views, and fields for Sidecar. Using Handlebars, you can make modifications to the display of content such as adding HTML or CSS.

For more information on the Handlebars library, please refer to their website at http://handlebarsjs.com.

Templates

The Handlebars templates are stored in the filesystem as .hbs files. These files are stored along with the view, layout, and field metadata and are loaded according to the inheritance you have selected in your controller. To view the list of available templates, or to see if a custom-created template is available, you can open your browser's console window and inspect the Handlebars.templates namespace.

Debugging Templates

When working with Handlebar templates, it can be difficult to identify where an issue is occurring or what a variable contains. To assist with troubleshooting this, you can use the log helper. The log helper will output the contents of this and the variable passed to it in your browser's console.

This is an example of using the logger in a handlebars template:

```
  {{log this}}
```

Helpers

Handlebar Helpers are a way of adding custom functionality to the templates. Helpers are located in the following places:

- ./sidecar/src/view/hbs-helpers.js : Sidecar uses these helpers by default
- ./include/javascript/sugar7/hbs-helpers.js : Additional helpers used by the base client

Creating Helpers

When working with Handlebar templates, you may need to create your helper. To do this, follow these steps:

1. Create a Handlebars helper file in the ./custom/ directory. For this example, we will create two functions to convert a string to uppercase or lowercase:

./custom/JavaScript/my-handlebar-helpers.js

```
  /**
 * Handlebars helpers.
 *
 * These functions are to be used in handlebars templates.
 * @class Handlebars.helpers
 * @singleton
 */
(function(app) {
    app.events.on("app:init", function() {

        /**
         * convert a string to upper case
         */
        Handlebars.registerHelper("customUpperCase", function (text)
        {
            return text.toUpperCase();
        });

        /**
         * convert a string to lower case
         */
        Handlebars.registerHelper("customLowerCase", function (text)
        {
            return text.toLowerCase();
        });

    });
})(SUGAR.App);
```

2. Next, create a JSGrouping extension in ./custom/Extension/application/Ext/JSGroupings/. Name the file uniquely for your customization. For this example, we will create:

./custom/Extension/application/Ext/JSGroupings/my-handlebar-helpers.php

```
  <?php

//Loop through the groupings to find include/javascript/sugar_grp7.min
.js
foreach($js_groupings as $key => $groupings) {
    foreach($groupings as $file => $target) {
        if ($target == 'include/javascript/sugar_grp7.min.js') {
            //append the custom helper file
            $js_groupings[$key]['custom/JavaScript/my-handlebar-
helpers.js'] = 'include/javascript/sugar_grp7.min.js';
        }

        break;
    }
}
```

3. Finally, navigate to Admin > Repair and perform the following two repair sequences to include the changes:
    - Quick Repair and Rebuild
    - Rebuild JS Groupings.

You can now use your custom helpers in the HBS files by using:

```
  {{customUpperCase "MyString"}}
{{customLowerCase "MyString"}}
```

Note: You can also access the helpers function from your browsers developer console using Handlebars.helpers

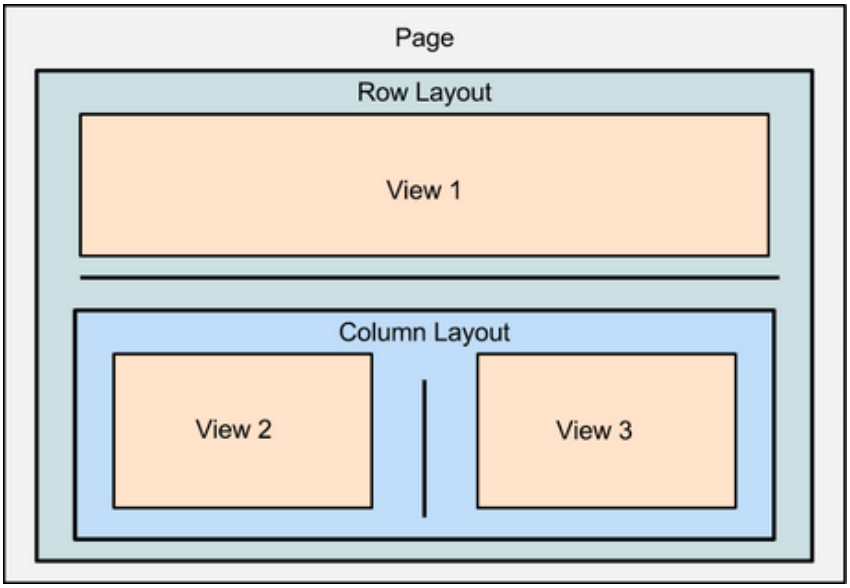
---

Layouts

Overview

Layouts are component plugins that define the overall layout and positioning of the page. Layouts replace the previous concept of MVC views and are used system-wide to generate rows, columns, bootstrap fluid layouts, and pop-ups by wrapping

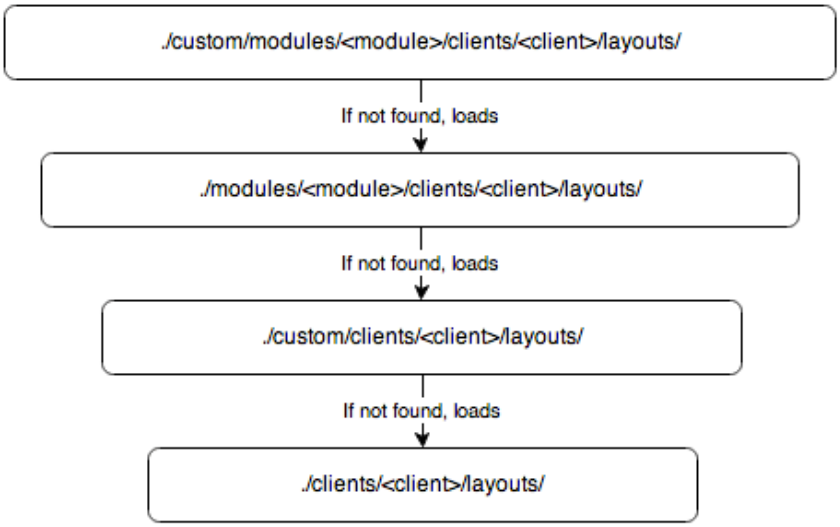
and placing multiple views or nested layouts on a page.



Layout components are typically made up of a controller JavaScript file (.js) and a PHP file (.php), however, layout types vary and are not dependent on having both files.

Hierarchy Diagram

The layout components are loaded in the following manner:



Note: The Sugar application client type is "base". For more information on the various client types, please refer to the User Interface page.

Sidecar Layout Routing

Sidecar uses routing to determine where to direct the user. To route the user to a specific page in Sugar, refer to the following default URL formats:

Behavior	URL Format
Route the user to the list layout for a module	http://{site url}/#<module>/
Route the user to the record layout for a specific record	http://{site url}/#<module>/f82d09cb-48cd-a1fb-beae-521cf39247b5
Route the user to a custom layout for the module	http://{site url}/#<module>/layout/<layout>

Layout Example

The list layout, located in ./clients/base/layouts/list/, handles the layout for the list view. The sections below outline the various files that render this view.

JavaScript

The file list.js, shown below, contains the JavaScript used to place the layout content.

./clients/base/layouts/list/list.js

```
/**
* Layout that places components using bootstrap fluid layout divs
* @class View.Layouts.ListLayout
* @extends View.FluidLayout
*/

({
    /**
     * Places a view's element on the page.
     * @param {View.View} comp
     * @protected
     * @method
     */

    _placeComponent: function(comp, def) {
        var size = def.size || 12;

        // Helper to create boiler plate layout containers
        function createLayoutContainers(self) {
            // Only creates the containers once
            if (!self.$el.children()[0]) {
```

```
                comp.$el.addClass('list');
            }
        }
        createLayoutContainers(this);
        // All components of this layout will be placed within the
        // innermost container div.
        this.$el.append(comp.el);
    }
})
```

Layout Definition

The layout definition is contained as an array in list.php. This layout definition contains four views:

- massupdate
- massaddtolist
- recordlist
- list-bottom

./clients/base/layouts/list/list.php

```
<?php

$viewdefs['base']['layout']['list'] = array(
    'components' =>
    array(
        array(
            'view' => 'massupdate',
        ),
        array(
            'view' => 'massaddtolist',
        ),
        array(
            'view' => 'recordlist',
            'primary' => true,
        ),
        array(
            'view' => 'list-bottom',
        ),
    ),
    'type' => 'simple',
    'name' => 'list',
    'span' => 12,
```

```
);
```

Application

For information on working with layouts, please refer to the Creating Layouts and Overriding Layouts pages for practical examples.

Creating Layouts

Overview

This example explains how to create a custom layout to define the various components that load on a page.

Creating the Layout

This example creates a component named "my-layout", which will render a custom view named "my-view".

./custom/clients/base/layouts/my-layout/my-layout.php

```
<?php
    $viewdefs['base']['layout']['my-layout'] = array(
        'type' => 'simple',
        'components' => array(
            array(
                'view' => 'my-view',
            ),
        ),
    );
```

Creating a View

The view component will render the actual content we want to see on the page. The view below will display a clickable cube icon that will spin when clicked by the user.

./custom/clients/base/views/my-view/my-view.js

```
({
    className: 'my-view tcenter',
    cubeOptions: {
        spin: false
    },
    events: {
        'click .sugar-cube': 'spinCube'
    },
    spinCube: function() {
        this.cubeOptions.spin = !this.cubeOptions.spin;
        this.render();
    }
})
```

./custom/clients/base/views/my-view/my-view.hbs

```
<style>
    div.my-view
    {
        padding-top: 5%;
    }
    div.my-view .sugar-cube
    {
        fill:#bbbbbb;
        height:200px;
        width:200px;
        display: inline;
    }

</style>

<h1>My View</h1>

{{{subFieldTemplate 'sugar-cube' 'detail' cubeOptions}}}

<p>Click to spin the cube!</p>
```

Once the files are in place, navigate to Admin > Repair and perform a Quick Repair and Rebuild.

Navigating to the Layout

To see this new layout and view, navigate to http://{site url}/#<module>/layout/my-layout.

Overriding Layouts

Overview

This page explains how to override a stock layout component. For this example, we will extend the stock record view and create a custom view named "my-record" that will be used in our record layout's override. This example involves two steps:

1. Override the Layout
2. Extend the View

These steps are explained in the following sections.

Overriding the Layout

First, copy ./clients/base/layouts/record/record.php to ./custom/clients/base/layouts/record/record.php. Once copied, modify the following line from:

```
  'view' => 'record',
```

To:

```
'view' => 'my-record',
```

That line will change the record layout from using the base record.js view, ./clients/base/views/record/record.js, to instead use a custom view that we will create in ./custom/clients/base/views/my-record/my-record.js. At this point, the custom layout override should be very similar to the example below:

./custom/clients/base/layouts/record/record.php

```
  <?php

$viewdefs['base']['layout']['record'] = array(
```

```
    'components' => array(
        array(
            'layout' => array(
                'type' => 'default',
                'name' => 'sidebar',
                'components' => array(
                    array(
                        'layout' => array(
                            'type' => 'base',
                            'name' => 'main-pane',
                            'css_class' => 'main-pane span8',
                            'components' => array(
                                array(
                                    'view' => 'my-record',
                                    'primary' => true,
                                ),
                                array(
                                    'layout' => 'extra-info',
                                ),
                                array(
                                    'layout' => array(
                                        'type' => 'filterpanel',
                                        'last_state' => array(
                                            'id' => 'record-
filterpanel',
                                            'defaults' => array(
                                                'toggle-
view' => 'subpanels',
                                            ),
                                        ),
                                        'refresh_button' => true,
                                        'availableToggles' => array(
                                            array(
                                                'name' => 'subpanels',
                                                'icon' => 'fa-table',
                                                'label' => 'LBL_DATA_V
IEW',
                                            ),
                                            array(
                                                'name' => 'list',
                                                'icon' => 'fa-table',
                                                'label' => 'LBL_LISTVI
EW',
                                            ),
                                            array(
                                                'name' => 'activitystr
```

```
eam',
                                                    'icon' => 'fa-clock-
o',
                                                    'label' => 'LBL_ACTIVI
TY_STREAM',
                                                ),
                                            ),
                                            'components' => array(
                                                array(
                                                    'layout' => 'filter',
                                                    'xmeta' => array(
                                                        'layoutType' => ''
,
                                                    ),
                                                    'loadModule' => 'Filte
rs',
                                                ),
                                                array(
                                                    'view' => 'filter-
rows',
                                                ),
                                                array(
                                                    'view' => 'filter-
actions',
                                                ),
                                                array(
                                                    'layout' => 'activitys
tream',
                                                    'context' =>
                                                    array(
                                                        'module' => 'Activ
ities',
                                                    ),
                                                ),
                                                array(
                                                    'layout' => 'subpanels
',
                                                ),
                                            ),
                                        ),
                                    ),
                                ),
                            ),
                        ),
                        array(
                            'layout' => array(
```

```
                            'type' => 'base',
                            'name' => 'dashboard-pane',
                            'css_class' => 'dashboard-pane',
                            'components' => array(
                                array(
                                    'layout' => array(
                                        'type' => 'dashboard',
                                        'last_state' => array(
                                            'id' => 'last-visit',
                                        )
                                    ),
                                    'context' => array(
                                        'forceNew' => true,
                                        'module' => 'Home',
                                    ),
                                    'loadModule' => 'Dashboards',
                                ),
                            ),
                        ),
                    ),
                    array(
                        'layout' => array(
                            'type' => 'base',
                            'name' => 'preview-pane',
                            'css_class' => 'preview-pane',
                            'components' => array(
                                array(
                                    'layout' => 'preview',
                                ),
                            ),
                        ),
                    ),
                ),
            ),
        ),
    ),
);
```

Extending the View

For this example, we will extend the stock record view and create a custom view named my-record that will be used in our record layouts override.

./custom/clients/base/views/my-record/my-record.js

```
  ({
    extendsFrom: 'RecordView',

    initialize: function (options) {
        this._super("initialize", [options]);


        //log this point
        console.log("**** Override called");
    }
})
```

Once the files are in place, navigate to Admin > Repair > Quick Repair and Rebuild.

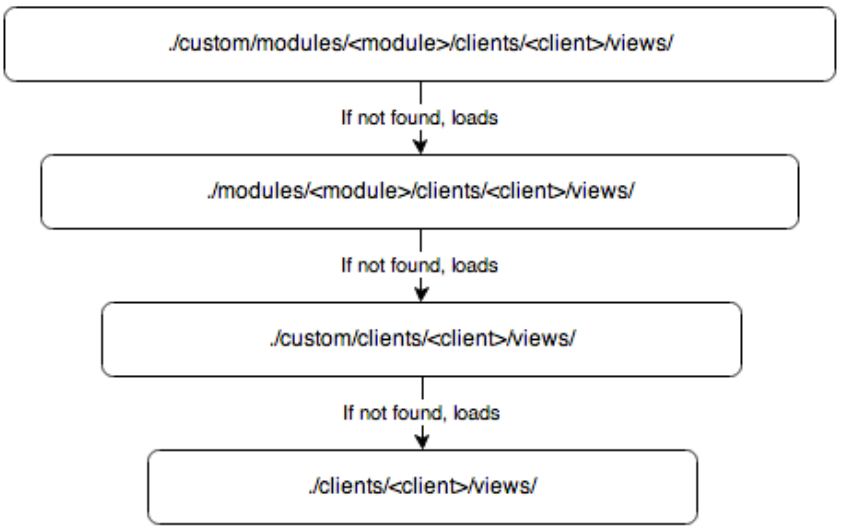
Views

Overview

Views are component plugins that render data from a context. View components may contain field components and are typically made up of a controller JavaScript file (.js) and at least one Handlebars template (.hbs).

Load Order Hierarchy Diagram

The view components are loaded in the following manner:



Note: The Sugar application's client type is "base". For more information on the various client types, please refer to the User Interface page.

Components

Views are made up of a controller and a Handlebar template.

Controller

The view's controller is what controls the view in how data is loaded, formatted, and manipulated. The controller is the JavaScript file named after the view. A controller file can be found in any of the directories shown in the hierarchy diagram above. In the example of the record view, the main controller file is located in ./clients/base/views/record/record.js and any modules extending this controller will have a file located in ./modules/<module>/clients/base/views/record/record.js.

Handlebar Template

The views template is built on Handlebars and is what adds the display markup for the data. The template is typically named after the view or an action in the view. In the example of the record view, the main template us located in ./clients/base/views/record/record.hbs. This template will take the data fetched from the REST API to render the display for the user. More information on templates can be found in the Handlebars section.

Extending Views

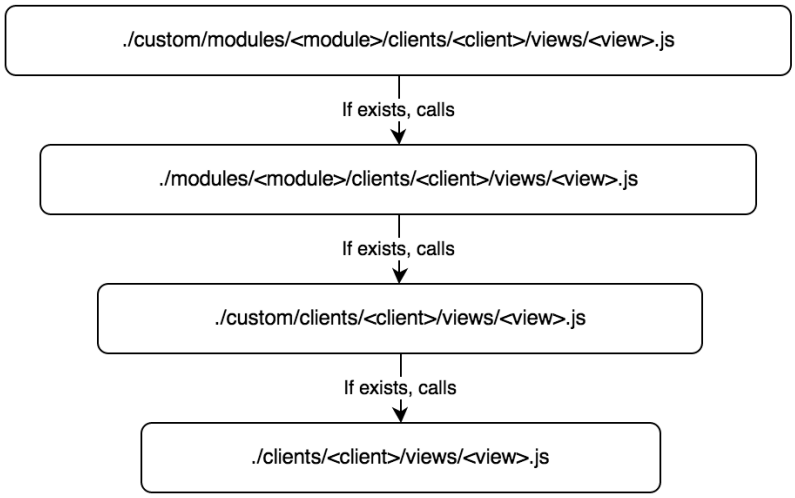
When working with module views, it is important to understand the difference between overriding and extending a view. Overriding is essentially creating or copying a view to be used by your application that is not extending its parent. By default, some module views already extend the core ./clients/base/views/record/record.js controller. An example of this is the accounts RecordView

./modules/Accounts/clients/base/views/record/record.js

```
({
  extendsFrom: 'RecordView',

  /**
   * @inheritdoc
   */
  initialize: function(options) {
      this.plugins = _.union(this.plugins || [], ['HistoricalSummary
```

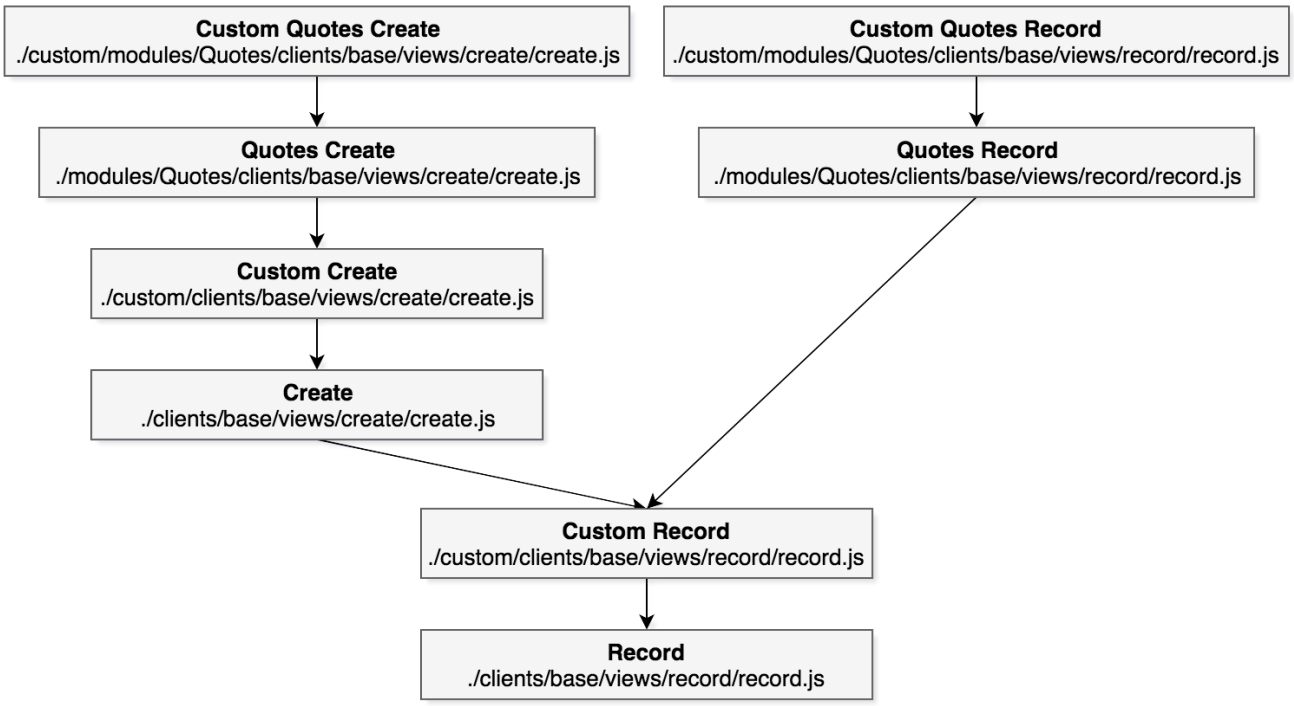
```
']);
        this._super('initialize', [options]);
    }
})
```

As you can see, this view has the property: extendsFrom: 'RecordView'. This property tells Sidecar that the view is going to extend its parent RecordView. In addition to this, you can see that the initialize method is also calling this._super('initialize', [options]);. Calling this._super tells Sidecar to execute the parent function. The major benefit of doing this is that any updates to ./clients/base/views/record/record.js will be reflected for the module without any modifications being made to ./modules/Accounts/clients/base/views/record/record.js. You should note that when using extendsFrom, the parent views are called similarly to the load hierarchy:



Create View and Record View Inheritance

The diagram below demonstrates the inheritance of the create and record views for the Quotes module. This inheritance structure is the same for stock and custom modules alike.



Basic View Example

A simple view for beginners is the access-denied view. The view is located in ./clients/base/views/access-denied/ and is what handles the display for restricted access. The sections below will outline the various files that render this view.

Controller

The access-denied.js, shown below, controls the manipulation actions of the view.

./clients/base/views/access-denied/access-denied.js

```
({
  className: 'access-denied tcenter',
  cubeOptions: {spin: false},
  events: {
      'click .sugar-cube': 'spinCube'
  },

  spinCube: function() {
      this.cubeOptions.spin = !this.cubeOptions.spin;
      this.render();
  }
```

```
})
```

Attributes

Attribute	Description
className	The CSS class to apply to the view.
cubeOptions	A set of options that are passed to the spinCube function when called.
events	A list of the view events. This view executes the spinCube function when the sugar cube is clicked.
spinCube	Function to control the start and stop of the cube spinning.

Handlebar Template

The access-denied.hbs file defines the format of the views content. As this view is used for restricting access, it displays a message to the user describing the restriction.

./clients/base/views/access-denied/access-denied.hbs

```
  <div class="error-message">
    <h1>{{str 'ERR_NO_VIEW_ACCESS_TITLE'}}</h1>
    <p>{{str 'ERR_NO_VIEW_ACCESS_REASON'}}</p>
    <p>{{str 'ERR_NO_VIEW_ACCESS_ACTION'}}</p>
</div>

{{{subFieldTemplate 'sugar-cube' 'detail' cubeOptions}}}
```

Helpers

Name	Description
str	Handlebars helper to render the label string
subFieldTemplate	Handlebars helper to render the cube content

Cookbook Examples

When working with views, you may find the follow cookbook examples helpful:

- Adding Buttons to the Record View
- Adding Field Validation to the Record View
- Passing_Data_to_Templates

Metadata

Overview

This page is an overview of the metadata framework for Sidecar modules.

View Metadata Framework

A module's view-specific metadata can be found in the modules view file:

```
./modules/<module>/clients/<client>/views/<view>/<view>.php
```

Any edits made in Admin > Studio will be reflected in the file:

```
./custom/modules/<module>/clients/<client>/views/<view>/<view>.php
```

Note: The Sugar application's client type is "base". For more information on the various client types, please refer to the User Interface page.

Note: In the case of metadata, custom view metadata files are respected over the stock view metadata files.

View Metadata

The Sidecar views metadata is very similar to that of the MVC metadata, however, there are some basic differences. All metadata for Sidecar follows the format:

```
$viewdefs['<module>']['base']['view']['<view>'] = array();
```

An example of this is the account's record layout shown below:

./modules/Accounts/clients/base/views/record/record.php

```php
<?php

$viewdefs['Accounts']['base']['view']['record'] = array(
    'panels' => array(
        array(
            'name' => 'panel_header',
            'header' => true,
            'fields' => array(
                array(
                    'name'          => 'picture',
                    'type'          => 'avatar',
                    'width'         => 42,
                    'height'        => 42,
                    'dismiss_label' => true,
                    'readonly'      => true,
                ),
                'name',
                array(
                    'name' => 'favorite',
                    'label' => 'LBL_FAVORITE',
                    'type' => 'favorite',
                    'dismiss_label' => true,
                ),
                array(
                    'name' => 'follow',
                    'label'=> 'LBL_FOLLOW',
                    'type' => 'follow',
                    'readonly' => true,
                    'dismiss_label' => true,
                ),
            )
        ),
        array(
            'name' => 'panel_body',
            'columns' => 2,
            'labelsOnTop' => true,
            'placeholders' => true,
            'fields' => array(
                'website',
                'industry',
                'parent_name',
                'account_type',
                'assigned_user_name',
                'phone_office',
            ),
        ),
```

```
        array(
            'name' => 'panel_hidden',
            'hide' => true,
            'columns' => 2,
            'labelsOnTop' => true,
            'placeholders' => true,
            'fields' => array(
                array(
                    'name' => 'fieldset_address',
                    'type' => 'fieldset',
                    'css_class' => 'address',
                    'label' => 'Billing Address',
                    'fields' => array(
                        array(
                            'name' => 'billing_address_street',
                            'css_class' => 'address_street',
                            'placeholder' => 'LBL_BILLING_ADDRESS_STRE
ET',
                        ),
                        array(
                            'name' => 'billing_address_city',
                            'css_class' => 'address_city',
                            'placeholder' => 'LBL_BILLING_ADDRESS_CITY
',
                        ),
                        array(
                            'name' => 'billing_address_state',
                            'css_class' => 'address_state',
                            'placeholder' => 'LBL_BILLING_ADDRESS_STAT
E',
                        ),
                        array(
                            'name' => 'billing_address_postalcode',
                            'css_class' => 'address_zip',
                            'placeholder' => 'LBL_BILLING_ADDRESS_POST
ALCODE',
                        ),
                        array(
                            'name' => 'billing_address_country',
                            'css_class' => 'address_country',
                            'placeholder' => 'LBL_BILLING_ADDRESS_COUN
TRY',
                        ),
                    ),
                ),
                array(
```

```
                    'name' => 'fieldset_shipping_address',
                    'type' => 'fieldset',
                    'css_class' => 'address',
                    'label' => 'Shipping Address',
                    'fields' => array(
                        array(
                            'name' => 'shipping_address_street',
                            'css_class' => 'address_street',
                            'placeholder' => 'LBL_SHIPPING_ADDRESS_STR
EET',
                        ),
                        array(
                            'name' => 'shipping_address_city',
                            'css_class' => 'address_city',
                            'placeholder' => 'LBL_SHIPPING_ADDRESS_CIT
Y',
                        ),
                        array(
                            'name' => 'shipping_address_state',
                            'css_class' => 'address_state',
                            'placeholder' => 'LBL_SHIPPING_ADDRESS_STA
TE',
                        ),
                        array(
                            'name' => 'shipping_address_postalcode',
                            'css_class' => 'address_zip',
                            'placeholder' => 'LBL_SHIPPING_ADDRESS_POS
TALCODE',
                        ),
                        array(
                            'name' => 'shipping_address_country',
                            'css_class' => 'address_country',
                            'placeholder' => 'LBL_SHIPPING_ADDRESS_COU
NTRY',
                        ),
                        array(
                            'name' => 'copy',
                            'label' => 'NTC_COPY_BILLING_ADDRESS',
                            'type' => 'copy',
                            'mapping' => array(
                                'billing_address_street' => 'shipping_
address_street',
                                'billing_address_city' => 'shipping_ad
dress_city',
                                'billing_address_state' => 'shipping_a
ddress_state',
```

```
                                'billing_address_postalcode' => 'shipp
ing_address_postalcode',
                                'billing_address_country' => 'shipping
_address_country',
                            ),
                        ),
                    ),
                ),
                array(
                    'name' => 'phone_alternate',
                    'label' => 'LBL_OTHER_PHONE',
                ),
                'email',
                'phone_fax',
                'campaign_name',
                array(
                    'name' => 'description',
                    'span' => 12,
                ),
                'sic_code',
                'ticker_symbol',
                'annual_revenue',
                'employees',
                'ownership',
                'rating',
                array(
                    'name' => 'date_entered_by',
                    'readonly' => true,
                    'type' => 'fieldset',
                    'label' => 'LBL_DATE_ENTERED',
                    'fields' => array(
                        array(
                            'name' => 'date_entered',
                        ),
                        array(
                            'type' => 'label',
                            'default_value' => 'LBL_BY',
                        ),
                        array(
                            'name' => 'created_by_name',
                        ),
                    ),
                ),
                'team_name',
                array(
                    'name' => 'date_modified_by',
```

```
                    'readonly' => true,
                    'type' => 'fieldset',
                    'label' => 'LBL_DATE_MODIFIED',
                    'fields' => array(
                        array(
                            'name' => 'date_modified',
                        ),
                        array(
                            'type' => 'label',
                            'default_value' => 'LBL_BY',
                        ),
                        array(
                            'name' => 'modified_by_name',
                        ),
                    ),
                ),
            ),
        ),
    ),
);
```

The metadata for a given view can be accessed using app.metadata.getView within your controller. An example fetching the view metadata for the Accounts RecordView is shown below:

```
app.metadata.getView('Accounts', 'record');
```

You should note that this can also be accessed in your browser's console window by using the global App Identifier:

```
App.metadata.getView('Accounts', 'record');
```

Fields

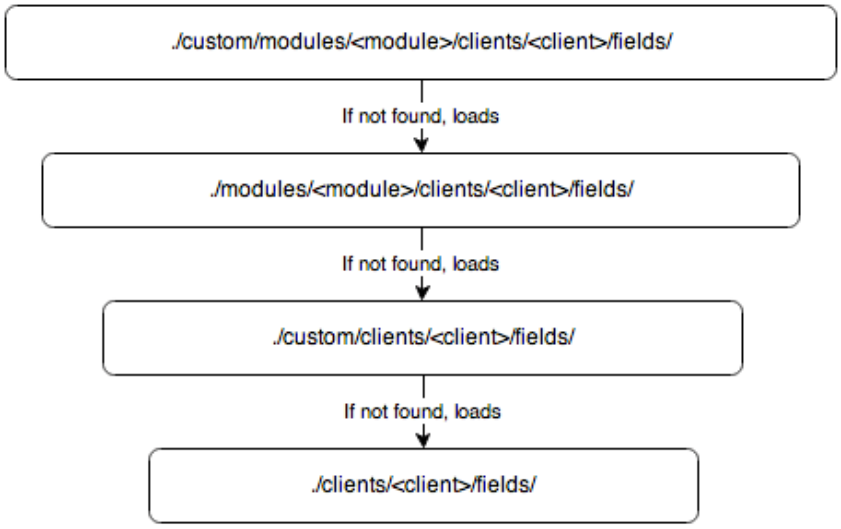
Overview

Fields are component plugins that render and format field values. They are made up of a controller JavaScript file (.js) and at least one Handlebars template (.hbt).

For more information regarding the data handling of a field, please refer the data framework fields documentation. For information on creating custom field types, please refer the Creating Custom Field Types cookbook example.

Hierarchy Diagram

The field components are loaded in the following manner:



Note: The Sugar application's client type is "base". For more information on the various client types, please refer to the User Interface page.

Field Example

The bool field, located in ./clients/base/fields/bool/, handles the display of checkbox boolean values. The sections below outline the various files that render this field type.

Controller

The bool.js, file is shown below, overrides the base _render function to disable the field. The format and unformat functions handle the manipulation of the field's value.

./clients/base/fields/bool/bool.js

```
({
  _render: function() {
      app.view.Field.prototype._render.call(this);

      if(this.tplName === 'disabled') {
```

```
            this.$(this.fieldTag).attr("disabled", "disabled");
        }
    },
    unformat:function(value){
        value = this.$el.find(".checkbox").prop("checked") ? "1" : "0"
;
        return value;
    },
    format:function(value){
        value = (value=="1") ? true : false;
        return value;
    }
})
```

Attributes

Attribute	Description
_render	Function to render the field.
unformat	Function to dynamically check the checkbox based on the value.
format	Function to format the value for storing in the database.

Handlebar Templates

The edit.hbs file defines the display of the control when the edit view is used. This layout is for displaying the editable form element that renders a clickable checkbox control for the user.

./clients/base/fields/bool/edit.hbs

```
  {{#if def.text}}
    <label>
        <input type="checkbox" class="checkbox"{{#if value}} checked{{
/if}}> {{str def.text this.module}}
    </label>
{{else}}
    <input type="checkbox" class="checkbox"{{#if value}} checked{{/if}
}>
{{/if}}
```

Helpers

Helpers	Description
str	Handlebars helper to render the label string.

The detail.hbs file defines the display of the control when the detail view is used. This layout is for viewing purposes only so the control is disabled by default.

./clients/base/fields/bool/detail.hbs

```
  <input type="checkbox" class="checkbox"{{#if value}} checked{{/if}}
disabled>
```

The list.hbs file defines the display of the control when the list view is used. This view is also for viewing purposes only so the control is disabled by default.

./clients/base/fields/bool/list.hbs

```
  <input type="checkbox" class="checkbox"{{#if value}} checked{{/if}}
disabled>
```

Cookbook Examples

When working with fields, you may find the follow cookbook examples helpful:

- Creating Custom Field Types
- Converting Address' Country Field to a Dropdown

Subpanels

Overview

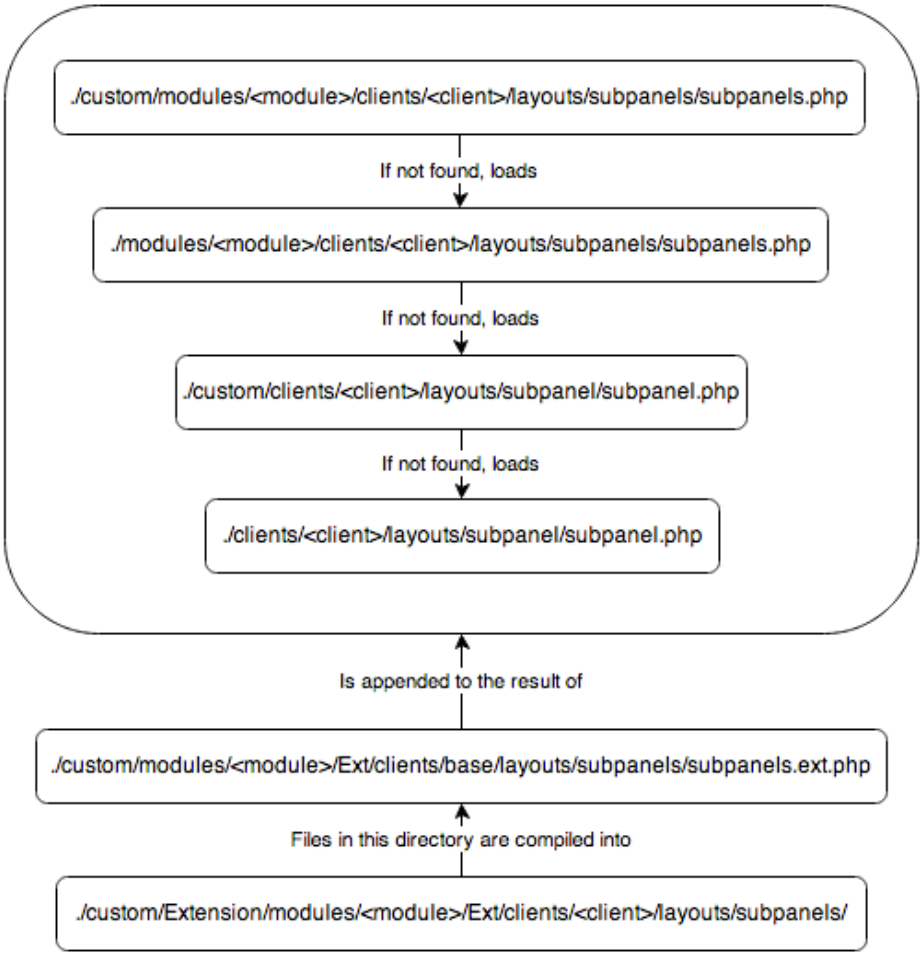
For Sidecar, Sugar's subpanel layouts have been modified to work as simplified metadata. This page is an overview of the metadata framework for subpanels.

The reason for this change is that previous versions of Sugar generated the metadata from various sources such as the SubPanelLayout and MetaDataManager classes. This eliminates the need for generating and processing the layouts and allows the metadata to be easily loaded to Sidecar.

Note: Modules running in backward compatibility mode do not use the Sidecar subpanel layouts as they use the legacy MVC framework.

Hierarchy Diagram

When loading the Sidecar subpanel layouts, the system processes the layout in the following manner:



Note: The Sugar application's client type is "base". For more information on the various client types, please refer to the User Interface page.

Subpanels and Subpanel Layouts

Sugar contains both a subpanels (plural) layout and a subpanel (singular) layout. The subpanels layout contains the collection of subpanels, whereas the subpanel layout renders the actual subpanel widget.

An example of a stock module's subpanels layout is:

./modules/Bugs/clients/base/layouts/subpanels/subpanels.php

```
<?php

$viewdefs['Bugs']['base']['layout']['subpanels'] = array (
  'components' => array (
      array (
          'layout' => 'subpanel',
          'label' => 'LBL_DOCUMENTS_SUBPANEL_TITLE',
          'context' => array (
              'link' => 'documents',
          ),
      ),
      array (
          'layout' => 'subpanel',
          'label' => 'LBL_CONTACTS_SUBPANEL_TITLE',
          'context' => array (
              'link' => 'contacts',
          ),
      ),
      array (
          'layout' => 'subpanel',
          'label' => 'LBL_ACCOUNTS_SUBPANEL_TITLE',
          'context' => array (
              'link' => 'accounts',
          ),
      ),
      array (
          'layout' => 'subpanel',
          'label' => 'LBL_CASES_SUBPANEL_TITLE',
          'context' => array (
              'link' => 'cases',
          ),
      ),
  ),
  'type' => 'subpanels',
  'span' => 12,
);
```

You can see that the layout incorporates the use of the subpanel layout for each module. As most of the subpanel data is similar, this approach allows us to use less duplicate code. The subpanel layout, shown below, shows the three views that make up the subpanel widgets users see.

./clients/base/layouts/subpanel/subpanel.php

```
<?php

$viewdefs['base']['layout']['subpanel']  = array (
    'components' => array (
        array (
            'view' => 'panel-top',
        )
        array (
            'view' => 'subpanel-list',
        ),
        array (
            'view' => 'list-bottom',
        ),
    ),
    'span' => 12,
    'last_state' => array(
        'id' => 'subpanel'
    ),
);
```

Adding Subpanel Layouts

When a new relationship is deployed from Studio, the relationship creation process will generate the layouts using the extension framework. You should note that for stock relationships and custom deployed relationships, layouts are generated for both Sidecar and Legacy MVC Subpanel formats. This is done to ensure that any related modules, whether in Sidecar or Backward Compatibility mode, display a related subpanel as expected.

Sidecar Layouts

Custom Sidecar layouts, located in ./custom/Extension/modules/<module>/Ext/clients/<client>/layouts/subpanels/, are compiled into ./custom/modules/<module>/Ext/clients/<client>/layouts/subpanels/subpanels.ext.php using the extension framework. When a relationship is saved, layout files are created for both the "base" and "mobile" client types.

For example, deploying a 1:M relationship from Bugs to Leads will generate the following Sidecar files:

./custom/Extension/modules/Bugs/Ext/clients/base/layouts/subpanels/bugs_leads_1

_Bugs.php

```
<?php

$viewdefs['Bugs']['base']['layout']['subpanels']['components'][] = arr
ay (
  'layout' => 'subpanel',
  'label' => 'LBL_BUGS_LEADS_1_FROM_LEADS_TITLE',
  'context' =>
  array (
    'link' => 'bugs_leads_1',
  ),
);
```

./custom/Extension/modules/Bugs/Ext/clients/mobile/layouts/subpanels/bugs_leads_1_Bugs.php

```
<?php

$viewdefs['Bugs']['mobile']['layout']['subpanels']['components'][] = a
rray (
  'layout' => 'subpanel',
  'label' => 'LBL_BUGS_LEADS_1_FROM_LEADS_TITLE',
  'context' =>
  array (
    'link' => 'bugs_leads_1',
  ),
);
```

Note: The additional legacy MVC layouts generated by a relationships deployment are described below.

Legacy MVC Subpanel Layouts

Custom Legacy MVC Subpanel layouts, located in ./custom/Extension/modules/<module>/Ext/Layoutdefs/, are compiled into ./custom/modules/<module>/Ext/Layoutdefs/layoutdefs.ext.php using the extension framework. You should also note that when a relationship is saved, wireless layouts, located in ./custom/Extension/modules/<module>/Ext/WirelessLayoutdefs/, are created and compiled into ./custom/modules/<module>/Ext/Layoutdefs/layoutdefs.ext.php.

An example of this is when deploying a 1-M relationship from Bugs to Leads, the following layoutdef files are generated:

./custom/Extension/modules/Bugs/Ext/Layoutdefs/bugs_leads_1_Bugs.php

```
<?php

$layout_defs["Bugs"]["subpanel_setup"]['bugs_leads_1'] = array (
  'order' => 100,
  'module' => 'Leads',
  'subpanel_name' => 'default',
  'sort_order' => 'asc',
  'sort_by' => 'id',
  'title_key' => 'LBL_BUGS_LEADS_1_FROM_LEADS_TITLE',
  'get_subpanel_data' => 'bugs_leads_1',
  'top_buttons' =>
  array (
    0 =>
    array (
      'widget_class' => 'SubPanelTopButtonQuickCreate',
    ),
    1 =>
    array (
      'widget_class' => 'SubPanelTopSelectButton',
      'mode' => 'MultiSelect',
    ),
  ),
);
```

./custom/Extension/modules/Bugs/Ext/WirelessLayoutdefs/bugs_leads_1_Bugs.php

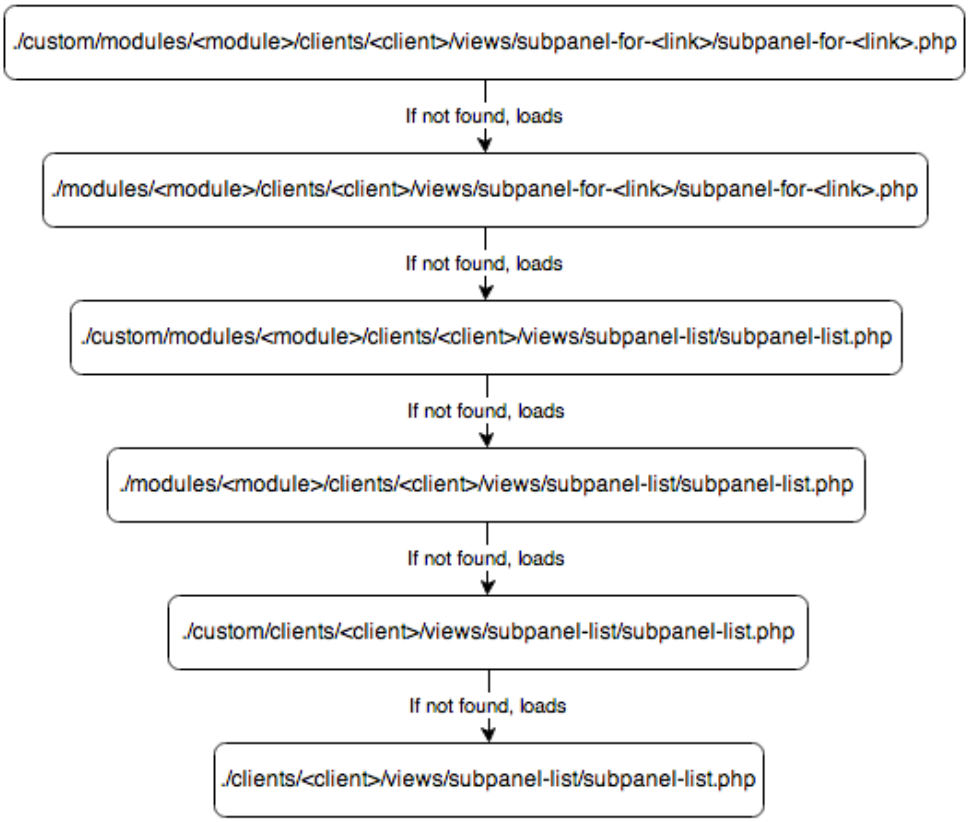
```
<?php

$layout_defs["Bugs"]["subpanel_setup"]['bugs_leads_1'] = array (
  'order' => 100,
  'module' => 'Leads',
  'subpanel_name' => 'default',
  'title_key' => 'LBL_BUGS_LEADS_1_FROM_LEADS_TITLE',
  'get_subpanel_data' => 'bugs_leads_1',
);
```

Fields Metadata

Sidecar's subpanel field layouts are initially defined by the subpanel list-view metadata.

Hierarchy Diagram

The subpanel list metadata is loaded in the following manner:



Note: The Sugar application's client type is "base". For more information on the various client types, please refer to the User Interface page.

Subpanel List Views

By default, all modules come with a default set of subpanel fields for when they are rendered as a subpanel. An example of this is can be found in the Bugs module:

./modules/Bugs/clients/base/views/subpanel-list/subpanel-list.php

```
<?php

$subpanel_layout['list_fields'] = array (
  'full_name' =>
  array (
    'type' => 'fullname',
```

```
    'link' => true,
    'studio' =>
    array (
      'listview' => false,
    ),
    'vname' => 'LBL_NAME',
    'width' => '10%',
    'default' => true,
  ),
  'date_entered' =>
  array (
    'type' => 'datetime',
    'studio' =>
    array (
      'portaleditview' => false,
    ),
    'readonly' => true,
    'vname' => 'LBL_DATE_ENTERED',
    'width' => '10%',
    'default' => true,
  ),
  'refered_by' =>
  array (
    'vname' => 'LBL_LIST_REFERED_BY',
    'width' => '10%',
    'default' => true,
  ),
  'lead_source' =>
  array (
    'vname' => 'LBL_LIST_LEAD_SOURCE',
    'width' => '10%',
    'default' => true,
  ),
  'phone_work' =>
  array (
    'vname' => 'LBL_LIST_PHONE',
    'width' => '10%',
    'default' => true,
  ),
  'lead_source_description' =>
  array (
    'name' => 'lead_source_description',
    'vname' => 'LBL_LIST_LEAD_SOURCE_DESCRIPTION',
    'width' => '10%',
    'sortable' => false,
    'default' => true,
```

```
  ),
  'assigned_user_name' =>
  array (
    'name' => 'assigned_user_name',
    'vname' => 'LBL_LIST_ASSIGNED_TO_NAME',
    'widget_class' => 'SubPanelDetailViewLink',
    'target_record_key' => 'assigned_user_id',
    'target_module' => 'Employees',
    'width' => '10%',
    'default' => true,
  ),
  'first_name' =>
  array (
    'usage' => 'query_only',
  ),
  'last_name' =>
  array (
    'usage' => 'query_only',
  ),
  'salutation' =>
  array (
    'name' => 'salutation',
    'usage' => 'query_only',
  ),
);
```

To modify this layout, navigate to Admin > Studio > {Parent Module} > Subpanels > Bugs and make your changes. Once saved, Sugar will generate ./custom/modules /Bugs/clients/<client>/views/subpanel-for-<link>/subpanel-for-<link>.php which will be used for rendering the fields you selected.

You should note that, just as Sugar mimics the Sidecar layouts in the legacy MVC framework for modules in backward compatibility, it also mimics the field list in ./modules/<module>/metadata/subpanels/default.php and ./custom/modules/<module>/metadata/subpanels/default.php. This is done to ensure that any related modules, whether in Sidecar or Backward Compatibility mode, display the same field list as expected.

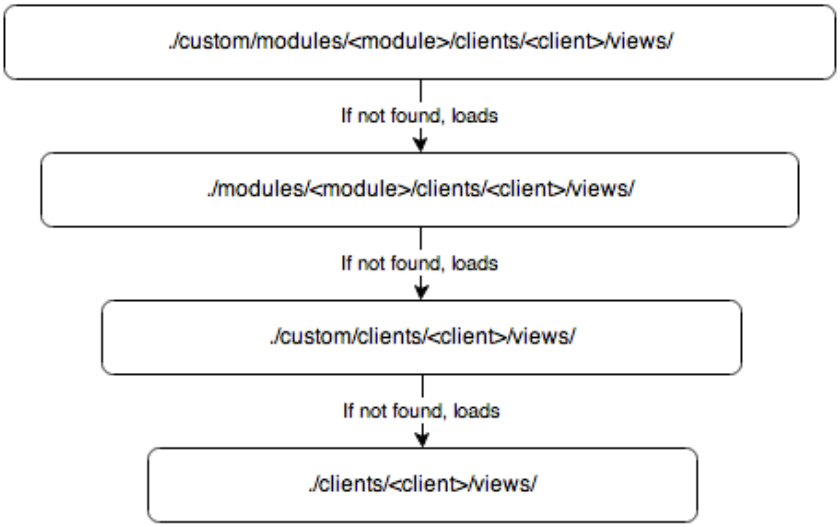
Dashlets

Overview

Dashlets are special view-component plugins that render data from a context and make use of the Dashlet plugin. They are typically made up of a controller JavaScript file (.js) and at least one Handlebars template (.hbs).

Hierarchy Diagram

Sugar loads the dashlet view components in the following manner:



Note: The Sugar application's client type is "base". For more information on the various client types, please refer to the User Interface page.

Dashlet Views

The are three views when working with dashlets: Preview, Dashlet View, and Configuration View. The following sections discuss the differences between views.

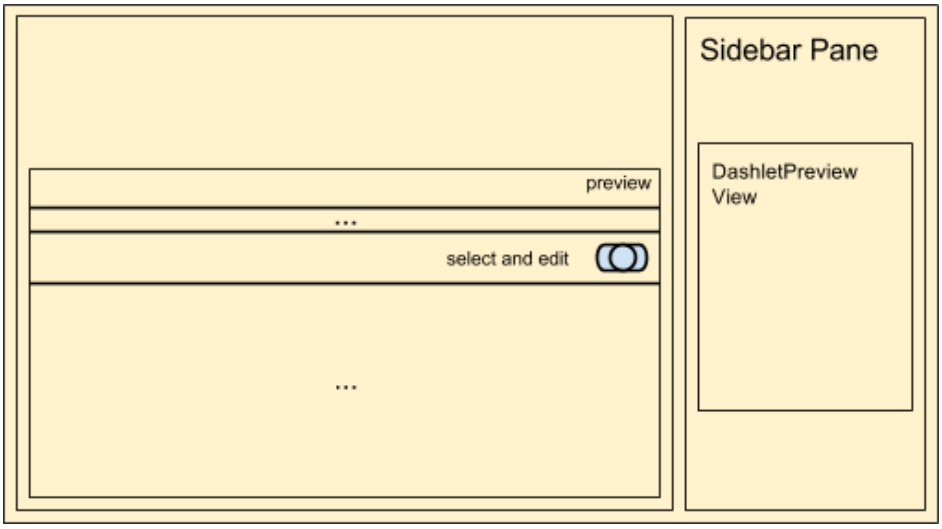
Preview

The preview view is used when selecting dashlets to add to your homepage. Preview variables in the metadata will be assigned to the custom model variables.

```
  'preview' => array(
    'key1' => 'value1',
),
```

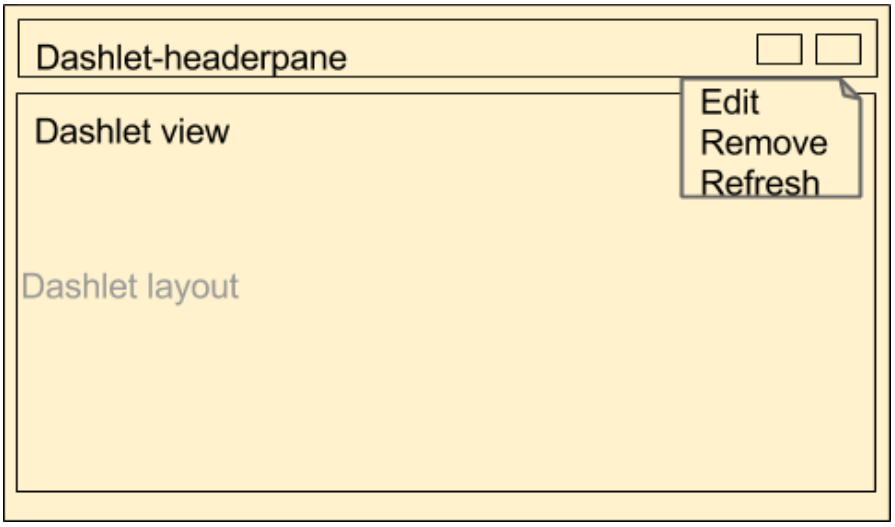
The values in the preview metadata can be retrieved using:

```
this.model.get("key1");
```



Dashlet View

The dashlet view will render the content for the dashlet. It will also contain the settings for editing, removing, and refreshing the dashlet.



Configuration View

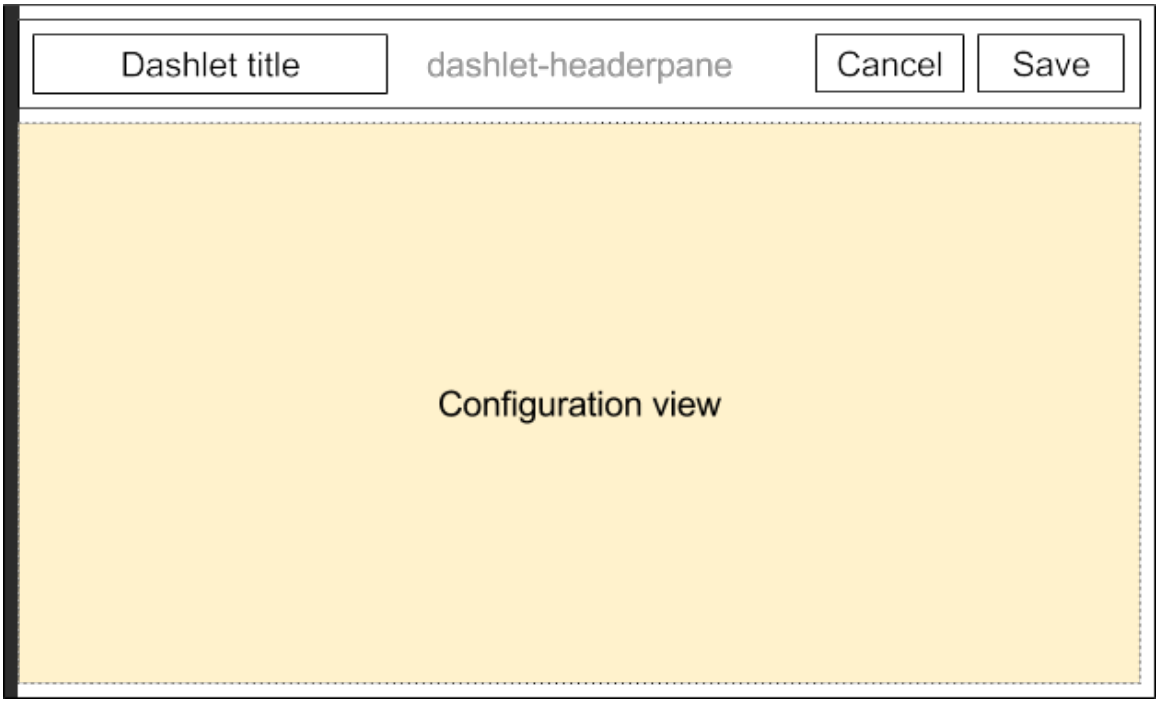
The configuration view is displayed when a user clicks the 'edit' option on the dashlet frame's drop-down menu. Config variables in the metadata will be assigned to the custom model variables

```
'config' => array(
  //key value pairs of attributes
  'key1' => 'value1',
```

```
),
```

The values in the config metadata can be retrieved using:

```
this.model.get("key1");
```



Dashlet Example

The RSS feed dashlet, located in ./clients/base/views/rssfeed/, handles the display of RSS feeds to the user. The sections below outline the various files that render this dashlet.

Metadata

The Dashlet view contains the 'dashlets' metadata:

Parameters	Type	Required	Description
label	String	yes	The name of the dashlet

description	String	no	A description of the dashlet
config	Object	yes	Pre-populated variables in the configuration view **Note**: Config variables in the metadata are assigned to the custom model variables.
preview	Object	yes	Pre-populated variables in the preview
filter	Object	no	Filter for display

The RSS feed dashlets metadata is located in:

./clients/base/views/rssfeed/rssfeed.php

```
  <?php

/*
 * Your installation or use of this SugarCRM file is subject to the ap
plicable
 * terms available at
 * http://support.sugarcrm.com/Resources/Master_Subscription_Agreement
s/.
 * If you do not agree to all of the applicable terms or do not have t
he
 * authority to bind the entity as an authorized representative, then
do not
 * install or use this SugarCRM file.
 *
 * Copyright (C) SugarCRM Inc. All rights reserved.
 */

$viewdefs['base']['view']['rssfeed'] = array(
    'dashlets' => array(
        array(
            'label' => 'LBL_RSS_FEED_DASHLET',
            'description' => 'LBL_RSS_FEED_DASHLET_DESCRIPTION',
            'config' => array(
                'limit' => 5,
                'auto_refresh' => 0,
```

```
            ),
            'preview' => array(
                'limit' => 5,
                'auto_refresh' => 0,
                'feed_url' => 'http://blog.sugarcrm.com/feed/',
            ),
        ),
    ),
    'panels' => array(
        array(
            'name' => 'panel_body',
            'columns' => 2,
            'labelsOnTop' => true,
            'placeholders' => true,
            'fields' => array(
                array(
                    'name' => 'feed_url',
                    'label' => 'LBL_RSS_FEED_URL',
                    'type' => 'text',
                    'span' => 12,
                    'required' => true,
                ),
                array(
                    'name' => 'limit',
                    'label' => 'LBL_RSS_FEED_ENTRIES_COUNT',
                    'type' => 'enum',
                    'options' => 'tasks_limit_options',
                ),
                array(
                    'name' => 'auto_refresh',
                    'label' => 'LBL_DASHLET_REFRESH_LABEL',
                    'type' => 'enum',
                    'options' => 'sugar7_dashlet_reports_auto_refresh_
options',
                ),
            ),
        ),
    ),
);
```

Controller

The rssfeed.js controller file, shown below, contains the JavaScript to render the news articles on the dashlet. The Dashlet view must include 'Dashlet' plugin and

can override initDashlet to add additional custom process while it is initializing.

./clients/base/views/rssfeed/rssfeed.js

```
  /*
 * Your installation or use of this SugarCRM file is subject to the ap
plicable
 * terms available at
 * http://support.sugarcrm.com/Resources/Master_Subscription_Agreement
s/.
 * If you do not agree to all of the applicable terms or do not have t
he
 * authority to bind the entity as an authorized representative, then
do not
 * install or use this SugarCRM file.
 *
 * Copyright (C) SugarCRM Inc. All rights reserved.
 */
/**
 * RSS Feed dashlet consumes an RSS Feed URL and displays it's content
 as a list
 * of entries.
 *
 * The following items are configurable.
 *
 * - {number} limit Limit imposed to the number of records pulled.
 * - {number} refresh How often (minutes) should refresh the data coll
ection.
 *
 * @class View.Views.Base.RssfeedView
 * @alias SUGAR.App.view.views.BaseRssfeedView
 * @extends View.View
 */
({
    plugins: ['Dashlet'],

    /**
     * Default options used when none are supplied through metadata.
     *
     * Supported options:
     * - timer: How often (minutes) should refresh the data collection
.
     * - limit: Limit imposed to the number of records pulled.
     *
     * @property {Object}
     * @protected
```

```
     */
    _defaultOptions: {
        limit: 5,
        auto_refresh: 0
    },

    /**
     * @inheritdoc
     */
    initialize: function(options) {
        options.meta = options.meta || {};
        this._super('initialize', [options]);
        this.loadData(options.meta);
    },

    /**
     * Init dashlet settings
     */
    initDashlet: function() {
        // We only need to handle this if we are NOT in the configure
screen
        if (!this.meta.config) {
            var options = {};
            var self = this;
            var refreshRate;

            // Get and set values for limits and refresh
            options.limit = this.settings.get('limit') || this._defaul
tOptions.limit;
            this.settings.set('limit', options.limit);

            options.auto_refresh = this.settings.get('auto_refresh') |
| this._defaultOptions.auto_refresh;
            this.settings.set('auto_refresh', options.auto_refresh);

            // There is no default for this so there's no pointing in
setting from it
            options.feed_url = this.settings.get('feed_url');

            // Set the refresh rate for setInterval so it can be check
ed ahead
            // of time. 60000 is 1000 miliseconds times 60 seconds in
a minute.
            refreshRate = options.auto_refresh * 60000;

            // Only set up the interval handler if there is a refreshR
```

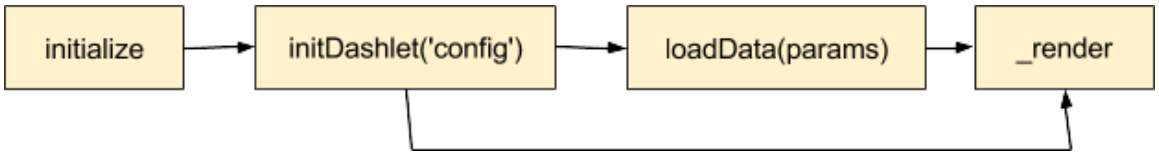
```
ate higher
            // than 0
            if (refreshRate > 0) {
                if (this.timerId) {
                    clearInterval(this.timerId);
                }
                this.timerId = setInterval(_.bind(function() {
                    if (self.context) {
                        self.context.resetLoadFlag();
                        self.loadData(options);
                    }
                }, this), refreshRate);
            }
        }

        // Validation handling for individual fields on the config
        this.layout.before('dashletconfig:save', function() {
            // Fields on the metadata
            var fields = _.flatten(_.pluck(this.meta.panels, 'fields')
);

            // Grab all non-valid fields from the model
            var notValid = _.filter(fields, function(field) {
                return field.required && !this.dashModel.get(field.nam
e);
            }, this);

            // If there no invalid fields we are good to go
            if (notValid.length === 0) {
                return true;
            }

            // Otherwise handle notification of invalidation
            _.each(notValid, function(field) {
                 var fieldOnView = _.find(this.fields, function(comp,
cid) {
                    return comp.name === field.name;
                 });

                 fieldOnView.model.trigger('error:validation:' + field
.name, {required: true});
            }, this);

            // False return tells the drawer that it shouldn't close
            return false;
        }, this);
```

```
    },

    /**
     * Handles the response of the feed consumption request and sets d
ata from
     * the result
     *
     * @param {Object} data Response from the rssfeed API call
     */
    handleFeed: function (data) {
        if (this.disposed) {
            return;
        }

        // Load up the template
        _.extend(this, data);
        this.render();
    },

    /**
     * Loads an RSS feed from the RSS Feed endpoint.
     *
     * @param {Object} options The metadata that drives this request
     */
    loadData: function(options) {
        if (options && options.feed_url) {
            var callbacks = {success: _.bind(this.handleFeed, this), e
rror: _.bind(this.handleFeed, this)},
                limit = options.limit || this._defaultOptions.limit,
                params = {feed_url: options.feed_url, limit: limit},
                apiUrl = app.api.buildURL('rssfeed', 'read', '', param
s);

            app.api.call('read', apiUrl, {}, callbacks);
        }
    },

    /**
     * @inheritdoc
     *
     * New model related properties are injected into each model:
     *
     * - {Boolean} overdue True if record is prior to now.
     */
    _renderHtml: function() {
        if (this.meta.config) {
```

```
            this._super('_renderHtml');
            return;
        }

        this._super('_renderHtml');
    }
})
```

Workflow

When triggered, the following procedure will render the view area:



Retrieving Data

Use the following commands to retrieve the corresponding data:

Data Location	Element	Command
Main pane	Record View	this.model
	Record View	this.context.parent.get("model")
	List View	this.context.parent.get("collection")
Metadata		this.dashletConfig['metadata_key']
Module vardefs		app.metadata.getModule("ModuleName")
Remote data	Bean	new app.data.createBean("Module") new app.data.createBeanCollection("Module")
	RestAPI	app.api.call(method, url, data, callbacks, options)
	Ajax Call	$.ajax()
User inputs		this.settings.get("custom_key")

Handlebar Template

The rssfeed.hbs template file defines the content of the view. This view is used for rendering the markup rendering in the dashlet content.

./clients/base/views/rssfeed/rssfeed.hbs

```
{{!--
/*
 * Your installation or use of this SugarCRM file is subject to the ap
plicable
 * terms available at
 * http://support.sugarcrm.com/Resources/Master_Subscription_Agreement
s/.
 * If you do not agree to all of the applicable terms or do not have t
he
 * authority to bind the entity as an authorized representative, then
do not
 * install or use this SugarCRM file.
 *
 * Copyright (C) SugarCRM Inc. All rights reserved.
 */
--}}
{{#if feed}}
    <div class="rss-feed">
        <h4>
            {{#if feed.link}}<a href="{{feed.link}}">{{/if}}
            {{feed.title}}
            {{#if feed.link}}</a>{{/if}}
        </h4>
        <ul>
        {{#each feed.entries}}
            <li class="news-article">
                <a href="{{link}}">{{title}}</a>
                {{#if author}} - {{str "LBL_RSS_FEED_AUTHOR"}} {{autho
r}}{{/if}}
            </li>
        {{/each}}
        </ul>
    </div>
{{else}}
    <div class="block-footer">
        {{#if errorThrown}}
        {{str "LBL_NO_DATA_AVAILABLE"}}
```

```
        {{else}}
        {{loading 'LBL_ALERT_TITLE_LOADING'}}
        {{/if}}
    </div>
{{/if}}
```

Drawers

Overview

The drawer layout widget, located in ./clients/base/layouts/drawer/, is used to display a window of additional content to the user. This window can then be closed to display the content the user was previously viewing.

Methods

app.drawer.open(layoutDef, onClose)

The app.drawer.open(layoutDef, onClose) method displays a new content window over the current view.

Parameters

Name	Required	Description
layoutDef.layout	yes	The id of the layout to load.
layoutDef.context	no	Additional data you would like to pass to the drawer. Data passed in can be retrieved from the view using: this.context.get('<data key>'); **Note**: Be very careful about what you pass in as data to the drawer. As the data is passed in by reference, when the drawer is closed, the context is destroyed.
onClose	no	Optional callback handler

		for when the drawer is closed.

Example

```
app.drawer.open({
    layout: 'my-layout',
    context: {
        myData: data
    },
},
function() {
    //on close, throw an alert
    alert('Drawer closed.');
});
```

app.drawer.close(callbackOptions)

The app.drawer.close(callbackOptions) method dismisses the topmost drawer.

Parameters

Name	Required	Description
callbackOptions	no	Any parameters passed into the close method will be passed to the callback.

Standard Example

```
app.drawer.close();
```

Callback Example

```
//open drawer
app.drawer.open({
    layout: 'my-layout',
},
function(message1, message2) {
    alert(message1);
    alert(message2);
});

//close drawer
```

```
app.drawer.close('message 1', 'message 2');
```

app.drawer.load(options)

Loads a new layout into an existing drawer.

Parameters

Name	Description
options.layout	The id of the layout to load.

Example

```
app.drawer.load({
    layout: 'my-second-layout',
});
```

app.drawer.reset(triggerBefore)

The app.drawer.reset(triggerBefore) method destroys all drawers at once. By default, whenever the application is routed to another page, reset() is called.

Parameters

Name	Required	Description
triggerBefore	no	Determines whether to triggerBefore. Defaults to false.

Example

```
app.drawer.reset();
```

Alerts

Overview

The alert view widget, located in ./clients/base/views/alert/, displays helpful information such as loading messages, notices, and confirmation messages to the user.

Methods

app.alert.show(id, options)

The app.alert.show(id, options) method displays an alert message to the user with the options provided.

Parameters

Name	Description
id	The id of the alert message. Used for dismissing specific messages.
options.level	The alert level
options.title	The alert's title, which corresponds to the alert's level
options.messages	The message that the user sees **Note**: Process alerts do not display messages.
options.autoClose	Whether or not to auto-close the alert popup
options.onClose	Callback handler for closing confirmation alerts when clicking the x
options.onConfirm	Callback handler for confirming confirmation alerts
options.onCancel	Callback handler for canceling confirmation alerts
options.onLinkClick	Callback handler for click actions on a link inside of the alert

Default Alert Values

Alert Level	Alert Appearance	Alert Title
info	blue	"Notice"
success	green	"Success"
warning	yellow	"Warning!"
error	red	"Error"
process	loading message	"Loading..."
confirmation	confirmation dialog	"Warning"

Alert Examples

Standard Alert

```
  app.alert.show('message-id', {
    level: 'success',
    messages: 'Task completed!',
    autoClose: true
});
```

Confirmation Alert

```
  app.alert.show('message-id', {
    level: 'confirmation',
    messages: 'Confirm?',
    autoClose: false,
    onConfirm: function(){
         alert("Confirmed!");
    },
    onCancel: function(){
        alert("Cancelled!");
    }
});
```

Process Alert

```
  app.alert.show('message-id', {
    level: 'process',
    title: 'In Process...' //change title to modify display from 'Load
ing...'
});
```

app.alert.dismiss(id)

The app.alert.dismiss(id) method dismisses an alert message from view based on the message id.

Parameters

Name	Description
id	The id of the alert message to dismiss.

Example

```
app.alert.dismiss('message-id');
```

app.alert.dismissAll

The app.alert.dismissAll dismisses all alert messages from view.

Example

```
app.alert.dismissAll();
```

Testing in Console

To test alerts, you can trigger them in your browser's developer tools by using the global App variable as shown below:

```
App.alert.show('message-id', {
  level: 'success',
  messages: 'Successful!',
  autoClose: false
});
```

Language

Overview

The language library, located in ./sidecar/src/core/language.js, is used to manage the user's display language as well as fetch labels and lists. For more information on customizing languages, please visit the language framework documentation.

Methods

app.lang.get(key, module, context)

The app.lang.get(key, module, context) method fetches a string for a given key. The method searches the module strings first and then falls back to the app strings. If the label is a template, it will be compiled and executed with the given context.

Parameters

Name	Required	Description
key	yes	The key of the string to retrieve
module	no	The Sugar module that the label belongs to
context	no	Template context

Example

```
app.lang.get('LBL_NAME', 'Accounts');
```

app.lang.getAppString(key)

The app.lang.getAppString(key) method retrieves an application string for a given key.

Parameters

Name	Required	Description
key	yes	The key of the string to retrieve

Example

```
app.lang.getAppString('LBL_MODULE');
```

app.lang.getAppListStrings(key)

The app.lang.getAppListStrings(key) method retrieves an application list string or object.

Parameters

Name	Required	Description
key	yes	The key of the string to retrieve

Example

```
app.lang.getAppListStrings('sales_stage_dom');
```

app.lang.getModuleSingular(moduleKey)

The app.lang.getModuleSingular(moduleKey) method retrieves an application list string or object.

Parameters

Name	Required	Description
moduleKey	yes	The module key of the singular module label to retrieve

Example

```
app.lang.getModuleSingular("Accounts");
```

app.lang.getLanguage()

The app.lang.getLanguage() method retrieves the current user's language key.

Example

```
app.lang.getLanguage();
```

app.lang.updateLanguage(languageKey)

The app.lang.updateLanguage(languageKey) method updates the current user's language key.

Parameters

Name	Required	Description
languageKey	yes	Language key of the language to set for the user

Example

```
app.lang.updateLanguage('en_us');
```

Testing in Console

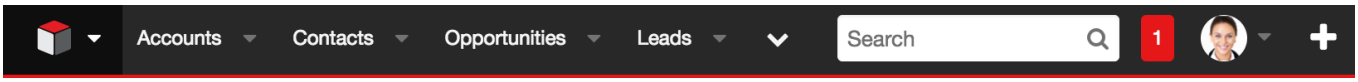
To test out the language library, you can trigger actions in your browsers developer tools by using the global Apps variable as shown below:

```
App.lang.getAppListStrings('sales_stage_dom');
```

MegaMenu

Overview

The MegaMenu is the header navigation bar located at the top of every Sugar page. It is the primary tool used to navigate the front end of the Sugar application.



Layout Components

The MegaMenu layout, located in ./clients/base/layouts/header/header.php, is composed of the module-list and quicksearch layouts as well as the notifications, profileactions and quickcreate views. To customize these components, you can create your own layout override in ./custom/clients/base/layouts/header/header.php to reference your own custom components.

Link Actions

The following properties define the navigation, display, and visibility of all links in the system:

Name	Description
acl_action	The ACL action is used to verify the user has access to a specific action required for the link
acl_module	The ACL module is used to verify if the user has access to a specific module required for the link
icon	The bootstrap icon to display next to the link (the full list of icons are listed in Admin > Styleguide > Core Elements >

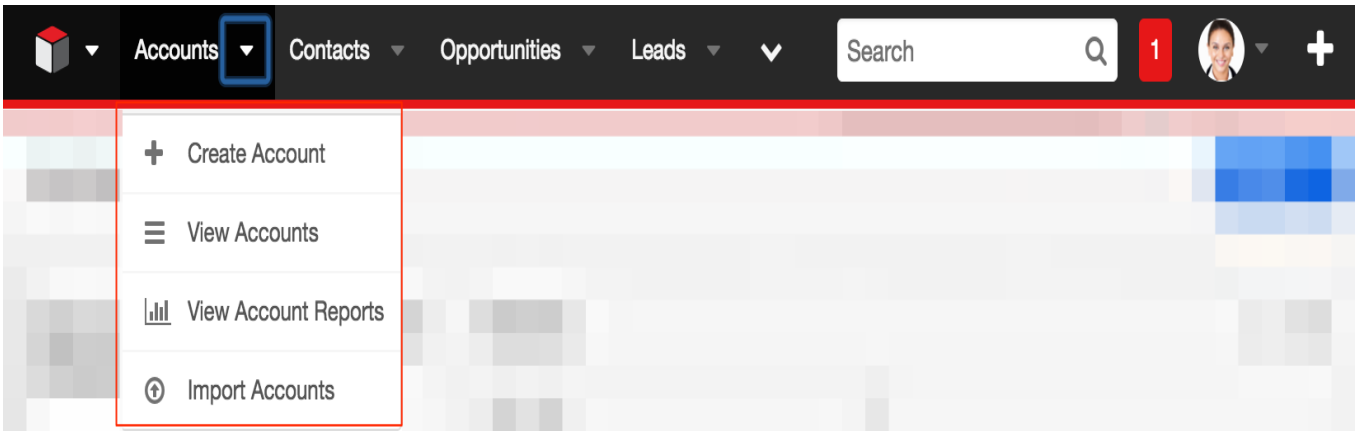
	Base CSS > Icons)
label	The label key that contains your link's display text
openwindow	Specifies whether or not the link should open in a new window
route	The route to direct the user. For sidecar modules, this is #<module>, but modules in backward compatibility mode are routed as #bwc/index.php?module=<module>. **Note**: External links require the full URL as well as openwindow set to true.
submenu	An array of sub-navigation links **Note**: Sub-navigation links contain these same basic link properties.

Module Links

Module links are the top-level links for each available module, represented as tabs in the navigation bar (or, sometimes, the overflow menu). When a top-level link is clicked, the user is directed to the selected module's list view layout. These top-level links are also expandable elements that contain sub-navigation menu links, which are outlined in the following section.

Module Action Links

The module action links are displayed to the user when they click the down arrow next to a module link.



Adding Module Action Links

The example below demonstrates how to add a module action link that points to the Leads module. To define your own module action link, you must create your own label extension for the link's display label:

./custom/Extension/application/Ext/Language/en_us.addModuleLink.php

```
<?php

//create the links label
$app_strings['LNK_LEADS_C'] = 'View Leads';
```

Now create the module action link extension:

./custom/Extension/modules/<module>/Ext/clients/base/menus/header/addModuleLink.php

```
<?php

$viewdefs['<module>']['base']['menu']['header'][] = array(
    'route'=>'#Leads',
    'label' =>'LNK_LEADS_C',
    'acl_module'=>'Leads',
    'icon' => 'icon-user',
```

```
);
```

Once you have created the extension files, navigate to Admin > Repair > Quick Repair and Rebuild. This will append your profile action item to the existing list of links.

Note: You may need to refresh the page to see the new profile menu items.

Removing Module Action Links

To remove a module action link, loop through the list of module actions and remove the item by one of its properties. For your reference, the stock module actions can be found in ./modules/<module>/clients/base/menus/header/header.php.

./custom/Extension/modules/<module>/Ext/clients/base/menus/header/removeModuleLink.php

```
if (isset($viewdefs['<module>']['base']['menu']['header'])) {
    foreach ($viewdefs['<module>']['base']['menu']['header'] as $key =
> $moduleAction) {
        //remove the link by label key
        if (in_array($moduleAction['label'], array('<link label key>')
)) {
            unset($viewdefs['<module>']['base']['menu']['header'][$key
]);
        }
    }
}
```

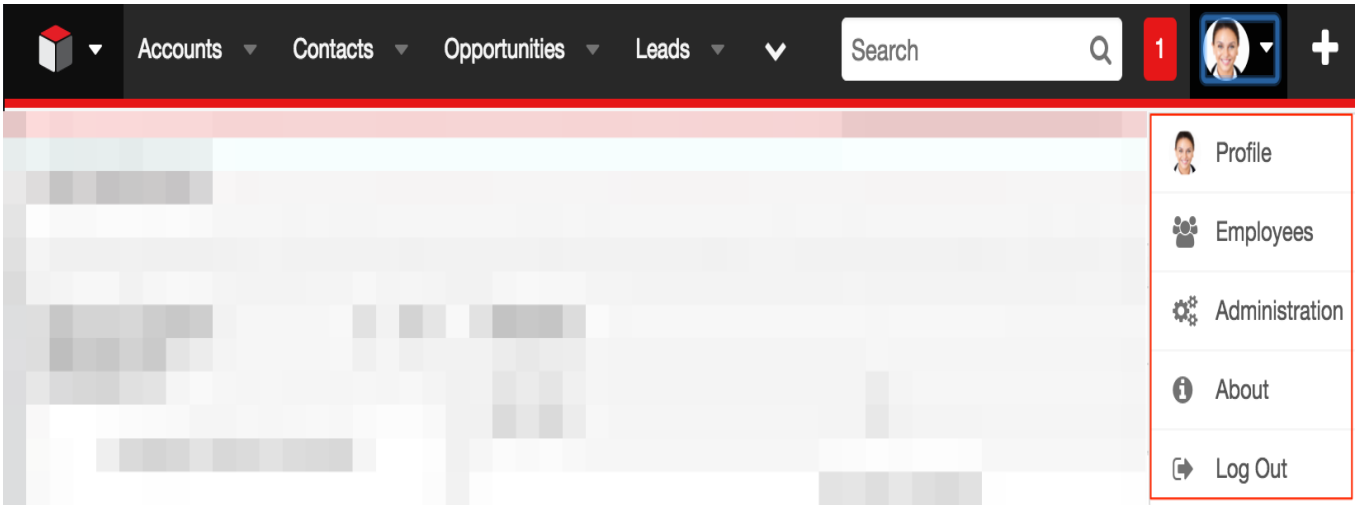
Once you have created the extension files, navigate to Admin > Repair > Quick Repair and Rebuild. This will remove the menu action item from the existing list of links.

Note: You may need to refresh the page to see the module menu items removed.

Profile Action Links

Profile actions are the links listed under the user's profile menu on the right side of the MegaMenu. Profile action extension files are located in ./custom/Extension/application/Ext/clients/base/views/profileactions/ and are compiled into

./custom/application/Ext/clients/base/views/profileactions/profileactions.ext.php.



Adding Profile Action Links

The example below demonstrates how to add a profile action link to the Styleguide. To define your own profile action link, create your own label extension for the link's display label.

./custom/Extension/application/Ext/Language/en_us.addProfileActionLink.php

```
<?php

//create the links label
$app_strings['LNK_STYLEGUIDE_C'] = 'Styleguide';
```

Next, create the profile action link extension:

./custom/Extension/application/Ext/clients/base/views/profileactions/addProfileActionLink.php

```
<?php
```

```
$viewdefs['base']['view']['profileactions'][] = array(
    'route' => '#Styleguide',
    'label' => 'LNK_STYLEGUIDE_C',
    'icon' => 'icon-link',
);
```

Once you have created the extension files, navigate to Admin > Repair > Quick Repair and Rebuild. This will append your profile action item to the existing list of links.

Note: You may need to refresh the page to see the new profile menu items.

Removing Profile Action Links

To remove a profile action link, loop through the list of profile actions and remove the item by one of its properties. For your reference, the stock profile actions can be found in ./clients/base/views/profileactions/profileactions.php.

./custom/Extension/application/Ext/clients/base/views/profileactions/removeProfileActionLink.php

```
<?php

if (isset($viewdefs['base']['view']['profileactions'])) {
    foreach ($viewdefs['base']['view']['profileactions'] as $key => $profileAction) {
        //remove the link by label key
        if (in_array($profileAction['label'], array('LNK_ABOUT'))) {
            unset($viewdefs['base']['view']['profileactions'][$key]);
        }
    }
}
```

Once you have created the extension files, navigate to Admin > Repair > Quick Repair and Rebuild. This will remove the profile action item from the existing list of links.

Note: You may need to refresh the page to see the profile menu items removed.

Administration Links

Overview

Administration links are the shortcut URLs found on the Administration page in the Sugar application. Developers can create additional administration links using the extension framework.

The global links extension directory is located at ./custom/Extension/modules/Administration/Ext/Administration/. After a Quick Repair and Rebuild, the PHP files in this directory are compiled into ./custom/modules/Administration/Ext/Administration/administration.ext.php. Additional information on this can be found in the extensions Administration section of the Extension Framework documentation. The current links defined in the administration section can be found in ./modules/Administration/metadata/adminpaneldefs.php.

Example

The following example will create a new panel on the Admin page:

./custom/Extension/modules/Administration/Ext/Administration/<file>.php

```
  <?php

    $admin_option_defs = array();
    $admin_option_defs['Administration']['<section key>'] = array(
        //Icon name. Available icons are located in ./themes/default/i
mages
        'Administration',

        //Link name label
        'LBL_LINK_NAME',

        //Link description label
        'LBL_LINK_DESCRIPTION',

        //Link URL - For Sidecar modules
        'javascript:void(parent.SUGAR.App.router.navigate("<module>/<p
ath>", {trigger: true}));',

        //Alternatively, if you are linking to BWC modules
        //'./index.php?module=<module>&action=<action>',
    );

    $admin_group_header[] = array(
        //Section header label
```

```
        'LBL_SECTION_HEADER',

        //$other_text parameter for get_form_header()
        '',

        //$show_help parameter for get_form_header()
        false,

        //Section links
        $admin_option_defs,

        //Section description label
        'LBL_SECTION_DESCRIPTION'
    );
```

To define labels for administration links in the new panel:

./custom/Extension/modules/Administration/Ext/Language/en_us.<name>.php

```
  <?php

    $mod_strings['LBL_LINK_NAME'] = 'Link Name';
    $mod_strings['LBL_LINK_DESCRIPTION'] = 'Link Description';
    $mod_strings['LBL_SECTION_HEADER'] = 'Section Header';
    $mod_strings['LBL_SECTION_DESCRIPTION'] = 'Section Description';
```

Finally, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and the panel will appear on the Admin page.

Legacy MVC

Overview

The legacy MVC Architecture.

You should note that the MVC architecture is being deprecated and is being replaced with sidecar. Until the framework is fully deprecated, modules set in backward compatibility mode will still use the MVC framework.

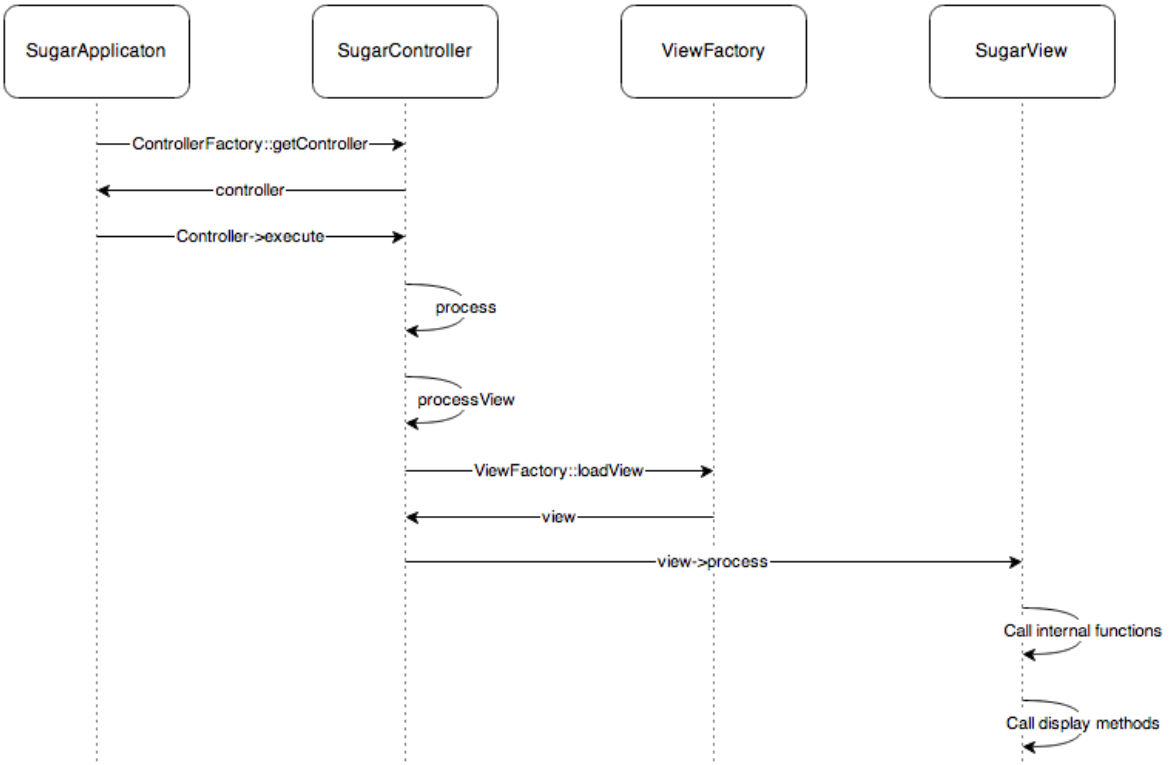
Model-View-Controller (MVC) Overview

A model-view-controller, or MVC, is a design philosophy that creates a distinct separation between business-logic and display logic.

- **Model** : This is the data object built by the business/application logic needed to present in the user interface. For Sugar, it is represented by the SugarBean and all subclasses of the SugarBean.
- **View** : This is the display layer which is responsible for rendering data from the Model to the end-user.
- **Controller** : This is the layer that handles user events such as "Save" and determines what business logic actions to take to build the model, and which view to load for rendering the data to end users.

SugarCRM MVC Implementation

The following is a sequence diagram that highlights some of the main components involved within the Sugar MVC framework.



View

Overview

Displaying information to the browser.

What are Views?

Views, otherwise known as actions, are typically used to render views or to process logic. Views are not just limited to HTML data. You can send JSON encoded data as part of a view or any other structure you wish. As with the controllers, there is a default class called SugarView which implements much of the basic logic for views, such as handling of headers and footers.

There are five main actions for a module:

- Display Actions
    - **Detail View**: A detail view displays a read-only view of a particular record. Usually, this is accessed via the list view. The detail view displays the details of the object itself and related items (subpanels).
      Subpanels are miniature list views of items that are related to the parent object. For example, Tasks assigned to a Project, or Contacts for an Opportunity will appear in subpanels in the Project or Opportunity detail view. The file ./<module>/metadata/detailviewdefs.php defines a module's detail view page layout. The file ./<module>/metadata/subpaneldefs.php defines the subpanels that are displayed in the module's detail view page.
    - **Edit View**: The edit view page is accessed when a user creates a new record or edits details of an existing one. Edit view can also be accessed directly from the list view. The file ./<module>/metadata/editviewdefs.php defines a module's edit view page layout.
    - **List View**: This Controller action enables the search form and search results for a module. Users can perform actions such as delete, export, update multiple records (mass update), and drill into a specific record to view and edit the details. Users can see this view by default when they click one of the module tabs at the top of the page. Files in each module describe the contents of the list and search view.
- Process Actions
    - **Save**: This Controller action is processed when the user clicks Save in the record's edit view.
    - **Delete**: This action is processed when the user clicks "Delete" in the detail view of a record or in the detail view of a record listed in a subpanel.

Implementation

Class File Structure

- ./include/MVC/Views/SugarView.php
- ./include/MVC/Views/view.<view>.php
- ./custom/include/MVC/Views/view.<view>.php
- ./modules/<module>/views/view.<view>.php
- ./custom/modules/<module>/views/view.<view>.php

Class Loading

The ViewFactory class loads the view based off the the following sequence loading the first file it finds:

- ./custom/modules/<module>/views/view.<view>.php
- ./modules/<module>/views/view.<view>.php
- ./custom/include/MVC/View/view.<view>.php
- ./include/MVC/Views/view.<view>.php

Methods

There are two main methods to override within a view:

- **preDisplay()**: This performs pre-processing within a view. This method is relevant only for extending existing views. For example, the include/MVC/View/views/view.edit.php file uses it, and enables developers who wishes to extend this view to leverage all of the logic done in preDisplay() and either override the display() method completely or within your own display() method call parent::display().
- **display()**: This method displays the data to the screen. Place the logic to display output to the screen here.

Creating Views

Creating a new/view action consists of a controller action and a view file. The first step is to define your controller action. If the module does not contain a controller.php file in ./modules/<module>/ you will create the following file:

./custom/modules/<module>/controller.php

```
  <?php

class <module>Controller extends SugarController
{
    function action_MyView()
    {
```

```
        $this->view = 'myview';
    }
}
```

More information on controllers can be found in the Controller section.

The next step is to define your view file. This example extends the ViewDetail class but you can extend any of the classes you choose in ./include/MVC/View/views/.

./custom/modules/<module>/views/view.newview.php

```
  <?php

require_once 'include/MVC/View/views/view.detail.php' ;

class <module>ViewMyView extends ViewDetail
{
    function display()
    {
        echo 'This is my new view<br>';
    }
}
```

Overriding Views

The following section will demonstrate how to extend and override a view. When overriding existing actions and views, you won't need to make any changes to the controller. This approach will be very similar for any view you may choose to modify. If the module you are extending the view for does not contain an existing view in its modules views directory (./modules/<module>/views/), you will need to extend the views base class. Otherwise, you will extend the view class found within the file.

In the case of a detail view, you would check for the file ./modules/<module>/views/view.detail.php. If this file does not exist, you will create ./custom/modules/<module>/views/view.detail.php and extend the base ViewDetail class with the name <module>ViewDetail.

./custom/modules/<module>/views/view.detail.php

```
  <?php
```

```
require_once('include/MVC/View/views/view.detail.php');

class <module>ViewDetail extends ViewDetail
{
    function display()
    {
        echo 'This is my addition to the DetailView<br>';

        //call parent display method
        parent::display();
    }
}
```

If ./modules/<module>/views/view.detail.php does exist, you would create ./custom/modules/<module>/views/view.detail.php and extend the base <module>ViewDetail class with the name Custom<module>ViewDetail.

./custom/modules/<module>/views/view.detail.php

```
  <?php

require_once('modules/<module>/views/view.detail.php');

class Custom<module>ViewDetail extends <module>ViewDetail
{
    function display()
    {
        echo 'This is my addition to the DetailView<br>';

        //call parent display method
        parent::display();
    }
}
```

Display Options for Views

The Sugar MVC provides developers with granular control over how the screen looks when a view is rendered. Each view can have a config file associated with it. In the case of an edit view, the developer would create the file ./customs/modules/<module>/views/view.edit.config.php . When the edit view is rendered, this config file will be picked up. When loading the view, ViewFactory class will merge the view config files from the following possible locations with

precedence order (high to low):

- ./customs/modules/<module>/views/view.<view>.config.php
- ./modules/<module>/views/view.<view>.config.php
- ./custom/include/MVC/View/views/view.<view>.config.php
- ./include/MVC/View/views/view.<view>.config.php

Implementation

The format of these files is as follows:

```
  $view_config = array(
    'actions' =>
        array(
            'popup' => array(
            'show_header' => false,
            'show_subpanels' => false,
            'show_search' => false,
            'show_footer' => false,
            'show_JavaScript' => true,
        ),

    ),
    'req_params' => array(
        'to_pdf' => array(
            'param_value' => true,
            'config' => array(
                'show_all' => false
            ),
        ),
    ),
);
```

To illustrate this process, let us take a look at how the 'popup' action is processed. In this case, the system will go to the actions entry within the view_config and determine the proper configuration. If the request contains the parameter to_pdf, and is set to be true, then it will automatically cause the show_all configuration parameter to be set false, which means none of the options will be displayed.

Controller

Overview

The basic actions of a module.

Controllers

The main controller, named SugarController, addresses the basic actions of a module from EditView and DetailView to saving a record. Each module can override this SugarController by adding a controller.php file into its directory. This file extends the SugarController, and the naming convention for the class is: <module>Controller

Inside the controller, you define an action method. The naming convention for the method is: action_<action name>

There are more fine-grained control mechanisms that a developer can use to override the controller processing. For example, if a developer wanted to create a new save action, there are three places where they could possibly override.

- **action_save:** This is the broadest specification and gives the user full control over the save process.
- **pre_save:** A user could override the population of parameters from the form.
- **post_save:** This is where the view is being set up. At this point, the developer could set a redirect URL, do some post-save processing, or set a different view.

Upgrade-Safe Implementation

You can also add a custom Controller that extends the module's Controller if such a Controller already exists. For example, if you want to extend the Controller for a module, you should check if that module already has a module-specific controller. If so, you extend from that controller class. Otherwise, you extend from SugarController class. In both cases, you should place the custom controller class file in ./custom/modules/<module>/Controller.php instead of the module directory. Doing so makes your customization upgrade-safe.

File Structure

- ./include/MVC/Controller/SugarController.php
- ./include/MVC/Controller/ControllerFactory.php
- ./modules/<module>/Controller.php
- ./custom/modules/<module>/controller.php

Implementation

If the module does not contain a controller.php file in ./modules/<module>/, you will create the following file:

./custom/modules/<module>/controller.php

```
class <module>Controller extends SugarController
{
    function action_<action>()
    {
        $this->view = '<action lowercase>';
    }
}
```

If the module does contain a controller.php file, you will need to extend it by doing the following:

./custom/modules/<module>/controller.php

```
require_once 'modules/<module>/controller.php';

class Custom<module>Controller extends <module>Controller
{
    function action_<action>()
    {
        $this->view = '<action lowercase>';
    }
}
```

Note: When creating or moving files you will need to rebuild the file map.

More information on rebuilding the file map can be found in the SugarAutoLoader.

Mapping Actions to Files

You can choose not to provide a custom action method as defined above, and instead, specify your mappings of actions to files in $action_file_map. Take a look at ./include/MVC/Controller/action_file_map.php as an example:

```
$action_file_map['subpanelviewer'] = 'include/SubPanel/SubPanelViewer.
php';
$action_file_map['save2'] = 'include/generic/Save2.php';
$action_file_map['deleterelationship'] = 'include/generic/DeleteRelati
onship.php';
$action_file_map['import'] = 'modules/Import/index.php';
```

Here the developer has the opportunity to map an action to a file. For example, Sugar uses a generic sub-panel file for handling subpanel actions. You can see above that there is an entry mapping the action 'subpanelviewer' to ./include/SubPanel/SubPanelViewer.php.

The base SugarController class loads the action mappings in the following path sequence:

- ./include/MVC/Controller
- ./modules/<module>
- ./custom/modules/<module>
- ./custom/include/MVC/Controller

Each one loads and overrides the previous definition if in conflict. You can drop a new action_file_map in the later path sequence that extends or overrides the mappings defined in the previous one.

Upgrade-Safe Implementation

If you want to add custom action_file_map.php to an existing module that came with the SugarCRM release, you should place the file at ./custom/modules/<module>/action_file_map.php

File Structure

- ./include/MVC/Controller/action_file_map.php
- ./modules/<module>/action_file_map.php
- ./custom/modules/<module>/action_file_map.php

Implementation

```
$action_file_map['soapRetrieve'] = 'custom/SoapRetrieve/soap.php';
```

Classic Support (Not Recommended)

Classic support allows you to have files that represent actions within your module. Essentially, you can drop in a PHP file into your module and have that be handled as an action. This is not recommended, but is considered acceptable for backward compatibility. The better practice is to take advantage of the action_<action> structure.

File Structure

- ./modules/<module>/<action>.php

Controller Flow Overview

For example, if a request comes in for DetailView the controller will handle the request as follows:

1. Start in index.php and load the SugarApplication instance.
2. SugarApplication instantiates the SugarControllerFactory.
3. SugarControllerFactory loads the appropriate Controller.
4. SugarControllerFactory checks for ./custom/modules/<module>/Controller.php.
    1. If not found, check for ./modules/<module>/Controller.php.
    2. If not found, load SugarController.php.
5. Calls on the appropriate action.
    1. Look for ./custom/modules/<module>/<action>.php. If found and ./custom/modules/<module>/views/view.<action>.php is not found, use this view.
    2. If not found check for modules/<module>/<action>.php. If found and ./modules/<module>/views/view.<action>.php is not found, then use the ./modules/<module>/<action>.php action.
    3. If not found, check for the method action_<action> in the controller.
    4. If not found, check for an action_file_mapping.
    5. If not found, report error "Action is not defined".

Metadata

Overview

An overview of the legacy MVC metadata framework.

You should note that the MVC architecture is being deprecated and is being replaced with sidecar. Until the framework is fully deprecated, modules set in backward compatibility mode will still use the legacy MVC framework.

Metadata Framework

Background

Metadata is defined as information about data. In Sugar, metadata refers to the framework of using files to abstract the presentation and business logic found in the system. The metadata framework is described in definition files that are processed using PHP. The processing usually includes the use of Smarty templates for rendering the presentation and JavaScript libraries to handle some business logic that affects conditional displays, input validation, and so on.

Application Metadata

All application modules are defined in the modules.php file. It contains several variables that define which modules are active and usable in the application.

The file is located under the '<sugar root>/include' folder. It contains the $moduleList() array variable which contains the reference to the array key to look up the string to be used to display the module in the tabs at the top of the application. The coding standard is for the value to be in the plural of the module name; for example, Contacts, Accounts, Widgets, and so on.

The $beanList() array stores a list of all active beans (modules) in the application. The $beanList entries are stored in a 'name' => 'value' fashion with the 'name' value being in the plural and the 'value' being in the singular of the module name. The 'value' of a $beanList() entry is used to lookup values in our next modules.php variable, the $beanFiles() array.

The $beanFiles variable is also stored in a 'name' => 'value' fashion. The 'name', typically in singular, is a reference to the class name of the object, which is looked up from the $beanList 'value', and the 'value' is a reference to the class file.

The remaining relevant variables in the modules.php file are the $modInvisList variable which makes modules invisible in the regular user interface (i.e., no tab appears for these modules), and the $adminOnlyList which is an extra level of security for modules that are accessible only by administrators through the Admin page.

Module Metadata

The following table lists the metadata definition files found in the modules/[module]/metadata directory, and a brief description of their purpose within the system.

File	Description

additionalDetails.php	Used to render the popup information displayed when a user hovers the mouse cursor over a row in the List View.
editviewdefs.php	Used to render a record's EditView.
detailviewdefs.php	Used to render a record's DetailView.
listviewdefs.php	Used to render the List View display for a module.
metafiles.php	Used to override the location of the metadata definition file to be used. The EditView, DetailView, List View, and Popup code check for the presence of these files.
popupdefs.php	Used to render and handle the search form and list view in popups.
searchdefs.php	Used to render a module's basic and advanced search form displays.
sidecreateviewdefs.php	Used to render a module's quick create form shown in the side shortcut panel.
subpaneldefs.php	Used to render a module's subpanels shown when viewing a record's DetailView.

SearchForm Metadata

The search form layout for each module is defined in the module's metadata file searchdefs.php. A sample of the Accounts searchdefs.php appears as:

```
<?php

  $searchdefs['Accounts'] = array(
      'templateMeta' => array(
          'maxColumns' => '3',
          'widths' => array(
              'label' => '10',
              'field' => '30'
          )
      ),
      'layout' => array(
          'basic_search' => array(
              'name',
              'billing_address_city',
              'phone_office',
              array(
```

```
            'name' => 'address_street',
            'label' => 'LBL_BILLING_ADDRESS',
            'type' => 'name',
            'group' => 'billing_address_street'
        ),
        array(
            'name' => 'current_user_only',
            'label' => 'LBL_CURRENT_USER_FILTER',
            'type'=>'bool'
        ),
    ),
    'advanced_search' => array(
        'name',
        array(
            'name' => 'address_street',
            'label' => 'LBL_ANY_ADDRESS',
            'type' => 'name'
        ),
        array(
            'name' => 'phone',
            'label' => 'LBL_ANY_PHONE',
            'type' => 'name'
        ),
        'website',
        array(
            'name' => 'address_city',
            'label' => 'LBL_CITY',
            'type' => 'name'
        ),
        array(
            'name' => 'email',
            'label' =>'LBL_ANY_EMAIL',
            'type' => 'name'
        ),
        'annual_revenue',
        array(
            'name' => 'address_state',
            'label' =>'LBL_STATE',
            'type' => 'name'
        ),
        'employees',
        array(
            'name' => 'address_postalcode',
            'label' =>'LBL_POSTAL_CODE',
            'type' => 'name'
        ),
```

```
                array(
                    'name' => 'billing_address_country',
                    'label' =>'LBL_COUNTRY',
                    'type' => 'name'
                ),
                'ticker_symbol',
                'sic_code',
                'rating',
                'ownership',
                array(
                    'name' => 'assigned_user_id',
                    'type' => 'enum',
                    'label' => 'LBL_ASSIGNED_TO',
                    'function' => array(
                        'name' =>'get_user_array',
                        'params' => array(false)
                    )
                ),
                'account_type',
                'industry',
            ),
        ),
    );

?>
```

The searchdefs.php file contains the Array variable $searchDefs with one entry. The key is the name of the module as defined in $moduleList array defined in include/modules.php. The $searchDefsarray is another array that describes the search form layout and fields.

The 'templateMeta' key points to another array that controls the maximum number of columns in each row of the search form ('maxColumns'), as well as layout spacing attributes as defined by 'widths'. In the above example, the generated search form files will allocate 10% of the width spacing to the labels and 30% for each field respectively.

The 'layout' key points to another nested array which defines the fields to display in the basic and advanced search form tabs. Each individual field definition maps to a SugarField widget. See the SugarField widget section for an explanation about SugarField widgets and how they are rendered for the search form, DetailView, and EditView.

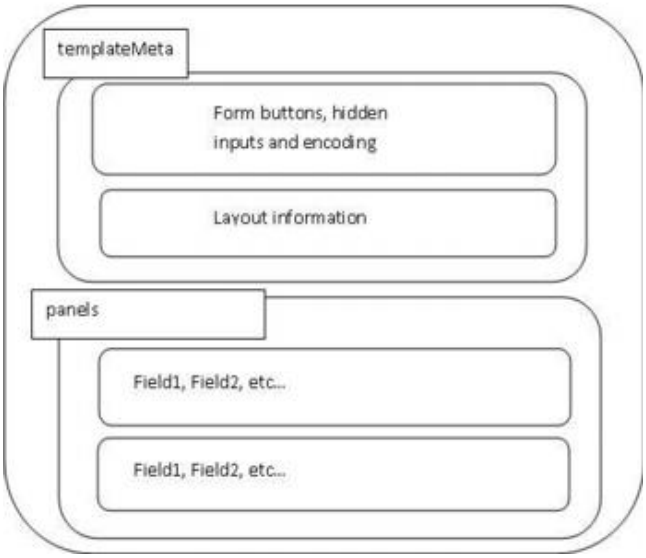
The searchdefs.php file is invoked from the MVC framework whenever a module's

list view is rendered (see include/MVC/View/views/view.list.php). Within view.list.php, checks are made to see if the module has defined a SearchForm.html file. If this file exists, the MVC will run in classic mode and use the aforementioned include/SearchForm/SearchForm.php file to process the search form. Otherwise, the new search form processing is invoked using include/SearchForm/SearchForm2.php and the searchdefs.php file is scanned for, first under the custom/modules/[module]/metadata directory and then in modules/[module]/metadata.

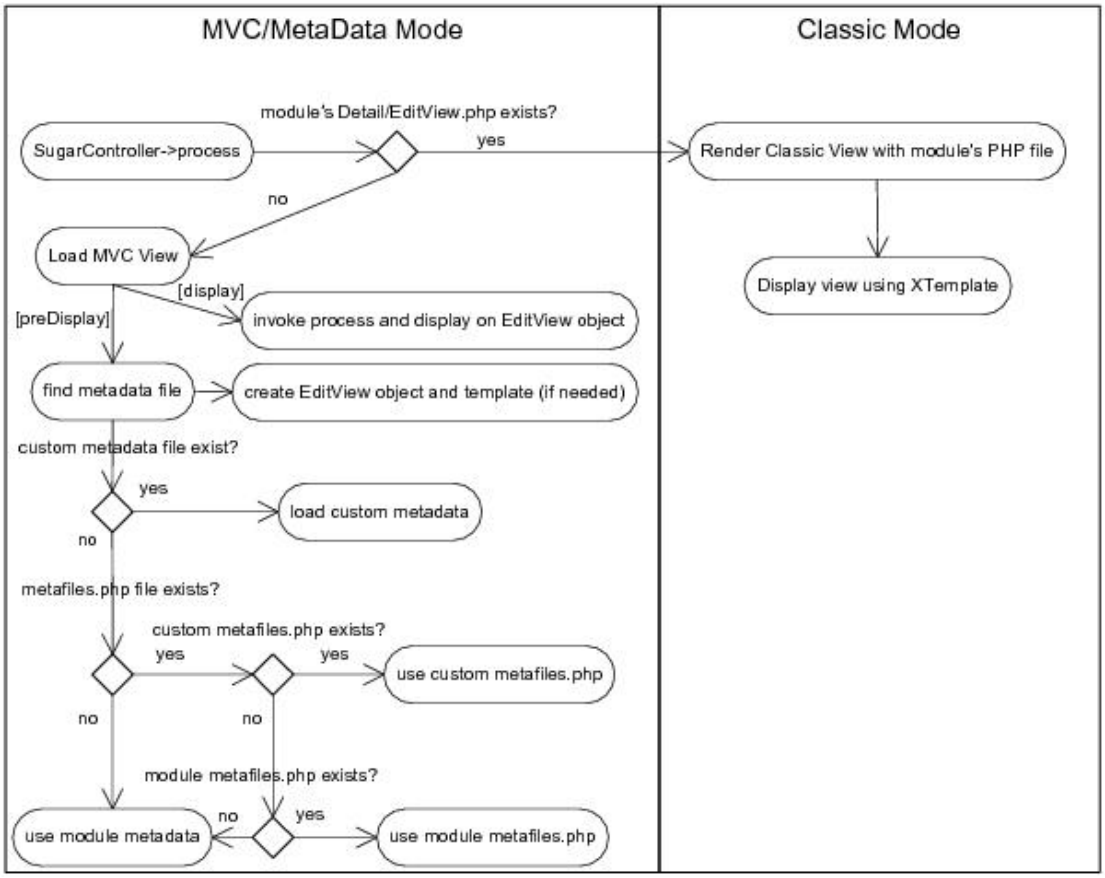
The processing flow for the search form using the metadata subpaneldefs.php file is similar to that of EdiView and DetailView.

DetailView and EditView Metadata

Metadata files are PHP files that declare nested Array values that contain information about the view, such as buttons, hidden values, field layouts, and more. Following is a visual diagram representing how the Array values declared in the Metadata file are nested:



The following diagram highlights the process of how the application determines which Metadata file is to be used when rendering a request for a view:



The "Classic Mode" on the right hand side of the diagram represents the SugarCRM pre-5.x rendering of a Detail/Editview. This section will focus on the MVC/Metadata mode.

When the view is first requested, the preDisplay method will attempt to find the correct Metadata file to use. Typically, the Metadata file will exist in the [root level]/modules/[module]/metadata directory, but in the event of edits to a layout through the Studio interface, a new Metadata file will be created and placed in the [root level]/custom/modules/[module]/metadata directory. This is done so that changes to layouts may be restored to their original state through Studio, and also to allow changes made to layouts to be upgrade-safe when new patches and upgrades are applied to the application. The metafiles.php file that may be loaded allows for the loading of Metadata files with alternate naming conventions or locations. An example of the metafiles.php contents can be found for the Accounts module (though it is not used by default in the application).

```
  $metafiles['Accounts'] = array(
    'detailviewdefs' => 'modules/Accounts/metadata/detailviewdefs.php'
,
    'editviewdefs' => 'modules/Accounts/metadata/editviewdefs.php',
    'ListViewdefs' => 'modules/Accounts/metadata/ListViewdefs.php',
    'searchdefs' => 'modules/Accounts/metadata/searchdefs.php',
```

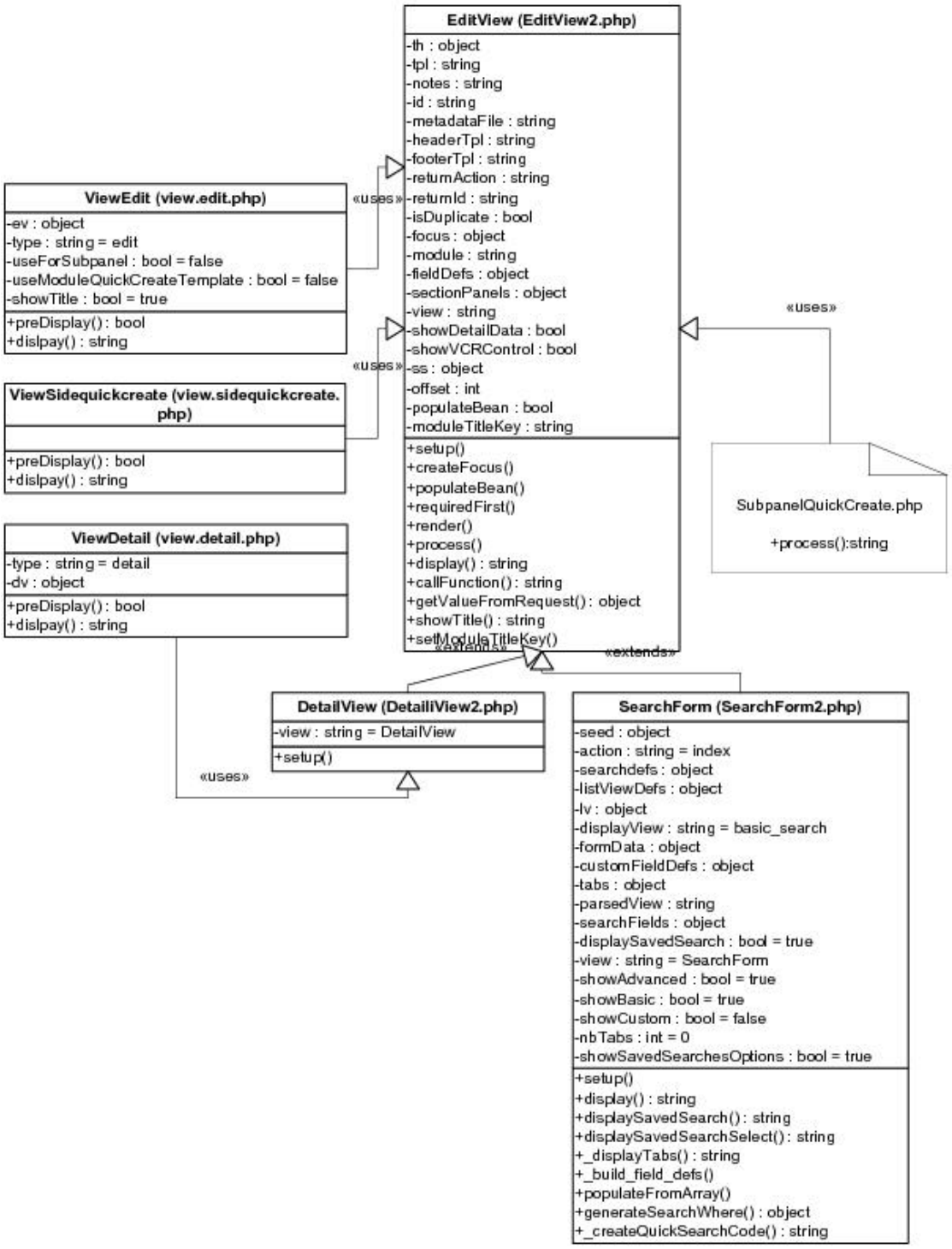
```
    'popupdefs' => 'modules/Accounts/metadata/popupdefs.php',
    'searchfields' => 'modules/Accounts/metadata/SearchFields.php',
);
```

After the Metadata file is loaded, the preDisplay method also creates an EditView object and checks if a Smarty template file needs to be built for the given Metadata file. The EditView object does the bulk of the processing for a given Metadata file (creating the template, setting values, setting field level ACL controls if applicable, etc.). Please see the EditView process diagram for more detailed information about these steps.

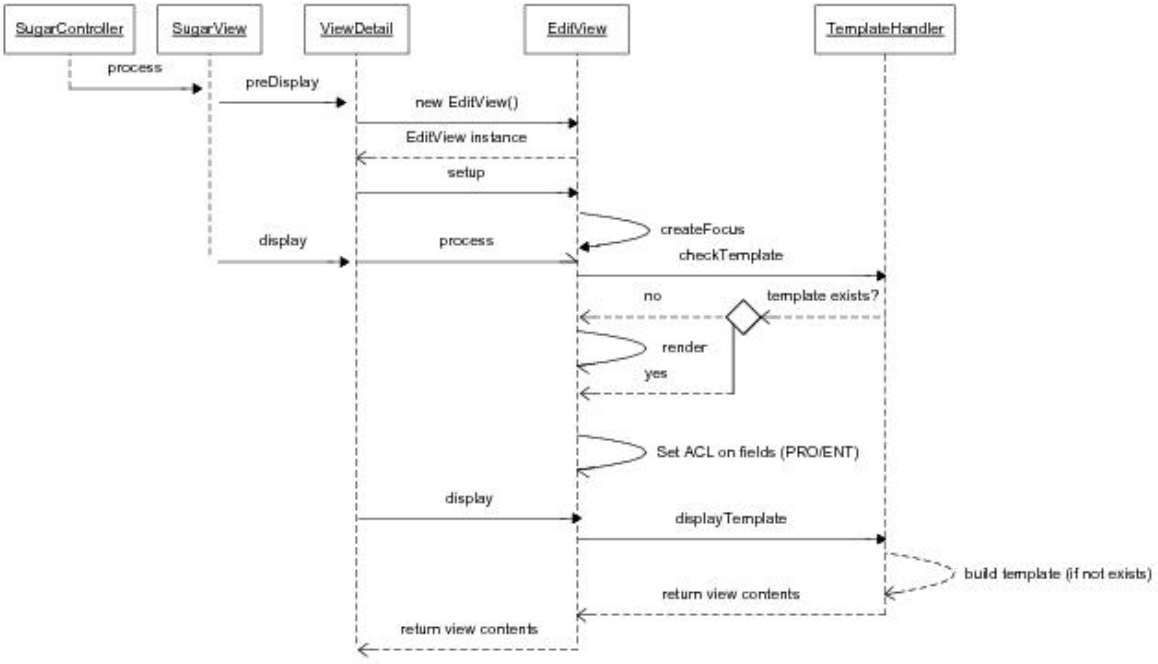
After the preDisplay method is called in the view code, the display method is called, resulting in a call to the EditView object's process method, as well as the EditView object's display method.

The EditView class is responsible for the bulk of the Metadata file processing and creation of the resulting display. The EditView class also checks to see if the resulting Smarty template is already created. It also applies the field level ACL controls for Sugar Sell, Serve, Ultimate, Enterprise, Corporate, and Professional.

The classes responsible for displaying the Detail View and SearchForm also extend and use the EditView class. The ViewEdit, ViewDetail and ViewSidequickcreate classes use the EditView class to process and display their contents. Even the file that renders the quick create form display (SubpanelQuickCreate.php) uses the EditView class. DetailView (in DetailView2.php) and SearchForm (in SearchForm2.php) extend the EditView class while SubpanelQuickCreate.php uses an instance of the EditView class. The following diagram highlights these relationships.



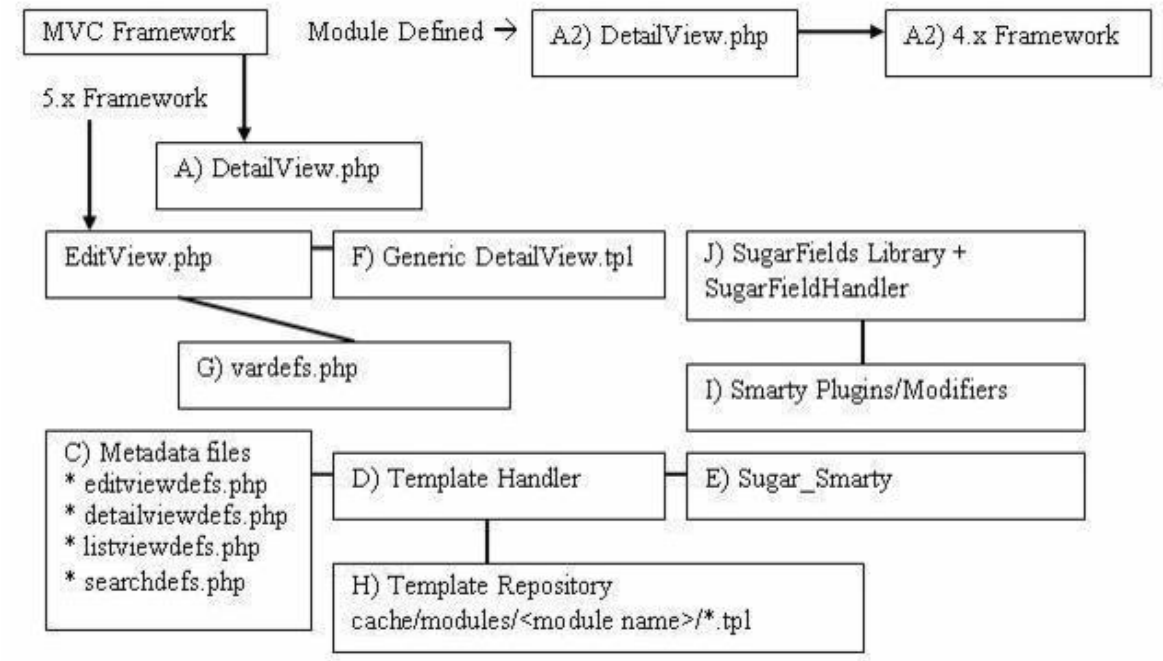
The following diagram highlights the EditView class's main responsibilities and their relationships with other classes in the system. We will use the example of a DetailView request although the sequence will be similar for other views that use the EditView class.



One thing to note is the EditView class's interaction with the TemplateHandler class. The TemplateHandler class is responsible for generating a Smarty template in the cache/modules/<module> directory. For example, for the Accounts module, the TemplateHandler will create the Smarty file, cache/modules/Accounts/DetailView.tpl, based on the Metadata file definition and other supplementary information from the EditView class. The TemplateHandler class actually uses Smarty itself to generate the resulting template that is placed in the aforementioned cache directory.

Some of the modules that are available in the SugarCRM application also extend the ViewDetail class. One example of this is the DetailView for the Projects module. As mentioned in the MVC section, it is possible to extend the view classes by placing a file in the modules/<module>/views directory. In this case, a view.detail.php file exists in the modules/Projects/views folder. This may serve as a useful example in studying how to extend a view and apply additional field/layout settings not provided by the EditView class.

The following diagram shows the files involved with the DetailView example in more detail:



A high level processing summary of the components for DetailViews follows:

The MVC framework receives a request to process the DetaiView.php (A) action for a module. For example, a record is selected from the list view shown on the browser with URL:

index.php?action=DetailView&module=Opportunities&record=46af9843-ccdf-f489-8833

At this point the new MVC framework checks to see if there is a DetailView.php (A2) file in the modules/Opportunity directory that will override the default DetailView.php implementation. The presence of a DetailView.php file will trigger the "classic" MVC view. If there is no DetailView.php (A2) file in the directory, the MVC will also check if you have defined a custom view to handle the DetailView rendering in MVC (that is, checks if there is a file modules/Opportunity/views/view.detail.php). See the documentation for the MVC architecture for more information. Finally, if neither the DetailView.php (A2) nor the view.detail.php exists, then the MVC will invoke include/DetailView/DetailView.php (A).

The MVC framework (see views.detail.php in include/MVC/View/views folder) creates an instance of the generic DetailView (A)

```
  // Call DetailView2 constructor
$dv = new DetailView2();
```

```
// Assign by reference the Sugar_Smarty object created from MVC
// We have to explicitly assign by reference to back support PHP 4.x
$dv->ss =& $this->ss;

// Call the setup function
$dv->setup($this->module, $this->bean, $metadataFile, 'include/DetailV
iew/DetailView.tpl');

// Process this view
$dv->process();

// Return contents to the buffer
echo $dv->display();
```

When the setup method is invoked, a TemplateHandler instance (D) is created. A check is performed to determine which detailviewdefs.php metadata file to used in creating the resulting DetailView. The first check is performed to see if a metadata file was passed in as a parameter. The second check is performed against the custom/studio/modules/[Module] directory to see if a metadata file exists. For the final option, the DetailView constructor will use the module's default detailviewdefs.php metadata file located under the modules/[Module]/metadata directory. If there is no detailviewdefs.php file in the modules/[Module]/metadata directory, but a DetailView.html exists, then a "best guess" version is created using the metadata parser file in include/SugarFields/Parsers/DetailViewMetaParser.php (not shown in diagram).

The TemplateHandler also handles creating the quick search (Ajax code to do look ahead typing) as well as generating the JavaScript validation rules for the module. Both the quick search and JavaScript code should remain static based on the definitions of the current definition of the metadata file. When fields are added or removed from the file through Studio, this template and the resulting updated quick search and JavaScript code will be rebuilt.

It should be noted that the generic DetailView (A) defaults to using the generic DetailView.tpl smarty template file (F). This may also be overridden through the constructor parameters. The generic DetailView (A) constructor also retrieves the record according to the record id and populates the $focus bean variable.

The process() method is invoked on the generic DetailView.php instance:

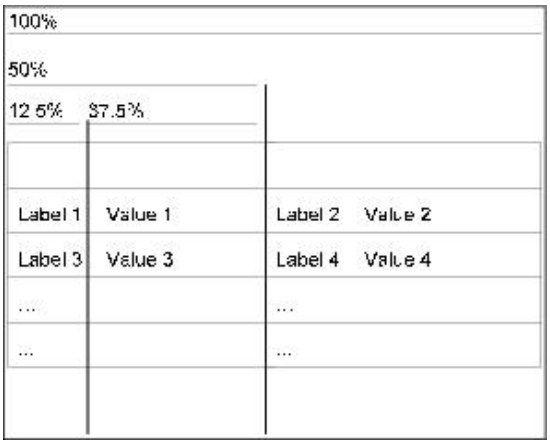
```
  function process()
{
    //Format fields first
    if($this->formatFields)
```

```
    {
        $this->focus->format_all_fields();
    }

    parent::process();
}
```

This, in turn, calls the EditView->process() method since DetailView extends from EditView. The EditView->process() method will eventually call the EditView->render() method to calculate the width spacing for the DetailView labels and values. The number of columns and the percentage of width to allocate to each column may be defined in the metadata file. The actual values are rounded as a total percentage of 100%. For example, given the templateMeta section's maxColumns and widths values:

```
  'templateMeta' => array(
    'maxColumns' => '2',
    'widths' => array(
        array(
            'label' => '10',
            'field' => '30'
        ),
        array(
            'label' => '10',
            'field' => '30'
        )
    ),
),
```

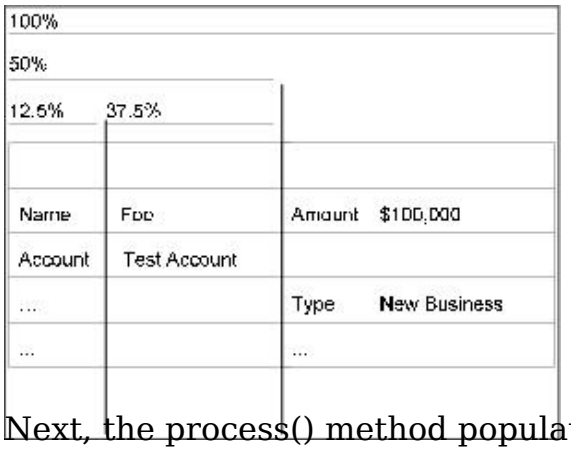
We can see that the labels and fields are mapped as a 1-to-3 ratio. The sum of the widths only equals a total of 80 (10 + 30 x 2) so the actual resulting values written to the Smarty template will be at a percentage ratio of 12.5-to-37.5. The resulting fields defined in the metadata file will be rendered as a table with the column widths as defined:



The actual metadata layout will allow for variable column lengths throughout the displayed table. For example, the metadata portion defined as:

```
  'panels' => array(
    'default' => array(
        array(
            'name',
            array(
                'name' => 'amount',
                'label' => '{$MOD.LBL_AMOUNT} ({$CURRENCY})',
            ),
        ),
        array(
            'account_name',
        ),
        array(
            '',
            'opportunity_type',
        )
    )
)
```

This specifies a default panel under the panels section with three rows. The first row has two fields (name and amount). The amount field has some special formatting using the label override option. The second row contains the account_name field and the third row contains the opportunity_type column.



Next, the process() method populates the $fieldDefs array variable with the vardefs.php file (G) definition and the $focus bean's value. This is done by calling the toArray () method on the $focus bean instance and combining these values with the field definition specified in the vardefs.php file (G).

The display() method is then invoked on the generic DetailView instance for the final step.
When the display() method is invoked, variables to the DetailView.tpl Smarty template are assigned and the module's HTML code is sent to the output buffer.

Before HTML code is sent back, the TemplateHandler (D) first performs a check to see if an existing DetailView template already exists in the cache respository (H). In this case, it will look for file cache/modules/Opportunity/DetailView.tpl. The operation of creating the Smarty template is expensive so this operation ensures that the work will not have to be redone. As a side note, edits made to the DetailView or EditView through the Studio application will clear the cache file and force the template to be rewritten so that the new changes are reflected.

If the cache file does not exist, the TemplateHandler (D) will create the template file and store it in the cache directory. When the fetch() method is invoked on the Sugar_Smarty class (E) to create the template, the DetailView.tpl file is parsed.

Examples

Legacy MVC metadata examples.

Hiding the Quotes Module PDF Buttons

Overview

How to hide the PDF buttons on a Quote.

The PDF buttons on quotes are rendered differently than the standard buttons on most layouts. Since these buttons can't be removed directly from the DetailView in the detailviewdefs, the best approach is using jQuery to hide the buttons.

Note: This customization is only applicable for the quotes module as it is in backward compatibility mode.

Hidding the PDF Buttons

This approach involves modifying the detailviewdefs.php in the custom/modules/Quotes/metadata directory to include a custom JavaScript file. If a custom detailviewdefs.php file doesn't exist, you will need to create it through Studio or by manually coping the detailviewdefs.php from the Quotes stock module metadata directory.

First, we will create a javascript file, say removePdfBtns.js, in the ./custom/modules/Quotes directory. This javascript file will contain the jQuery statements to hide the Quotes "Download PDF" and "Email PDF" buttons on the DetailView of the Quote.

./custom/modules/Quotes/removePdfBtns.js

```
SUGAR.util.doWhen("typeof $ != 'undefined'", function(){
    YAHOO.util.Event.onDOMReady(function(){
        $("#pdfview_button").hide();
        $("#pdfemail_button").hide();
    });
});
```

Next, we will modify the custom detailviewdefs.php file to contain the 'includes' array element in the templateMeta array as follows:

./custom/modules/Quotes/metadata/detailviewdefs.php

```
$viewdefs['Quotes'] = array (
  'DetailView' =>
  array (
    'templateMeta' =>
    array (
      'form' =>
      array (
        'closeFormBeforeCustomButtons' => true,
        'buttons' =>
```

```
        array (
          0 => 'EDIT',
          1 => 'SHARE',
          2 => 'DUPLICATE',
          3 => 'DELETE',
          4 =>
          array (
            'customCode' => '<form action="index.php" method="POST" na
me="Quote2Opp" id="form">
                    <input type="hidden" name="module" value="Quotes">
                    <input type="hidden" name="record" value="{$fields
.id.value}">
                    <input type="hidden" name="user_id" value="{$curre
nt_user->id}">
                    <input type="hidden" name="team_id" value="{$field
s.team_id.value}">
                    <input type="hidden" name="user_name" value="{$cur
rent_user->user_name}">
                    <input type="hidden" name="action" value="QuoteToO
pportunity">
                    <input type="hidden" name="opportunity_subject" va
lue="{$fields.name.value}">
                    <input type="hidden" name="opportunity_name" value
="{$fields.name.value}">
                    <input type="hidden" name="opportunity_id" value="
{$fields.billing_account_id.value}">
                    <input type="hidden" name="amount" value="{$fields
.total.value}">
                    <input type="hidden" name="valid_until" value="{$f
ields.date_quote_expected_closed.value}">
                    <input type="hidden" name="currency_id" value="{$f
ields.currency_id.value}">
                    <input id="create_opp_from_quote_button" title="{$
APP.LBL_QUOTE_TO_OPPORTUNITY_TITLE}"
                        class="button" type="submit" name="opp_to_quot
e_button"
                        value="{$APP.LBL_QUOTE_TO_OPPORTUNITY_LABEL}"
{$DISABLE_CONVERT}></form>',
          ),
        ),
        'footerTpl' => 'modules/Quotes/tpls/DetailViewFooter.tpl',
      ),
      'maxColumns' => '2',
      'widths' =>
      array (
        0 =>
```

```
      array (
        'label' => '10',
        'field' => '30',
      ),
      1 =>
      array (
        'label' => '10',
        'field' => '30',
      ),
    ),
    'includes' =>
    array (
      0 =>
      array (
        'file' => 'custom/modules/Quotes/removePdfBtns.js',
      ),
    ),
    'useTabs' => false,
    'tabDefs' =>
    array (
      'LBL_QUOTE_INFORMATION' =>
      array (
        'newTab' => false,
        'panelDefault' => 'expanded',
      ),
      'LBL_PANEL_ASSIGNMENT' =>
      array (
        'newTab' => false,
        'panelDefault' => 'expanded',
      ),
    ),
  ),
  ...
```

Finally, navigate to:

```
Admin > Repair > Quick Repair and Rebuild
```

The buttons will then be removed from the DetailView layouts.

Manipulating Buttons on Legacy MVC Layouts

Overview

How to add custom buttons to the EditView and DetailView layouts.

Note: This customization is only applicable for modules in backward compatibility mode.

Metadata

Before adding buttons to your layouts, you will need to understand how the metadata framework is used. Detailed information on the metadata framework can be found in the Legacy Metadata section.

Custom Layouts

Before you can add a button to your layout, you will first need to make sure you have a custom layout present. The stock layouts are located in ./modules/<module>/metadata/ and must be recreated in ./custom/modules/<module>/metadata/.

There are two ways to recreate a layout in the custom directory if it does not already exist. The first is to navigate to:

```
Studio > {Module} > Layouts > {View}
```

Once there, you can click the "Save & Deploy" button. This will create the layoutdef for you. Alternatively, you can also manually copy the layoutdef from the stock folder to the custom folder.

Editing Layouts

When editing layouts you have three options in having your changes reflected in the UI.

Developer Mode

You can turn on Developer Mode:

```
Admin > System Settings
```

Developer Mode will remove the caching of the metadata framework. This will cause your changes to be reflected when the page is refreshed. Make sure this setting is deactivated when you are finished with your customization.

Quick Repair and Rebuild

You can run a Quick Repair and Rebuild:

```
Admin > Repair > Quick Repair and Rebuild
```

Doing this will rebuild the cache for the metadata.

Saving & Deploying the Layout in Studio

You may also choose to load the layout in studio and then save & deploy it:

```
Admin > Studio > {Module} > Layouts > {View}
```

This process can be a bit confusing, however, once a layout is changed, you can then choose to load the layout in studio and then click "Save & Deploy" . This will rebuild the cache for that specific layout. Please note that any time you change the layout, you will have to reload the Studio layout view before deploying in order for this to work correctly.

Adding Custom Buttons

When adding buttons, there are several things to consider when determining how the button should be rendered. The following sections will outline these scenarios when working with the accounts editviewdefs located in ./custom/modules/Accounts/metadata/editviewdefs.php.

JavaScript Actions

If you are adding a button solely to execute JavaScript (no form submissions), you can do so by adding the button HTML to:

```
$viewdefs['<Module>']['<View>']['templateMeta']['form']['buttons']
```

Example

```
<?php
$viewdefs['Accounts'] =
array (
  'DetailView' =>
  array (
    'templateMeta' =>
    array (
      'form' =>
      array (
        'buttons' =>
        array (
          0 => 'EDIT',
          1 => 'DUPLICATE',
          2 => 'DELETE',
          3 => 'FIND_DUPLICATES',
          4 => 'CONNECTOR',
          5 =>
          array (
            'customCode' => '<input id="JavaScriptButton" title="JavaS
cript Button" class="button" type="button" name="JavaScriptButton" val
ue="JavaScript Button" onclick="alert(\'Button JavaScript\')">',
          ),
        ),
      ),
```

Submitting the Stock View Form

If you intend to submit the stock layout form ('formDetailView' or 'formEditView') to a new action, you can do so by adding a the button HTML with an onclick event as shown below to:

```
$viewdefs['<Module>']['<View>']['templateMeta']['form']['buttons']
```

Example

```
<?php

$viewdefs['Accounts'] =

array (

  'DetailView' =>

  array (

    'templateMeta' =>

    array (

      'form' =>

      array (

        'hidden' =>

        array (

          0 => '<input type="hidden" id="customFormField" name="custom
FormField" value="">',

        ),

        'buttons' =>

        array (

          0 => 'EDIT',
```

```
          1 => 'DUPLICATE',

          2 => 'DELETE',

          3 => 'FIND_DUPLICATES',

          4 => 'CONNECTOR',

          5 =>

          array (

            'customCode' => '<input id="SubmitStockFormButton" title="
Submit Stock Form Button" class="button" type="button" name="SubmitSto
ckFormButton" value="Submit Stock Form Button" onclick="var _form = do
cument.getElementById(\'formDetailView\'); _form.customFormField.value
 = \'CustomValue\'; _form.action.value = \'CustomAction\'; SUGAR.ajaxU
I.submitForm(_form);">',

          ),

        ),

      ),
```

You should note in this example that there is also a 'hidden' index. This is where you should add any custom hidden inputs:

```
  $viewdefs['<Module>'][
  '<View>'][
  'templateMeta'][
  'form'][
  'hidden']
```

Submitting Custom Forms

If you intend to submit a custom form, you will first need to set 'closeFormBeforeCustomButtons' to true. This will close out the current views form and allow you to create your own.

```
$viewdefs['<Module>']['<View>']['templateMeta']['form']['closeFormBefo
reCustomButtons']
```

Next, you will add the form and button HTML as shown below to:

```
$viewdefs['<Module>']['<View>']['templateMeta']['form']['buttons']
```

Example

```
<?php

$viewdefs['Accounts'] =

array (

  'DetailView' =>

  array (

    'templateMeta' =>

    array (

      'form' =>

      array (

        'closeFormBeforeCustomButtons' => true,

        'buttons' =>

        array (

          0 => 'EDIT',

          1 => 'DUPLICATE',

          2 => 'DELETE',

          3 => 'FIND_DUPLICATES',

          4 => 'CONNECTOR',
```

```
          5 =>

          array (

            'customCode' => '<form action="index.php" method="POST" na
me="CustomForm" id="form"><input type="hidden" name="customFormField"
name="customFormField" value="CustomValue"><input id="SubmitCustomForm
Button" title="Submit Custom Form Button" class="button" type="submit"
 name="SubmitCustomFormButton" value="Submit Custom Form Button"></for
m>',

          ),

        ),

      ),
```

Removing Buttons

To remove a button from the detail view will require modifying the ./modules/<module>/metadata/detailviewdefs.php.

The code is originally defined as:

```
$viewdefs[$module_name] = array (

    'DetailView' => array (

        'templateMeta' => array (

            'form' => array (

                'buttons' => array (

                    'EDIT',

                    'DUPLICATE',

                    'DELETE',

                    'FIND_DUPLICATES'

                ),
```

```
            ),
```

To remove one or more buttons, simply remove the 'buttons' attribute(s) that you do not want on the view.

```
$viewdefs[$module_name] = array (

    'DetailView' => array (

        'templateMeta' => array (

            'form' => array (

                'buttons' => array (

                    'DELETE',

                ),

            ),
```

Manipulating Layouts Programmatically

Overview

How to manipulate and merge layouts programmatically.

Note: This customization is only applicable for modules in backward compatibility mode.

The ParserFactory

The ParserFactory can be used to manipulate layouts such as editviewdefs or detailviewdefs. This is a handy when creating custom plugins and needing to merge changes into an existing layout. The following example will demonstrate

how to add a button to the detail view:

```
  <?php

//Instantiate the parser factory for the Accounts DetailView.
require_once('modules/ModuleBuilder/parsers/ParserFactory.php');
$parser = ParserFactory::getParser('detailview', 'Accounts');

//Button to add
$button = array(
    'customCode'=>'<input type="button" name="customButton" value="Cus
tom Button">'
);

//Add button into the parsed layout
array_push($parser->_viewdefs['templateMeta']['form']['buttons'], $but
ton);

//Save the layout
$parser->handleSave(false);
```

Modifying Layouts to Display Additional Columns

Overview

How to add additional columns to layouts.

By default, the editview, detailview, and quickcreate layouts for each module display two columns of fields. The number of columns to display can be customized on a per-module basis with the following steps.

Note: This customization is only applicable for modules in backward compatibility mode.

Resolution

SugarCloud

First, you will want to ensure your layouts are deployed in the custom directory. If you have not previously customized your layouts via Studio, go to Admin > Studio

> {Module Name} > Layouts. From there, select each layout you wish to add additional columns to and click 'Save & Deploy'. This action will create a corresponding layout file under the ./custom/modules/{Module Name}/metadata/ directory. The files will be named editviewdefs.php, detailviewdefs.php, and quickcreatedefs.php depending on the layouts deployed.

To access your custom files, go to Admin > Diagnostic Tool, uncheck all the boxes except for "SugarCRM Custom directory" and then click "Execute Diagnostic". This will generate an archive of your instance's custom directory to download, and you will find the layout files in the above path. Open the custom layout file, locate the 'maxColumns' value, and change it to the number of columns you would like to have on screen:

```
  'maxColumns' => '3',
```

Once that is updated, locate the 'widths' array to define the spacing for your new column(s). You should have a label and field entry for each column in your layout:

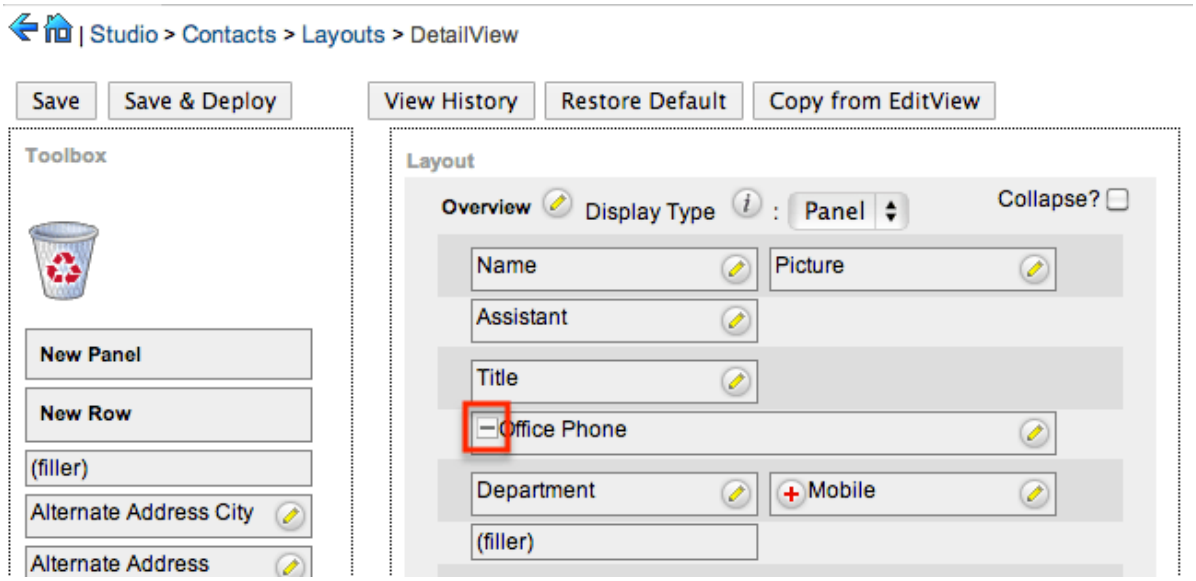
```
  'widths' => array (
    0 => array (
        'label' => '10',
        'field' => '30',
    ),
    1 => array (
        'label' => '10',
        'field' => '30',
    ),
    2 => array (
        'label' => '10',
        'field' => '30',
    ),
),
```

After this is completed, you will need to create a module-loadable package to install the changes on your SugarCloud instance. More information on creating this package can be found in Creating an Installable Package that Creates New Fields. To upload and install the package, go to Admin > Module Loader.

Note: Sugar Sell Essentials customers do not have the ability to upload custom file packages to Sugar using Module Loader.

Once the installation completes, you can navigate to Studio and add fields to your

new column in the layout. For any rows that already contain two fields, the second field will automatically span the second and third column. Simply click the minus (-) icon to contract the field to one column and expose the new column space:



After you have added the desired fields in Studio, click 'Save & Deploy', and you are ready to go!

On-Site

First, you will want to ensure your layouts are deployed in the custom directory. If you have not previously customized your layouts via Studio, go to Admin > Studio > {Module Name} > Layouts. From there, select each layout you wish to add additional columns to and click 'Save & Deploy'. This action will create a corresponding layout file under the ./custom/modules/{Module Name}/metadata/ directory. The files will be named editviewdefs.php, detailviewdefs.php, and quickcreatedefs.php depending on the layouts deployed.

Next, open the custom layout file, locate the 'maxColumns' value, and change it to the number of columns you would like to have on screen:

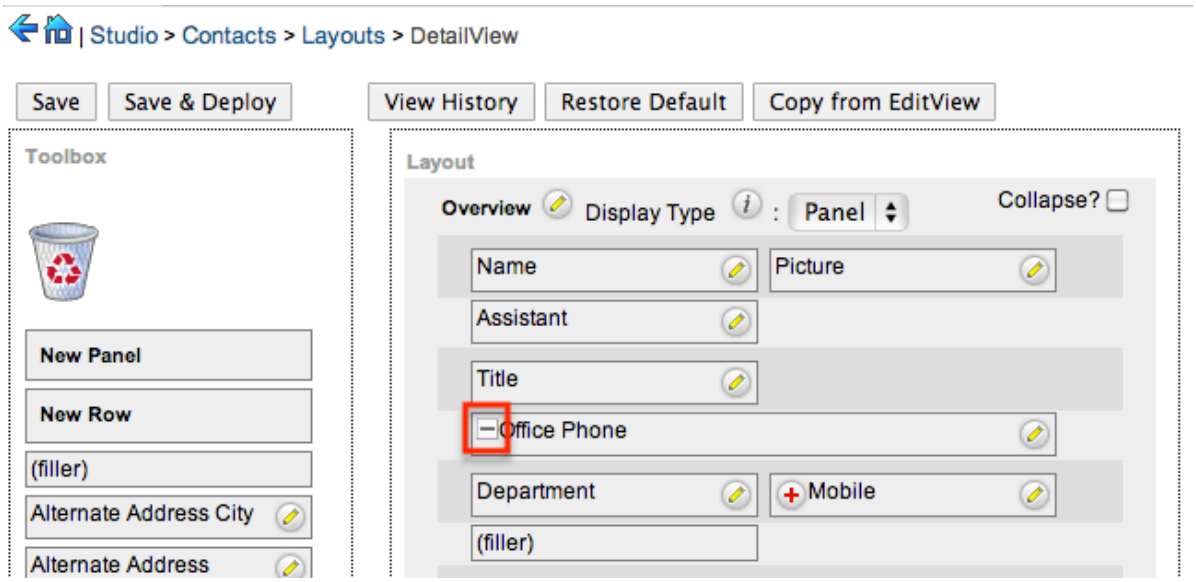
```
'maxColumns' => '3',
```

Once that is updated, locate the 'widths' array to define the spacing for your new column(s). You should have a label and field entry for each column in your layout:

```
'widths' => array (
  0 => array (
```

```
        'label' => '10',
        'field' => '30',
    ),
    1 => array (
        'label' => '10',
        'field' => '30',
    ),
    2 => array (
        'label' => '10',
        'field' => '30',
    ),
),
```

Once this is completed, you can navigate to Studio and add fields to your new column in the layout. For any rows that already contain two fields, the second field will automatically span the second and third column. Simply click the minus (-) icon to contract the field to one column and expose the new column space:



After you have added the desired fields in Studio, click 'Save & Deploy', and you are ready to go!

Examples

Provides an overview of example MVC customizations.

Changing the ListView Default Sort Order

Overview

This article addresses the need to customize the advanced search layout options for modules in backward compatibility mode to change the default sort order from ascending to descending.

Customization Information

This customization is only for modules in backward compatibility mode and involves creating custom files that extend stock files. You should note that this customization does not address all scenarios within the view that may assign a sort order.

Extending the Search Form

First, we will need to extend the SearchForm class. To do this, we will create a CustomSearchForm class that extends the original SearchForm class located in ./include/SearchForm/SearchForm2.php. We will then override the _displayTabs method to check the $_REQUEST['sortOrder'] and default it to descending if it isn't set.

./custom/include/SearchForm/SearchForm2.php

```
  <?php

require_once 'include/SearchForm/SearchForm2.php';

class CustomSearchForm extends SearchForm
{
    /**
     * displays the tabs (top of the search form)
     *
     * @param string $currentKey key in $this->tabs to show as the cur
rent tab
     *
     * @return string html
     */
    function _displayTabs($currentKey)
    {
        //check and set the default sort order
        if (!isset($_REQUEST['sortOrder']))
        {
```

```
            $_REQUEST['sortOrder'] = 'DESC';
        }

        return parent::_displayTabs($currentKey);;
    }
}

?>
```

Extending the List View

Next, we will need to extend the ListView. We will create a ViewCustomList class that extends the original ListView located in ./include/MVC/View/views/view.list.php. In the ViewCustomList class, we will override the prepareSearchForm and getSearchForm2 methods to call the CustomSearchForm class.

./custom/include/MVC/View/views/view.customlist.php

```
  <?php

require_once 'include/MVC/View/views/view.list.php';

class ViewCustomList extends ViewList
{

    function prepareSearchForm()
    {
        $this->searchForm = null;

        //search
        $view = 'basic_search';

        if(!empty($_REQUEST['search_form_view']) && $_REQUEST['search_
form_view'] == 'advanced_search')

            $view = $_REQUEST['search_form_view'];

        $this->headers = true;

        if(!empty($_REQUEST['search_form_only']) && $_REQUEST['search_
form_only'])
            $this->headers = false;
        elseif(!isset($_REQUEST['search_form']) || $_REQUEST['search_f
```

```
orm'] != 'false')
        {
            if(isset($_REQUEST['searchFormTab']) && $_REQUEST['searchF
ormTab'] == 'advanced_search')
            {
                $view = 'advanced_search';
            }
            else
            {
                $view = 'basic_search';
            }
        }

        $this->view = $view;

        $this->use_old_search = true;

        if (SugarAutoLoader::existingCustom('modules/' . $this->module
 . '/SearchForm.html') &&

            !SugarAutoLoader::existingCustom('modules/' . $this->modul
e . '/metadata/searchdefs.php')) {
            require_once('include/SearchForm/SearchForm.php');
            $this->searchForm = new SearchForm($this->module, $this->s
eed);

            }  else {

            $this->use_old_search = false;
            //Updated to require the extended CustomSearchForm class
            require_once('custom/include/SearchForm/SearchForm2.php');

            $searchMetaData = SearchForm::retrieveSearchDefs($this->mo
dule);

            $this->searchForm = $this->getSearchForm2($this->seed, $th
is->module, $this->action);
            $this->searchForm->setup($searchMetaData['searchdefs'], $s
earchMetaData['searchFields'], 'SearchFormGeneric.tpl', $view, $this->
listViewDefs);
            $this->searchForm->lv = $this->lv;
        }
    }

    /**
     * Returns the search form object
```

```
     *
     * @return SearchForm
     */
    protected function getSearchForm2($seed, $module, $action = "index
")
    {
        //Updated to use the extended CustomSearchForm class
        return new CustomSearchForm($seed, $module, $action);
    }

}

?>
```

Extending the Sugar Controller

Finally, we will create a CustomSugarController class that extends the orginal SugarController located in ./include/MVC/Controller/SugarController.php. We will then need to override the do_action and post_action methods to execute their parent methods as well as the action_listview method to assign the custom view to the view attribute.

./custom/include/MVC/Controller/SugarController.php

```
  <?php

/**
* Custom SugarCRM controller
* @api
*/
class CustomSugarController extends SugarController
{

    /**
     * Perform the specified action.
     * This can be overridde in a sub-class
     */
    private function do_action()
    {
        return parent::do_action();
    }

    /**
     * Perform an action after to the specified action has occurred.
```

```
     * This can be overridde in a sub-class
     */
    private function post_action()
    {
        return parent::post_action();
    }

    /**
     * Perform the listview action
     */
    protected function action_listview()
    {
        parent::action_listview();

        //set the new custom view
        $this->view = 'customlist';
    }
}

?>
```

Data Framework

The Sugar application comprises core elements such as modules, fields, vardefs, and other foundational components. The Data Framework pages document how these core elements are modeled and implemented in Sugar.

Modules

Overview

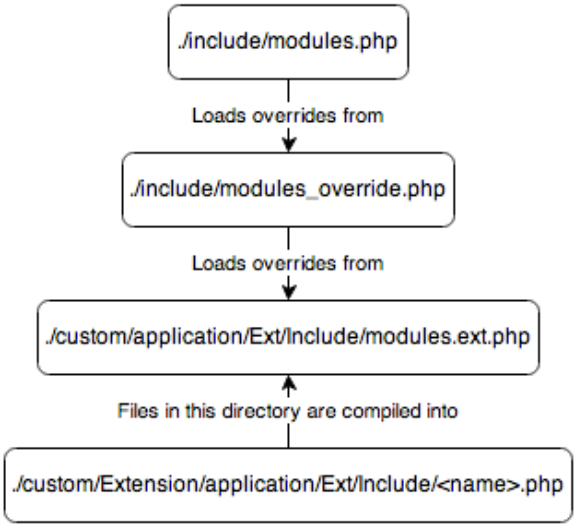
How modules are defined and used within the system

Module Definitions

The module definitions, defined in ./include/modules.php, determine how modules are displayed and used throughout the application. Any custom metadata, whether from a plugin or a custom module, should be loaded through the Include extension. Prior to 6.3.x, module definitions could be added by creating the file ./include/modules_override.php. This method of creating module definitions is still compatible but is not recommended from a best practices standpoint.

Hierarchy Diagram

The modules metadata are loaded in the following manner:



$moduleList

The $moduleList is an array containing a list of modules in the system. The format of the array is to have a numeric index and a value of the modules unique key.

```
$moduleList[] = 'Accounts';
```

$beanList

The $beanList variable is an array that stores a list of all active beans (modules) in the application. The format of the array is array('<bean plural name>' => '<bean singular name>');. The $beanList key is used to lookup values in the $beanFiles variable.

```
$beanList['Accounts'] = 'Account';
```

$beanFiles

The $beanFiles variable is an array used to reference the class files for a bean. The format of the array is array('<bean singular name>' => '<relative class file>');. The bean name, stored in singular form, is a reference to the class name of the object, which is looked up from the $beanList 'key'.

```
$beanFiles['Account'] = 'modules/Accounts/Account.php';
```

$modInvisList

The $modInvisList variable removes a module from the navigation tab in the MegaMenu, reporting, and it's subpanels under related modules.To enable a hidden module for reporting, you can use $report_include_modules. To enable a hidden modules subpanels on related modules, you can use $modules_exempt_from_availability_check. The

```
$modInvisList[] = 'Prospects';
```

$modules_exempt_from_availability_check

The $modules_exempt_from_availability_check variable is used in conjunction with $modInvisList. When a module has been removed from the MegaMenu view with $modInvisList, this will allow for the display of the modules subpanels under related modules.

```
$modules_exempt_from_availability_check['OAuthKeys'] = 'OAuthKeys';
```

$report_include_modules

The $report_include_modules variable is used in conjunction with $modInvisList. When a module has been hidden with $modInvisList, this will allow for the module to be enabled for reporting.

```
$report_include_modules['Prospects'] = 'Prospect';
```

$adminOnlyList

The $adminOnlyList variable is an extra level of security for modules that are can be accessed only by administrators through the Admin page. Specifying all will restrict all actions to be admin only.

```
$adminOnlyList['PdfManager'] = array(
    'all' => 1
);
```

$bwcModules

The $bwcModules variable determines which modules are in backward compatibility mode. More information on backward compatibility can be found in the Backward Compatibility section.

Models

Overview

Each module in Sugar is extending the Sugar Model. This model is determined by the SugarBean, which contains methods to create, read/retrieve, update, and delete records in the database and any subclass of the SugarBean. Many of the common Sugar modules also use the SugarObjects class, which is explained in the next section.

SugarObject Templates

Sugar objects extend the concept of subclassing a step further and allow you to subclass the vardefs. This includes inheriting of fields, relationships, indexes, and language files. However, unlike subclassing, you are not limited to a single inheritance. For example, if there were a Sugar object for fields used in every module (e.g. id, deleted, or date_modified), you could have the module inherit from both the Basic Sugar object and the Person Sugar object.

To further illustrate this, let's say the Basic object type has a field 'name' with length 10 and the Company object has a field 'name' with length 20. If you inherit from Basic first and then Company, your field will inherit the Company object's length of 20. However, the module-level setting always overrides any values provided by Sugar objects, so, if the field 'name' in your module has been set to length 60, then the field's length will ultimately be 60.

There are six types of SugarObject templates:

- **Basic** : Contains the basic fields required by all Sugar modules
- **Person** : Based on the Contacts, Prospects, and Leads modules
- **Issue** : Based on the Bugs and Cases modules
- **Company** : Based on the Accounts module
- **File** : Based on the Documents module
- **Sale** : Based on the Opportunities module

SugarObject Interfaces

In addition to the object templates above, "assignable" object templates can be used for modules with records that should contain an Assigned To field. Although every module does not use this, most modules allow assignment of records to users. SugarObject interfaces allow us to add the assignable attribute to these modules.

SugarObject interfaces and SugarObject templates are very similar to one another, but the main distinction is that templates have a base class you can subclass while interfaces do not. If you look into the file structure, templates include many additional files, including a full metadata directory. This is used primarily for Module Builder.

File Structure

- ./include/SugarObjects/interfaces
- ./include/SugarObjects/templates

Implementation

There are two things you need to do to take advantage of Sugar objects:

1. Subclass the SugarObject class you wish to extend for your class:

```
class MyClass extends Person
{
    function MyClass()
    {
        parent::Person();
    }
}
```

2. Add the following line to the end of the vardefs.php file:

```
VardefManager::createVardef('MyClass', 'MyClass', array('default
', 'assignable', 'team_security', 'person'));
```

   This snippet tells the VardefManager to create a cache of the MyClass vardefs with the addition of all the default fields, assignable fields, team security fields, and all fields from the person class.

Performance Considerations

VardefManager caches the generated vardefs into a single file that is loaded at runtime. If that file is not found, Sugar loads the vardefs.php file, located in your module's directory. The same is true for language files. This caching also includes data for custom fields and any vardef or language extensions that are dropped into the extension framework.

Cache Files

- ./cache/modules/<module>/<object_name>vardefs.php
- ./cache/modules/<module>/languages/en_us.lang.php

SugarBean

Overview

The SugarBean class supports all the model operations to and from the database. Any module that interacts with the database utilizes the SugarBean class, which contains methods to create, read/retrieve, update, and delete records in the database (CRUD), as well as fetch related records.

CRUD Handling

The SugarBean handles the most basic functions of the Sugar database, allowing you to create, retrieve, update, and delete data.

Creating and Retrieving Records

The BeanFactory class is used for bean loading. The class should be used whenever you are creating or retrieving bean objects. It will automatically handle any classes required for the bean. More information on this can be found in the BeanFactory section.

Obtaining the Id of a Recently Saved Bean

For new records, a GUID is generated and assigned to the record id field. Saving new or existing records returns a single value to the calling routine, which is the id attribute of the saved record. The id has the following format:

```
aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee
```

You can retrieve a newly created record's id with the following:

```
  //Create bean
$bean = BeanFactory::newBean($module);

//Populate bean fields
$bean->name = 'Example Record';

//Save
$bean->save();

//Retrieve the bean id
$record_id = $bean->id;
```

Saving a Bean with a Specific ID

Saving a record with a specific id requires the id and new_with_id attribute of the bean to be set. When doing so, it is important that you use a globally unique identifier. Failing to do so may result in unexpected behavior in the system. An example setting an id is shown below:

```
  //Create bean
$bean = BeanFactory::newBean($module);

//set the new flag
$bean->new_with_id = true;

//Set the record id with a static id
$id = '38c90c70-7788-13a2-668d-513e2b8df5e1';
$bean->id = $id;

//or have the system generate an id for you
//$id = Sugarcrm\Sugarcrm\Util\Uuid::uuid1()
//$bean->id = $id;

//Populate bean fields
$bean->name = 'Example Record';

//Save
$bean->save();
```

Setting new_with_id to true prevents the save method from creating a new id value and uses the assigned id attribute. If the id attribute is empty and the new_with_id attribute is set to true, the save will fail. If you would like for the system to

generate an id for you, you can use Sugarcrm\Sugarcrm\Util\Uuid::uuid1().

Distinguishing New from Existing Records

To identify whether or not a record is new or existing, you can check the fetched_rows property. If the $bean has a fetched_row, it was loaded from the database. An example is shown below:

```
  if (!isset($bean->fetched_row['id'])) {
    //new record
} else {
    //existing record
}
```

If you are working with a logic hook such as before_save or after_save, you should check the arguments.isUpdate property to determine this as shown below:

```
  <?php

if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry Poin
t');

class logic_hooks_class
{
    function hook_method($bean, $event, $arguments)
    {
        if (isset($arguments['isUpdate']) && $arguments['isUpdate'] ==
 false) {
            //new record
        } else {
            //existing record
        }
    }
}

?>
```

Retrieving a Bean by Unique Field

Sometimes developers have a need to fetch a record based on a unique field other than the id. In previous versions you were able to use the retrieve_by_string_fields() method to accomplish this, however, that has now

been deprecated. To search and fetch records, you should use the SugarQuery builder. An example of this is shown below:

```
  require_once('include/SugarQuery/SugarQuery.php');
$sugarQuery = new SugarQuery();

//fetch the bean of the module to query
$bean = BeanFactory::newBean('<modules>');

//create first query
$sql = new SugarQuery();
$sql->select('id');
$sql->from($bean);
$sql->Where()->equals('<field>', '<unique value>');

$result = $sql->execute();

$count = count($result);

if ($count == 0) {
    //no results were found
} elseif ($count == 1) {
    //one result was found
    $bean = BeanFactory::getBean('<module>', $result[0]['id']);
} else {
    //multiple results were found
}
```

Updating a Bean

You can update a bean by fetching a record and then updating the property:

```
  //Retrieve bean
$bean = BeanFactory::retrieveBean($module, $id);

//Fields to update
$bean->name = 'Updated Name';

//Save
$bean->save();
```

Note: Disabling row-level security when accessing a bean should be set to

true only when it is absolutely necessary to bypass security, for example, when updating a Lead record from a custom Entry Point. An example of accessing the bean while bypassing row security is:

```
  $bean = BeanFactory::retrieveBean($module, $record_id, array('disabl
e_row_level_security' => true));
```

Updating a Bean Without Changing the Date Modified

The SugarBean class contains an attribute called update_date_modified, which is set to true when the class is instantiated and means that the date_modified attribute is updated to the current database date timestamp.
Setting update_date_modified to false would result in the date_modified attribute not being set with the current database date timestamp.

```
  //Retrieve bean
$bean = BeanFactory::retrieveBean($module, $id);

//Set modified flag
$bean->update_date_modified = false;

//Fields to update
$bean->name = 'Updated Name';

//Save
$bean->save();
```

Note: Disabling row-level security when accessing a bean should be set to true only when it is absolutely necessary to bypass security, for example, when updating a Lead record from a custom Entry Point. An example of accessing the bean while bypassing row security is:

```
  $bean = BeanFactory::retrieveBean($module, $record_id, array('disabl
e_row_level_security' => true));
```

Deleting a Bean

Deleting a bean can be done by fetching it then calling the mark_deleted() method which makes sure any relationships with related records are removed as well:

```
  //Retrieve bean
$bean = BeanFactory::retrieveBean($module, $id);

//Set deleted to true
$bean->mark_deleted($bean->id);

//Save
$bean->save();
```

Note: Disabling row-level security when accessing a bean should be set to true only when it is absolutely necessary to bypass security, for example, when updating a Lead record from a custom Entry Point. An example of accessing the bean while bypassing row security is:

```
  $bean = BeanFactory::retrieveBean($module, $record_id, array('disabl
e_row_level_security' => true));
```

Fetching Relationships

This section explains how the SugarBean class can be used to fetch related information from the database.

Fetching Related Records

To fetch related records, load the relationship using the link:

```
  //If relationship is loaded
if ($bean->load_relationship($link)) {
    //Fetch related beans
    $relatedBeans = $bean->$link->getBeans();
}
```

An example of this is to load the contacts related to an account:

```
  //Load Account
$bean = BeanFactory::getBean('Accounts', $id);

//If relationship is loaded
if ($bean->load_relationship('contacts')) {
```

```
    //Fetch related beans
    $relatedBeans = $bean->contacts->getBeans();
}
```

Fetching Related Record IDs

To fetch **only related record IDs**, load the relationship using the link:

```
  //If relationship is loaded
if ($bean->load_relationship($link)) {
    //Fetch related record IDs
    $relatedBeans = $bean->$link->get();
}
```

An example of this is to load the record IDs of contacts related to an account:

```
  //Load Account
$bean = BeanFactory::getBean('Accounts', $id);

//If relationship is loaded
if ($bean->load_relationship('contacts')) {
    //Fetch related beans
    $relatedBeans = $bean->contacts->get();
}
```

Fetching a Parent Record

Fetching a parent record is similar to fetching child records in that you will still need to load the relationship using the link:

```
  //If relationship is loaded
if ($bean->load_relationship($link)) {
    //Fetch related beans
    $relatedBeans = $bean->$link->getBeans();

    $parentBean = false;
    if (!empty($relatedBeans)) {
        //order the results
        reset($relatedBeans);
```

```
        //first record in the list is the parent
        $parentBean = current($relatedBeans);
    }
}
```

An example of this is to load a contact and fetch its parent account:

```
  //Load Contact
$bean = BeanFactory::getBean('Contacts', $id);

//If relationship is loaded
if ($bean->load_relationship('accounts')) {
    //Fetch related beans
    $relatedBeans = $bean->accounts->getBeans();

    $parentBean = false;
    if (!empty($relatedBeans)) {
        //order the results
        reset($relatedBeans);

        //first record in the list is the parent
        $parentBean = current($relatedBeans);
    }
}
```

Customizing Core SugarBeans

Overview

This article covers how to extend core SugarBean objects to implement custom code. This can be used to alter the stock assignment notification message or to add logic to change the default modules behavior.

Customizing a Core Module

The best approach to customizing a core SugarBean object is to become familiar with how the core Bean operates and only modify the bean logic where absolutely necessary. After understanding where you wish to make your customization in the module Bean class, you can extend the class in the custom directory. In order to avoid duplicating code from the core Bean class, it is always best to extend the class rather than duplicate the class to the custom directory.

Extending the SugarBean

For this example, we will look at modifying the Leads SugarBean object. To extend the core Leads bean, create the following file:

./custom/modules/Leads/CustomLead.php

```
  <?php

require_once 'modules/Leads/Lead.php';

class CustomLead extends Lead
{
    /**
     * Saves the record
     * - You can use this method to implement before save or after sav
e logic
     *
     * @param bool $check_notify
     * @return string
     */
    function save($check_notify = FALSE)
    {
        $id = parent::save($check_notify);
        return $id;
    }

    /**
     * Retrieves a record
     * - You can use this method to set properties when fetching a bea
n
     *
     * @param string $id
     * @param bool $encode
     * @param bool $deleted
     * @return $this
     */
    function retrieve($id = '-1', $encode = true, $deleted = true)
    {
        return parent::retrieve($id, $encode, $deleted);
    }

    /**
     * Calls custom logic events
     * - You can use this method to watch for specific logic hook even
ts
```

```
     *
     * @param $event
     * @param array $arguments
     */
    function call_custom_logic($event, $arguments = array())
    {
        parent::call_custom_logic($event, $arguments);
    }
}
```

Register the Custom Class

Once the custom SugarBean class file has been created, register that class to be used by the module so that the BeanFactory knows which class to use. To register the class you will create the following file in the ./custom/Extension/ directory:

./custom/Extension/application/Ext/Include/customLead.php

```
  <?php

/**
* The $objectList array, maps the module name to the Vardef property
* By default only a few core modules have this defined, since their Cl
ass/Object names differs from their Vardef Property
**/
$objectList['Leads'] = 'Lead';

// $beanList maps the Bean/Module name to the Class name
$beanList['Leads'] = 'CustomLead';

// $beanFiles maps the Class name to the PHP Class file
$beanFiles['CustomLead'] = 'custom/modules/Leads/CustomLead.php';
```

Note: The $objectList array only needs to be set on those modules that do not have it set already. You can view ./include/modules.php to see the core modules that have it defined already.

Once the registration file is in place, go to Admin > Repairs, and run a Quick Repair and Rebuild so that the system starts using the custom class.

Things to Note

Overriding a core SugarBean class comes with some caveats:

1. The custom class is only used when the product core code uses the BeanFactory.
    - For most circumstances, Sugar is set up to use BeanFactory, however, legacy code or specific logic that is pairs core modules together may use direct calls to a core SugarBean class, which would then cause the custom class to not be used. In those scenarios, it is recommended to use a logic look instead.
2. Extending the Cases module doesn't affect the email Import process.
3. If the $objectList property is not defined for the module, the custom object will be used, however, things like Studio will no longer work correctly.
    - For the example above, we defined the $objectList property for the Leads module to make sure that Studio continued working as expected. Without this definition, if you navigate to Admin > Studio, fields and relationships will not render properly.

Implementing Custom SugarBean Templates

Overview

This article covers how to implement custom SugarBean templates, which can then be extended by custom modules. It also covers vardef templates, which can more portable and used by multiple modules. By default, Sugar comes with a few templates such as Company, Person, Issue, and File templates.

Creating a Custom SugarBean Template

The following example will create a custom Bean template that can be used by custom modules to shrink down the overall size of the Bean model, by only implementing the two fields required by all modules, the id field and the deleted field. In order to do this, we will have to override some core SugarBean methods, as the base SugarBean is statically configured to do things like Save/Delete the model with the date_entered, date_modified, modified_user_id and created_user_id fields. In order to accommodate custom Beans that wish to use those fields, this template will also have properties for configuring those types of fields so that they are populated using the default SugarBean logic.

To create a template for use by custom modules, you create a class that extends from SugarBean
in ./custom/include/SugarObjects/templates/<template_name>/<class_name>.php. For this example, we will create the "bare" template with the following file:

./custom/include/SugarObjects/templates/bare/BareBean.php

```
  <?php

class BareBean extends SugarBean
{
    /**
     * The configured modified date field
     * @var string|false
     */
    protected $_modified_date_field = false;

    /**
     * The configured modified user field
     * @var string|false
     */
    protected $_modified_user_field = false;

    /**
     * The configured created date field
     * @var string|false
     */
    protected $_created_date_field = false;

    /**
     * The configured created user field
     * @var string|false
     */
    protected $_created_user_field = false;

    /**
     * Get the field name where modified date is stored, if in use by
Module
     * @return string|false
     */
    public function getModifiedDateField()
    {
        if ($this->_modified_date_field !== FALSE){
            $field = $this->_modified_date_field;
            if (isset($this->field_defs[$field]) && $this->field_defs[
$field]['type'] == 'datetime'){
```

```
                return $field;
            }
        }
        return FALSE;
    }

    /**
     * Override the stock setModifiedDate method
     *  - Check that field is in use by this Module
     *  - If in use, set the configured field to current time
     * @inheritdoc
     **/
    public function setModifiedDate($date = '')
    {
        global $timedate;

        $field = $this->getModifiedDateField();
        if ($field !== FALSE){
            // This code was duplicated from the stock SugarBean::setM
odifiedDate
            if ($this->update_date_modified || empty($this->$field)) {
                // This only needs to be calculated if it is going to
be used
                if (empty($date)) {
                    $date = $timedate->nowDb();
                }

                $this->$field = $date;
            }
        }
    }

    /**
     * Get the field name where modified user ID is stored, if in use
by Module
     * @return string|false
     **/
    public function getModifiedUserField()
    {
        if ($this->_modified_user_field !== FALSE){
            $field = $this->_modified_user_field;
            if (isset($this->field_defs[$field]) && $this->field_defs[
$field]['type'] == 'assigned_user_name'){
                return $field;
            }
        }
```

```
        return FALSE;
    }

    /**
     * Override the stock setModifiedUser Method
     * - Check that field is in use by this Module
     * - If in use, set the configured field to user_id
     * @inheritdoc
     * @param User|null $user [description]
     */
    public function setModifiedUser(User $user = null)
    {
        global $current_user;
        $field = $this->getModifiedUserField();
        if ($field !== FALSE){
            // If the update date modified by flag is set then carry o
ut this directive
            if ($this->update_modified_by) {
                // Default the modified user id to the default
                $this->$field = 1;

                // If a user was not presented, default to the current
 user
                if (empty($user)) {
                    $user = $current_user;
                }

                // If the user is set, use it
                if (!empty($user)) {
                    $this->$field = $user->id;
                }
            }
        }
    }

    /**
     * Get the field name where created date is stored, if in use by M
odule
     * @return string|false
     */
    public function getCreatedDateField()
    {
        if ($this->_created_date_field !== FALSE){
            $field = $this->_created_date_field;
            if (isset($this->field_defs[$field]) && $this->field_defs[
$field]['type'] == 'datetime'){
```

```
                return $field;
            }
        }
        return FALSE;
    }

    /**
     * Get the field name where created user ID is stored, if in use b
y Module
     * @return string|false
     */
    public function getCreatedUserField()
    {
        if ($this->_created_user_field !== FALSE){
            $field = $this->_created_user_field;
            if (isset($this->field_defs[$field]) && $this->field_defs[
$field]['type'] == 'assigned_user_name'){
                return $field;
            }
        }
        return FALSE;
    }

    /**
     * Override the stock setCreateData method
     * - Code was duplicated from stock, to accommodate not having cre
ated date or created user fields
     * @inheritdoc
     */
    public function setCreateData($isUpdate, User $user = null)
    {
        if (!$isUpdate) {
            global $current_user;

            $field = $this->getCreatedDateField();
            if ($field !== FALSE){
                //Duplicated from SugarBean::setCreateData with modifi
cations for dynamic field name
                if (empty($this->$field)) {
                    $this->$field = $this->getDateModified();
                }

                if (empty($this->$field)){
                    global $timedate;
                    $this->$field = $timedate->nowDb();
                }
```

```
            }

            $field = $this->getCreatedUserField();
            if ($field !== FALSE){
                //Duplicated from SugarBean::setCreateData with modifi
cations for dynamic field name
                if ($this->set_created_by == true) {
                    // created by should always be this user
                    // unless it was set outside of the bean
                    if ($user) {
                        $this->$field = $user->id;
                    } else {
                        $this->$field = isset($current_user) ? $curren
t_user->id : "";
                    }
                }
            }

            if ($this->new_with_id == false) {
                $this->id = Sugarcrm\Sugarcrm\Util\Uuid::uuid1();
            }
        }
    }

    /**
     * Get the Date Modified fields value
     * @return mixed|false
     */
    public function getDateModified()
    {
        if ($this->_modified_date_field !== FALSE){
            $field = $this->_modified_date_field;
            return $this->$field;
        }
        return FALSE;
    }

    /**
     * Get the Date Created Fields value
     * @return mixed|false
     */
    public function getDateCreated()
    {
        if ($this->_created_date_field !== FALSE){
            $field = $this->_created_date_field;
            return $this->$field;
```

```
        }
        return FALSE;
    }

    /**
     * @inheritdoc
     */
    public function mark_deleted($id)
    {
        $date_modified = $GLOBALS['timedate']->nowDb();
        if (isset($_SESSION['show_deleted'])) {
            $this->mark_undeleted($id);
        } else {
            // Ensure that Activity Messages do not occur in the conte
xt of a Delete action (e.g. unlink)
            // and do so for all nested calls within the Top Level Del
ete Context
            $opflag = static::enterOperation('delete');
            $aflag = Activity::isEnabled();
            Activity::disable();
            // call the custom business logic
            $custom_logic_arguments['id'] = $id;
            $this->call_custom_logic("before_delete", $custom_logic_ar
guments);
            $this->deleted = 1;

            if (isset($this->field_defs['team_id'])) {
                if (empty($this->teams)) {
                    $this->load_relationship('teams');
                }

                if (!empty($this->teams)) {
                    $this->teams->removeTeamSetModule();
                }
            }

            $this->mark_relationships_deleted($id);

            $updateFields = array('deleted' => 1);

            $field = $this->getModifiedDateField();
            if ($field !== FALSE){
                $this->setModifiedDate();
                $updateFields[$field] = $this->$field;
            }
```

```
            $field = $this->getModifiedUserField();
            if ($field !== FALSE){
                $this->setModifiedUser();
                $updateFields[$field] = $this->$field;
            }

            $this->db->updateParams(
                $this->table_name,
                $this->field_defs,
                $updateFields,
                array('id' => $id)
            );

            if ($this->isFavoritesEnabled()) {
                SugarFavorites::markRecordDeletedInFavorites($id, $dat
e_modified);
            }

            // Take the item off the recently viewed lists
            $tracker = BeanFactory::newBean('Trackers');
            $tracker->makeInvisibleForAll($id);

            SugarRelationship::resaveRelatedBeans();

            // call the custom business logic
            $this->call_custom_logic("after_delete", $custom_logic_arg
uments);
            if(static::leaveOperation('delete', $opflag) && $aflag) {
                Activity::enable();
            }
        }
    }

    /**
     * @inheritdoc
     */
    public function mark_undeleted($id)
    {
        // call the custom business logic
        $custom_logic_arguments['id'] = $id;
        $this->call_custom_logic("before_restore", $custom_logic_argum
ents);

        $this->deleted = 0;
        $modified_date_field = $this->getModifiedDateField();
        if ($modified_date_field !== FALSE){
```

```
            $this->setModifiedDate();
        }

        $query = "UPDATE {$this->table_name} SET deleted = ?".(!$modif
ied_date_field?"":",$modified_date_field = ?")." WHERE id = ?";
        $conn = $this->db->getConnection();
        $params = array($this->deleted);
        if ($modified_date_field){
            $params[] = $this->$modified_date_field;
        }
        $params[] = $id;
        $conn->executeQuery($query, $params);

        // call the custom business logic
        $this->call_custom_logic("after_restore", $custom_logic_argume
nts);
    }
}
```

With the Bean template in place, we can define the default vardefs for the template by creating the following file:

./custom/include/SugarObjects/templates/bare/vardefs.php

```
  <?php

$vardefs = array(
    'audited' => false,
    'favorites' => false,
    'activity_enabled' => false,
    'fields' => array(
        'id' => array(
            'name' => 'id',
            'vname' => 'LBL_ID',
            'type' => 'id',
            'required' => true,
            'reportable' => true,
            'duplicate_on_record_copy' => 'no',
            'comment' => 'Unique identifier',
            'mandatory_fetch' => true,
        ),
        'deleted' => array(
            'name' => 'deleted',
            'vname' => 'LBL_DELETED',
```

```
            'type' => 'bool',
            'default' => '0',
            'reportable' => false,
            'duplicate_on_record_copy' => 'no',
            'comment' => 'Record deletion indicator'
        )
    ),
    'indices' => array(
        'id' => array(
            'name' => 'idx_' . preg_replace('/[^a-
z0-9_\-]/i', '', strtolower($module)) . '_pk',
            'type' => 'primary',
            'fields' => array('id')
        ),
        'deleted' => array(
            'name' => 'idx_' . strtolower($table_name) . '_id_del',
            'type' => 'index',
            'fields' => array('id', 'deleted')
        )
    ),
    'duplicate_check' => array(
        'enabled' => false
    )
);
```

Note: This vardef template also shrinks the SugarBean template down even further, by defaulting auditing, favorites, and activity stream to false.

Using a Custom Bean Template

With the custom SugarBean template in place, we can now implement the template on a custom module. Typically if you deployed a module from Module Builder, there would be two classes generated by Sugar, "custom_Module_sugar" and "custom_Module". The "_sugar" generated class extends from the Bean template that was selected in Module Builder, and the primary Bean class is what is utilized in the system and where your development would take place. For the above example template that is created, the assumption is that you are writing the Bean class, rather than using the generated classes by Module Builder, or at the very least removing the intermediary "_sugar" class generated by Module Builder and extending from the desired template. If you deployed a module via Module Builder, and are going to switch to a custom template, please note that some of the stock templates contain internal logic for the setup of fields for that template and that would need to be duplicated if you intend to continue using those fields. The stock templates, located in ./include/SugarObjects/templates/, are available for your

reference to copy over any of the required logic to your custom Bean class. With all that being said, let us take a look at a custom Bean class that extends from our custom template.

This example will use the "custom_Module" module which is a module that will be used for Tagging records. Since the module is just storing tags, a slimmed down Bean works well as we only need a "name" field to store those tags. The following class would implement the custom Bean template created above.

./modules/custom_Module/custom_Module.php

```
  <?php

require_once 'custom/include/SugarObjects/templates/bare/BareBean.php'
;

class custom_Module extends BareBean {

    public $new_schema = true;
    public $module_dir = 'custom_Module';
    public $object_name = 'custom_Module';
    public $table_name = 'custom_Module';
    public $importable = true;

    public $id;
    public $name;
    public $deleted;

    public function __construct(){
        parent::__construct();
    }

    public function bean_implements($interface){
        switch($interface){
            case 'ACL': return true;
        }
        return false;
    }

}
```

With the modules SugarBean class created, the other thing that needs to be implemented is the vardefs.php file:

./modules/custom_Module/vardefs.php

```
  <?php

$module = 'custom_Module';
$table_name = strtolower($module);

$dictionary[$module] = array(
    'table' => $table_name,
    'fields' => array(
        'name' => array(
            'name' => 'name',
            'vname' => 'LBL_NAME',
            'type' => 'username',
            'link' => true,
            'dbType' => 'varchar',
            'len' => 255,
            'unified_search' => true,
            'full_text_search' => array(),
            'required' => true,
            'importable' => 'required',
            'duplicate_merge' => 'enabled',
            //'duplicate_merge_dom_value' => '3',
            'merge_filter' => 'selected',
            'duplicate_on_record_copy' => 'always',
        ),
    ),
    'relationships' => array (
    ),
    'optimistic_locking' => false,
    'unified_search' => false,
);

VardefManager::createVardef(
    $module,
    $module,
    //Specify the bare template to be used to create the vardefs
    array('bare')
);
```

With these files implemented, the "custom_Module" module would implement the "bare" template we created.

Creating Custom Vardef Templates

For some customizations, you might not need a SugarBean template as you may not be implementing logic that needs to be shared across multiple module's Bean classes. However, you may have field definitions that are common across multiple modules that would be beneficial for implementing as Vardef Templates. To create a vardef template, a file as follows, ./custom/include/SugarObjects/implements/<template_name>/vardefs.php.

Continuing on with our example custom_Module module from above, we might want to have a creation date on this custom tags module since our 'bare' SugarBean template does not come with one by default. We could easily just add one in the modules vardef file, but for our example purposes, we know that we will use our 'bare' SugarBean template on other customizations, and on some of those we might also want to include a creation date. To implement the vardef template for the creation date field, we create the following:

./custom/include/SugarObjects/implements/date_entered/vardefs.php

```
  <?php

$vardefs = array(
    'fields' => array(
        'date_entered' => array(
            'name' => 'date_entered',
            'vname' => 'LBL_DATE_ENTERED',
            'type' => 'datetime',
            'group' => 'created_by_name',
            'comment' => 'Date record created',
            'enable_range_search' => true,
            'options' => 'date_range_search_dom',
            'studio' => array(
                'portaleditview' => false,
            ),
            'duplicate_on_record_copy' => 'no',
            'readonly' => true,
            'massupdate' => false,
            'full_text_search' => array(
                'enabled' => true,
                'searchable' => false
            ),
        )
    ),
    'indices' => array(
        'date_entered' => array(
```

```
            'name' => 'idx_' . strtolower($table_name) . '_date_entere
d',
            'type' => 'index',
            'fields' => array('date_entered')
        )
    )
);
```

Using a Custom Vardef Template

Once the vardef template is in place, you can use the template by adding it to the 'uses' array property of your module vardefs. Continuing with our example module custom_Module, we can update the vardefs file as follows:

./modules/custom_Module/vardefs.php

```
  <?php

$module = 'custom_Module';
$table_name = strtolower($module);

$dictionary[$module] = array(
    'table' => $table_name,
    'fields' => array(
        'name' => array(
            'name' => 'name',
            'vname' => 'LBL_NAME',
            'type' => 'username',
            'link' => true,
            'dbType' => 'varchar',
            'len' => 255,
            'unified_search' => true,
            'full_text_search' => array(),
            'required' => true,
            'importable' => 'required',
            'duplicate_merge' => 'enabled',
            //'duplicate_merge_dom_value' => '3',
            'merge_filter' => 'selected',
            'duplicate_on_record_copy' => 'always',
        ),
    ),
    'relationships' => array (
    ),
    //Add the desired vardef templates to this list
```

```
    'uses' => array(
        'date_entered'
    ),
    'optimistic_locking' => false,
    'unified_search' => false,
);

VardefManager::createVardef(
    $module,
    $module,
    //Specify the bare template to be used to create the vardefs
    array('bare')
);
```

After a Quick Repair and Rebuild, a SQL statement should be generated to update the table of the module with the new date_entered field that was added to the vardefs using the vardef template.

BeanFactory

Overview

The BeanFactory class, located in ./data/BeanFactory.php, is used for loading an instance of a SugarBean. This class should be used any time you are creating or retrieving bean objects. It will automatically handle any classes required for the bean.

Creating a SugarBean Object

newBean()

To create a new, empty SugarBean, use the newBean() method. This method is typically used when creating a new record for a module or to call properties of the module's bean object.

```
  $bean = BeanFactory::newBean($module);
```

newBeanByName()

Used to fetch a bean by its beanList name.

```
$bean = BeanFactory::newBeanByName($name);
```

Retrieving a SugarBean Object

getBean()

The getBean() method can be used to retrieve a specific record from the database. If a record id is not passed, a new bean object will be created.

```
$bean = BeanFactory::getBean($module, $record_id);
```

Note: Disabling row-level security when accessing a bean should be set to true only when it is absolutely necessary to bypass security, for example when updating a Lead record from a custom Entry Point. An example of accessing the bean while bypassing row security is:

```
$bean = BeanFactory::getBean($module, $record_id, array('disable_row
_level_security' => true));
```

retrieveBean()

The retrieveBean() method can also be used to retrieve a specific record from the database. The difference between this method and getBean() is that null will be returned instead of an empty bean object if the retrieve fails.

```
$bean = BeanFactory::retrieveBean($module, $record_id);
```

Note: Disabling row-level security when accessing a bean should be set to true only when it is absolutely necessary to bypass security, for example, when updating a Lead record from a custom Entry Point. An example of accessing the bean while bypassing row security is:

```
$bean = BeanFactory::retrieveBean($module, $record_id, array('disabl
e_row_level_security' => true));
```

Retrieving Module Keys

getObjectName()

The getObjectName() method will return the object name / dictionary key for a given module. This is normally the same as the bean name, but may not be for some modules such as Cases which has a key of 'aCase' and a name of 'Case'.

```
$moduleKey = BeanFactory::getObjectName($moduleName);
```

getBeanName()

The getBeanName() method will retrieve the bean class name given a module name.

```
$moduleClass = BeanFactory::getBeanName($module);
```

Vardefs

Overview

Vardefs (Variable Definitions) provide the Sugar application with information about SugarBeans. Vardefs specify information on the individual fields, relationships between beans, and the indexes for a given bean.

Each module that contains a SugarBean file has a vardefs.php file located in it, which specifies the fields for that SugarBean. For example, the vardefs for the Contact bean are located in ./modules/Contacts/vardefs.php.

Dictionary Array

Vardef files create an array called $dictionary, which contains several entries including tables, fields, indices, and relationships.

- **table** : The name of the database table (usually, the name of the module) for this bean that contains the fields
- **audited** : Specifies whether the module has field-level auditing enabled
- **duplicate_check** : Determines if duplicate checking is enabled on the module, and what duplicate check strategy will be used if enabled.
- **fields** : A list of fields and their attributes

- **indices** : A list of indexes that should be created in the database
- **optimistic_locking** : Determines whether the module has optimistic locking enabled
    - Optimistic locking prevents loss of data by using the bean's modified date to ensure that it is not being modified simultaneously by more than one person or process.
- **unified_search** : Determines whether the module can be searched via Global Search
    - This setting defaults to false and has no effect if all of the fields in the fields array have the 'unified_search' field attribute set to false.
- **unified_search_default_enabled** : Determines whether the module should be searched by default for new users via Global Search
    - This setting defaults to true but has no effect if unified_search is set to false.
- **visibility** : A list of visibility classes enabled on the module

Duplicate Check Array

The duplicate_check array contains two properties, that control if duplicate checking is enabled on the module, and which duplicate check strategy will be used to check for duplicates.

The two properties for the array are as follows:

Name	Type	Description
enabled	Boolean	Specifies whether or not the Bean is utilizing the duplicate check framework
<class_name>	Array	<class_name> is the name of the duplicate check strategy class that is handling the duplicate checking. It is set to an array of Metadata, specific to the strategy defined in the key. Review the Duplicate Check Framework for further information.

Fields Array

The fields array contains one array for each field in the SugarBean. At the top level

of this array, the key is the name of the field, and the value is an array of attributes about that field.

The list of possible attributes are as follows:

- **name** : The name of the field
- **vname** : The language pack ID for the label of this field
- **type** : The type of the attribute
    - **assigned_user_name** : A linked user name
    - **bool** : A boolean value
    - **char** : A character array
    - **date** : A date value with no time
    - **datetime** : A date and time
    - **email** : An email address field
    - **enum** : An enumeration (dropdown list from the language pack)
    - **id :** A database GUID
    - **image** : A photo-type field
    - **link** : A link through an explicit relationship. This attribute should only be used when working with related fields. It does not make the field render as a link.
    - **name** : A non-database field type that concatenates other field values
    - **phone** : A phone number field to utilize with callto:// links
    - **relate** : Related bean
    - **team_list** : A team-based ID
    - **text :** A text area field
    - **url** : A hyperlinked field on the detail view
    - **varchar** : A variable-sized string
- **table** : The database table the field comes from.
    - The table attribute is only needed to join fields from another table outside of the module in focus.
- **isnull** : Whether the field can contain a null value
- **len** : The length of the field (number of characters if a string)
- **options** : The name of the enumeration (dropdown list) in the language pack for the field
- **dbtype** : The database type of the field (if different than the type)
- **reportable** : Determines whether the field will be available in the Reports and Workflow modules
- **default :** The default value for this field. Default values for the record are populated by default on create for the record view (for Sidecar modules) and edit view (for Legacy modules) layout but can be modified by users. The Default Value option is available for all data type fields except HTML, Image, Flex Relate, and Relate.
- **massupdate** : Determines whether the field will show up in the mass-update panel on its module's list view
    - Some field types are restricted from mass update regardless of this

setting.

- **rname** : For relate-type fields, the field from the related variable that contains the text
- **id_name** : For relate-type fields, the field from the bean that stores the ID for the related bean
- **source** : Set to 'non-db' if the field value does not come from the database
    - The source attribute can be used for calculated values or values retrieved in some other way.
- **sort_on** : The field to sort by if multiple fields are used in the presentation of field's information
- **fields** : For concatenated values, an array containing the fields that are concatenated
- **db_concat_fields** : For concatenated values, an array containing the fields to concatenate in the database
- **unified_search** : Determines whether the field can be searched via Global Search
    - This has no effect if the dictionary array setting 'unified_search' is not set to true.
- **enable_range_search** : For date, datetime, and numeric fields, determines if this field should be searchable by range in module searches
- **dependency** : Allows a field to have a predefined formula to control the field's visibility
- **studio** : Controls the visibility of the field in the Studio editor
    - If set to false, then the field will not appear in any studio screens for the module. Otherwise, you may specify an Array of view keys from which the field's visibility should be removed (e.g. array('listview'=>false) will hide the field in the listview layout screen).

The following example illustrates a standard ID field for a bean:

```
  'id' => array (
    'name' => 'id',
    'vname' => 'LBL_ID',
    'type' => 'id',
    'required' => true,
),
```

Indices Array

This array contains a list of arrays that are used to create indices in the database. The fields in this array are:

- **name** : The unique name of the index
- **type** : The type of index (primary, unique, or index)
- **fields** : An ordered array of the fields to index
- **source** : Set to 'non-db' if you are creating an index for added application functionality such as duplication checking on imports

The following example creates a primary index called 'userspk' on the 'id' column:

```
  array(
    'name' => 'userspk',
    'type' => 'primary',
    'fields'=> array('id')
),
```

Relationships Array

The relationships array specifies relationships between beans. Like the indices array entries, it is a list of names with array values.

- **lhs_module** : The module on the left-hand side of the relationship
- **lhs_table** : The table on the left-hand side of the relationship
- **lhs_key** : The primary key column of the left-hand side of the relationship
- **rhs_module** : The module on the right-hand side of the relationship
- **rhs_table** : The table on the right-hand side of the relationship
- **rhs_key** : The primary key column of the right-hand side of the relationship
- **relationship_type** : The type of relationship (e.g. one-to-many, many-to-many)
- **relationship_role_column** : The type of relationship role
- **relationship_role_column_value** : Defines the unique identifier for the relationship role

The following example creates a relationship between a contact and the contact that they report to. The reports_to_id field maps to the id of the record that belongs to the higher-ranked contact. This is a one-to-many relationship in that a contact may only report to one person, but many people may report to the same contact.

```
  'contact_direct_reports' => array(
    'lhs_module' => 'Contacts',
    'lhs_table' => 'contacts',
    'lhs_key' => 'id',
    'rhs_module' => 'Contacts',
```

```
    'rhs_table' => 'contacts',
    'rhs_key' => 'reports_to_id',
    'relationship_type' => 'one-to-many'
),
```

Visibility Array

The visibility array specifies the row level visibility classes that are enabled on the bean. Each entry in the array, is a key-value pair, where the key is the name of the Visibility class and the value is set to boolean True. More information on configuring custom Visibility strategies can be found in the Architecture section under Visibility Framework.

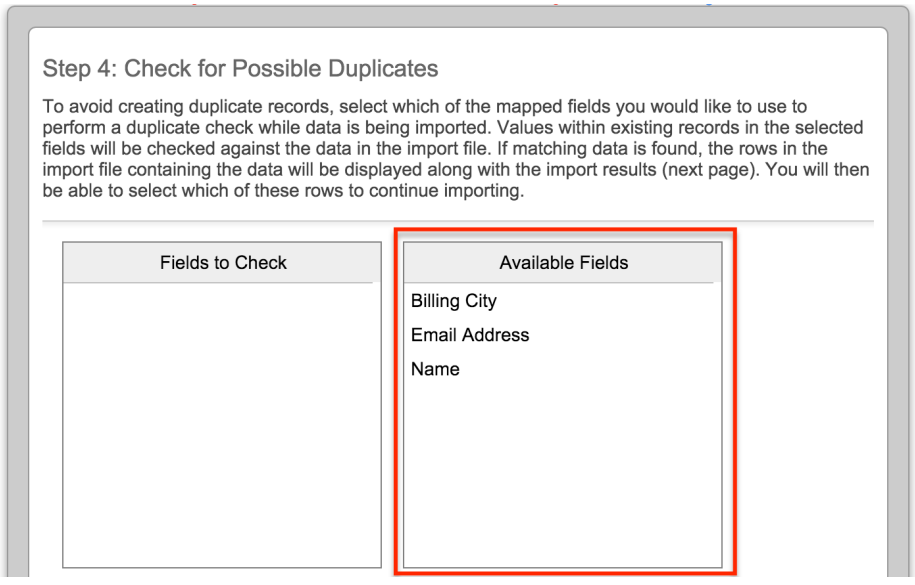
Extending Vardefs

More information about extending and overriding vardefs can be found in the Extensions Framework documentation under Vardefs.

Specifying Custom Indexes for Import Duplicate Checking

Overview

When importing records to Sugar via the Import Wizard, users can select which of the mapped fields they would like to use to perform a duplicate check and thereby avoid creating duplicate records. This article explains how to enable an additional field or set of fields for selection in this step.



Resolution

The import wizard's duplicate check operates based on indices defined for that module. You can create a non-database index to check for a field. It is important that it is non-database as single column indices on your database can hamper overall performance. The following is an example to add the home phone field to the Contact module's duplicate check.

First, create the following file from the root directory of your Sugar installation on the web server:

```
./custom/Extension/modules/Contacts/Ext/Vardefs/custom_import_index.ph
p
```

When creating the file, keep in mind the following requirements:

- The name of the file is not important, as long as it ends with a .php extension.
- The rest of the directory path is case sensitive so be sure to create the directories as shown.
- If you are creating the import index for a module other than Contacts, then substitute the corresponding directory name with that module.
- Ensure that the entire directory path and file have the correct ownership and sufficient permissions for the web server to access the file.

The contents of the file should look similar to the following code:

```
  <?php

$dictionary['Contact']['indices'][] = array(
     'name' => 'idx_home_phone_cstm',
     'type' => 'index',
     'fields' => array(
         0 => 'phone_home',
     ),
     'source' => 'non-db',
);
```

Please note that the module name in line 2 of the code is singular (i.e. Contact, not Contacts). If you are unsure of what to enter for the module name, you can verify the name by opening the

./cache/modules/<module_name>/<module_name>vardefs.php file. The second line of that file will have text like the following:

```
$GLOBALS["dictionary"]["Contact"] = array (
```

The parameter following "dictionary" is the same parameter you should use in the file defining the custom index. To verify duplicates against a combination of fields (i.e. duplicates will only be flagged if the values of multiple fields match those of an existing record), then simply add the desired fields to the 'fields' array in the code example.

Finally, navigate to Admin > Repair > Quick Repair and Rebuild to enable the custom index for duplicate verification when importing records in the module.

Working With Indexes

Overview

Sugar provides a simple method for creating custom indexes through the vardef framework. Indexes can be built on one or more fields within a module. Indexes can be saved down to the database level or made available only in the application for functions such as Import Duplicate Checking.

Index Metadata

Indexes have the following metadata options that can be configured per index:

Key	Value	Description
name	string	A Unique identifier to define the index. Best practices recommend indexes start with idx and contain the suffix cstm to avoid conflicting with a stock index. **Note** : Some databases have restrictions on the length of index names. Please check with your database vendor to avoid

		any issues.
type	string	All indexes should use the type of "index"
fields	array	A PHP array of the fields for the index to utilize
source	string	Specify as "non-db" to avoid creating the index in the database

Creating Indexes

Stock indexes are initially defined in the module's vardefs file under the indices array. For reference, you can find them using the vardef path of your module. The path will be ./modules/<module>/vardefs.php.

Custom indexes should be created using the Extension Framework. First, create a PHP file in the extension directory of your desired module. The path should similar to ./custom/Extension/modules/<module>/Ext/Vardefs/<name>.php.

In the new file, add the appropriate $dictionary reference to define the custom index:

```
  <?php

$dictionary['<module>']['indices'][] = array(
 'name' => '<index name>',
 'type' => 'index',
 'fields' => array(
  'field1',
  'field2',
 )
);
```

Note : For performance reasons, it is not recommended to create an index on a single field unless the source is set to non-db.

Once installed,you will need to navigate to Admin > Repair > Quick Repair and Rebuild to enable the custom index. You will need to execute any scripts generated by the rebuild process.

Removing Indexes

Stock indexes are initially defined in the module's vardefs file under the indices array. For reference, you can find them using the vardef path of your module. The path will be ./modules/<module>/vardefs.php.

Stock indexes should be removed using the Extension Framework. First, create a PHP file in the extension directory of your desired module. The path should similar to ./custom/Extension/modules/<module>/Ext/Vardefs/<name>.php.

In the new file, loop through the existing 'indices' sub-array of the $dictionary to locate the stock index to remove, and use unset() to remove it from the array.

Example

The following is an example to remove the idx_calls_date_start index from the Call module's vardefs.

First, create ./custom/Extension/modules/Calls/Ext/Vardefs/remove_idx_calls_date_start.php from the root directory of your Sugar installation on the web server. When creating the file, keep in mind the following requirements:

- The name of the file is not important, as long as it ends with a .php extension.
- The rest of the directory path is case sensitive so be sure to create the directories as shown.
- If you are removing the index for a module other than Calls, then substitute the corresponding directory name with that module.
- Ensure that the entire directory path and file have the correct ownership and sufficient permissions for the web server to access the file.

The contents of the file should look similar to the following code:

```
  <?php

$call_indexes = $dictionary['Call']['indices'];
$remove_index = "idx_calls_date_start";

foreach($call_indexes as $index_key => $index_item) {
    if( $index_item['name'] == $remove_index ) {
        unset($dictionary['Call']['indices'][$index_key]);
    }
}
```

Note : Removing the reference to the index from the module's indices array does not actually remove the index from the module's database table. Removing the

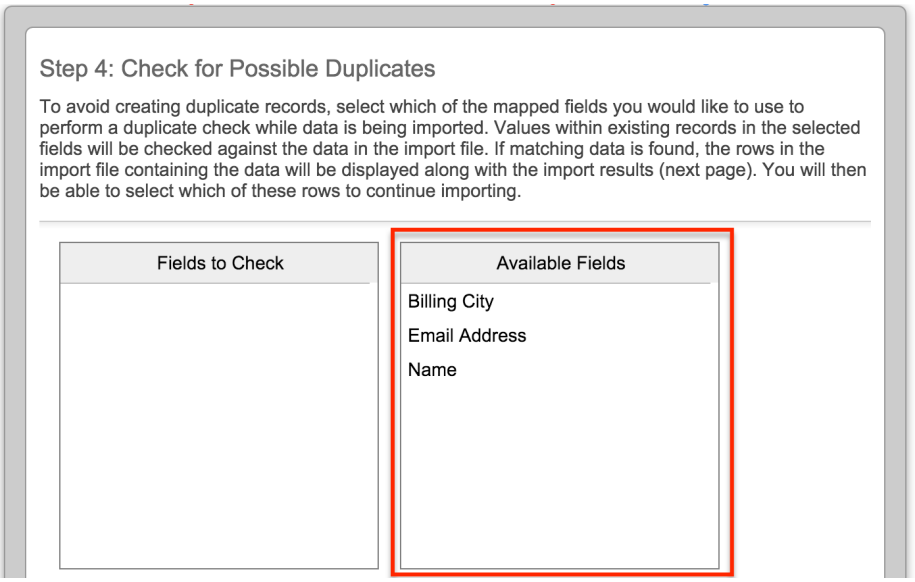
reference from the indices array ensures that the index is not added back to the module's database table when performing any future Quick Repair and Rebuilds. The database index must be removed directly at the database level. On MySQL, with the current example, this could be done with a query like:

```
ALTER TABLE calls DROP INDEX idx_calls_date_start;
```

Once installed,you will need to navigate to Admin > Repair > Quick Repair and Rebuild to remove the index from the $dictionary array. You will need to execute any scripts generated by the rebuilding process.

Creating Indexes for Import Duplicate Checking

When importing records to Sugar via the Import Wizard, users can select which of the mapped fields they would like to use to perform a duplicate check and thereby avoid creating duplicate records. The following instructions explain how to enable an additional field or set of fields for selection in this step.



Example

The following is an example to add the home phone field to the Contact module's duplicate check.

First, create ./custom/Extension/modules/Contacts/Ext/Vardefs/custom_import_index.php from the root directory of your Sugar installation on the web server. When creating the file, keep in mind the following requirements:

- The name of the file is not important, as long as it ends with a .php extension.
- The rest of the directory path is case sensitive so be sure to create the directories as shown.
- If you are creating the import index for a module other than Contacts, then substitute the corresponding directory name with that module.
- Ensure that the entire directory path and file have the correct ownership and sufficient permissions for the web server to access the file.

The contents of the file should look similar to the following code:

```
  <?php

$dictionary['Contact']['indices'][] = array(
     'name' => 'idx_home_phone_cstm',
     'type' => 'index',
     'fields' => array(
         0 => 'phone_home',
     ),
     'source' => 'non-db',
);
```

Please note that the module name in line 2 of the code is singular (i.e. Contact, not Contacts). If you are unsure of what to enter for the module name, you can verify the name by opening the ./cache/modules/<module_name>/<module_name>vardefs.php file. The second line of that file will have text like the following:

```
  $GLOBALS["dictionary"]["Contact"] = array (...);
```

The parameter following "dictionary" is the same parameter you should use in the file defining the custom index. To verify duplicates against a combination of fields (i.e. duplicates will only be flagged if the values of multiple fields match those of an existing record), then simply add the desired fields to the 'fields' array in the code example.

Finally, navigate to Admin > Repair > Quick Repair and Rebuild to enable the custom index for duplicate verification when importing records in the module.

Fields

Overview

How fields interact with the various aspects of Sugar.

SugarField Widgets

The SugarField widgets, located in ./include/SugarFields/Fields/ , define the data formatting and search query structure for the various field types. They also define the rendering of fields for modules running in backward compatibility mode. When creating or overriding field widgets, developers should place their customization in ./custom/include/SugarFields/Fields/. For information on how Sidecar renders fields, please refer to the fields section in our user interface documentation. Creating Custom Fields

Implementation

All fields for a module are defined within vardefs. Within this definition, the type attribute will determine all of the logic applied to the field. For example, the Contacts module has a 'Do Not Call' field. In the vardefs, this field is defined as follows:

```
'do_not_call' => array (
    'name' => 'do_not_call', // the name of the field
    'vname' => 'LBL_DO_NOT_CALL', // the label for the field name
    'type' => 'bool', // the fields type
    'default' => '0', // the fields default value
    'audited'=>true, // whether the field is audited
    'duplicate_on_record_copy' => 'always', // whether to duplicate th
e fields value when being copied
    'comment' => 'An indicator of whether contact can be called' // ad
min context of the field
),
```

The bool type field is rendered in the UI from the ./clients/base/fields/bool/bool.js field controller which renders the appropriate handlebars template as defined by the users current view for sidecar enabled modules. When the user saves data, the controller formats the data for the API and passes it to an endpoint. Once the data is received by the server, The SugarField definition calls any additional logic in the apiSave function to format the data for saving to the database. The same concept is applied in the apiFormatField function when retrieving data from the database to be passed back to the user interface through the API. For modules running in backward compatibility mode, the bool field is rendered using the Smarty .tpl) templates located in ./include/SugarFields/Fields/Bool/.

While the vardefs define the default type for a field, this value can be overridden in the metadata of the view rendering the field. The example being that in ./custom/modules/Contacts/clients/base/views/record/record.php, you can modify the do_not_call field array to point to a custom field type you have created. For more information on creating custom field types, please refer to Creating Custom Fields documentation.

Relationships

Overview

Relationships are the basis for linking information within the system. This page explains the various aspects of relationships. For information on custom relationships, please refer to the Custom Relationships documentation.

Definitions

Relationships are initially defined in the module's vardefs file under the relationships array. For reference, you can find them using the vardef path ./modules/<module>/vardefs.php.

Database Structure

In Sugar, most relationships are stored using a joining table. This applies to both one-to-many (1:M) relationships as well as many-to-many (M:M) relationships. An example of this is the relationship between Accounts and Opportunities where there are three tables: accounts, accounts_opportunities, and opportunities. You will find that the joining table, accounts_opportunities, will contain the fields needed in order to establish the relationship link.

The fields on the accounts_opportunities table are listed below:

Fields	Description
id	A unique identifier for the relationship row (not typically used)
opportunity_id	The ID for the related opportunity record. This is named uniquely based on the relationship
account_id	The ID for the related account record.

	This is named uniquely based on the relationship
date_modified	The date the row was last modified
deleted	Whether or not the relationship still exists

Relationship Cache

All relationships in Sugar are compiled into the cache directory ./cache/Relationships/relationships.cache.php. If needed, the relationships cache can be rebuilt by navigating to Admin > Repair > Rebuild Relationships.

Custom Relationships

Overview

This page needs an overview

Creating Custom Relationships

Relationships are initially defined in the module's vardefs file under the relationships array. For reference, you can find them using the vardef path as follows:

```
./modules/<module>/vardefs.php
```

Custom relationships are created in a different way using the Extension Framework. The process requires two steps which are explained in the following sections:

1. Defining the Relationship MetaData
2. Defining the Relationship in the TableDictionary

Defining the Relationship MetaData

The definitions for custom relationships will be found in a path similar to:

```
./custom/metadata/<relatonship name>MetaData.php
```

This file will contain the $dictionary information needed for the system to generate the relationship. The $dictionary array will contain the following:

Index	Type	Description
true_relationship_type	String	The relationship's structure (possible values: 'one-to-many' or 'many-to-many')
from_studio	Boolean	Whether the relationship was created in Studio
table	String	The name of the table that is created in the database to contain the link ids
fields	Array	An array of fields to be created on the relationship join table
indices	Array	The list of indexes to be created
relationships	Array	An array defining relationships
relationships.<rel>	Array	The array defining the relationship
relationships.<rel>.lhs_module	String	Left-hand module (should match $beanList index)
relationships.<rel>.lhs_table	String	Left-hand table name
relationships.<rel>.lhs_key	String	The key to use from the table on the left
relationships.<rel>.rhs_module	String	Right-hand module (should match $beanList index)
relationships.<rel>.rhs_table	String	Right-hand table name
relationships.<rel>.rhs_key	String	The key to use from the table on the right
relationships.<rel>.relationship_type	String	The relationship type, typically stored as 'many-to-many'
relationships.<rel>.join_table	String	The join table

relationships.<rel>.join_key_lhs	String	Left table key, should exist in table field definitions above
relationships.<rel>.join_key_rhs	String	Right table key, should exist in table field definitions above

MetaData Example

Creating a custom 1:M relationship between Accounts and Contacts will yield the following metadata file:

./custom/metadata/accounts_contacts_1MetaData.php

```
<?php

// created: 2013-09-20 15:15:47

$dictionary["accounts_contacts_1"] = array (
  'true_relationship_type' => 'one-to-many',
  'from_studio' => true,
  'relationships' =>
  array (
    'accounts_contacts_1' =>
    array (
      'lhs_module' => 'Accounts',
      'lhs_table' => 'accounts',
      'lhs_key' => 'id',
      'rhs_module' => 'Contacts',
      'rhs_table' => 'contacts',
      'rhs_key' => 'id',
      'relationship_type' => 'many-to-many',
      'join_table' => 'accounts_contacts_1_c',
      'join_key_lhs' => 'accounts_contacts_1accounts_ida',
      'join_key_rhs' => 'accounts_contacts_1contacts_idb',
    ),
  ),
  'table' => 'accounts_contacts_1_c',
  'fields' =>
  array (
    0 =>
    array (
      'name' => 'id',
      'type' => 'varchar',
      'len' => 36,
    ),
```

```
  1 =>
  array (
    'name' => 'date_modified',
    'type' => 'datetime',
  ),
  2 =>
  array (
    'name' => 'deleted',
    'type' => 'bool',
    'len' => '1',
    'default' => '0',
    'required' => true,
  ),
  3 =>
  array (
    'name' => 'accounts_contacts_1accounts_ida',
    'type' => 'varchar',
    'len' => 36,
  ),
  4 =>
  array (
    'name' => 'accounts_contacts_1contacts_idb',
    'type' => 'varchar',
    'len' => 36,
  ),
),
'indices' =>
array (
  0 =>
  array (
    'name' => 'accounts_contacts_1spk',
    'type' => 'primary',
    'fields' =>
    array (
      0 => 'id',
    ),
  ),
  1 =>
  array (
    'name' => 'accounts_contacts_1_ida1',
    'type' => 'index',
    'fields' =>
    array (
      0 => 'accounts_contacts_1accounts_ida',
    ),
  ),
```

```
    2 =>
    array (
      'name' => 'accounts_contacts_1_alt',
      'type' => 'alternate_key',
      'fields' =>
      array (
        0 => 'accounts_contacts_1contacts_idb',
      ),
    ),
  ),
);
```

Defining the Relationship in the TableDictionary

Once a relationship's metadata has been created, the metadata file will have a reference placed in the TableDictionary:

```
./custom/Extension/application/Ext/TableDictionary/<relationship name>
.php
```

This file will contain an 'include' reference to the metadata file:

```
<?php

    include('custom/metadata/<relationship name>MetaData.php');

?>
```

TableDictionary Example

The custom 1:M relationship between Accounts and Contacts will yield the following TableDictionary file:

./custom/Extension/application/Ext/TableDictionary/accounts_contacts_1.php

```
<?php

    //WARNING: The contents of this file are auto-generated

    include('custom/metadata/accounts_contacts_1MetaData.php');
```

?>

If you have created the relationship through Studio, the files above will be automatically created. If you are manually creating the files, run a Quick Repair and Rebuild and run any SQL scripts generated. The Quick Repair and Rebuild will rebuild the file map and relationship cache as well as populate the relationship in the relationships table.

Subpanels

Overview

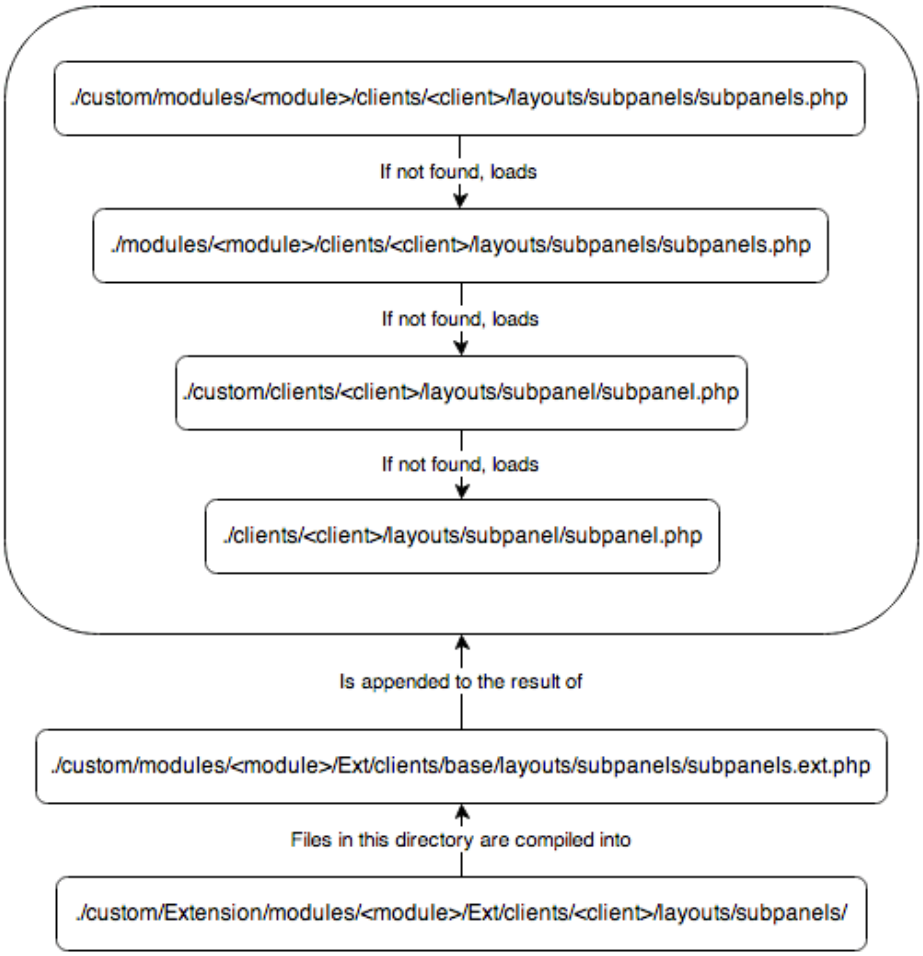
For Sidecar, Sugar's subpanel layouts have been modified to work as simplified metadata. This page is an overview of the metadata framework for subpanels.

The reason for this change is that previous versions of Sugar generated the metadata from various sources such as the SubPanelLayout and MetaDataManager classes. This eliminates the need for generating and processing the layouts and allows the metadata to be easily loaded to Sidecar.

Note: Modules running in backward compatibility mode do not use the Sidecar subpanel layouts as they use the legacy MVC framework.

Hierarchy Diagram

When loading the Sidecar subpanel layouts, the system processes the layout in the following manner:



Note: The Sugar application's client type is "base". For more information on the various client types, please refer to the User Interface page.

Subpanels and Subpanel Layouts

Sugar contains both a subpanels (plural) layout and a subpanel (singular) layout. The subpanels layout contains the collection of subpanels, whereas the subpanel layout renders the actual subpanel widget.

An example of a stock module's subpanels layout is:

./modules/Bugs/clients/base/layouts/subpanels/subpanels.php

```
  <?php

$viewdefs['Bugs']['base']['layout']['subpanels'] = array (
  'components' => array (
      array (
          'layout' => 'subpanel',
```

```
            'label' => 'LBL_DOCUMENTS_SUBPANEL_TITLE',
            'context' => array (
                'link' => 'documents',
            ),
        ),
        array (
            'layout' => 'subpanel',
            'label' => 'LBL_CONTACTS_SUBPANEL_TITLE',
            'context' => array (
                'link' => 'contacts',
            ),
        ),
        array (
            'layout' => 'subpanel',
            'label' => 'LBL_ACCOUNTS_SUBPANEL_TITLE',
            'context' => array (
                'link' => 'accounts',
            ),
        ),
        array (
            'layout' => 'subpanel',
            'label' => 'LBL_CASES_SUBPANEL_TITLE',
            'context' => array (
                'link' => 'cases',
            ),
        ),
    ),
    'type' => 'subpanels',
    'span' => 12,
);
```

You can see that the layout incorporates the use of the subpanel layout for each module. As most of the subpanel data is similar, this approach allows us to use less duplicate code. The subpanel layout, shown below, shows the three views that make up the subpanel widgets users see.

./clients/base/layouts/subpanel/subpanel.php

```
  <?php

$viewdefs['base']['layout']['subpanel']  = array (
    'components' => array (
        array (
            'view' => 'panel-top',
```

```
        )
        array (
            'view' => 'subpanel-list',
        ),
        array (
            'view' => 'list-bottom',
        ),
    ),
    'span' => 12,
    'last_state' => array(
        'id' => 'subpanel'
    ),
);
```

Adding Subpanel Layouts

When a new relationship is deployed from Studio, the relationship creation process will generate the layouts using the extension framework. You should note that for stock relationships and custom deployed relationships, layouts are generated for both Sidecar and Legacy MVC Subpanel formats. This is done to ensure that any related modules, whether in Sidecar or Backward Compatibility mode, display a related subpanel as expected.

Sidecar Layouts

Custom Sidecar layouts, located in ./custom/Extension/modules/<module>/Ext/clients/<client>/layouts/subpanels/, are compiled into ./custom/modules/<module>/Ext/clients/<client>/layouts/subpanels/subpanels.ext.php using the extension framework. When a relationship is saved, layout files are created for both the "base" and "mobile" client types.

For example, deploying a 1:M relationship from Bugs to Leads will generate the following Sidecar files:

./custom/Extension/modules/Bugs/Ext/clients/base/layouts/subpanels/bugs_leads_1_Bugs.php

```
  <?php

$viewdefs['Bugs']['base']['layout']['subpanels']['components'][] = array (
  'layout' => 'subpanel',
  'label' => 'LBL_BUGS_LEADS_1_FROM_LEADS_TITLE',
```

```
  'context' =>
  array (
    'link' => 'bugs_leads_1',
  ),
);
```

./custom/Extension/modules/Bugs/Ext/clients/mobile/layouts/subpanels/bugs_leads_1_Bugs.php

```
  <?php

$viewdefs['Bugs']['mobile']['layout']['subpanels']['components'][] = a
rray (
  'layout' => 'subpanel',
  'label' => 'LBL_BUGS_LEADS_1_FROM_LEADS_TITLE',
  'context' =>
  array (
    'link' => 'bugs_leads_1',
  ),
);
```

Note: The additional legacy MVC layouts generated by a relationships deployment are described below.

Legacy MVC Subpanel Layouts

Custom Legacy MVC Subpanel layouts, located in ./custom/Extension/modules/<module>/Ext/Layoutdefs/, are compiled into ./custom/modules/<module>/Ext/Layoutdefs/layoutdefs.ext.php using the extension framework. You should also note that when a relationship is saved, wireless layouts, located in ./custom/Extension/modules/<module>/Ext/WirelessLayoutdefs/, are created and compiled into ./custom/modules/<module>/Ext/Layoutdefs/layoutdefs.ext.php.

An example of this is when deploying a 1-M relationship from Bugs to Leads, the following layoutdef files are generated:

./custom/Extension/modules/Bugs/Ext/Layoutdefs/bugs_leads_1_Bugs.php

```
  <?php

$layout_defs["Bugs"]["subpanel_setup"]['bugs_leads_1'] = array (
```

```
  'order' => 100,
  'module' => 'Leads',
  'subpanel_name' => 'default',
  'sort_order' => 'asc',
  'sort_by' => 'id',
  'title_key' => 'LBL_BUGS_LEADS_1_FROM_LEADS_TITLE',
  'get_subpanel_data' => 'bugs_leads_1',
  'top_buttons' =>
  array (
    0 =>
    array (
      'widget_class' => 'SubPanelTopButtonQuickCreate',
    ),
    1 =>
    array (
      'widget_class' => 'SubPanelTopSelectButton',
      'mode' => 'MultiSelect',
    ),
  ),
);
```

./custom/Extension/modules/Bugs/Ext/WirelessLayoutdefs/bugs_leads_1_Bugs.php

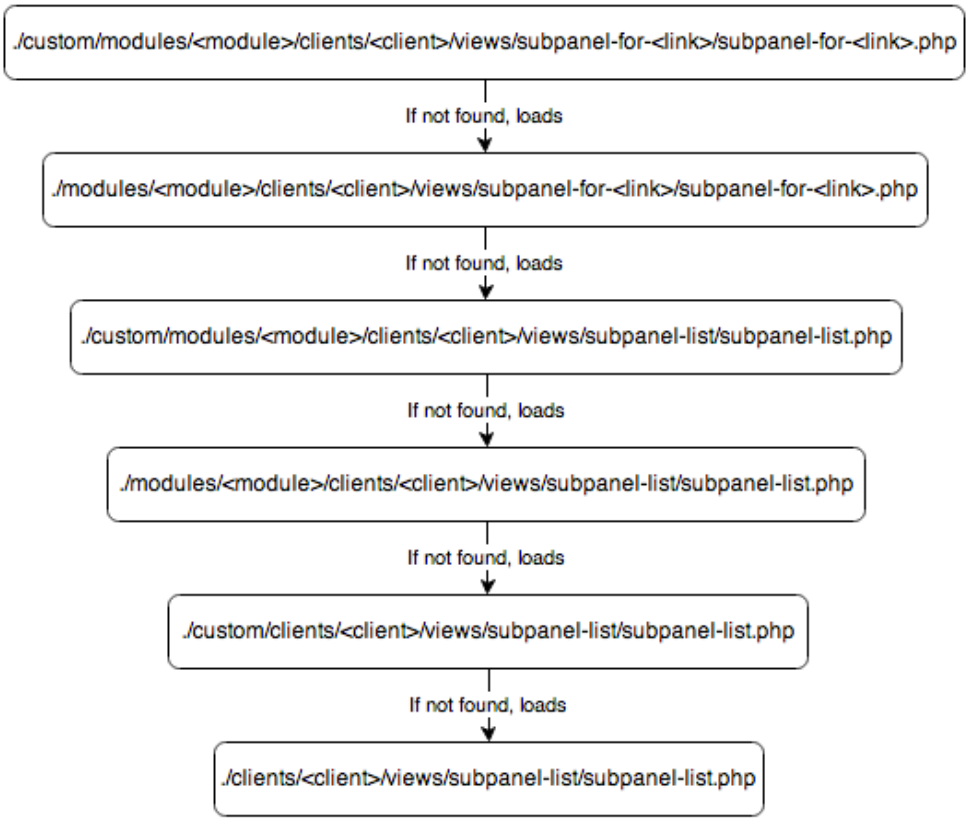
```
  <?php

$layout_defs["Bugs"]["subpanel_setup"]['bugs_leads_1'] = array (
  'order' => 100,
  'module' => 'Leads',
  'subpanel_name' => 'default',
  'title_key' => 'LBL_BUGS_LEADS_1_FROM_LEADS_TITLE',
  'get_subpanel_data' => 'bugs_leads_1',
);
```

Fields Metadata

Sidecar's subpanel field layouts are initially defined by the subpanel list-view metadata.

Hierarchy Diagram

The subpanel list metadata is loaded in the following manner:



Note: The Sugar application's client type is "base". For more information on the various client types, please refer to the User Interface page.

Subpanel List Views

By default, all modules come with a default set of subpanel fields for when they are rendered as a subpanel. An example of this is can be found in the Bugs module:

./modules/Bugs/clients/base/views/subpanel-list/subpanel-list.php

```
  <?php

$subpanel_layout['list_fields'] = array (
  'full_name' =>
  array (
    'type' => 'fullname',
    'link' => true,
    'studio' =>
    array (
      'listview' => false,
    ),
    'vname' => 'LBL_NAME',
    'width' => '10%',
```

```
  'default' => true,
),
'date_entered' =>
array (
  'type' => 'datetime',
  'studio' =>
  array (
    'portaleditview' => false,
  ),
  'readonly' => true,
  'vname' => 'LBL_DATE_ENTERED',
  'width' => '10%',
  'default' => true,
),
'refered_by' =>
array (
  'vname' => 'LBL_LIST_REFERED_BY',
  'width' => '10%',
  'default' => true,
),
'lead_source' =>
array (
  'vname' => 'LBL_LIST_LEAD_SOURCE',
  'width' => '10%',
  'default' => true,
),
'phone_work' =>
array (
  'vname' => 'LBL_LIST_PHONE',
  'width' => '10%',
  'default' => true,
),
'lead_source_description' =>
array (
  'name' => 'lead_source_description',
  'vname' => 'LBL_LIST_LEAD_SOURCE_DESCRIPTION',
  'width' => '10%',
  'sortable' => false,
  'default' => true,
),
'assigned_user_name' =>
array (
  'name' => 'assigned_user_name',
  'vname' => 'LBL_LIST_ASSIGNED_TO_NAME',
  'widget_class' => 'SubPanelDetailViewLink',
  'target_record_key' => 'assigned_user_id',
```

```
    'target_module' => 'Employees',
    'width' => '10%',
    'default' => true,
  ),
  'first_name' =>
  array (
    'usage' => 'query_only',
  ),
  'last_name' =>
  array (
    'usage' => 'query_only',
  ),
  'salutation' =>
  array (
    'name' => 'salutation',
    'usage' => 'query_only',
  ),
);
```

To modify this layout, navigate to Admin > Studio > {Parent Module} > Subpanels > Bugs and make your changes. Once saved, Sugar will generate ./custom/modules/Bugs/clients/<client>/views/subpanel-for-<link>/subpanel-for-<link>.php which will be used for rendering the fields you selected.

You should note that, just as Sugar mimics the Sidecar layouts in the legacy MVC framework for modules in backward compatibility, it also mimics the field list in ./modules/<module>/metadata/subpanels/default.php and ./custom/modules/<module>/metadata/subpanels/default.php. This is done to ensure that any related modules, whether in Sidecar or Backward Compatibility mode, display the same field list as expected.

Database

Overview

All Sugar products support the MySQL and Microsoft SQL Server databases. Sugar Enterprise and Sugar Ultimate also support the DB2 and Oracle databases. In general, Sugar uses only common database functionality, and the application logic is embedded in the PHP code. Sugar does not use or recommend database triggers or stored procedures. This design simplifies coding and testing across different database vendors. The only implementation difference across the various supported databases is column types.

Primary Keys, Foreign Keys, and GUIDs

By default, Sugar uses globally unique identification values (GUIDs) for primary keys for all database records. Sugar provides a Sugarcrm\Sugarcrm\Util\Uuid::uuid1() utility function for creating these GUIDs in the following format: aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee. The primary key's column length is 36 characters.

The GUID format and value has no special meaning (relevance) in Sugar other than the ability to match records in the database. Sugar links two records (such as an Accounts record with a Contacts record) with a specified ID in the record type relationship table (e.g. accounts_contacts).

Primary keys in Sugar may contain any unique string such as a GUID algorithm, a key that has some meaning (e.g. bean type first, followed by info), an external key, or auto-incrementing numbers converted to strings. Sugar chose GUIDs over auto-incrementing keys to enable easier data synchronization across databases and avoid primary-key collisions.

You can also import data from a previous system with one primary key format and make all new records in Sugar use the GUID primary key format. All keys must be stored as globally unique strings with no more than 36 characters.

Notice If multiple records between modules contain matching ids, you may experience undesired behaviors within the system.

To implement a new primary key method or to import data with a different primary key format (based on the existing GUID mechanism for new records), keep in mind the following rules of primary key behavior:

- **Quote characters** : Sugar expects primary keys to be string types and will format the SQL with quotes. If you change the primary key types to an integer type, SQL errors may occur since Sugar stores all ID values in quotes in the generated SQL. The database may be able to ignore this issue. MySQL running in Safe mode experiences issues, for instance.
- **Case sensitivity** : The ID values abc and ABC are treated the same in MySQL but represent different values in Oracle. When migrating data to Sugar, some CRM systems may use case-sensitive strings as their IDs on export. If this is the case, and you are running MySQL, you must run an algorithm on the data to make sure all of the IDs are unique. One simple algorithm is to MD5 the ID values that they provide. A quick check will let you know if there is a problem. If you imported 80,000 leads and there are only 60,000 in the system, some may have been lost due to non-unique primary keys caused by case insensitivity.

- **Key size** : Sugar only tracks the first 36 characters in the primary key. Any replacement primary key will either require changing all of the ID columns with one of an appropriate size or to make sure you do not run into any truncation or padding issues. MySQL in some versions has had issues with Sugar where the IDs were not matching because it was adding spaces to pad the row out to the full size. MySQL's handling of char and varchar padding has changed in later versions. To protect against this, make sure the GUIDs are not padded with blanks in the database by removing any leading or trailing space characters.

Indexes

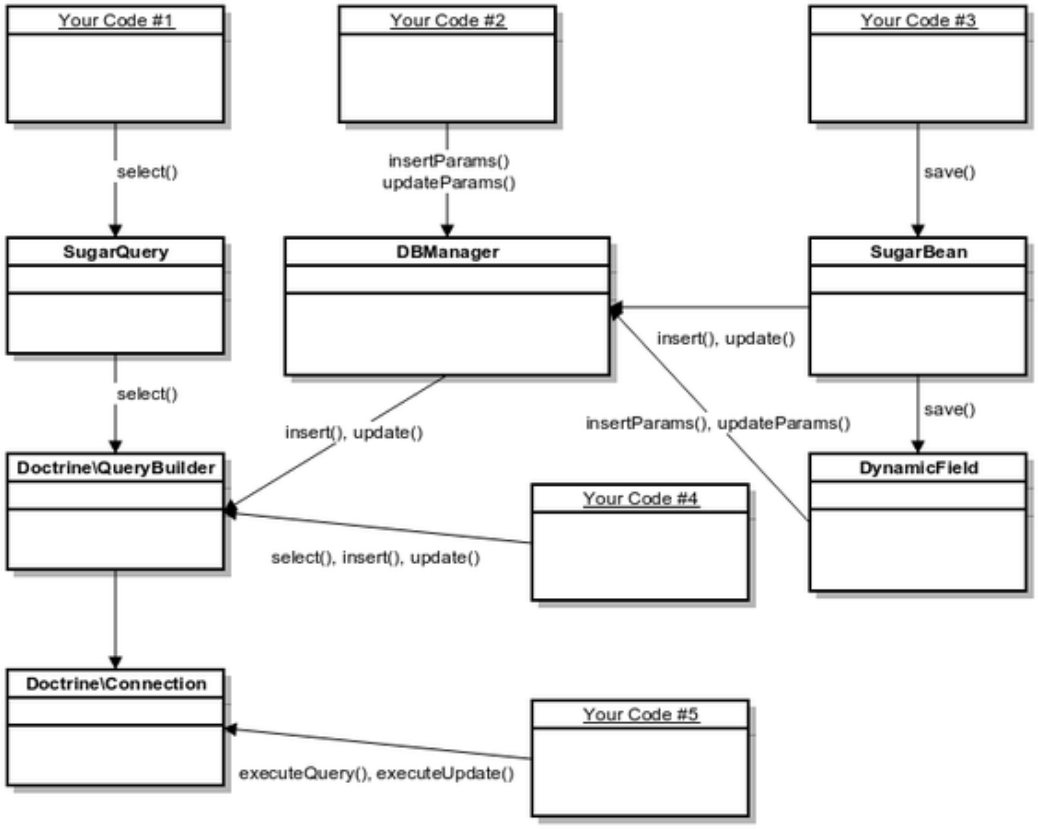
Indexes can be defined in the main or custom vardefs.php for a module in an array under the key indices. See below for an example of defining several indices:

```
  'indices' => array(
     array(
         'name' => 'idx_modulename_name',
         'type' => 'index',
         'fields' => array('name'),
     ),
     array(
         'name' => 'idx_modulename_assigned_deleted',
         'type' => 'index',
         'fields' => array('assigned_user_id', 'deleted'),
     ),
 ),
```

The name of the index must start with idx_ and must be unique across the database. Possible values for type include primary for a primary key or index for a normal index. The fields list matches the column names used in the database.

Doctrine

In order to provide robust support for Prepared Statements, which provide more security and better database access performance, Sugar 7.9 has adopted parts of Doctrine's Database Abstraction Layer, especially the QueryBuilder class, for working with prepared statements. The picture below shows how Sugar objects like DBManager and SugarQuery utilize Doctrine to provide this functionality, while still using the same toolset that has existed in Sugar 7.



DBManager

The DBManager class will use Doctrine QueryBuilder for building INSERT and UPDATE queries.

SugarQuery

The SugarQuery class will use Doctrine QueryBuilder for building SELECT queries.

SugarBean

The SugarBean class will continue to use DBManager class for saving all fields.

DBManager

Overview

The DBManager Object provides an interface for working with the database. As of Sugar 7.9, there are some deprecated methods that have been removed from the

system that are outlined in the Release Notes.

Instantiating the DBManager Object

The DBManagerFactory class, located in ./include/database/DBManagerFactory.php, can help instantiate a DBManager object using the getInstance() method.

```
  $db = \DBManagerFactory::getInstance();
```

For best practices, we recommend using the global DBManager Object:

```
  $GLOBALS['db']
```

Querying The Database

As of Sugar 7.9, there is support for prepared statements. The following sections outline the legacy usage and the new prepared statement usage.

SELECT queries

For select queries that do not have a dynamic portion of the where clause, you can use the query() method on the DBManager object. For queries that are accepting data passed into the system in the where clause, the following examples demonstrate how best utilize the new Prepared Statement functionality.

Legacy:

```
  $id = '1234-abcde-fgh45-6789';
$query = 'SELECT * FROM accounts WHERE id = ' . $GLOBALS['db']->quoted
($id);
$results = $GLOBALS['db']->query($query);
```

Best Practice:

Use the getConnection() method to retrieve a Doctrine Connection Object which handles prepared statements.

```
  $id = '1234-abcde-fgh45-6789';
$query = 'SELECT * FROM accounts WHERE id = ?';
```

```
$conn = $GLOBALS['db']->getConnection();
$stmt = $conn->executeQuery($query, array($id));
```

In the case that query logic is variable or conditionally built then it makes sense to use Doctrine QueryBuilder directly.

Legacy:

```
  $query = 'SELECT * FROM accounts';
if ($status !== null) {
 $query .= ' WHERE status = ' . $GLOBALS['db']->quoted($status);
}
$results = $GLOBALS['db']->query($query);
```

Best Practice:

Use the getConnection() method to retrieve the Doctrine Connection Object, and then use the createQueryBuilder() method on the Connection Object to retrieve the QueryBuilder Object.

```
  $builder = $GLOBALS['db']->getConnection()->createQueryBuilder();
$builder->select('*')->from('accounts');
if ($status !== null) {
 $builder->where('status = ' . $builder->createPositionalParameter($st
atus)));
}
$stmt = $builder->execute();
```

Retrieving Results

Legacy:

After using the query() method, such as in the Legacy code examples above, you can use the fetchByAssoc() method to retrieve results. The query() method will submit the query and retrieve the results while the fetchByAssoc() method will iterate through the results:

```
  $sql = "SELECT id FROM accounts WHERE deleted = 0";
$result = $GLOBALS['db']->query($sql);

while($row = $GLOBALS['db']->fetchByAssoc($result) )
```

```
{
    //Use $row['id'] to grab the id fields value
    $id = $row['id'];
}
```

Best Practice:

When using Prepared Statements, both the Doctrine Query Builder and the Doctrine Connection Object will return a Doctrine\DBAL\Portability\Statement Object to allow iterating through the results of the query. You can use the fetch() or fetchAll() methods to retrieve results.

fetchAll() Example

The fetchAll() method will return the entire result set as an array, with each index containing a row of data.

```
  $id = '1234-abcde-fgh45-6789';
$query = 'SELECT * FROM accounts WHERE id = ?';
$conn = $GLOBALS['db']->getConnection();
$stmt = $conn->executeQuery($query, array($id));
foreach($stmt->fetchAll() as $row){
 $id = $row['id']
 //do other stuff...
}
```

fetch() Example

The fetch() method will return the next index in the result set.

```
  $id = '1234-abcde-fgh45-6789';
$query = 'SELECT * FROM accounts WHERE id = ?';
$conn = $GLOBALS['db']->getConnection();
$stmt = $conn->executeQuery($query, array($id));
while($row = $stmt->fetch()){
 $id = $row['id']
 //do other stuff...
}
```

Retrieving a Single Result

To retrieve a single result from the database, such as a specific record field, you can use the getOne() method for Legacy query usage.

```
  $sql = "SELECT name FROM accounts WHERE id = '{$id}'";
$name = $GLOBALS['db']->getOne($sql);
```

Limiting Results

To limit the results of a query, you can add a limit to the SQL string or for legacy query usage you can use the limitQuery() method on the DBManager Object:

Legacy:

```
  $sql = "SELECT id FROM accounts WHERE deleted = 0";
$offset = 0;
$limit = 1;

$result = $GLOBALS['db']->limitQuery($sql, $offset, $limit);

while($row = $GLOBALS['db']->fetchByAssoc($result) )
{
    //Use $row['id'] to grab the id fields value
    $id = $row['id'];
}
```

Prepared Statements:

When using the Doctrine Query Builder, you can limit the results of the query by using the setMaxResults() method.

```
  $builder = $GLOBALS['db']->getConnection()->createQueryBuilder();
$builder->select('*')->from('accounts');
if ($status !== null) {
 $builder->where('status = ' . $builder->createPositionalParameter($st
atus)));
}
$builder->setMaxResults(2);
$stmt = $builder->execute();
```

INSERT queries

INSERT queries can be easily performed using DBManager class.

Legacy:

```
  $query = 'INSERT INTO table (foo, bar) VALUES ("foo", "bar")';
$GLOBALS['db']->query($query);
```

Best Practice:

```
  $fieldDefs = $GLOBALS['dictionary']['table']['fields'];
$GLOBALS['db']->insertParams('table', $fieldDefs, array('foo' => 'foo'
,'bar' => 'bar'));
```

UPDATE queries

When updating records with known IDs or a set of records with simple filtering criteria, then DBManager can be used:

Legacy:

```
  $query = 'UPDATE table SET foo = "bar" WHERE id = ' . $GLOBALS['db']
->quoted($id);
$GLOBALS['db']->query($query);
```

Best Practice:

```
  $fieldDefs = $GLOBALS['dictionary']['table']['fields'];
$GLOBALS['db']->updateParams('table', $fieldDefs, array('foo' => 'bar'
,), array('id' => $id) );
```

For more complex criteria or when column values contain expressions or references to other fields in the table then Doctrine QueryBuilder can be used.

Legacy:

```
  $query = 'UPDATE table SET foo = "bar" WHERE foo = "foo" OR foo IS N
ULL';
$GLOBALS['db']->execute($query);
```

Best Practice:

```
  $query = 'UPDATE table SET foo = ? WHERE foo = ? OR foo IS NULL';
$conn = $GLOBALS['db']->getConnection();
$stmt = $conn->executeQuery($query, array('bar', 'foo'));
```

Generating SQL Queries from SugarBean

To have Sugar automatically generate SQL queries, you can use the following methods from the bean class.

Select Queries

To create a select query you can use the create_new_list_query() method:

```
  $bean = BeanFactory::newBean($module);

$order_by = '';
$where = '';
$fields = array(
    'id',
    'name',
);

$sql = $bean->create_new_list_query($order_by, $where, $fields);
```

Count Queries

You can also run the generated SQL through the create_list_count_query() method to generate a count query:

```
  $bean = BeanFactory::newBean('Accounts');

$sql = "SELECT * FROM accounts WHERE deleted = 0";

$count_sql = $bean->create_list_count_query($sql);
```

SugarQuery

Overview

SugarQuery, located in ./include/SugarQuery/SugarQuery.php, provides an object-oriented approach to working with the database. This allows developers to generate the applicable SQL for a Sugar system without having to know which database backend the instance is using. SugarQuery supports all databases supported by Sugar.

Note: SugarQuery only supports reading data from the database at this time (i.e. SELECT statements).

Setup

To use SugarQuery, simply create a new SugarQuery object.

```
$sugarQuery = new SugarQuery();
```

Basic Usage

Using the SugarQuery object to retrieve records or generate SQL queries is very simple. At a minimum you need to set the Module you are working with, using the from() method, however, there are helper methods for just about any operation you would need in a SQL query. The methods listed below will outline the major methods you should consider utilizing on the SugarQuery object in order to achieve your development goals.

from()

The from() method is used to set the primary module the SugarQuery object will be querying from. It is also used to set some crucial options for the query, such as whether Team Security should be used or if only non-deleted records should be queried. The following example will set the SugarQuery object to query from the Accounts module.

```
$sugarQuery->from(BeanFactory::newBean('Accounts'));
```

Arguments

Name	Type	Required	Description
$bean	SugarBean Object	true	The SugarBean object for a specified module. The SugarBean

			object does not have to be a blank or new Bean as seen in the example above, but can be a previously instantiated SugarBean object.
$options	Array	false	An associative array that can specify any of the following options: • **alias** - *string* - The alias for the module table in the generated SQL query • **team_security** - *boolean* - Whether or not Team Security should be added to the generated SQL query • **add_deleted** - *boolean* - Whether or not 'deleted' = 0 should be added to Where clause of generated SQL query

Returns

SugarQuery Object

Allows for method chaining on the SugarQuery object.

select()

The example above demonstrates the most basic example of retrieving records from a module. The select() method can be used on the SugarQuery object to specify the specific fields you wish to retrieve from the query.

```
//Alter the Selected Fields
$sugarQuery->select(array('id', 'name'));
```

Arguments

Name	Type	Required	Description
$fields	Array	false	Sets the fields that should be added to the SELECT portion of the SQL query

Returns

SugarQuery_Builder_Select Object

You cannot chain SugarQuery methods off of the select() method, however, you can use the returned Select object to modify the SELECT portion of the statement. Review the SugarQuery_Builder_Select object in ./include/SugarQuery/Builder/Select.php for additional information on usage.

where()

To add a WHERE clause to the query, use the where() method to generate the Where object, and then use method chaining with the various helper methods to add conditions. To add a WHERE clause for records with the name field containing the letter "I", you could add the following code.

```
//add the where clause
$sugarQuery->where()->contains('name', 'I');
```

Arguments

None

Returns

SugarQuery_Builder_Where Object

Allows for method chaining on the Where object as shown above. Review the SugarQuery Conditions documentation for a full spectrum of where() method usage.

Relationships

join()

To add data from a related module to the SugarQuery, use the join() method. Adding to the same SugarQuery code example in this page, the following code would add the JOIN from Accounts module tables to Contacts table:

```
//add join
$sugarQuery->join('contacts');
```

Arguments

Name	Type	Required	Description
$link_name	String	true	The name of the relationship
$options	Array	false	An associative array that can specify any of the following options: • **alias** - *string* - The alias for the module table in the generated SQL query • **relatedJoin** - *string* - If joining to a secondary table

			(related to a related module), such as joining on Opportunities related to Contacts, when querying from Accounts, you can specify either the name of the relationship or the alias you specified for that relationship table.

Returns

SugarQuery_Builder_Join Object

Allows for method chaining on the SugarQuery_Builder_Join Object, to add additional conditions to the WHERE clause of the SQL condition.

joinTable()

If you were using the joinRaw() method in previous versions of Sugar, this is the replacement method which allows for joining to a related table in SugarQuery. Adding to the same SugarQuery code example in this page, the following code would add the JOIN from Accounts module tables to the accounts_contacts table:

```
//add join
$sugarQuery->joinTable('accounts_contacts', array('alias' => 'ac'))->on()
    ->equalsField('accounts.id','ac.account_id')
    ->equals('ac.primary_account',1);
```

Arguments

Name	Type	Required	Description
$table_name	String	true	The name of the database table to join.
$options	Array	false	An associative array that can specify any of the following options: • **alias** - *string* - The alias for the module table in the generated SQL query

Returns

SugarQuery_Builder_Join Object

Allows for method chaining on the SugarQuery_Builder_Join Object, to add additional conditions to the ON clause using the on() method.

Altering Results

Altering the result set of a query can help the performance, as well as be crucial to finding the correct data. The following methods provide ways to limit the result set and change the order.

distinct()

To group the query on a field, you can use the corresponding distinct() method.

```
//add group by
$sugarQuery->distinct(true);
```

Arguments

Name	Type	Required	Description

$value	Boolean	true	Set whether or not the DISTINCT statement should be added to the query

Returns

Current *SugarQuery Object*

Allows for method chaining on the SugarQuery Object.

limit()

To limit the results of the query, you can use the limit() method.

```
//set the limit
$sugarQuery->limit(10);
```

Arguments

Name	Type	Required	Description
$number	Integer	true	The max amount of rows that should be returned by the query

Returns

Current *SugarQuery Object*

Allows for method chaining on the SugarQuery Object.

offset()

Adding a limit to the query limits the rows returned, however when doing so, you may need to alter the offset of the query to account for pagination or access other portions of the result set. To set an offset, you can use the offset() method.

```
//set the offset
$sugarQuery->offset(5);
```

Arguments

Name	Type	Required	Description
$number	Integer	true	The offset amount of rows, or starting point, of the result

Returns

Current *SugarQuery Object*

Allows for method chaining on the SugarQuery Object.

orderBy()

To order the query on a field, you can use the corresponding orderBy() method. This method can be called multiple times, to add multiple fields to the order by clause of the query.

```
//add group by
$sugarQuery->orderBy('account_type');
```

Arguments

Name	Type	Required	Description
$column	String	true	The field you want the query to be grouped on
$direction	String	false	Sets the direction of sorting. Must be 'ASC' or 'DESC'. The default is 'DESC'.

Returns

Current *SugarQuery Object*

Allows for method chaining on the SugarQuery Object.

Execution

Once you have the SugarQuery object setup and configured for your statement,

you will want to retrieve the results of the query, or simply get the generated query for the object. The following methods are used for executing the SugarQuery object.

execute()

To query the database for a result set, you will use the execute() method. The execute() method will retrieve the results and return them as a raw string, db object, json, or an array depending on the $type parameter. By default, results are returned as an array. An example of fetching records from an account is below:

```
//fetch the result
$result = $sugarQuery->execute();
```

The execute() function will return an array of results that you can iterate through as shown below:

```
Array
(
    [0] => Array
    (
        [id] => f39593da-3f88-3059-4f18-524b4d23d07b
        [name] => International Art Inc
    )
)
```

Note: An empty resultset will return an empty array.

Arguments

Name	Type	Required	Description
$type	String	false	How you want the results of the Query returned. Can be one of the following options: • **db** - Returns the result directly

			from the DatabaseManager resource • **array** - *Default* - Returns the results as a formatted array • **json** - Returns the results encoded as JSON

Returns

Default: *Array.* See above argument details for details on other Return options.

compile()

If you want to log the query being generated or want to output the query without running it during development, the compile() method is what should be used retrieve the Prepared Statement. You can then retrieve the Prepared Statement Object to retrieve the Parameterized SQL and the Parameters. For further information on Prepared Statement usage, see our Database documentation.

```
//get the compiled prepared statement
$preparedStmt = $sugarQuery->compile();

//Retrieve the Parameterized SQL
$sql = $preparedStmt->getSQL();

//Retrieve the parameters as an array
$parameters = $preparedStmt->getParameters();
```

Arguments

No arguments

Returns

Object

The compiled SQL Query built by the SugarQuery object.

SugarQuery Conditions

Overview

Learn about the various methods that can be utilized with SugarQuery to add conditional statements to a query.

Where Clause

Manipulating, the WHERE clause of a SugarQuery object is crucial for getting the correct results. To create a WHERE clause on the query, use the where() method on the SugarQuery object, as outlined in the SugarQuery documentation. Once you have the Where object, you can utilize the following methods on the Where object to add conditional statements.

equals() | notEquals()

Used to equate a field to a given value. Wildcards will not work with this function, as it is looking for an exact match.

```
//add equals
$SugarQuery->where()->equals('name','Test');

//add Not Equals
$SugarQuery->where()->notEquals('name','Tester');
```

Arguments

Name	Type	Required	Description
$field	*String*	true	The field you are checking against
$value	*String*	true	The value the field should be equal to

Returns

SugarQuery_Builder_Where Object

Allows for method chaining on the Where object to add additional conditions.

equalsField() | notEqualsField()

Used to equate a field to another field in the result set.

```
//add an Equals Field statement
$SugarQuery->where()->equalsField('industry','account_type');

//add a Not Equals Field statement
$SugarQuery->where()->notEqualsField('name','account_type');
```

Arguments

Name	Type	Required	Description
$field	*String*	true	The field you are checking against
$field	*String*	true	The other field you want the first field to be equal to

Returns

SugarQuery_Builder_Where Object

Allows for method chaining on the Where object to add additional conditions.

isEmpty() | isNotEmpty()

Used to check if a field is or isn't empty.

```
//add an isEmpty statement
$SugarQuery->where()->isEmpty('industry');

//add an isNotEmpty statement
$SugarQuery->where()->isNotEmpty('name');
```

Arguments

Name	Type	Required	Description
$field	*String*	true	The field you are checking against

Returns

SugarQuery_Builder_Where Object

Allows for method chaining on the Where object to add additional conditions.

isNull() | notNull()

Used to check if a field is or isn't equal to NULL.

```
//add an isNull statement
$SugarQuery->where()->isNull('industry');

//add a notNull statement
$SugarQuery->where()->notNull('name');
```

Arguments

Name	Type	Required	Description
$field	*String*	true	The field you are checking against

Returns

SugarQuery_Builder_Where Object

Allows for method chaining on the Where object to add additional conditions.

contains() | notContains()

Used to check if a field has or doesn't have a specified string in its value. Utilizes the LIKE statement, and wildcards on both sides of the provided string.

```
//add an isNull statement
$SugarQuery->where()->contains('name','Test');

//add a notNull statement
$SugarQuery->where()->notContains('industry','Test');
```

Arguments

Name	Type	Required	Description
$field	*String*	true	The field you are checking against

$value	*String*	true	The string being searched for in the value of the field

Returns

SugarQuery_Builder_Where Object

Allows for method chaining on the Where Object to add additional conditions.

starts() | ends()

Similar to the above contains() method, these methods use the LIKE statement in the SQL query and wildcards for searching for a specified string in the field's value. However, the starts() and ends() methods only wildcard the right side and the left side, respectively. The following example demonstrates searching for records where the Name field starts with A, and ends with E.

```
//add an starts and ends statement
$SugarQuery->where()->starts('name','A')->ends('name','e');
```

Arguments

Name	Type	Required	Description
$field	*String*	true	The field you are checking against
$value	*String*	true	The string being searched for in the value of the field

Returns

SugarQuery_Builder_Where Object

Allows for method chaining on the Where object to add additional conditions.

in() | notIn()

Used to check if a field's value is or isn't one of a set of specified values. The following examples look for records where the industry field is in a list of values, and not in a separate list of values.

```
$values = array(
    'Support',
```

```
    'Sales',
    'Engineering'
);

//add in statement
$SugarQuery->where()->in('industry',$values);

$values = array(
    'Marketing',
    'Accounting'
);

//add NotIn Statement
$SugarQuery->where()->notIn('industry',$values);
```

Arguments

Name	Type	Required	Description
$field	*String*	true	The field you are checking
$values	*Array*	true	The array of values which the field is being checked against

Returns

SugarQuery_Builder_Where Object

Allows for method chaining on the Where object to add additional conditions.

between()

Used primarily for numeric type fields, to check if the value is greater than the minimum number specified and less than the maximum number specified. The following code would check for records where the employees field is between 50 and 1000.

```
//add Between statement
$SugarQuery->where()->between('employees',50,1000);
```

Arguments

Name	Type	Required	Description
$field	*String*	true	The field you are checking against
$min	*Number*	true	The lowest number the field's value should be
$max	*Number*	true	The highest number the field's value should be

Returns

SugarQuery_Builder_Where Object

Allows for method chaining on the Where object to add additional conditions.

lt() | lte() | gt() | gte()

These methods are primarily for numeric fields, to check if a field's value is less than (<), less than or equal (<=), greater than (>), or greater than or equal (>=) to a specified value.

```
//Add Less Than Statement
$SugarQuery->where()->lt('gross_revenue',1000000);

//Add Less Than or Equal to Statement
$SugarQuery->where()->lte('net_revenue','500000');

//Add Greater Than Statement
$SugarQuery->where()->gt('gross_revenue',500000);

//Add Greater Than or Equal to Statement
$SugarQuery->where()->gte('net_revenue',100000);
```

Arguments

Name	Type	Required	Description
$field	*String*	true	The field you are checking against
$value	*Number*	true	The numeric value for comparison

Returns

SugarQuery_Builder_Where Object

Allows for method chaining on the Where object to add additional conditions.

dateRange()

Used to check if a field's value is between a preset date range from the current time. See the TimeDate documentation on the available date range keys.

```
//add DateRange statement
$SugarQuery->where()->dateRange('date_modified','last_30_days');
```

Arguments

Name	Type	Required	Description
$field	*String*	true	The field you are checking against
$value	*String*	true	The string specifying the date range key that will be used for comparison. Example 'next_7_days'

Returns

SugarQuery_Builder_Where Object

Allows for method chaining on the Where object to add additional conditions.

dateBetween()

To group the query on a field, you can use the corresponding groupBy() method. This method can be called multiple times, to add multiple fields to the grouping of the query.

```
//add group by
$SugarQuery->where()->dateBetween('date_created',array('2016-01-01','2016-03-01'));
```

Arguments

Name	Type	Required	Description
$field	*String*	true	The field you are checking against
$value	*Array*	true	An array containing the minimum date in the first key, and the maximum date in the second.

Returns

SugarQuery_Builder_Where Object

Allows for method chaining on the Where Object to add additional conditions.

Combinations

Now that you have reviewed all of the available conditional statements for SugarQuery, you may want to combine them using AND and OR all within the same query. By default when the where() method is called, chained conditional methods will be added with AND to the where clause. You can specify an OR where clause on the main SugarQuery object by using the orWhere() method, which works the same as the where() method, just adds conditional statements with OR instead. The following methods allow for adding internal AND and OR logic to conditional statements on the Where object.

queryAnd()

To start a group of conditional statements that should all evaluate to True, use the queryAnd() method. For example, if you want to query for Accounts, where the name contains 'Test' AND description contains 'Test', you might use the following code:

```
$SugarQuery = new SugarQuery();
$SugarQuery->select(array('name'));
$SugarQuery->from(BeanFactory::newBean('Accounts'));

//Using queryAnd
$SugarQuery->where()->queryAnd()->contains('name','Test')->contains('d
escription','Test');
```

The above use of queryAnd() method isn't entirely needed, as the main Where object would be using AND for all conditions anyway, but it does group the two

conditions inside of their own parenthesis in the compiled query, as shown below, to demonstrate how it can be used for altering query logic.

```
SELECT  accounts.name name FROM accounts WHERE accounts.deleted = 0 AN
D (accounts.name LIKE '%Test%' AND accounts.description LIKE '%Test%')
```

queryOr()

To start a group of conditional statements that should evaluate to true, if any condition is true, you can use the queryOr() method. For example, if you want to query for Accounts, where the name contains 'Test' or where the description contains 'Test', you might use the following code:

```
$SugarQuery = new SugarQuery();
$SugarQuery->select(array('name'));
$SugarQuery->from(BeanFactory::newBean('Accounts'));

//Using queryOr
$SugarQuery->where()->queryOr()->contains('name','Test')->contains('de
scription','Test');
```

This will group the two conditions inside of their own parenthesis in the compiled query. If either of the conditions is True, it will return a record. An example is shown below.

```
SELECT  accounts.name name FROM accounts WHERE accounts.deleted = 0 AN
D (accounts.name LIKE '%Test%' OR accounts.description LIKE '%Test%')
```

Advanced Techniques

Overview

Learn about some of the advanced methods that SugarQuery has to offer, that are not as commonly used.

Get First Record

Getting the first record in a result set, can be accomplished by using the limit() method. The getOne() method is similar in that it gets the first record, but it also returns the first piece of data for that record.

getOne()

Get the first piece of data on the first record returned by the generated query. In this example, we want the 'name' from the Account with a given ID.

```
$SugarQuery = new SugarQuery();
$SugarQuery->select(array('name'));
$SugarQuery->from(BeanFactory::newBean('Accounts'));
$SugarQuery->where()->equals('id',$id);

//Get the Name of the account
$accountName = $SugarQuery->getOne();
```

Aggregates

setCountQuery()

Currently, the only method available for creating an aggregate column, is the setCountQuery() method on the SugarQuery_Builder_Select Object. You can add this method to your select() method chain, to add count(0) as record_count to the SQL SELECT statement.

```
$SugarQuery = new SugarQuery();
$SugarQuery->select(array('name'))->setCountQuery();
$SugarQuery->from(BeanFactory::newBean('Accounts'));
$SugarQuery->groupByRaw('accounts.name');
```

The above example will output the following prepared statement when using compile():

```
SELECT accounts.name, COUNT(0) AS record_count FROM accounts WHERE acc
ounts.deleted = ? GROUP BY accounts.name, accounts.name
```

Parameters

```
array (
    [1] => 0
)
```

Arguments

No arguments

Returns

SugarQuery_Builder_Select Object

Allows for method chaining on the Select Object.

Joins

Joining to tables and joining via SugarBean relationships is outlined in the SugarQuery documentation, however the SugarQuery_Builder_Join Object has a few helpful methods not mentioned there.

joinName()

If you are not using a custom alias for the relationship or table, you may want to retrieve the generated name used by SugarQuery to add a conditions or join to.

```
$SugarQuery = new SugarQuery();
$SugarQuery->from(BeanFactory::getBean('Accounts'));
$contacts = $SugarQuery->join('contacts')->joinName();
$SugarQuery->select(array("$contacts.full_name"));
$SugarQuery->where()->equals('industry', 'Media');
```

The above example will output the following prepared statement when using compile():

```
SELECT jt0_contacts.salutation rel_full_name_salutation, jt0_contacts.
first_name rel_full_name_first_name, jt0_contacts.last_name rel_full_n
ame_last_name FROM accounts INNER JOIN accounts_contacts jt1_accounts_
contacts ON (accounts.id = jt1_accounts_contacts.account_id) AND (jt1_
accounts_contacts.deleted = ?) INNER JOIN contacts jt0_contacts ON (jt
0_contacts.id = jt1_accounts_contacts.contact_id) AND (jt0_contacts.de
leted = ?) WHERE (accounts.industry = ?) AND (accounts.deleted = ?)
```

Parameters:

```
array (
```

```
    [1] => 0
    [2] => 0
    [3] => Media
    [4] => 0
)
```

Arguments

No arguments

Returns

string

The name used in Query to identify the joined table

Unions

Unions allow joining multiple queries with the same selected fields to be combined during output. You can use Unions in SugarQuery by using the union() method.

union()

To add a union, you can use the corresponding union() method. The example below will join two SQL queries:

```
//Fetch the bean of the module to query
$bean = BeanFactory::newBean('Accounts');

//Specify fields to fetch
$fields = array(
    'id',
    'name'
);

//Create first query
$sq1 = new SugarQuery();
$sq1->select($fields);
$sq1->from($bean, array('team_security' => false));
$sq1->Where()->in('account_type', array('Customer'));
```

```
//Create second query
$sq2 = new SugarQuery();
$sq2->select($fields);
$sq2->from($bean, array('team_security' => false));
$sq2->Where()->in('account_type', array('Investor'));

//Create union
$sqUnion = new SugarQuery();
$sqUnion->union($sq1);
$sqUnion->union($sq2);
$sqUnion->limit(5);
```

The above example will output the following prepared statement when using compile():

```
SELECT accounts.id, accounts.name FROM accounts WHERE (accounts.accoun
t_type IN (?)) AND (accounts.deleted = ?) UNION ALL SELECT accounts.id
, accounts.name FROM accounts WHERE (accounts.account_type IN (?)) AND
 (accounts.deleted = ?) LIMIT 5
```

Parameters:

```
array (
    [1] => Customer
    [2] => 0
    [3] => Investor
    [4] => 0
)
```

Arguments

Name	Type	Required	Description
$select	*SugarQuery*	true	The SugarQuery Object you wish to add to the UNION query
$all	*Boolean*	false	Whether to use UNION ALL or just UNION in the query. The default

			value is TRUE.

Returns

SugarQuery_Builder_Union Object

Allows for method chaining on the Union Object.

Having

When using aggregates in a query, you might want to filter out values based on a condition. SugarQuery provides the having() method for adding HAVING clause to the query.

having()

To use the having() method, you have to build a SugarQuery_Builder_Condition Object and set the field, operator, and value that condition is based on.

```
$SugarQuery = new SugarQuery();
$SugarQuery->from(BeanFactory::getBean('Accounts'));
$SugarQuery->join('contacts', array('alias' => 'industryContacts'));
$SugarQuery->join('opportunities', array('relatedJoin' => 'industryCon
tacts', 'alias' => 'contactsOpportunities'));
$SugarQuery->select()->setCountQuery();
$SugarQuery->where()->equals('contactsOpportunities.sales_stage', 'clo
sed');
$havingCondition = new SugarQuery_Builder_Condition($SugarQuery);
$havingCondition->setField('contactsOpportunities.amount')->setOperato
r('>')->setValues('1000');
$SugarQuery->having($havingCondition);
```

The above example will output the following prepared statement when using compile():

```
SELECT COUNT(0) AS record_count FROM accounts INNER JOIN accounts_cont
acts jt0_accounts_contacts ON (accounts.id = jt0_accounts_contacts.acc
ount_id) AND (jt0_accounts_contacts.deleted = ?) INNER JOIN contacts i
ndustryContacts ON (industryContacts.id = jt0_accounts_contacts.contac
t_id) AND (industryContacts.deleted = ?) INNER JOIN opportunities_cont
acts jt1_opportunities_contacts ON jt1_opportunities_contacts.deleted
= ? INNER JOIN opportunities contactsOpportunities ON (contactsOpportu
nities.id = jt1_opportunities_contacts.opportunity_id) AND (contactsOp
portunities.deleted = ?) WHERE (contactsOpportunities.sales_stage = ?)
```

```
 AND (accounts.deleted = ?) HAVING contactsOpportunities.amount > ?
```

Parameters:

```
array (
    [1] => 0
    [2] => 0
    [3] => 0
    [4] => 0
    [5] => closed
    [6] => 0
    [7] => 1000
)
```

Arguments

Name	Type	Required	Description
$condition	*SugarQuery_Builder_Condition*	true	The conditional object used to generate the HAVING clause

Returns

SugarQuery_Builder_Having Object

Allows for method chaining on the Having Object to add additional conditions.

Raw Methods

The SugarQuery Object has a few helper methods that allow raw SQL statement parts to be passed into. This allows for more complex statements or edge case scenarios where a helper function may not have met the requirements for the query.

whereRaw()

To add to the WHERE clause of SugarQuery Object with raw SQL syntax, you can utilize the whereRaw() method. This method will append the specified statement, to the WHERE clause using an AND operator, and will wrap the entire statement in parenthesis. The following is an example use with the output:

```
$SugarQuery = new SugarQuery();
$SugarQuery->select(array('name'));
$SugarQuery->from(BeanFactory::newBean('Accounts'));
$SugarQuery->whereRaw("name LIKE '%T%'");
```

The above example will output the following prepared statement when using compile():

```
SELECT accounts.name FROM accounts WHERE (name LIKE '%T%') AND (accoun
ts.deleted = ?)
```

Parameters:

```
array (
    [1] => 0
)
```

Arguments

Name	Type	Required	Description
$sql	*String*	true	The WHERE clause SQL to be appended to the where clause on the SugarQuery object. All conditions passed in are wrapped in parenthesis and appended using AND (if other conditions exist on where clause).

Returns

SugarQuery_Builder_Where Object

Allows for method chaining on the Where object.

groupByRaw()

To add multiple fields to the GROUP BY statement on the SugarQuery Object, it may be easiest to use the groupByRaw() method.

```
$SugarQuery = new SugarQuery();
$SugarQuery->select(array('account_type', 'industry'));
$SugarQuery->from(BeanFactory::newBean('Accounts'));
$SugarQuery->groupByRaw("accounts.account_type,accounts.industry");
```

The above example will output the following prepared statement when using compile():

```
SELECT accounts.account_type, accounts.industry FROM accounts WHERE ac
counts.deleted = ? GROUP BY accounts.account_type,accounts.industry
```

Parameters:

```
array (
    [1] => 0
)
```

Arguments

Name	Type	Required	Description
$sql	*String*	true	The GROUP BY statement, without the GROUP BY keyword.

Returns

SugarQuery Object

Allows for method chaining on the SugarQuery Object.

orderByRaw()

Using the oderBy() method only allows for adding a single field to the SugarQuery object at a time. In some cases, you might consider using the orderByRaw()

method to add multiple fields or the entire ORDER BY statement to the SugarQuery object.

```
$SugarQuery = new SugarQuery();
$SugarQuery->select(array('name'));
$SugarQuery->from(BeanFactory::newBean('Accounts'));
$SugarQuery->orderByRaw("accounts.name DESC, accounts.date_modified");
```

The above example will output the following prepared statement when using compile():

```
SELECT accounts.name FROM accounts WHERE accounts.deleted = ? ORDER BY
 accounts.name DESC, accounts.date_modified DESC, accounts.id DESC
```

Parameters:

```
array (
    [1] => 0
)
```

Arguments

Name	Type	Required	Description
$sql	String	true	The ORDER BY statement, without the ORDER BY keyword.

Returns

SugarQuery Object

Allows for method chaining on the SugarQuery Object.

havingRaw()

Using the havingRaw() method allows for adding a having statement to the SugarQuery object.

```
$SugarQuery = new SugarQuery();
```

```
$SugarQuery->from(BeanFactory::getBean('Accounts'));
$SugarQuery->join('contacts', array('alias' => 'industryContacts'));
$SugarQuery->join('opportunities', array('relatedJoin' => 'industryCon
tacts', 'alias' => 'contactsOpportunities'));
$SugarQuery->select()->setCountQuery();
$SugarQuery->where()->equals('contactsOpportunities.sales_stage', 'clo
sed');
$SugarQuery->havingRaw("contactsOpportunities.amount > 1000");
```

The above example will output the following prepared statement when using compile():

```
SELECT COUNT(0) AS record_count FROM accounts INNER JOIN accounts_cont
acts jt0_accounts_contacts ON (accounts.id = jt0_accounts_contacts.acc
ount_id) AND (jt0_accounts_contacts.deleted = ?) INNER JOIN contacts i
ndustryContacts ON (industryContacts.id = jt0_accounts_contacts.contac
t_id) AND (industryContacts.deleted = ?) INNER JOIN opportunities_cont
acts jt1_opportunities_contacts ON jt1_opportunities_contacts.deleted
= ? INNER JOIN opportunities contactsOpportunities ON (contactsOpportu
nities.id = jt1_opportunities_contacts.opportunity_id) AND (contactsOp
portunities.deleted = ?) WHERE (contactsOpportunities.sales_stage = ?)
 AND (accounts.deleted = ?) HAVING contactsOpportunities.amount > 1000
```

Parameters:

```
array (
    [1] => 0
    [2] => 0
    [3] => 0
    [4] => 0
    [5] => closed
    [6] => 0
)
```

Arguments

Name	Type	Required	Description
$sql	String	true	The HAVING statement, without the

			HAVING keyword.

Returns

SugarQuery Object

Allows for method chaining on the SugarQuery Object.

Architecture

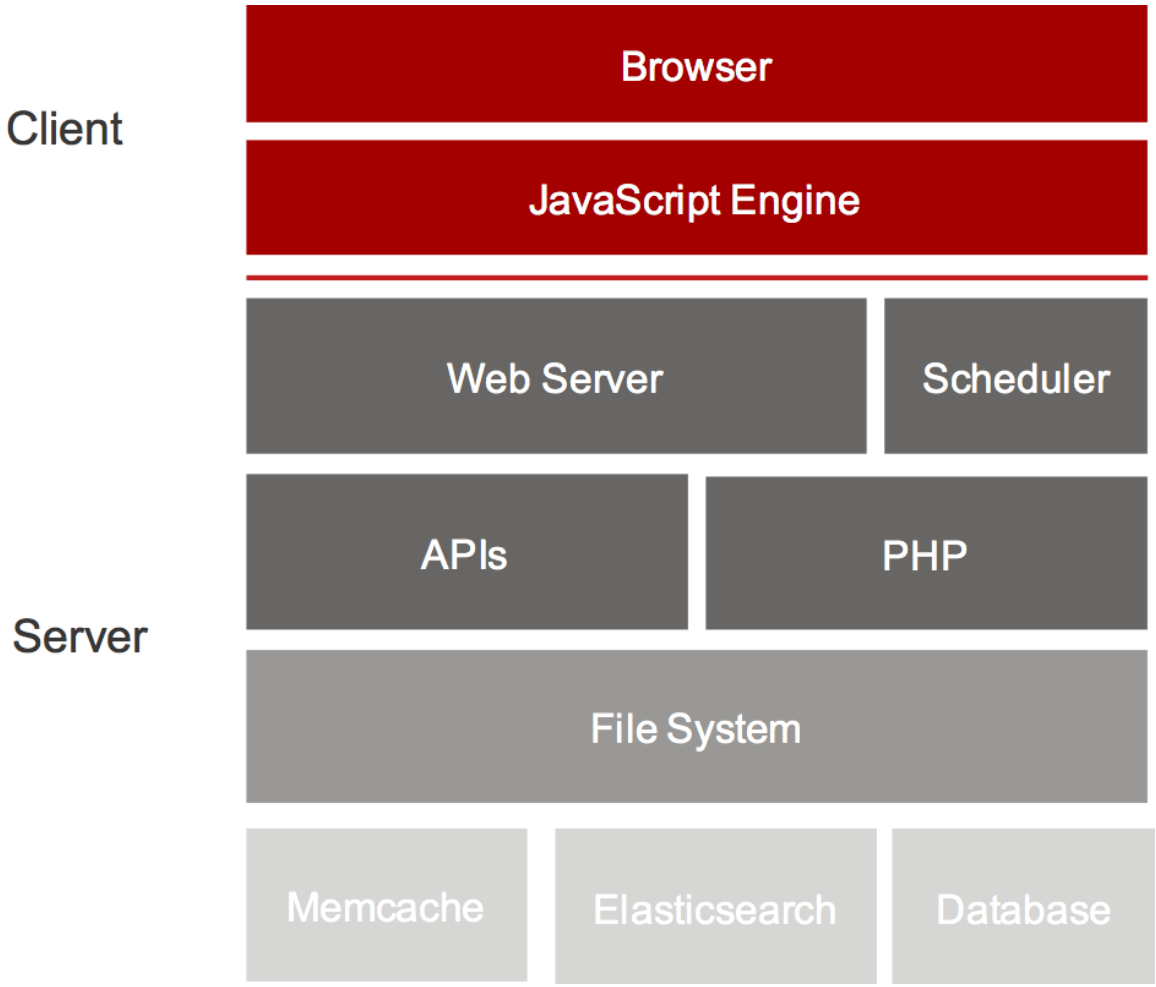
Overview

This section of Sugar's Developer Guide begins with a high-level overview of the Sugar platform's architecture and contains documentation on granular concepts in Sugar such as logic hooks, caching, logging, extensions, job queue, and more.

Please continue to the bottom of this page or use the navigation on the left to explore the related content.

Platform

Sugar® is built on open standards and technology such as HTML5, PHP, and JavaScript, and runs on a variety of free and open-source technology like Linux, MySQL, and Elasticsearch. The Sugar platform also supports common proprietary databases such as Oracle, IBM DB2, and Microsoft SQL Server.



All of Sugar's customers and partners have access to source code that they can choose to deploy on-premise or utilize Sugar's cloud service for a SaaS deployment.

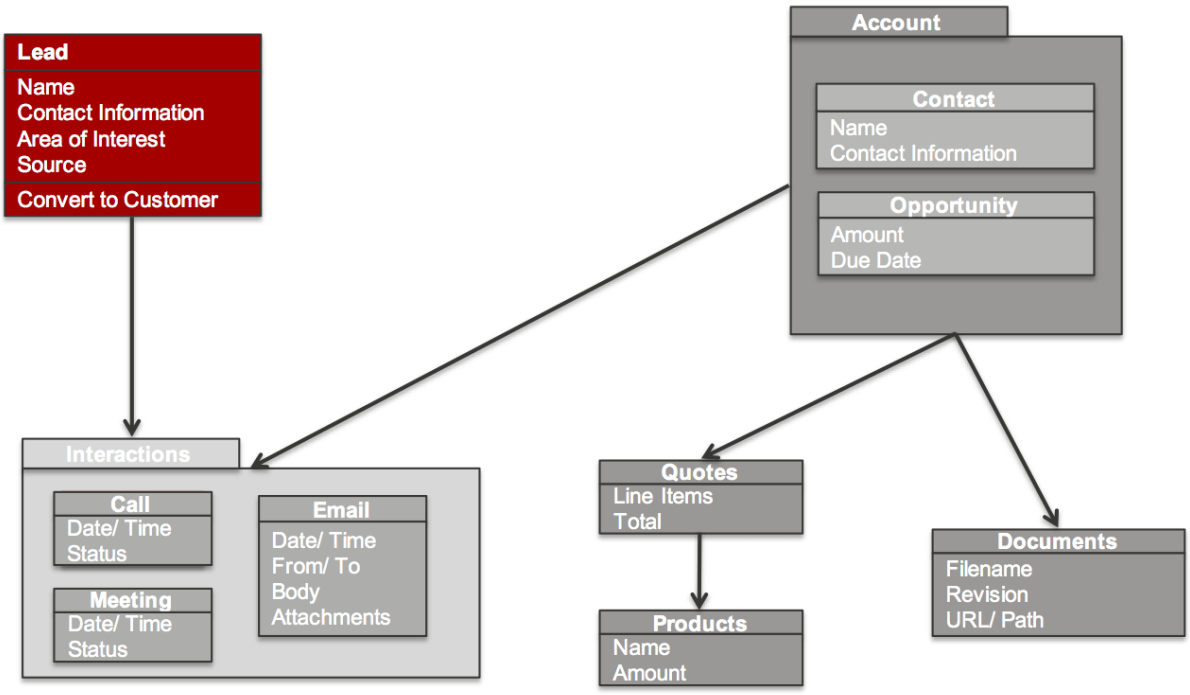
Out of the box, Sugar uses a consistent platform across all clients and devices (e.g. mobile, web, plug-ins, etc.).

Front-End Framework

Our clients are based on a front-end framework called Sidecar. Sidecar is built on open source technology: Backbone.js, jQuery, Handlebars.js, and Bootstrap. The Sidecar framework provides a responsive UI (to support a variety of form factors) and uses modern, single-page client architecture. Sugar clients connect to Sugar server application via our client REST API. The REST API is implemented in PHP and drives server-side business logic and interacts with a database. If it can be accomplished via one of our clients, then its equivalent functionality can be accomplished using our REST API.

The Sugar platform uses modules. Modules are a vertically integrated application component that is traditionally organized around a single feature or record type (or underlying database table). For example, contact records are managed via a Contacts module that contains all the business logic, front-end interface definitions, REST APIs, data schema, and relationships with other modules.

Custom modules can be created and deployed as needed in order to add new features to a Sugar application instance.



Metadata

Sugar's modules are defined primarily using Metadata. There are two types of metadata definitions within Sugar: Vardefs, which define the data model for Sugar modules; and Viewdefs, which define the user interface components that are used

with a module.

Sugar Metadata is implemented as PHP files that can be modified directly by a Sugar Developer making filesystem changes, or indirectly through the use of Sugar Studio and Module Builder by a Sugar Administrator.

Metadata allows you to configure solutions instead of having to write countless lines of custom code in order to implement common customizations such as adding custom fields, calculated values, and changing user interface layouts.

Extensions

Beyond metadata, Sugar is highly customizable and includes an extensive Extensions Framework that provides Sugar Developers the capability to contribute to pre-defined extension points within the application in a way that is upgrade-safe and will not conflict with other customizations that exist in the system.

Autoloader

Overview

The autoloader is an API that allows the unified handling of customizations and customizable metadata while reducing the number of filesystem accesses and improving performance.

SugarAutoLoader

The SugarAutoLoader class, located in ./include/utils/autoloader.php, keeps a map of files within the Sugar directory that may be loaded.

Included File Extensions

The autoloader will only map files with the following extensions:

- bmp
- css
- gif
- hbs
- html
- ico
- jpg
- js
- less

- override
- php
- png
- tif
- tpl
- xml

*All other file extensions are excluded from the mapping.

Class Loading Directories

The autoloader will scan and autoload classes in the following directories:

- ./clients/base/api/
- ./data/duplicatecheck/
- ./data/Relationships/
- ./data/visibility/
- ./include/
- ./include/api/
- ./include/CalendarEvents/
- ./include/SugarSearchEngine/
- ./modules/Calendar/
- ./modules/Mailer/

Ignored Directories

The following directories in Sugar are ignored by the autoloader mapping:

- ./.idea/
- ./cache/
- ./custom/backup/
- ./custom/blowfish/
- ./custom/Extension/
- ./custom/history/
- ./custom/modulebuilder/
- ./docs/
- ./examples/
- ./portal/
- ./tests/
- ./upload/
- ./vendor/bin/
- ./vendor/HTMLPurifier/
- ./vendor/log4php/
- ./vendor/nusoap/
- ./vendor/pclzip/

- ./vendor/reCaptcha/
- ./vendor/ytree/

Configuration API

Overview

Methods to configure loading paths for the AutoLoader API.

addDirectory($dir)

Adds a directory to the directory map for loading classes. Directories added should include a trailing "/".

```
  SugarAutoloader::addDirectory('relative/file/path/');
```

addPrefixDirectory($prefix, $dir)

Adds a prefix and directory to the $prefixMap for loading classes by prefix.

```
  SugarAutoloader::addPrefixDirectory('myPrefix', 'relative/file/path/
');
```

File Check API

Overview

File check methods for use with the AutoLoader API.

existing(...)

Returns an array of filenames that exist in the file map. Accepts any number of arguments of which can be filename or array of filenames. If no files exist, empty array is returned.

```
$files = SugarAutoloader::existing('include/utils.php', 'include/TimeD
ate.php');
```

existingCustom(...)

This method accepts any number of arguments, each of which can be filename or array of filenames. It will return an array of filenames that exist in the file map, adding also files that exist when custom/ is prepended to them. If the original filename already had custom/ prefix, it is not prepended again. custom/ files are added to the list after the root directory files so that if included in order, they will override the data of the root file. If no files exist, empty array is returned.

```
$files = SugarAutoloader::existingCustom('include/utils.php', 'include
/TimeDate.php');
```

existingCustomOne(...)

Returns the last file of the result returned by existingCustom(), or null if none exist. Accepts any number of arguments of which can be filename or array of filenames. Since customized files are placed after the root files, it will return customized file if exists, otherwise root file.

```
$files = SugarAutoloader::existingCustomOne('include/utils.php');
```

You should note that the existingCustomOne() method can be used for loading inline PHP files. An example is shown below:

```
foreach(SugarAutoLoader::existingCustomOne('custom/myFile.php') as $fi
le)
{
    include $file;
}
```

Alternative to including inline PHP files, loading class files should be done using requireWithCustom() .

fileExists($filename)

Checks if a file exists in the file map. You should note that ".." is **not** supported by this function and any paths including ".." will return false. The path components should be separated by **/**. You should also note that multiple slashes are compressed and treated as single slash.

```
$file = 'include/utils.php';

if (SugarAutoloader::fileExists($file))
{
    require_once($file);
}
```

getDirFiles($dir, $get_dirs = false, $extension = null)

Retrieves the list of files existing in the file map under the specified directory. If no files are found, the method will return an empty array. By default, the method will return file paths, however, If $get_dirs is set to true, the method will return only directories. If $extension is set, it would return only files having that specific extension.

```
$files = SugarAutoloader::getDirFiles('include');
```

getFilesCustom($dir, $get_dirs = false, $extension = null)

Retrieves the list of files existing in the file map under the specified directory and under it's custom/ path. If no files are found it will return empty array. By default, the method will return file paths, however, If $get_dirs is set to true, the method will return only directories. If $extension is set, it would return only files having that specific extension.

```
$files = SugarAutoloader::getFilesCustom('include');
```

lookupFile($paths, $file)

Looks up a file in the list of given paths, including with and without custom/ prefix, and return the first match found. The custom/ directory is checked before root files. If no file is found, the method will return false.

```
$paths = array(
  'include',
  'modules',
);
```

```
$files = SugarAutoloader::lookupFile($paths, 'utils.php');
```

requireWithCustom($file, $both = false)

If a custom/ override of the file or the file exist, require_once it and return true, otherwise return false. If $both is set to true, both files are required with the root file being first and custom/ file being second. Unlike other functions, this function will actually include the file.

```
$file = SugarAutoloader::requireWithCustom('include/utils.php');
```

You should note that the requireWithCustom() method should be used for loading class files and not inline PHP files. Inline PHP files should be loaded using the existingCustomOne() method.

File Map Modification API

Overview

Methods to modify files in the AutoLoader API. All the functions below return true on success and false on failure.

addToMap($filename, $save = true)

Adds an existing file to the file map. If $save is true, the new map will be saved to the disk file map, otherwise, it will persist only until the end of the request. This method does not create the file on the filesystem.

```
  SugarAutoloader::addToMap('custom/myFile.php');
```

delFromMap($filename, $save = true)

Removes a file from the file map. If $filename points to a directory, this directory and all files under it are removed from the map. If $save is true, the new map will be saved to the disk file map, otherwise, it will persist only until the end of the request. This method does not delete the file from the filesystem.

```
  SugarAutoloader::delFromMap('custom/myFile.php');
```

put($filename, $data, $save = false)

Saves data to a file on the filesystem and adds it to the file map. If $save is true, the new map will be saved to the disk file map, otherwise, it will persist only until the end of the request.

```
  $file = 'custom/myFile.php';

SugarAutoloader::touch($file, true);

SugarAutoloader::put($file, '<?php /*file content*/ ?>', true);
```

touch($filename, $save = false)

Creates the specified file on the filesystem and adds it to the file map. If $save is true, the new map will be saved to the disk file map, otherwise, it will persist only until the end of the request.

```
  SugarAutoloader::touch('custom/myFile.php', true);
```

unlink($filename, $save = false)

Removes the specified file from the filesystem and from the current file map. If $save is true, the new map will be saved to the disk file map, otherwise, it will persist only until the end of the request.

```
  SugarAutoloader::unlink('custom/myFile.php', true);
```

Metadata API

Overview

Methods to load metadata for the AutoLoader API.

Metadata Loading

For the specific sets of metadata, such as detailviewdefs, editviewefs, listviewdefs, searchdefs, popupdefs, and searchfields, a special process is used to load the correct metadata file. You should note that the variable name for the defs, e.g. "detailviewdefs", is usually the same as variable name, except in the case of "searchfields" where it is "SearchFields".

The process is described below:

1. If ./custom/modules/{$module}/metadata/{$varname}.php exists, it is used as the data file.
2. If ./modules/{$module}/metadata/metafiles.php or ./custom/modules/{$module}/metadata/metafiles.php exists, it is loaded with the custom file being preferred. If the variable name exists in the data specified by the metafile, the corresponding filename is assumed to be the defs file name.
3. If the defs file name or its custom/ override exists, it is used as the data file (custom one is preferred).
4. If no file has been found yet, ./modules/{$module}/metadata/{$varname}.php is checked and if existing, it is used as the data file.
5. Otherwise, no metadata file is used.

loadWithMetafiles($module, $varname)

Returns the specified metadata file for a specific module. You should note that due to the scope nature of include(), this function does not load the actual metadata file but will return the file name that should be loaded by the caller.

```
  $metadataPath = SugarAutoloader::loadWithMetafiles('Accounts', 'edit
viewdefs');
```

loadPopupMeta($module, $metadata = null)

Loads popup metadata for either specified $metadata variable or "popupdefs" variable via loadWithMetafiles() and returns it. If no metadata found returns empty array.

```
  $popupMetadata = SugarAutoloader::loadPopupMeta('Accounts');
```

loadExtension($extname, $module = "application")

Returns the extension path given the extension name and module. For global extensions, the module should be "application" and may be omitted. If the extension has its own module, such as schedulers, it will be used instead of the $module parameter. You should note that due to the scope nature of include(), this function does not load the actual metadata file but return the file name that should be loaded by the caller. If no extension file exists it will return false.

```
  //The list of extensions can be found in ./ModuleInstall/extensions.
php
$extensionPath = SugarAutoloader::loadExtension('logichooks');
```

Caching

Overview

Much of Sugar's user interface is built dynamically using a combination of templates, metadata and language files. A file caching mechanism improves the performance of the system by reducing the number of static metadata and language files that need to be resolved at runtime. This cache directory stores the compiled files for JavaScript files, Handlebars templates, and language files.

In a stock instance, the cache is located in the ./cache/ directory. If you would like to move this directory to a new location, you can update the config parameter cache_dir in config.php or config_override.php to meet your needs. It is not advisable to move the cache to another network server as it may impact system performance.

Developer Mode

To prevent caching while developing, a developer may opt to turn on Developer Mode by navigating to Admin > System Settings > Advanced > Developer Mode. This will disable caching so that developers can test code-level customizations without the need to manually rebuild the cache, which is especially helpful when developing templates, metadata, or language files. The system automatically refreshes the file cache. Make sure to deactivate Developer Mode after completing customizations because this mode degrades system performance.

Uploads

Overview

The upload directory is used to store files uploaded for imports, attachments,

documents, and module loadable packages.

Uploads

The upload directory is used to store any files uploaded to Sugar. By default, anything uploaded to Sugar is stored in the ./upload/ directory. You can change this directory by updating the upload_dir configuration variable. Once uploaded, each file is stored in a subdirectory derived from the UUID filename. For example, the file 3657325a-bdd6-11eb-9a6c-08002723a3b8 will now be stored in the ./upload/25a/ subdirectory. These UID character locations were selected to ensure even distribution of files across the new subdirectories.

There are several file-size limits that affect uploads to Sugar:

Setting Name	Location	Description	Default
upload_max_filesize	php.ini	Maximum allowed size for uploaded files	2 MB
post_max_size	php.ini	Maximum size of POST data that PHP will accept	8 MB
upload_maxsize	config.php	The maximum individual file size that users can upload to modules that support file uploads	30 MB
max_aggregate_email_attachments_bytes	config.php	The maximum allowed size of all uploaded attachments added together for a single email message	10 MB

The lowest of the first three values above will be respected when an oversized file is uploaded to Sugar. The first two settings are the PHP server's upload_max_filesize and post_max_size, which are configured in your system's php.ini file. The third setting is the Sugar configuration for upload_maxsize, which will restrict the upload limit from within Sugar without affecting any other applications that may be running on your server. This limit can be easily adjusted in Sugar via Admin > System Settings but is only useful if it is set to a size smaller than the php.ini limits.

Finally, the max_aggregate_email_attachments_bytes setting will permit users to upload files as email attachments according to the previous size limits but restrict users from uploading more files to a single message than its configuration permits.

Note: PHP-defined file size settings and the upload directory cannot be modified for instances hosted on Sugar's cloud service.

Upload Extensions

By default, several extension types are restricted due to security issues. Any files uploaded with these extensions will have '.txt' appended to it. The restricted extensions are listed below:

- php
- php3
- php4
- php5
- pl
- cgi
- py
- asp
- cfm
- js
- vbs
- html
- htm

You can add or remove extensions to this list by modifying sugar configuration setting for upload_badext. You should note that this setting cannot be modified for instances hosted on Sugar's cloud service.

How Files Are Stored

Note Attachments

When a file is uploaded to Sugar attached to a note, the file will be moved to the upload directory with a GUID name matching that of the notes id. The attributes of the file, such as filename and file_mime_type, will be stored in the note record.

The SQL to fetch information about a notes attachment is shown below:

```
SELECT filename, file_mime_type FROM notes;
```

Email Attachments

Email attachments are stored the same way as note attachments. When an email is imported to Sugar, the file will be moved to the upload directory with a GUID file name matching that of the notes id. The attributes of the file, such as filename and file_mime_type, will be stored in the note record and the note will have a parent_type of 'Emails'. This relates the attachment to the content of the email.

The SQL to fetch information about an emails attachment is shown below:

```
  SELECT filename, file_mime_type FROM notes INNER JOIN emails ON note
s.parent_type = 'Emails' AND notes.parent_id = emails.id INNER JOIN em
ails_text ON emails.id = emails_text.email_id;
```

Picture Fields

Picture fields will upload the image to the upload directory with a GUID name and store the GUID in the database field on the record. An example of picture field can be found on the Contacts module.

For a contact, the id of the picture attachment can be found with the SQL below:

```
  SELECT picture FROM contacts;
```

Knowledge Base Attachments

When working with the Knowledge Base, files and images attached to the form will be created as a record in the /#EmbeddedFiles module. These files will be stored as <GUID> of EmbbededFiles record in the upload folder.
All other file enclosed to attachments field of KBContents will be also saved as Notes record in the upload folder.

The SQL to fetch information about a knowledge base attachment is shown below:

```
  select kb.id, kb.name, kb.revision, n.filename, n.file_mime_type, n.
file_ext from notes n, kbcontents kb where n.parent_type = "KBContents
" and n.parent_id = kb.id order by kb.name, kb.revision;
```

Module Loadable Packages

Module Loader packages are stored in the system differently than other uploads.

They are uploaded to the ./upgrades/module directory, each uploaded file is now stored in a subdirectory derived from the UUID filename. Existing files will be moved into subdirectories during upgrade. For example, the file 3657325a-bdd6-11eb-9a6c-08002723a3b8 will now be stored in the ./upload/25a/ subdirectory. These UID character locations were selected to ensure even distribution of files across the new subdirectories.

The details of the package, such as installation status and description, are stored in the upgrade_history table.

The SQL to fetch information about an installed package is shown below:

```
SELECT * FROM upgrade_history;
```

CSV Imports

When importing records into Sugar, the most recent uploaded CSV file is stored in the upload directory as IMPORT_<module>_<user id>. Once the import has been run, the results of the import are stored in ./upload/import/ directory using a predefined format using the current user's id. The files created will be as follows:

- **dupes_<user id>.csv** : The list of duplicate records found during the import
- **dupesdisplay_<user_id>.csv** : The HTML formatted CSV for display to the user after import
- **error_<user id>.csv** : The list of errors encountered during the import
- **errorrecords_<user_id>.csv** : The HTML formatted CSV for display to the user after import
- **errorrecordsonly_<user id>.csv** : The list of records that encountered an error
- **status_<user id>.csv** : Determines the status of the users import

---

Working with File Uploads

Overview

The UploadFile class handles the various tasks when uploading a file.

Retrieving a Files Upload Location

To retrieve a files upload path, you can use the get_upload_path method and pass in the file's GUID id.

```
  require_once 'include/upload_file.php';
UploadFile::get_upload_path($file_id);
```

This method will normally return the path as:

```
  upload://1d0fd9cc-02e5-f6cd-1426-51a509a63334
```

Retrieving a Files Full File System Location

To retrieve a files full system path, you can use the get_upload_path and real_path methods as shown below:

```
  require_once 'include/upload_file.php';
UploadFile::realpath(UploadFile::get_upload_path($file_id));
```

This method will normally return the path where they are uploaded to. Each uploaded file is now stored in a subdirectory derived from the UUID filename. Existing files will be moved into subdirectories during upgrade. For example, the file 3657325a-bdd6-11eb-9a6c-08002723a3b8 will now be stored in the ./upload/25a/ subdirectory. These UID character locations were selected to ensure even distribution of files across the new subdirectories.

Retrieving a Files Contents

As an alternative to using file_get_contents or sugar_file_get_contents, you can retrieve the contents of a file using the get_file_contents method as shown below:

```
  require_once 'include/upload_file.php';

$file = new UploadFile();

//get the file location
$file->temp_file_location = UploadFile::get_upload_path($file_id);
$file_contents = $file->get_file_contents();
```

Duplicating a File

To duplicate an uploaded file, you can use the duplicate_file method by passing in the files current id and the id you would like it copied to as shown below:

```
  require_once 'include/upload_file.php';

$uploadFile = new UploadFile();
$result = $uploadFile->duplicate_file($oldFileId, $newFileId);
```

Email

Overview

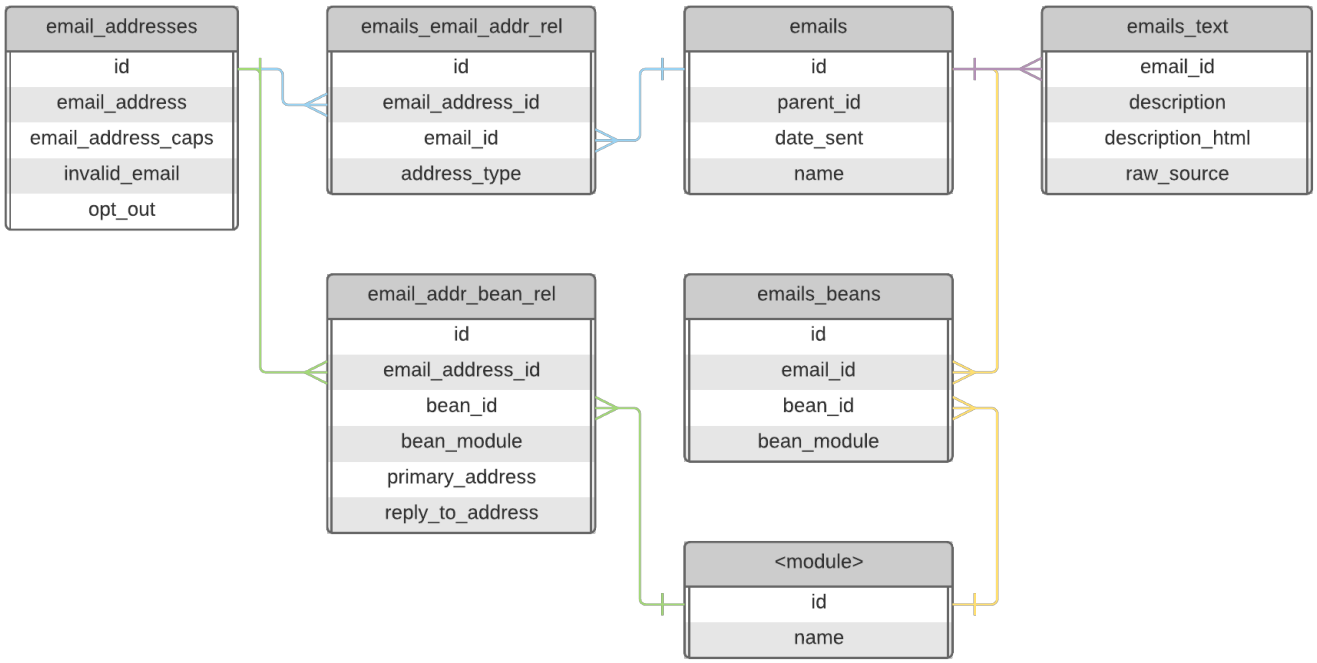
Outlines the relationships between emails, email addresses, and bean records.

Email Tables

Table Name	Description
email_addresses	Each record in this table represents an email address in the system. Note that the invalid_email column and opt_out column are stored on this table, meaning that they are treated as properties of the email address itself, not a relationship attribute between a given email address and related record. This means if a Lead opts-out of a campaign, this email address will be considered opted-out in all contexts, not just with further correspondence with that specific Lead.
email_addr_bean_rel	The email_addr_bean_rel table maintains the relationship between the email address and its parent module record (Contacts, Accounts, Leads, Users, etc) to determine which record of the given module the email address belongs to. Note that this relationship table also has the primary_address and reply_to_address columns to indicate whether a given email address for a given record is the primary email address, the reply to email address, or some other address for the contact. A

	contact can have one address flagged as primary, and one flagged as "Reply To". This can be the same email address or two different email addresses.
emails_email_addr_rel	The emails_email_addr_rel table maintains the relationships between the email address and email records. Note that this relationship table also has the address_type column to indicate if the email address is related to the email as a "To", "From", "CC", or "BCC" address. The valid values at the database level for this column are: from, to, cc, bcc.
emails	Each record in this table represents an email in the system. Note that emails fall outside the scope of this document, but are mentioned here for clarity, as the two modules are closely related.
emails_beans	Similar to the email_addr_bean_rel table, the emails_beans table maintains the relationship between an email record and any related records (Contacts, Accounts, Cases, etc.).
<module>	This is used to show how an email address record relates to a record in any module is set on bean_module. If bean_module is set to Contacts, <module> would be the contacts table.

The following diagram illustrates table relationships between email addresses and other modules, email addresses and email records, and email records and other modules.



Helper Queries

Retrieve the Primary Email Address of a Contact

The following query will fetch the email address given a specific contacts id:

```
  SELECT
  email_address
FROM email_addresses
JOIN email_addr_bean_rel eabr
  ON eabr.email_address_id = email_addresses.id
WHERE eabr.bean_module = "Contacts"
AND eabr.bean_id = "<contact id>"
AND email_addresses.invalid_email = 0
AND eabr.deleted = 0
AND eabr.primary_address = 1;
```

Retrieve All Records Related to an Email Address

The following query will fetch the id and module name of all records related to the specified email address:

```
  SELECT
```

```
  bean_module,
  bean_id
FROM email_addr_bean_rel eabr
JOIN email_addresses
  ON eabr.email_address_id = email_addresses.id
WHERE email_addresses.email_address = "<email address>"
AND eabr.deleted = 0;
```

Retrieve All Emails Sent From An Email Address

The following query will fetch all emails sent from a specified email address:

```
  SELECT
  emails.name,
  emails.date_sent
FROM emails
JOIN emails_email_addr_rel eear
  ON eear.email_id = emails.id
JOIN email_addresses
  ON eear.email_address_id = email_addresses.id
WHERE email_addresses.email_address = "<email address>"
AND eear.address_type = "from"
AND eear.deleted = 0
```

Cleanup Duplicate Email Addresses

The following queries will remove any duplicate email addresses.

First, create a temporary table with distinct records from the email_addr_bean_rel table:

```
  create table email_addr_bean_rel_tmp_ful
SELECT
  *
FROM email_addr_bean_rel
WHERE deleted = '0'
GROUP BY email_address_id,
         bean_module,
         bean_id
ORDER BY primary_address DESC;
```

Next, clear out the email_addr_bean_rel table:

```
  truncate email_addr_bean_rel;
```

Move the records from the temporary table back to email_addr_bean_rel:

```
  INSERT INTO email_addr_bean_rel
 SELECT
 *
 FROM email_addr_bean_rel_tmp;
```

Validate that all of the duplicates have been removed:

```
  SELECT
 COUNT(*) AS repetitions,
 date_modified,
 bean_id,
 bean_module
FROM email_addr_bean_rel
WHERE deleted = '0'
GROUP BY bean_id,
 bean_module,
 email_address_id
HAVING repetitions > 1;
```

Finally, remove the temporary table:

```
  drop table email_addr_bean_rel_tmp;
```

Email Address Validation

Sugar validates emails addresses according to the RFC 5321 and RFC 5322 standards. The following sections will detail how a developer can validate email addresses both server and client side.

Server Side

To validate an email address in a server-side context, the EmailAddress::isValidEmail() static method should be used. For example:

```
  $emailAddress = "test@example.com";
$isValid = EmailAddress::isValidEmail($emailAddress);
```

The EmailAddress::isValidEmail method leverages the PHPMailer library bundled with Sugar, specifically the PHPMailer::validateAddress method, which validates the address according to the RFC 5321 and RFC 5322 standards.

Client Side

To validate an email address client-side context, the app.utils.isValidEmailAddress() function can be used.

```
  var emailAddress = "test@example.com";
var isValid = app.utils.isValidEmailAddress(emailAddress);
```

Note: This function is more permissive and does not conform exactly to the RFC standards used on the server. As such, the email address will be validated again on the server when the record is saved, which could still fail validation.

Mailer Factory

Overview

The Mailer Factory, located in ./modules/Mailer/MailerFactory.php, helps developers generate outbound mailers for the system account as well as individual user accounts. The Mailer Factory is a replacement for SugarPHPMailer which is now deprecated.

Mailers

There are two types of outbound mailers: System and User. The follow sections will outline how to use each.

System Mailer

The system outbound mailer can be set using the getSystemDefaultMailer method.

This will set the mailer to use the system outbound email account.

Example

```
$mailer = MailerFactory::getSystemDefaultMailer();
```

User Mailer

The user outbound mailer can be set using the getMailerForUser method. This will set the mailer to use the outbound email account for a specific user.

Example

```
$user = BeanFactory::getBean("Users", 1);
$mailer = MailerFactory::getMailerForUser($user);
```

Populating the Mailer

Setting the Subject

To set the email subject, use the setSubject method. It accepts a plain text string.

Example

```
$mailer->setSubject("Test Mail Subject");
```

Setting the Body

Depending on your email type, you can use the setTextBody and/or setHtmlBody methods respectively to populate the content of the email body.

Example

```
// Text Body
$mailer->setTextBody("This is a text body message");

// HTML Body
$mailer->setHtmlBody("This is an <b>HTML</b> body message. <br> You ca
n use html tags.");
```

Note: The email HTML body is not necessary if you have populated the text body.

Adding Recipients

To add recipients to your email, you can use the addRecipientsTo, addRecipientsCc, or addRecipientsBcc methods . These methods require an EmailIdentity object as a parameter.

Example

```
$mailer->addRecipientsTo(new EmailIdentity('user1@yourcompany.crm', 'U
ser 1'));
$mailer->addRecipientsCc(new EmailIdentity('user2@yourcompany.crm', 'U
ser 2'));
$mailer->addRecipientsBcc(new EmailIdentity('user3@yourcompany.crm', '
User 3'));
```

Clearing Recipients

You can clear the current recipients specified in the mailer by using the clearRecipients method.

Example

```
$to = true;
$cc = true;
$bcc = true;

$mailer->clearRecipients($to, $cc, $bcc);
```

Adding Attachments

To add attachments, use the addAttachment method.

Example

```
$path = "/path/to/your/document";
$mailer->addAttachment(new Attachment($path));
```

Sending Emails

Once your email is populated, you can send it using the send method. The send method will return the content of the mail. If the Mailer Factory experiences an error, it will throw an exception. It is highly recommended to use a try and catch when sending emails.

Example

```
$mailSubject = "Test Mail Subject";
$mailHTML = "<h1>SugarCRM</h1><br> Test body message";
$mailTo = array(
    0 => array(
        'name' => 'Test User',
        'email' => 'test@yourcompany.crm',
    ),
    1 => array(
        'name' => 'Other Recipient',
        'email' => 'email@addres'
    )
);

$mailAttachment = "/path/to/pdf/files/document.pdf";

try {
    $mailer = MailerFactory::getSystemDefaultMailer();
    $mailTransmissionProtocol = $mailer->getMailTransmissionProtocol()
;
    $mailer->setSubject($mailSubject);
    $body = trim($mailHTML);
    $textOnly = EmailFormatter::isTextOnly($body);
    if ($textOnly) {
        $mailer->setTextBody($body);
    } else {
        $textBody = strip_tags(br2nl($body)); // need to create the pl
ain-text part
        $mailer->setTextBody($textBody);
        $mailer->setHtmlBody($body);
    }
    $mailer->clearRecipients();
    foreach ($mailTo as $mailTo) {
        $mailer->addRecipientsTo(new \EmailIdentity($mailTo['email'],
$mailTo['name']));
    }
    $mailer->addAttachment(new \Attachment($mailAttachment));
    $result = $mailer->send();
    if ($result) {
```

```
        // $result will be the body of the sent email
    } else {
        // an exception will have been thrown
    }
} catch (MailerException $me) {
    $message = $me->getMessage();
    switch ($me->getCode()) {
        case \MailerException::FailedToConnectToRemoteServer:
            $GLOBALS["log"]->fatal("BeanUpdatesMailer :: error sending
 email, system smtp server is not set");
            break;
        default:
            $GLOBALS["log"]->fatal("BeanUpdatesMailer :: error sending
 e-mail (method: {$mailTransmissionProtocol}), (error: {$message})");
            break;
    }
}
```

Email Addresses

Overview

Recommended approaches when working with email addresses in Sugar.

Client Side

Recommended approaches when accessing email addresses in Sugar from a client.

Sidecar

Sidecar is the JavaScript UI framework that users interact within their browsers.

Fetching Email Addresses in Sidecar

In Sidecar, the email field will return an array of email addresses and their properties for the record. Given the model, you can fetch it using:

```
  var emailAddresses = model.get('email');
```

Note: In the past, developers could use model.get("email1") to fetch the primary email address. While this currently does work, these legacy email fields are deprecated and may be subject to removal in an upcoming Sugar release.

Fetching a Primary Email Address in Sidecar

To fetch the primary email address for a bean, you can use app.utils.getPrimaryEmailAddress():

```
  var primaryEmailAddress = app.utils.getPrimaryEmailAddress(model);
```

Fetching an Email Address by Properties in Sidecar

To fetch an email address based on properties such as invalid_email, you can use app.utils.getEmailAddress(). This function will return the first email address that matches the options or an empty string if not found. An example is shown below:

```
  var emailAddress = app.utils.getEmailAddress(model, {invalid_email:
true});
```

If you have complex filtering rules, you can use _.find() to fetch an email address:

```
  var emailAddress = _.find(model.get('email'), function(emailAddress)
{
    if (emailAddress.invalid_email == true) {
        return emailAddress;
    }
});
```

Validating Email Addresses in Sidecar

To validate an email address, you can use app.utils.isValidEmailAddress():

```
  var isValid = app.utils.isValidEmailAddress(emailAddress);
```

Note: This function is more permissive and does not conform exactly to the RFC standards used on the server. As such, the email address will be validated again on the server when the record is saved, which could still fail validation. More information can be found in the email address validation section.

Iterating Email Address in Sidecar

To iterate through email addresses on a model, you can use _.each():

```
  _.each(model.get('email'), function(emailAddress) {
    console.log(emailAddress.email_address);
});
```

Updating Email Addresses in Sidecar

This section covers how to manipulate the email addresses for a model.

Adding Email Addresses in Sidecar

In Sidecar, you can add email addresses to a model using the custom function below:

```
  function addAddress(model, email) {
    var existingAddresses = model.get('email') ? app.utils.deepCopy(mo
del.get('email')) : [],
        dupeAddress = _.find(existingAddresses, function(address){
            return (address.email_address === email);
        }),
        success = false;

    if (_.isUndefined(dupeAddress)) {
        existingAddresses.push({
            email_address: email,
            primary_address: (existingAddresses.length === 0),
            opt_out: app.config.newEmailAddressesOptedOut || false
        });
        model.set('email', existingAddresses);
        success = true;
    }

    return success;
}
```

Removing Email Addresses in Sidecar

In Sidecar, you can remove email addresses from a model using the custom function below:

```
  function removeAddress(model, email) {
    var existingAddresses = app.utils.deepCopy(model.get('email'));
    var index = false;
    _.find(existingAddresses, function(emailAddress, idx){
```

```
        if (emailAddress.email_address === email) {
            index = idx;
            return true;
        }
    });
    var primaryAddressRemoved = false;

    if (index !== false) {
        primaryAddressRemoved = !!existingAddresses[index]['primary_ad
dress'];
    }

    //Reject this index from existing addresses
    existingAddresses = _.reject(existingAddresses, function (emailInf
o, i) { return i == index; });

    // If a removed address was the primary email, we still need at le
ast one address to be set as the primary email
    if (primaryAddressRemoved) {
        //Let's pick the first one
        var address = _.first(existingAddresses);
        if (address) {
            address.primary_address = true;
        }
    }

    model.set('email', existingAddresses);
    return primaryAddressRemoved;
}
```

Server Side

Recommended approaches when accessing email addresses in Sugar from the server.

SugarBean

The SugarBean is Sugars PHP core object model.

Fetching Email Addresses Using the SugarBean

Using the SugarBean, the $bean->emailAddress->addresses property will return an array of email addresses and its properties. The $bean->emailAddress property

makes use of the EmailAddress class which is located in ./modules/EmailAddresses/EmailAddress.php. An example is shown below:

```
  $emailAddresses = $bean->emailAddress->addresses;
```

Fetching a Primary Email Address Using the SugarBean

To fetch the primary email address for a bean, you can use $bean->emailAddress->getPrimaryAddress():

```
  $primaryEmailAddress = $bean->emailAddress->getPrimaryAddress($bean)
;
```

Another alternative is to use the email_addresses_primary relationship:

```
  $primaryEmailAddress = false;
if ($this->load_relationship('email_addresses_primary')) {
    $relatedBeans = $this->email_addresses_primary->getBeans();
    if (!empty($relatedBeans)) {
        $primaryEmailAddress = current($relatedBeans);
    }
}
```

You may also choose to iterate the email address list with a foreach(). An example function is shown below:

```
  function getPrimaryEmailAddress($bean)
{
    foreach ($bean->emailAddress->addresses as $emailAddress) {
        if ($emailAddress['primary_address'] == true) {
            return $emailAddress;
        }
    }

    return false;
}
```

Fetching an Email Address by Properties Using the SugarBean

To fetch an email address based on properties such as invalid_email, you can use a foreach():

```
  $result = false;
foreach ($bean->emailAddress->addresses as $emailAddress) {
    if ($emailAddress['invalid_email']) {
        $result = $emailAddress;
        break;
    }
}
```

Validating Email Addresses Using the SugarBean

To validate an email address, you can use $bean->emailAddress->isValidEmail():

```
  $isValid = $bean->emailAddress->isValidEmail($emailAddress);
```

Note: The EmailAddress::isValidEmail method leverages the PHPMailer library bundled with Sugar, specifically the PHPMailer::validateAddress method, which validates the address according to the RFC 5321 and RFC 5322 standards. More information can be found in the email address validation section.

Iterating Email Addresses Using the SugarBean

To iterate through email addresses on a bean, you can use foreach():

```
  foreach ($bean->emailAddress->addresses as $emailAddress) {
    $GLOBALS['log']->info($emailAddress['email_address']);
}
```

Fetching Beans by Email Address Using the SugarBean

To fetch all beans related to an email address you can use getBeansByEmailAddress():

```
  $beans = $bean->emailAddress->getBeansByEmailAddress($emailAddress);
```

If you don't have a bean available, you may choose to create a new EmailAddress object:

```
  $sea = BeanFactory::newBean('EmailAddresses');
$sea->getBeansByEmailAddress($emailAddress);
```

Updating Email Addresses Using the SugarBean

This section covers how to manipulate the email addresses for a bean.

Adding Email Addresses Using the SugarBean

To add an email address to the bean, you can use $bean->emailAddress->addAddress():

```
  $bean->emailAddress->addAddress('address@sugar.crm');
```

Note: The addAddress() function has additional parameters that are defaulted for determining if the email address is a primary, reply to, invalid, or opted out email address. You can also specify an id for the email address and whether or not the email address should be validated.

```
  function addAddress($addr, $primary=false, $replyTo=false, $invalid=
false, $optOut=false, $email_id = null, $validate = true)
```

Removing Email Addresses Using the SugarBean

To remove an email address you can use $bean->emailAddress->removeAddress():

```
  $bean->emailAddress->removeAddress('address@sugar.crm');
```

PDF Templates

Using the Sugar PDF templates, you can reference the primary email address of the bean using:

```
  {$fields.email_addresses_primary.email_address}
```

REST API

Sugar comes out of the box with an API that can be called from custom applications utilizing the REST interface. The API can be used to mass create and update records in Sugar with external data. For more information on the REST API in Sugar, please refer to the Web Services documentation.

Creating Email Addresses Using the REST API

When creating records in Sugar through the API, modules with relationships to email addresses can utilize the email link field to specify email addresses for a record. Using the email link field, you can specify multiple email addresses to assign to the record. You may specify the following additional information regarding each email address being added:

Property	Description
invalid_email	Specify this email address as being invalid
opt_out	Specify this email address as being opted out. More information on opt-outs can be found in the Emails documentation.
primary_address	Specify this email address as the primary

Using the **/<module> POST** endpoint, you can send the following JSON payload to create a contact record with multiple email addresses using the email link field:
POST URL: http://<site url>/rest/v<version>/Contacts

```
{
 "first_name":"Rob",
 "last_name":"Robertson",
 "email":[
    {
       "email_address":"rob.robertson@sugar.crm",
       "primary_address":"1",
       "invalid_email":"0",
       "opt_out":"0"
    },
    {
       "email_address":"rob@sugar.crm",
       "primary_address":"0",
       "invalid_email":"0",
       "opt_out":"1"
    }
 ]
```

```
}
```

For more information on the **/<module>/:record POST** endpoint, you can refer to your instance's help documentation found at:

```
http://<site url>/rest/v<version>/help
```

Or you can reference the <module> POST PHP example.

Updating Email Addresses Using the REST API

When updating existing records in Sugar through the API, modules with relationships to email addresses can use the email link field to specify email addresses for a record. Using the email link field, you can specify multiple email addresses to update the record with. You may specify the following additional information regarding each email address being added:

- **invalid_email** : Specify this email address as being invalid
- **opt_out** : Specify this email address as being opted out
- **primary_address** : Specify this email address as the primary

Using the **/<module>/:record PUT** endpoint, you can send the following JSON payload to update a contact record with multiple email addresses:
PUT URL: http://<site url>/rest/v<version>/Contacts/<record id>

```
  {
   "email":[
      {
         "email_address":"rob.robertson@sugar.crm",
         "primary_address":"1",
         "invalid_email":"0",
         "opt_out":"0"
      },
      {
         "email_address":"rob@sugar.crm",
         "primary_address":"0",
         "invalid_email":"0",
         "opt_out":"1"
      }
   ]
}
```

For more information on the **/<module>/:record PUT** endpoint, you can refer to your instance's help documentation found at:

```
http://<site url>/rest/v<version>/help
```

You want to reference the <module>/:record PUT PHP example.

Legacy Email Fields

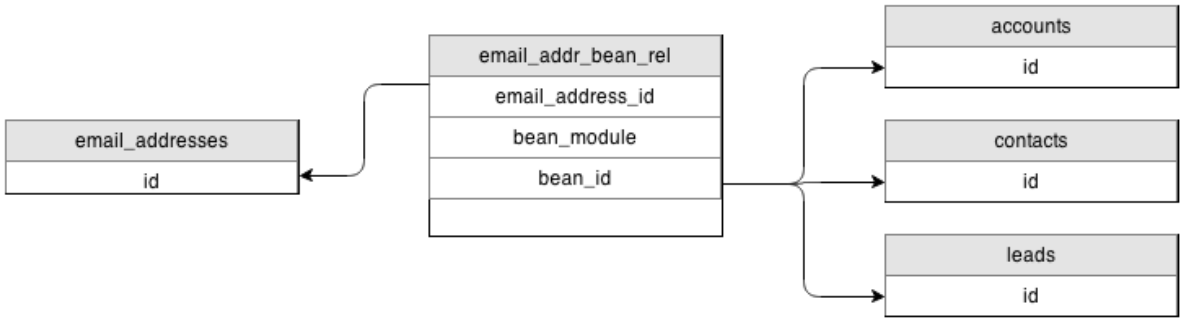
The legacy email fields in Sugar are deprecated and may be subject to removal in an upcoming Sugar release. When using the email1 field, the default functionality is to import the email address specified as the primary address.

Legacy Email Field	Description
email1	The text value of primary email address. Does not indicate if the email address is valid or opted-out.
email2	The text value of first non-primary email address. Does not indicate if the email address is valid or opted-out.

Note: For importing multiple email addresses with properties, you will need to use the email link field.

Creating Email Addresses Using Direct SQL

When importing records into Sugar directly via the database, it is important that you understand the data structure involved before loading data. Email addresses are not stored directly on the table for the module being imported in but are related via the email_addr_bean_rel table.



The table structure for email addresses can be seen from the database via the

following SQL statement:

```
  SELECT
 email_addr_bean_rel.bean_id,
 email_addr_bean_rel.bean_module,
 email_addresses.email_address
FROM email_addr_bean_rel
INNER JOIN email_addresses
 ON email_addresses.id = email_addr_bean_rel.email_address_id
 AND email_addr_bean_rel.deleted = 0
WHERE email_addresses.deleted = 0;
```

Checking for Duplicates

Email addresses can become duplicated in Sugar from a variety of sources including API calls, imports, and from data entry. There are a few ways to have the system check for duplicate contact records, but not many of those methods work for checking email addresses for duplicates. The following section will demonstrate how to find and clean up duplicate email addresses using SQL.

The following SQL query can be used to locate if any email addresses are being used against more than one bean record within Sugar:

```
  SELECT
 email_addresses.email_address,
 COUNT(*) AS email_address_count
FROM email_addr_bean_rel
INNER JOIN email_addresses
 ON email_addresses.id = email_addr_bean_rel.email_address_id
 AND email_addr_bean_rel.deleted = 0
WHERE email_addresses.deleted = 0
GROUP BY email_addresses.email_address
HAVING COUNT(*) > 1;
```

Note: If you convert a Lead record to a Contact record, both the Lead and the Contact will be related to the same Email Address and will return as having duplicates in this query. You can add the following line to the WHERE clause to filter the duplicate check down to only one bean type:

```
  AND email_addr_bean_rel.bean_module = 'Contacts'
```

Email addresses can not only be duplicated in the system but can occasionally be missing critical data. Each bean record with an email address assigned to it should have an email address designated the primary. The following query will locate any bean records that have at least one email address, where there is not an email address designated as the primary:

```
  SELECT
 email_addr_bean_rel.bean_module,
 email_addr_bean_rel.bean_id,
 COUNT(*) AS email_count,
 COUNT(primary_email_addr_bean_rel.id) AS primary_email_count
FROM email_addr_bean_rel
LEFT JOIN email_addr_bean_rel primary_email_addr_bean_rel
 ON primary_email_addr_bean_rel.bean_module = email_addr_bean_rel.bean
_module
 AND primary_email_addr_bean_rel.bean_id = email_addr_bean_rel.bean_id
 AND primary_email_addr_bean_rel.primary_address = '1'
 AND primary_email_addr_bean_rel.deleted = '0'
WHERE email_addr_bean_rel.deleted = '0'
GROUP BY email_addr_bean_rel.bean_module,
 email_addr_bean_rel.bean_id
HAVING primary_email_count < 1;
```

Note: If you are a SugarCloud customer, you can open up a case with Sugar Support to have this query run for you.

Removing Duplicates

If it is determined you have duplicate email addresses being used in your system, you can use the following query to clean up the records:

```
  START TRANSACTION;
CREATE
 TABLE email_addr_bean_rel_tmp
SELECT
 *
FROM email_addr_bean_rel
WHERE deleted = '0'
GROUP BY email_address_id,
 bean_module,
 bean_id
ORDER BY primary_address DESC;
TRUNCATE TABLE email_addr_bean_rel;
INSERT INTO email_addr_bean_rel
```

```
 SELECT
 *
 FROM email_addr_bean_rel_tmp;
SELECT
 COUNT(*) AS repetitions,
 date_modified,
 bean_id,
 bean_module
FROM email_addr_bean_rel
WHERE deleted = '0'
GROUP BY bean_id,
 bean_module,
 email_address_id
HAVING repetitions > 1;
COMMIT;
```

Note: If you are a SugarCloud customer, you can open up a case with Sugar Support to have this query run for you.

Logging

Overview

There are two logging systems implemented in the Sugar application: SugarLogger and PSR-3. PSR-3 is Sugar's preferred logger solution and should be used going forward.

PSR-3

PSR-3 compliant logging solution has been implemented based on PHP Monolog.

Log Levels

Log Level	Description
Debug	Logs events that help in debugging the application
Info	Logs informational messages and database queries
Warning	Logs potentially harmful events
Notice	Logs messages for deprecated methods

	that are still in use.
Error	Logs error events in the application
Alert	Logs severe error events that may cause the application to abort. This is the default and recommended level.
Critical	Logs events that may compromise the security of the application
Off	Turns off all logging

When you specify a logging level, the system will record messages for the specified level as well as all higher levels. For example, if you specify "Error", the system records all Error, Fatal, and Security messages. More information on logging levels can be found in the logger level configuration documentation.

Considerations

- When you are not troubleshooting Sugar, the log level should be set to Fatal in Admin > System Settings > Logger Settings to ensure that your environment is not wasting unnecessary resources to write to the Sugar log.

Logging Messages

The PSR-3 implementation in Sugar can also be used to log messages to the Sugar Log file. You can utilize the implementation to log to the Sugar log file using the default channel or you can specify your own custom channel if you want further control over when your custom logs should be displayed.

```
  use \Sugarcrm\Sugarcrm\Logger\Factory;

//Get the default Logger
$Logger = Factory::getLogger('default');
$Logger->debug('Debug level message');
$Logger->info('Info level message');
$Logger->notice('Notice level message');
$Logger->warning('Warning level message');
$Logger->error('Error level message');
$Logger->critical('Critical level message');
$Logger->alert('Alert level message');
$Logger->emergency('Emergency level message');

//Get a custom Log Channel
$Logger = Factory::getLogger('my_logger');
```

Note: For more information on using custom channels, adding custom log handlers and processors see the PSR-3 Logger documentation.

SugarLogger

The SugarLogger class, located in ./include/SugarLogger/SugarLogger.php, allows for developers and system administrators to log system events to a log file. Sugar then determines which events to write to the log based on the system's Log Level. This can be set in Admin > System Settings.

Log Levels

Log Level	Description
Debug	Logs events that help in debugging the application
Info	Logs informational messages and database queries
Warn	Logs potentially harmful events
Deprecated	Logs messages for deprecated methods that are still in use.
Error	Logs error events in the application
Fatal	Logs severe error events that may cause the application to abort. This is the default and recommended level.
Security	Logs events that may compromise the security of the application
Off	Logging is turned off

When you specify a logging level, the system will record messages for the specified level as well as all higher levels. For example, if you specify "Error", the system records all Error, Fatal, and Security messages. More information on logging levels can be found in the logger level documentation.

Considerations

- When you are not troubleshooting Sugar, the log level should be set to Fatal in Admin > System Settings > Logger Settings to ensure that your environment is not wasting unnecessary resources to write to the Sugar log.

Logging Messages

Using $GLOBALS['log']

How to log messages using $GLOBALS['log'] in the system.

```
  $GLOBALS['log']->debug('Debug level message');
$GLOBALS['log']->info('Info level message');
$GLOBALS['log']->warn('Warn level message');
$GLOBALS['log']->deprecated('Deprecated level message');
$GLOBALS['log']->error('Error level message');
$GLOBALS['log']->fatal('Fatal level message');
$GLOBALS['log']->security('Security level message');
```

For more information on the implementation, please refer to the SugarLogger documentation.

Using LoggerManager

How to log messages using the LoggerManager.

```
  $Logger = \LoggerManager::getLogger();
$Logger->debug('Debug level message');
$Logger->info('Info level message');
$Logger->warn('Warn level message');
$Logger->deprecated('Deprecated level message');
$Logger->error('Error level message');
$Logger->fatal('Fatal level message');
$Logger->security('Security level message');
```

For more information on the implementation, please refer to the SugarLogger documentation.

Log Rotation

The SugarLogger will automatically rotate the logs when the logger.file.maxSize configuration setting has been met or exceeded. When this happens, the Sugar log will be renamed with an integer. For example, if the Sugar log was named "sugarcrm.log, it will then be renamed "sugarcrm_1.log". The next log rotation after that would create "sugarcrm_2.log". This will occur until the logger.file.maxLogs configuration setting has been met. Once met, the log rollover will start over.

Debugging Messages with _ppl()

When developing, it may be beneficial for a developer to use _ppl() method to log a message to the Sugar log. The _ppl() method handles converting Objects, so you can quickly dump an entire object to the log while testing during development.

```
  _ppl('Debugging message');
```

This will write a message to the Sugar log that defines the message and file location. An example is shown below:

```
  ------------------------------ _ppLogger() output start ------------
-----------------

Debugging message

------------------------------ _ppLogger() output end ----------------
-------------

------------------------------ _ppLogger() file: myFile.php line#: 5--
--------------------------
```

Note: It is important that you remove _ppl() from your code for production use as it will affect system performance.

Creating Custom Loggers

Custom Loggers

Custom loggers, defined in ./custom/include/SugarLogger/, can be used to write log entries to a centralized application management tool or to write messages to a developer tool such as FirePHP.

To do this, you can create a new instance class that implements the LoggerTemplate interface. The following is an example of how to create a FirePHP logger.

./custom/include/SugarLogger/FirePHPLogger.php.

```
<?php
```

```
// change the path below to the path to your FirePHP install
require_once('/path/to/fb.php');

class FirePHPLogger implements LoggerTemplate
{
    /** Constructor */
    public function __construct()
    {
        if (
            isset($GLOBALS['sugar_config']['logger']['default'])
            && $GLOBALS['sugar_config']['logger']['default'] == 'FireP
HP'
        )
        {
            LoggerManager::setLogger('default','FirePHPLogger');
        }
    }

    /** see LoggerTemplate::log() */
    public function log($level, $message)
    {
        // change to a string if there is just one entry
        if ( is_array($message) && count($message) == 1 )
        {
            $message = array_shift($message);
        }

        switch ($level)
        {
            case 'debug':
                FB::log($message);
                break;
            case 'info':
                FB::info($message);
                break;
            case 'deprecated':
            case 'warn':
                FB::warn($message);
                break;
            case 'error':
            case 'fatal':
            case 'security':
                FB::error($message);
                break;
        }
```

```
    }
}
```

The only method that needs to be implemented by default is the log() method, which writes the log message to the backend. You can specify which log levels this backend can use in the constructor by calling the LoggerManager::setLogger() method and specifying the level to use for this logger in the first parameter; passing 'default' makes it the logger for all logging levels.

You will then specify your default logger as 'FirePHP' in your ./config_override.php file.

```
$sugar_config['logger']['default'] = 'FirePHP';
```

PSR-3 Logger

`Monolog\Handler\HandlerInterface`

Overview

As of Sugar 7.9, a PSR-3 compliant logging solution has been implemented based on PHP Monolog. Accessing the PSR-3 Logger Objects can be done by via the \Sugarcrm\Sugarcrm\Logger namespace. Currently, this logging implementation is only used in a few areas of the system to allow for more in-depth logging in those areas, see the Usage section for more details.

Architecture

The PSR-3 Logging solution is found in ./src/Logger directory, which is mapped to the \Sugarcrm\Sugarcrm\Logger namespace in the application. The following outlines the architecture and current implementations of those core objects.

Factory

The Logger Factory Object, namespaced as \Sugarcrm\Sugarcrm\Logger\Factory, is a singleton factory that creates and manages all Loggers currently being used by the system. The Logger Factory uses 'channels' to allow for multiple Loggers to be utilized and configured independently of default log level set for the system. It also allows for each channel to use different handlers, formatters, and processors. Check out the Usage section below for further details on configuration and usage.

Methods

getLogger($channel)

Returns the requested channels \Psr\Log\LoggerInterface implementation

Arguments

Name	Type	Description
$channel	String	The channel for which you are logging against

Example

```
  use \Sugarcrm\Sugarcrm\Logger\Factory;

$Logger = Factory::getLogger('default');
```

Handlers

Handlers are the primary object used to manage the logs. They control how logs are submitted and where the logs will go. Since the PSR-3 Logging solution is based on Monolog, there are a number of default Handlers included in Sugar, however, they are not set up in the Sugar architecture to be utilized right off the bat. Sugar utilizes a Factory implementation for the handlers so that the Logger Factory Object can build out specific Handlers for each channel based on the Sugar Config.

Factory Interface

The Factory Interface for Handlers is used to implement a Logging Handler, so that the Logger Factory can build out the configured handler for a channel, without a lot of work from external developers. The Handler Factory Interface is located in ./src/Logger/Handlers/Factory.php or in code at the \Sugarcrm\Sugarcrm\Logger\Handler\Factory namespace.

Methods

There is only one method to implement for a Handler Factory, which is the create() method. This will contain the necessary Logic to set up the Logger Handler

create($level, $config)

Arguments

Name	Type	Description

$level	String	The log level the Logger Handler will operate at
$config	Array	The config parameters set for the handler in the Sugar config file

Returns

The Monolog\Handler\HandlerInterface implementation

Implementations

By default, Sugar only comes with a single Handler implementation, which is the File Handler implementation. This is used to log directly to the ./sugarcrm.log file to mirror the functionality of the previous logging framework. For information on adding custom handlers, or implementing the built-in Monolog handlers inside of Sugar PSR-3 Logging framework, see the Customization section below.

Configuration

To configure the default logging handler for Sugar the following configuration setting is used:

```
  $sugar_config['logger']['handler'] = '<handler>';
```

You can also configure a different Logging Handler or Handlers for a specific channel:

```
  //Single Handler
$sugar_config['logger']['channels']['test_channel']['handlers'] = '<ha
ndler>';

//Multiple Channels
$sugar_config['logger']['channels']['test_channel']['handlers'] = arra
y('<handler1>','<handler2>');
```

To pass configuration properties to a Handler your configuration will need to look as follows:

```
  //For system handler
$sugar_config['logger']['handlers']['<handler>']['host'] = '127.0.0.1'
;
```

```
$sugar_config['logger']['handlers']['<handler>']['port'] = 12201;

//For channel handlers
$sugar_config['logger']['channels']['test_channel']['handlers']['<hand
ler>']['host'] = '127.0.0.1';
$sugar_config['logger']['channels']['test_channel']['handlers']['<hand
ler>']['port'] = 12201;
$sugar_config['logger']['channels']['test_channel']['handlers']['<hand
ler>']['level'] = 'debug';
```

Note: For more information, please refer to the logger.channels.channel.handlers documentation.

Formatters

Formatters are a component of the Handler Object. By default Sugar only comes with a single Formatter, which is the BackwardCompatibleFormatter used by the File Handler, which simply assures that Log messages are consistent with the legacy Logging functionality. Formatters are used and built out in accordance with the Monolog framework, and under the majority of circumstances, building out a custom formatter is not necessary. For more information on Formatters, you can review the Monolog repository.

Processors

Processors provide a way to add more information to Log messages, without having to hard code this information inside the Handler, so that it can be used only when necessary. For example, Sugar's PSR-3 Logging implementation provides two default Processors that can be enabled for a given channel or handler via configuration. These Processors provide functionality such as adding a stack trace or the web request information to the log message to provide further debugging context.

Factory Interface

The Factory Interface for processors is used to implement a Logging Processor, so that the Logger Factory can build out the configured handler for a channel, without a lot of work from external developers. The Processor Factory Interface is located in ./src/Logger/Processor/Factory.php or in code at \Sugarcrm\Sugarcrm\Logger\Handler\Factory namespace.

Methods

There is only one method to implement for a Processor Factory, which is

the create() method. This method will contain the necessary Logic to set up the Processor object.

create($config)

Arguments

Name	Type	Description
$config	Array	The config parameters set for the processor in the Sugar config file

Returns

Callable - See Customization section for an example of implementation, otherwise review the included Processor Factory implementations in code in ./src/Logger/Processor/Factory/.

Implementations

By default, Sugar comes with two Processor implementations, \Sugarcrm\Sugarcrm\Logger\Processor\BacktraceProcessor and \Sugarcrm\Sugarcrm\Logger\Processor\RequestProcessor.

BacktraceProcessor

As the name implies, the BacktraceProcessor appends a backtrace or stack trace to the log message output, to easily trace where and how the log message came from. The Factory implementation for this Processor lives in ./src/Logger/Processor/Factory/Backtrace.php, and is referenced in the sugar config as backtrace.

The following shows an example of the output that occurs when utilizing this processor:

```
  Fri Mar 23 09:24:19 2018 [90627][1][FATAL] <log message>; Call Trace
: \n0: /var/www/Ent/71110/custom/SugarQueryLogger.php:43 - Sugarcrm\Su
garcrm\Logger\BackwardCompatibleAdapter::fatal()\n2: /var/www/Ent/7111
0/include/utils/LogicHook.php:270\n3: /var/www/Ent/71110/include/utils
/LogicHook.php:160 - LogicHook::process_hooks()\n4: /var/www/Ent/71110
/data/SugarBean.php:6684 - LogicHook::call_custom_logic()\n5: /var/www
/Ent/71110/data/SugarBean.php:3317 - SugarBean::call_custom_logic()\n6
: /var/www/Ent/71110/clients/base/api/FilterApi.php:632 - SugarBean::f
etchFromQuery()\n7: /var/www/Ent/71110/clients/base/api/FilterApi.php:
397 - FilterApi::runQuery()\n8: /var/www/Ent/71110/include/api/RestSer
vice.php:257 - FilterApi::filterList()\n9: /var/www/Ent/71110/api/rest
.php:23 - RestService::execute()
```

Note: when viewing complex logs, you can use the following to print log entries in a more human readable format:

```
cat sugarcrm.log | sed s/\\n/\n/g
```

RequestProcessor

The RequestProcessor appends Session properties to the Log message that would help identify the request that caused the log message. The Factory implementation for this Processor lives in ./src/Logger/Processor/Factory/Request.php, and is referenced in the sugar config as request.

The following list of session properties are appended to the log:

- User ID
- Client ID (OAuth2 Client)
- Platform

The following shows an example of the output that occurs when utilizing this processor:

```
Mon Mar 26 09:48:58 2018 [5930][1][FATAL] <log message>; User ID=1;
Client ID=sugar; Platform=base
```

Configuration

To configure the default Logging Handler for Sugar to utilize the built-in processors:

```
//Configure a single Processor
$sugar_config['logger']['channels']['default']['processors'] = 'backtr
ace';

//Configure multiple Processors
$sugar_config['logger']['channels']['default']['processors'] = array('
backtrace','request');
```

You can also configure different channels to utilize the processors:

```
  //Single Processors
$sugar_config['logger']['channels']['<channel>']['processors'] = 'back
trace';

//Multiple Channels
$sugar_config['logger']['channels']['<channel>']['processors'] = array
('backtrace','request');
```

To pass configuration properties to a Processor your configuration will need to look as follows:

```
  $sugar_config['logger']['channels']['<channel>']['processors']['<pro
cessor>']['config_key'] = 'test_value';
```

Note: For more information, please refer to the logger.channels.channel.processors documentation.

Usage

As previously mentioned the Logger Factory allows for multiple channels to be configured independently of each other. The following examples will showcase how to use the Logger Factory to get a channel's Logger and use it in code, as well as how to configure the system to use multiple channels with different configurations.

Basic Usage

```
  use \Sugarcrm\Sugarcrm\Logger\Factory;

//Retrieve the default Logger
$DefaultLogger = Factory::getLogger('default');
$DefaultLogger->alert('This is a log message');
```

Configuring Channels

The following is an example of the ./config_override.php file that would configure two different channels at different log levels, using the logger.channel.channel.level configuration setting. These two channels would allow for portions of the code to Log messages at Debug (and higher) levels, and other portions to only log Info (and higher) levels.

```
  $config['logger']['channels']['default'] = array(
    'level' => 'alert'
);
$config['logger']['channels']['channel1'] = array(
    'level' => 'debug'
);
```

The following code example shows how to retrieve the above-configured channels and use the Logger for each channel.

```
  use \Sugarcrm\Sugarcrm\Logger\Factory;

//Retrieve the default Logger
$DefaultLogger = Factory::getLogger('default');
$DefaultLogger->info("This message will not display");

//Channel1 Logger
$Channel1Logger = Factory::getLogger('channel1');
$Channel1Logger->info("This message will display");
```

In the example above, assuming a stock system with the previously mentioned config values set in config_override.php, the default channel logger would be set to Alert level logging, and therefore would not add the info level log to the Log file, however, the channel11 Logger would add the log message to the Sugar log file, since it is configured at the info log level.

Default Channels

By default, Sugar has a few areas of the system that utilize a different channel than the default. The usage of these channels means that you can configure the log level and processors differently for those areas, without inundating the log file with lower level logs from other areas of the system.

Channel	Description
authentication	This logger channel is utilized by the AuthenticationController, and can show useful information about failed login attempts.
input_validation	This logger channel is utilized by the Input Validation framework and can display information regarding failed

	validations.
metadata	This logger channel is utilized by the MetadataManager and mainly provides information on when rebuilds occur.
rest	This channel is utilized by the RestService object.
db	This channel is utilized by the databse for query logging.

Customization

You can use custom channels where ever you would like in your customizations, and configure them as outlined above, but if you need to send logs somewhere other than the Sugar log file, you will need to build out your own Handler. The following will walk through adding a custom Logging Handler that utilizes Monologs built-in Chrome Logger.

Adding a Custom Handler

Create the following file in ./custom/src/Logger/Handler/Factory/ folder.

./custom/src/Logger/Handler/Factory/Chrome.php

```
  <?php

namespace Sugarcrm\Sugarcrm\custom\Logger\Handler\Factory;

use Monolog\Handler\ChromePHPHandler;
use Sugarcrm\Sugarcrm\Logger\Handler\Factory;

class Chrome implements Factory
{

    public function create($level, array $config)
    {
        return new ChromePHPHandler($level);
    }
}
```

Once you have the new class in place, you will need to run a Quick Repair and Rebuild so that the new class is auto-loaded correctly. Then you can configure the handler for use in the Sugar config:

```
  $sugar_config['logger']['channels']['<channel>']['handlers'] = 'Chro
me';
```

Note: For more information, please refer to the logger.channels.channel.handlers documentation.

SugarLogger

Overview

The SugarLogger is used for log reporting by the application. The following article outlines the LoggerTemplate interface as well as the LoggerManager object and explains how to programmatically use the SugarLogger.

LoggerTemplate

The LoggerManager manages those objects that implement the LoggerTemplate interface found in ./include/SugarLogger/LoggerTemplate.php.

Methods

log($method, $message)

The LoggerTemplate has a single method that should be implemented, which is the log() method.

Arguments

Name	Type	Description
$method	String	The method or level which the Logger should handle the provided message
$message	String	The logged message

Implementations

Sugar comes with two stock LoggerTemplate implementations that can be utilized for different scenarios depending on your needs. You can extend from these implementations or create your own class that implements the template as outlined in Creating Custom Loggers section.

SugarLogger

The SugarLogger class, found in ./include/SugarLogger/SugarLogger.php, is the default system logging class utilized throughout the majority of Sugar.

SugarPsrLogger

The SugarPsrLogger was added in Sugar 7.9 to accommodate the PSR-3 compliant logging implementation. This logger works just like the SugarLogger object, however it uses the Monolog implementation of Logger.

To configure the SugarPsrLogger as the default system logger, you can add the following to your configuration:

```
$sugar_config['logger']['default'] = 'SugarPsrLogger';
```

Log Level Mappings

PSR-3 defines the set of log levels that should be implemented for all PHP Applications, however, these are different from the log levels defined by the SugarLogger. Below is the list of SugarLogger log levels and their SugarPsrLogger compatible mapping. All calls to the SugarLogger log levels are mapped according to the table below. For example, when using the SugarPsrLogger class, all calls to the fatal() method will map to the Alert log level.

SugarLogger Level	SugarPsrLogger Level (PSR-3)
Debug	Debug
Info	Info
Warn	Warning
Deprecated	Notice
Error	Error
Fatal	Alert
Security	Critical

LoggerManager

The LoggerManager Object acts as a singleton factory that sets up the configured logging object for use throughout the system. This is the object stored in $GLOBALS['log'] in the majority of use cases in the system.

Methods

getLogger()

This method is used to get the currently configured Logger class.

setLevel($level)

You may find that you want to define the log level while testing your code without modifying the configuration. This can be done by using the setLevel() method.

Arguments

Name	Type	Description
$level	String	The method or level which the Logger should handle the provided message

Example

```
\LoggerManager::getLogger()->setLevel('debug');
```

Note: The use of setLevel should be removed from your code for production and it is important to note that it is restricted by package scanner as defined by the Module Loader Restrictions.

assert($message, $condition)

In your custom code you may want to submit a debug level log, should a condition not meet your expectations. You could do this with an if statement, otherwise you could just use the assert() method as it allows you to pass the debug message, and the condition to check, and if that condition is FALSE a debug level message is logged.

Arguments

Name	Type	Description
$message	String	The log message
$condition	Boolean	The condition to check

Example

```
$x = 1;
\LoggerManager::getLogger()->assert('X was not equal to 0!', $x==0)
```

wouldLog($level)

If in your customization you find that extra debugging is needed for particular area of code, and that extra work might have performance impacts on standard log levels, you can use the wouldLog() method to check the current log level before doing the extra work.

Arguments

Name	Type	Description
$level	String	The level to check against

Example

```
if (\LoggerManager::getLogger()->wouldLog('debug')) {
   //Do extra debugging
}
```

setLogger($level, $logger)

This method allows you to setup a second logger for a specific log level, rather than just using the same logger for all levels. Passing default as the level, will set the default Logger used by all levels in the system.

Arguments

Name	Type	Description
$level	String	The level to check against
$logger	String	The class name of an installed Logger

Example

```
//Set the debug level to a custom Logger
\LoggerManager::getLogger()->setLogger('debug', 'CustomLogger');

//Set all other levels to SugarPsrLogger
\LoggerManager::getLogger()->setLogger('default', 'SugarPsrLogger');
```

getAvailableLoggers()

Returns a list of the names of Loggers found/installed in the system.

Arguments

None

Example

```
$loggers = \LoggerManager::getLogger()->getAvailableLoggers();
```

getLoggerLevels()

Returns a list of the names of Loggers found/installed in the system.

Arguments

None

Example

```
$levels = \LoggerManager::getLogger()->getLoggerLevels();
```

Adding a Custom SugarLogger

Custom loggers are defined in ./custom/include/SugarLogger/, and can be used to write log entries to a centralized application management tool, to a developer tool such as FirePHP or even to a secondary log file inside the Sugar application.

The following is an example of how to create a FirePHP logger.

./custom/include/SugarLogger/FirePHPLogger.php.

```
<?php

// change the path below to the path to your FirePHP install
require_once('/path/to/fb.php');

class FirePHPLogger implements LoggerTemplate
{
    /** Constructor */
    public function __construct()
    {
        if (
            isset($GLOBALS['sugar_config']['logger']['default'])
            && $GLOBALS['sugar_config']['logger']['default'] == 'FireP
HP'
```

```
        )
        {
            LoggerManager::setLogger('default','FirePHPLogger');
        }
    }

    /** see LoggerTemplate::log() */
    public function log($level, $message)
    {
        // change to a string if there is just one entry
        if ( is_array($message) && count($message) == 1 ) {
            $message = array_shift($message);
        }

        switch ($level) {
            case 'debug':
                FB::log($message);
                break;
            case 'info':
                FB::info($message);
                break;
            case 'deprecated':
            case 'warn':
                FB::warn($message);
                break;
            case 'error':
            case 'fatal':
            case 'security':
                FB::error($message);
                break;
        }
    }
}
```

You will then specify your default logger as 'FirePHP' in your ./config_override.php file.

```
$sugar_config['logger']['default'] = 'FirePHP';
```

Logic Hooks

Overview

The Logic Hook framework allows you to append actions to system events such as when creating, editing, and deleting records.

Hook Definitions

A logic hook definition file defines the various actions to execute when an event is triggered. It is important to note that there are various ways to implement a logic hook. The following sections describe the different ways to implement a logic hook and when to use each.

Methodologies

Logic hook definitions can pertain to a specific module or to the entire application. Either way, you must decide if the logic hook definition will be implemented as an extension of or directly to the module or application. The following sections explain the difference between these methodologies.

Module Extension Hooks

Module extension hooks are located in ./custom/Extension/modules/<module>/Ext/LogicHooks/ and allow a developer to define hook actions that will be executed for the specified events in a given module. Extensions are best used when creating customizations that may be installed in various environments. They help prevent conflicting edits that occur when modifying ./custom/modules/<module>/logic_hooks.php. More information can be found in the Logic Hooks extension section.

Module Hooks

Module hooks are located in ./custom/modules/<module>/logic_hooks.php and allow a developer to define hook actions that will be executed for the specified events in a given module. This path can be used for private development, however, it is not recommended for use with distributed modules and plugins. For distributed code, please refer to using module extensions.

Application Extension Hooks

Application extension hooks are located in ./custom/Extension/application/Ext/LogicHooks/ and allow a developer to define hook actions that will be executed for all specified application-level events using the extension framework. More information can be found in the Logic Hooks extension section.

Application Hooks

Application hooks are located in ./custom/modules/logic_hooks.php and allow a developer to define hook actions that will be executed for the specified events in all modules. This path can be used for private development, however, it is not recommended for use with distributed modules and plugins. For distributed code, please refer to using application extensions.

Definition Properties

All logic hooks must have a $hook_version and $hook_array variable defined. The following sections cover each required variable.

hook_version

All logic hook definitions will define a $hook_version. This determines the version of the logic hook framework. Currently, the only supported hook version is 1.

```
$hook_version = 1;
```

hook_array

The logic hook definition will also contain a $hook_array. This defines the specific action to execute. The hook array will be defined as follows:

Name	Type	Description
event_name	String	The name of the event to append the action to
process_index	Integer	The order of action execution
description	String	A short description of the hook action
file_path	String	The path to the logic hook file in the ./custom/ directory, or null if using namespaces
class_name	String	The name of the logic hook action class including any namespaces
method_name	String	The name of the logic hook action method

Your definition should resemble the code block below:

```
<?php

  $hook_version = 1;

  $hook_array['<event_name>'][] = array(
      <process_index>, //Integer
      '<description>', //String
      '<file_path>', //String or null if using namespaces
      '<class_name>', //String
      '<method_name>', //String
  );
```

Hook Method

The hook action class can be located anywhere you choose. We recommended grouping the hooks with the module they are associated with in the ./custom/ directory. You can create a hook action method either with a namespace or without.

Namespaced Hooks

As of 7.7, developers can create namespaced logic hooks. When using namespaces, the file path in ./custom/ will be automatically built out when using the corresponding namespace. The filename defining the class must match the class name exactly to allow the autoloader to find the class definition.

Namespace	**File Path**
Sugarcrm\Sugarcrm\custom	./custom/
Sugarcrm\Sugarcrm\custom	./custom/src/
Sugarcrm\Sugarcrm\custom\any\path	./custom/any/path/
Sugarcrm\Sugarcrm\custom\any\path	./custom/src/any/path/

./custom/Extension/modules/Accounts/Ext/LogicHooks/<file>.php

```
<?php

  $hook_array['before_save'][] = array(
      1,
      'Hook description',
      null,
      'Sugarcrm\\Sugarcrm\\custom\\modules\\Accounts\\className',
      'methodName'
```

```
    );
?>
```

./custom/modules/Accounts/className.php

```
  <?php

namespace Sugarcrm\Sugarcrm\custom\modules\Accounts;

    class className
    {
        function methodName($bean, $event, $arguments)
        {
            //logic
        }
    }
?>
```

The logic hook method signature may contain different arguments depending on the hook event you have selected.

Hooks without Namespaces

./custom/Extension/modules/Accounts/Ext/LogicHooks/<file>.php

```
  <?php

    $hook_array['before_save'][] = array(
        1,
        'Hook description',
        'custom/modules/Accounts/customLogicHook.php',
        'className',
        'methodName'
    );
?>
```

./custom/modules/Accounts/customLogicHook.php

```
  <?php
```

```
    class className
    {
        function methodName($bean, $event, $arguments)
        {
            //logic
        }
    }
?>
```

The logic hook method signature may contain different arguments depending on the hook event you have selected.

Considerations

When using logic hooks, keep in mind the following best practices:

- As of PHP 5.3, objects are automatically passed by reference. When creating logic hook signatures, do not append the ampersand (&) to the $bean variable. Doing this will cause unexpected behavior.
- There is no hook that fires specifically for the ListView, DetailView or EditView events. Instead, use either the process_record or after_retrieve logic hooks.
- In order to compare new values with previous values, use fetched_row to determine whether a change has occurred:

  ```
  if ($bean->fetched_row['{field}'] != $bean->{field})
  {
      //logic
  }
  ```

- Make sure that the permissions on the logic_hooks.php file and the class file that it references are readable by the web server. Otherwise, Sugar will not read the files and your business logic extensions will not work. For example, *nix developers who are extending the Tasks logic should use the following command for the logic_hooks file and the same command for the class file that will be called:

  ```
  chmod +r ./custom/modules/Tasks/logic_hooks.php
  ```

- Make sure that the entire ./custom/ directory is writable by the web server or else the logic hooks code will not work properly.

Application Hooks

Application hooks execute logic when working with the global application.

---

after_entry_point

Overview

The after_entry_point application hook executes at the start of every request.

Definition

```
function after_entry_point($event, $arguments){}
```

Arguments

Name	Type	Description
event	String	The current event
arguments	Array	Additional information related to the event (typically empty)

Considerations

- The after_entry_point hook is a global logic hook where the logic hook reference must be placed in ./custom/modules/logic_hooks.php.
- This hook is executed at the start of every request at the end of ./include/entryPoint.php.
- This hook should not be used for any type of display output.
- The after_entry_point hook is ideally used for logging or loading libraries.
- Application hooks do not make use of the $bean argument.

Change Log

Version	Note
6.4.3	Added after_entry_point hook

Example

./custom/modules/logic_hooks.php

```
<?php

    $hook_version = 1;
    $hook_array = Array();

    $hook_array['after_entry_point'] = Array();
    $hook_array['after_entry_point'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'after_entry_point example',

        //The PHP file where your class is located.
        'custom/modules/application_hooks_class.php',

        //The class the method is in.
        'application_hooks_class',

        //The method to call.
        'after_entry_point_method'
    );

?>
```

./custom/modules/application_hooks_class.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class application_hooks_class
    {
        function after_entry_point_method($event, $arguments)
        {
            //logic
        }
    }

?>
```

after_load_user

Overview

The after_load_user hook executes after the current user is set for the current request. It handles actions that are dependent on the current user, such as setting ACLs or configuring user-dependent parameters.

Definition

```
function after_load_user($event, $arguments){}
```

Arguments

Name	Type	Description
event	String	The current event
arguments	Array	Additional information related to the event (typically empty)

Change Log

Version	Note
6.6.0	Added after_load_user hook

Example

./custom/modules/logic_hooks.php

```
<?php

  $hook_version = 1;
  $hook_array = Array();

  $hook_array['after_load_user'] = Array();
  $hook_array['after_load_user'][] = Array(
      //Processing index. For sorting the array.
      1,

      //Label. A string value to identify the hook.
      'after_load_user example',
```

```
        //tde PHP file where your class is located.
        'custom/modules/application_hooks_class.php',

        //tde class the method is in.
        'application_hooks_class',

        //tde method to call.
        'after_load_user_method'
    );

?>
```

./custom/modules/application_hooks_class.php

```
  <?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class application_hooks_class
    {
        function after_load_user_method($event, $arguments)
        {
            //logic
        }
    }

?>
```

after_session_start

Overview

The after_session_start hook executes before the user's session starts, but after the user's visibility rules have been set up and the after_load_user logic hook has executed.

Definition

```
  function after_session_start($event, $arguments){}
```

Arguments

Name	Type	Description
event	String	The current event
arguments	Array	Additional information related to the event (typically empty)

Change Log

Version	Note
6.4.3	Added after_session_start hook

Example

./custom/modules/logic_hooks.php

```
  <?php

    $hook_version = 1;
    $hook_array = Array();

    $hook_array['after_session_start'] = Array();
    $hook_array['after_session_start'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'after_session_start example',

        //The PHP file where your class is located.
        'custom/modules/application_hooks_class.php',

        //The class the method is in.
        'application_hooks_class',

        //The method to call.
        'after_session_start_method'
    );

?>
```

./custom/modules/application_hooks_class.php

```
  <?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class application_hooks_class
    {
        function after_session_start_method($event, $arguments)
        {
            //logic
        }
    }

?>
```

after_ui_footer

Overview

The after_ui_footer hook executes after the footer has been invoked for modules in backward compatibility mode. This logic hook has been deprecated and will be removed in a future release. For information on modifying the footer, please refer to Adding Buttons to the Application Footer.

Definition

```
function after_ui_footer($event, $arguments){}
```

Arguments

Name	Type	Description
event	String	The current event
arguments	Array	Additional information related to the event (typically empty)

Considerations

- The after_ui_footer hook is only applicable for modules in backward

compatibility mode.
- This is a global logic hook where the logic hook reference must be placed in ./custom/modules/logic_hooks.php.
- If you intend to write display logic to the screen in a module running in backward compatibility, you must first wrap the display logic in an IF condition to prevent the text breaking the Ajax page loads. This logic may vary and it is best to only run your code on specific pages rather than all pages.

```
if (!isset($_REQUEST["to_pdf"]) || $_REQUEST["to_pdf"] == false)
{
    //display logic
}
```

- This hook is executed on all page loads.
- Application hooks do not make use of the $bean argument.

Change Log

Version	Note
7.10.0.0	Deprecated after_ui_footer hook
5.0.0a	Added after_ui_footer hook

Example

./custom/modules/logic_hooks.php

```
<?php

    $hook_version = 1;
    $hook_array = Array();

    $hook_array['after_ui_footer'] = Array();
    $hook_array['after_ui_footer'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'after_ui_footer example',

        //The PHP file where your class is located.
        'custom/modules/application_hooks_class.php',
```

```
        //The class the method is in.
        'application_hooks_class',

        //The method to call.
        'after_ui_footer_method'
    );

?>
```

./custom/modules/application_hooks_class.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class application_hooks_class
    {
        function after_ui_footer_method($event, $arguments)
        {
            //display logic should check for $_REQUEST["to_pdf"]
        }
    }

?>
```

after_ui_frame

Overview

The after_ui_frame hook executes after the frame has been invoked and before the footer has been invoked for modules in backward compatibility mode. This logic hook has been deprecated and will be removed in a future release. For information on modifying the footer, please refer to Adding Buttons to the Application Footer.

Definition

```
function after_ui_frame($event, $arguments){}
```

Arguments

Name	Type	Description
event	String	The current event
arguments	Array	Additional information related to the event (typically empty)

Considerations

- This hook is only applicable for modules in backward compatibility mode.
- This hook is executed on most views such as the DetailView, EditView, and Listview.
- Application hooks do not make use of the $bean argument.
- This is logic hook can be used as a global reference (./custom/modules/logic_hooks.php) or as a module reference (./custom/modules/<module>/logic_hooks.php).

Change Log

Version	Note
7.10.0.0	Deprecated after_ui_frame hook
5.0.0a	Added after_ui_frame hook

Example

Module-Specific Hook

This hook can be used at the application level for all modules or limited to specific modules. An example limiting the hook for specific modules is shown below:

./custom/modules/<module>/logic_hooks.php

```
<?php

    $hook_version = 1;
    $hook_array = Array();

    $hook_array['after_ui_frame'] = Array();
    $hook_array['after_ui_frame'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'after_ui_frame example',
```

```
        //The PHP file where your class is located.
        'custom/modules/{module}/logic_hooks_class.php',

        //The class the method is in.
        'logic_hooks_class',

        //The method to call.
        'after_ui_frame_method'
    );

?>
```

./custom/modules/<module>/logic_hooks_class.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class logic_hooks_class
    {
        function after_ui_frame_method($event, $arguments)
        {
            //display logic
        }
    }

?>
```

Application Hook

This hook can be used at the application level for all modules or limited to specific modules. An example of executing the hook for all modules is shown below:

./custom/modules/logic_hooks.php

```
<?php

    $hook_version = 1;
    $hook_array = Array();
```

```
    $hook_array['after_ui_frame'] = Array();
    $hook_array['after_ui_frame'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'after_ui_frame example',

        //The PHP file where your class is located.
        'custom/modules/application_hooks_class.php',

        //The class the method is in.
        'application_hooks_class',

        //The method to call.
        'after_ui_frame_method'
    );

?>
```

./custom/modules/application_hooks_class.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class application_hooks_class
    {
        function after_ui_frame_method($event, $arguments)
        {
            //display logic
        }
    }

?>
```

entry_point_variables_setting

Overview

The entry_point_variables_setting application hook executes at the start of every

entry point.

Definition

```
function entry_point_variables_setting($event, $arguments){}
```

Arguments

Name	Type	Description
event	String	The current event
arguments	Array	Additional information related to the event (typically empty)

Considerations

- The entry_point_variables_setting hook is a global logic hook where the logic hook reference must be placed in ./custom/modules/logic_hooks.php.
- This hook is executed at the start of every request at the end of ./include/entryPoint.php.
- This hook does not make use of the $bean argument.

Change Log

Version	Note
6.4.3	Added entry_point_variables_setting hook

Example

./custom/modules/logic_hooks.php

```
<?php

    $hook_version = 1;
    $hook_array = Array();

    $hook_array['entry_point_variables_setting'] = Array();
    $hook_array['entry_point_variables_setting'][] = Array(
        //Processing index. For sorting the array.
        1,
```

```
        //Label. A string value to identify the hook.
        'entry_point_variables_setting example',

        //The PHP file where your class is located.
        'custom/modules/application_hooks_class.php',

        //The class the method is in.
        'application_hooks_class',

        //The method to call.
        'entry_point_variables_setting_method'
    );

?>
```

./custom/modules/application_hooks_class.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class application_hooks_class
    {
        function entry_point_variables_setting_method($event, $argumen
ts)
        {
            //logic
        }
    }

?>
```

handle_exception

Overview

The handle_exception logic hook executes when an exception is thrown.

Definition

```
function handle_exception($event, $exception){}
```

Arguments

Name	Type	Description
event	String	The current event
exception	Object	The exception object

Considerations

- This hook should not be used for any type of display output.
- You can retrieve the exception message by using $exception->getMessage(). All exception classes extend the PHP Exceptions class.

Change Log

Version	Note
6.4.4	Added handle_exception hook

Example

./custom/modules/logic_hooks.php

```
<?php

    $hook_version = 1;
    $hook_array = Array();

    $hook_array['handle_exception'] = Array();
    $hook_array['handle_exception'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'handle_exception example',

        //The PHP file where your class is located.
        'custom/modules/handle_exception_class.php',

        //The class the method is in.
        'handle_exception_class',

        //The method to call.
```

```
        'handle_exception_method'
    );

?>
```

./custom/modules/handle_exception_class.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class handle_exception_class
    {
        function handle_exception_method($event, $exception)
        {
            //logic with $exception->getMessage()
        }
    }

?>
```

server_round_trip

Overview

Executes at the end of every page.

Definition

```
function server_round_trip($event, $arguments){}
```

Arguments

Name	Type	Description
event	String	The current event
arguments	Array	Additional information related to the event (typically empty)

Considerations

- This is a global logic hook where the logic hook reference must be placed in ./custom/modules/logic_hooks.php.
- If you are intending to write display logic to the screen, you must first wrap the display logic in an if condition to prevent breaking the Ajax page loads:

```
if (!isset($_REQUEST["to_pdf"]) || $_REQUEST["to_pdf"] == false)
{
    //display logic
}
```

- This hook is executed on all page loads.
- Application hooks do not make use of the $bean argument.

Change Log

Version	Note
5.0.0a	Added server_round_trip hook.

Example

./custom/modules/logic_hooks.php

```
<?php

    $hook_version = 1;
    $hook_array = Array();

    $hook_array['server_round_trip'] = Array();
    $hook_array['server_round_trip'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'server_round_trip example',

        //The PHP file where your class is located.
        'custom/modules/application_hooks_class.php',

        //The class the method is in.
        'application_hooks_class',
```

```
        //The method to call.
        'server_round_trip_method'
    );

?>
```

./custom/modules/application_hooks_class.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class application_hooks_class
    {
        function server_round_trip_method($event, $arguments)
        {
            //display logic should check for $_REQUEST["to_pdf"]
        }
    }

?>
```

Module Hooks

Module hooks execute logic when working with Sugar modules (e.g. Accounts, Cases, etc.).

---

after_delete

Overview

The after_delete hook executes after a record is deleted.

Definition

```
function after_delete($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object
event	String	The current event
arguments	Array	Additional information related to the event
arguments.id	String	ID of the deleted record

Examples

Creating a Logic Hook using the Extension Framework

./custom/Extension/modules/<module>/Ext/LogicHooks/<file>.php

```
<?php

    $hook_array['after_delete'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'after_delete example',

        //The PHP file where your class is located.
        'custom/modules/<module>/logic_hooks_class.php',

        //The class the method is in.
        'logic_hooks_class',

        //The method to call.
        'after_delete_method'
    );

?>
```

./custom/modules/<module>/logic_hooks_class.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');
```

```
    class logic_hooks_class
    {
        function after_delete_method($bean, $event, $arguments)
        {
            //logic
        }
    }

?>
```

Creating a Core Logic Hook

Prior to Sugar 6.3.x, logic hooks could only be created using the following method. Please note that this approach is still valid but is not recommended when building plugins as it may conflict with existing customizations.

./custom/modules/<module>/logic_hooks.php

```
<?php

   $hook_version = 1;
   $hook_array = Array();

   $hook_array['after_delete'] = Array();
   $hook_array['after_delete'][] = Array(
      //Processing index. For sorting the array.
      1,

      //Label. A string value to identify the hook.
      'after_delete example',

      //The PHP file where your class is located.
      'custom/modules/<module>/after_delete_class.php',

      //The class the method is in.
      'after_delete_class',

      //The method to call.
      'after_delete_method'
   );

?>
```

./custom/modules/<module>/after_delete_class.php

```
<?php

   if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry P
oint');

   class after_delete_class
   {
      function after_delete_method($bean, $event, $arguments)
      {
         //logic
      }
   }

?>
```

after_fetch_query

Overview

The after_fetch_query logic hook executes after a sugar query has been executed.

Definition

```
function after_fetch_query($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object
event	String	The current event
arguments	Array	Additional information related to the event
arguments.beans	Array	An array of bean objects resulting from the query
arguments.fields	Array	An array of selected fields
arguments.rows	Array	An array representation of the selected beans

Change Log

Version	Note
7.7.0.0	Added after_fetch_query logic hook

Examples

Creating a Logic Hook using Extension Framework

./custom/Extension/modules/<module>/Ext/LogicHooks/<file>.php

```
<?php

    $hook_array['after_fetch_query'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'after_fetch_query example',

        //The PHP file where your class is located.
        'custom/modules/<module>/after_fetch_query_class.php',

        //The class the method is in.
        'after_fetch_query_class',

        //The method to call.
        'after_fetch_query_method'
    );

?>
```

./custom/modules/<module>/after_fetch_query_class.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class after_fetch_query_class
    {
        function after_fetch_query_method($bean, $event, $arguments)
```

```
        {
            //logic
        }
    }

?>
```

Creating a Core Logic Hook

Prior to Sugar 6.3.x, logic hooks could only be created using the following method. Please note that this approach is still valid but is not recommended when building plugins as it may conflict with existing customizations.

./custom/modules/<module>/logic_hooks.php

```
<?php

   $hook_version = 1;
   $hook_array = Array();

   $hook_array['after_fetch_query'] = Array();
   $hook_array['after_fetch_query'][] = Array(
      //Processing index. For sorting the array.
      1,

      //Label. A string value to identify the hook.
      'after_fetch_query example',

      //The PHP file where your class is located.
      'custom/modules/<module>/after_fetch_query_class.php',

      //The class the method is in.
      'logic_hooks_class',

      //The method to call.
      'after_fetch_query_method'
   );

?>
```

./custom/modules/<module>/after_fetch_query_class.php

```
<?php

   if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry P
oint');

   class after_fetch_query_class
   {
      function after_fetch_query_method($bean, $event, $arguments)
      {
         //logic
      }
   }

?>
```

after_relationship_add

Overview

The after_relationship_add hook executes after a relationship has been added between two records.

Definition

```
function after_relationship_add($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object
event	String	The current event
arguments	Array	Additional information related to the event
arguments.id	String	Module ID
arguments.module	String	Module name
arguments.related_id	String	Related module ID
arguments.related_module	String	Related module name
arguments.link	String	Link field name

arguments.relationship	String	Relationship name

Considerations

- This hook will be executed for each side of the relationship. An example is that if you add an association between an Account and Contact, the hook will be run for both.
- The arguments parameter will have additional information regarding the records being modified. The $bean variable will not contain this information.

Change Log

Version	Note
6.0.0	Added after_relationship_add hook.

Examples

Creating a Logic Hook using the Extension Framework

./custom/Extension/modules/<module>/Ext/LogicHooks/<file>.php

```
<?php

    $hook_array['after_relationship_add'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'after_relationship_add example',

        //The PHP file where your class is located.
        'custom/modules/{module}/after_relationship_add_class.php',

        //The class the method is in.
        'after_relationship_add_class',

        //The method to call.
        'after_relationship_add_method'
    );

?>
```

./custom/modules/<module>/after_relationship_add_class.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class after_relationship_add_class
    {
        function after_relationship_add_method($bean, $event, $argumen
ts)
        {
            //logic
        }
    }

?>
```

Creating a Core Logic Hook

Prior to Sugar 6.3.x, logic hooks could only be created using the following method. Please note that this approach is still valid but is not recommended when building plugins as it may conflict with existing customizations.

./custom/modules/<module>/logic_hooks.php

```
<?php

   $hook_version = 1;
   $hook_array = Array();

   $hook_array['after_relationship_add'] = Array();
   $hook_array['after_relationship_add'][] = Array(
      //Processing index. For sorting the array.
      1,

      //Label. A string value to identify the hook.
      'after_relationship_add example',

      //The PHP file where your class is located.
      'custom/modules/<module>/after_relationship_add_class.php',

      //The class the method is in.
      'after_relationship_add_class',
```

```
      //The method to call.
      'after_relationship_add_method'
   );

?>
```

./custom/modules/<module>/after_relationship_add_class.php

```
<?php

   if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry P
oint');

   class after_relationship_add_class
   {
      function after_relationship_add_method($bean, $event, $arguments
)
      {
         //logic
      }
   }

?>
```

after_relationship_delete

Overview

The after_relationship_delete hook executes after a relationship between two records has been deleted.

Definition

```
function after_relationship_delete($bean, $event, $arguments){}
```

Arguments

Name	Type	Description

bean	Object	The bean object
event	String	The current event
arguments	Array	Additional information related to the event
arguments.id	String	Module ID
arguments.module	String	Module name
arguments.related_id	String	Related module ID
arguments.related_module	String	Related module name
arguments.link	String	Link field name
arguments.relationship	String	Relationship name

Considerations

- This hook will be executed for each side of the relationship. For example, if you delete an association between an account and contact, the hook will run for both.
- The 'arguments' parameter will have additional information regarding the records being modified. The $bean variable will not contain this information.

Change Log

Version	Note
6.0.0	Added after_relationship_delete hook

Examples

Creating a Logic Hook using the Extension Framework

./custom/Extension/modules/<module>/Ext/LogicHooks/<file>.php

```
<?php

    $hook_array['after_relationship_delete'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'after_relationship_delete example',

        //The PHP file where your class is located.
        'custom/modules/<module>/after_relationship_delete_class.php',
```

```
        //The class the method is in.
        'after_relationship_delete_class',

        //The method to call.
        'after_relationship_delete_method'
    );

?>
```

./custom/modules/<module>/after_relationship_delete_class.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class after_relationship_delete_class
    {
        function after_relationship_delete_method($bean, $event, $argu
ments)
        {
            //logic
        }
    }

?>
```

Creating a Core Logic Hook

Prior to Sugar 6.3.x, logic hooks could only be created using the following method. Please note that this approach is still valid but is not recommended when building plugins as it may conflict with existing customizations.

./custom/modules/<module>/logic_hooks.php

```
<?php

   $hook_version = 1;
   $hook_array = Array();

   $hook_array['after_relationship_delete'] = Array();
```

```
   $hook_array['after_relationship_delete'][] = Array(
      //Processing index. For sorting the array.
      1,

      //Label. A string value to identify the hook.
      'after_relationship_delete example',

      //The PHP file where your class is located.
      'custom/modules/<module>/after_relationship_delete_class.php',

      //The class the method is in.
      'after_relationship_delete_class',

      //The method to call.
      'after_relationship_delete_method'
   );

?>
```

./custom/modules/<module>/after_relationship_delete_class.php

```
<?php

   if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry P
oint');

   class after_relationship_delete_class
   {
      function after_relationship_delete_method($bean, $event, $argume
nts)
      {
         //logic
      }
   }

?>
```

after_restore

Overview

The after_restore hook executes after a record gets undeleted (i.e. the deleted

field's value changes from 1 to 0).

Definition

```
function after_restore($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object
event	String	The current event
arguments	Array	Additional information related to the event (typically empty)

Examples

Creating a Logic Hook using the Extension Framework

./custom/Extension/modules/<module>/Ext/LogicHooks/<file>.php

```
<?php

    $hook_array['after_restore'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'after_restore example',

        //The PHP file where your class is located.
        'custom/modules/<module>/after_restore_class.php',

        //The class the method is in.
        'after_restore_class',

        //The method to call.
        'after_restore_method'
    );

?>
```

./custom/modules/<module>/after_restore_class.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class after_restore_class
    {
        function after_restore_method($bean, $event, $arguments)
        {
            //logic
        }
    }

?>
```

Creating a Core Logic Hook

Prior to Sugar 6.3.x, logic hooks could only be created using the following method. Please note that this approach is still valid but is not recommended when building plugins as it may conflict with existing customizations.

./custom/modules/<module>/logic_hooks.php

```
<?php

   $hook_version = 1;
   $hook_array = Array();

   $hook_array['after_restore'] = Array();
   $hook_array['after_restore'][] = Array(
      //Processing index. For sorting the array.
      1,

      //Label. A string value to identify the hook.
      'after_restore example',

      //The PHP file where your class is located.
      'custom/modules/<module>/after_restore_class.php',

      //The class the method is in.
      'after_restore_class',
```

```
      //The method to call.
      'after_restore_method'
   );

?>
```

./custom/modules/<module>/after_restore_class.php

```
<?php

   if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry P
oint');

   class after_restore_class
   {
      function after_restore_method($bean, $event, $arguments)
      {
         //logic
      }
   }

?>
```

after_retrieve

Overview

The after_retrieve hook executes after a record has been retrieved from the database.

Definition

```
function after_retrieve($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object
event	String	The current event

arguments	Array	Additional information related to the event
arguments.id	String	ID of the record
arguments.encode	Boolean	Whether or not to encode

Considerations

- The after_retrieve hook does not fire when a new record is being created.
- Calling save on the bean in this hook can cause adverse side effects if not handled correctly and should be avoided. (i.e. $bean->save())

Examples

Creating a Logic Hook using the Extension Framework

./custom/Extension/modules/<module>/Ext/LogicHooks/<file>.php

```
<?php

    $hook_array['after_retrieve'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'after_retrieve example',

        //The PHP file where your class is located.
        'custom/modules/<module>/after_retrieve_class.php',

        //The class the method is in.
        'after_retrieve_class',

        //The method to call.
        'after_retrieve_method'
    );

?>
```

./custom/modules/{module}/{module}_hook.php

```
<?php
```

```
    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class after_retrieve_class
    {
        function after_retrieve_method($bean, $event, $arguments)
        {
            //logic
        }
    }

?>
```

Creating a Core Logic Hook

Prior to Sugar 6.3.x, logic hooks could only be created using the following method. Please note that this approach is still valid but is not recommended when building plugins as it may conflict with existing customizations.

./custom/modules/<module>/logic_hooks.php

```
<?php

   $hook_version = 1;
   $hook_array = Array();

   $hook_array['after_retrieve'] = Array();
   $hook_array['after_retrieve'][] = Array(
      //Processing index. For sorting the array.
      1,

      //Label. A string value to identify the hook.
      'after_retrieve example',

      //The PHP file where your class is located.
      'custom/modules/<module>/after_retrieve_class.php',

      //The class the method is in.
      'after_retrieve_class',

      //The method to call.
      'after_retrieve_method'
   );
```

```
?>
```

./custom/modules/<module>/after_retrieve_class.php

```
<?php

   if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry P
oint');

   class after_retrieve_class
   {
      function after_retrieve_method($bean, $event, $arguments)
      {
         //logic
      }
   }

?>
```

after_save

Overview

The after_save hook executes after a record is saved.

Definition

```
    function after_save($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object
event	String	The current event
arguments	Array	Additional information related to the event (typically empty)
arguments.isUpdate	Boolean	Whether or not the record

		is newly created or not. True is an existing record being saved. False is a new record being created
arguments.dataChanges	Array	A list of the fields in the auditable fields on the bean, including the ones that changed and the ones that did not. **Note**: This argument is deprecated and it is recommended to use arguments.stateChanges instead.
arguments.stateChanges	Array	Contains the fields which were changed comparing to the previous bean state

Considerations

- Calling save on the bean in this hook will cause an infinite loop if not handled correctly. (i.e: $bean->save())

Examples

Creating a Logic Hook using the Extension Framework

./custom/Extension/modules/<module>/Ext/LogicHooks/<file>.php

```
<?php

$hook_array['after_save'][] = Array(
    //Processing index. For sorting the array.
    1,

    //Label. A string value to identify the hook.
    'after_save example',

    //The PHP file where your class is located.
    'custom/modules/<module>/after_save_class.php',

    //The class the method is in.
    'after_save_class',
```

```
        //The method to call.
        'after_save_method'
    );

?>
```

./custom/modules/<module>/after_save_class.php

```
    <?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class after_save_class
    {
        function after_save_method($bean, $event, $arguments)
        {
            //logic
        }
    }

?>
```

Creating a Core Logic Hook

Prior to Sugar 6.3.x, logic hooks could only be created using the following method. Please note that this approach is still valid but is not recommended when building plugins as it may conflict with existing customizations.

./custom/modules/<module>/logic_hooks.php

```
    <?php

   $hook_version = 1;
   $hook_array = Array();

   $hook_array['after_save'] = Array();
   $hook_array['after_save'][] = Array(
      //Processing index. For sorting the array.
      1,

      //Label. A string value to identify the hook.
```

```
      'after_save example',

      //The PHP file where your class is located.
      'custom/modules/<module>/after_save_class.php',

      //The class the method is in.
      'after_save_class',

      //The method to call.
      'after_save_method'
   );

?>
```

./custom/modules/<module>/after_save_class.php

```
    <?php

   if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry P
oint');

   class after_save_class
   {
      function after_save_method($bean, $event, $arguments)
      {
         //logic
      }
   }

?>
```

before_delete

Overview

The before_delete logic hook executes before a record is deleted.

Definition

```
function before_delete($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object
event	String	The current event
arguments	Array	Additional information related to the event
arguments.id	String	ID of the record to delete

Examples

Creating a Logic Hook using the Extension Framework

./custom/Extension/modules/<module>/Ext/LogicHooks/<file>.php

```
<?php

    $hook_array['before_delete'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'before_delete example',

        //The PHP file where your class is located.
        'custom/modules/<module>/before_delete_class.php',

        //The class the method is in.
        'before_delete_class',

        //The method to call.
        'before_delete_method'
    );

?>
```

./custom/modules/<module>/before_delete_class.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');
```

```
    class before_delete_class
    {
        function before_delete_method($bean, $event, $arguments)
        {
            //logic
        }
    }

?>
```

Creating a Core Logic Hook

Prior to Sugar 6.3.x, logic hooks could only be created using the following method. Please note that this approach is still valid but is not recommended when building plugins as it may conflict with existing customizations.

./custom/modules/<module>/logic_hooks.php

```
<?php

   $hook_version = 1;
   $hook_array = Array();

   $hook_array['before_delete'] = Array();
   $hook_array['before_delete'][] = Array(
      //Processing index. For sorting the array.
      1,

      //Label. A string value to identify the hook.
      'before_delete example',

      //The PHP file where your class is located.
      'custom/modules/<module>/before_delete_class.php',

      //The class the method is in.
      'before_delete_class',

      //The method to call.
      'before_delete_method'
   );

?>
```

./custom/modules/<module>/before_delete_class.php

```
<?php

   if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry P
oint');

   class before_delete_class
   {
      function before_delete_method($bean, $event, $arguments)
      {
         //logic
      }
   }

?>
```

before_fetch_query

Overview

The before_fetch_query logic hook executes before a sugar query has been executed.

Definition

```
function before_fetch_query($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object
event	String	The current event
arguments	Array	Additional information related to the event
arguments.query	Object	The query to be executed
arguments.fields	Array	An array of selected fields

Change Log

Version	Note
7.7.0.0	Added before_fetch_query logic hook

Examples

Creating a Logic Hook using Extension Framework

./custom/Extension/modules/<module>/Ext/LogicHooks/<file>.php

```
<?php

    $hook_array['before_fetch_query'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'before_fetch_query example',

        //The PHP file where your class is located.
        'custom/modules/<module>/before_fetch_query_class.php',

        //The class the method is in.
        'before_fetch_query_class',

        //The method to call.
        'before_fetch_query_method'
    );

?>
```

./custom/modules/<module>/before_fetch_query_class.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class before_fetch_query_class
    {
        function before_fetch_query_method($bean, $event, $arguments)
        {
            //logic
        }
```

```
    }

?>
```

Creating a Core Logic Hook

Prior to Sugar 6.3.x, logic hooks could only be created using the following method. Please note that this approach is still valid but is not recommended when building plugins as it may conflict with existing customizations.

./custom/modules/<module>/logic_hooks.php

```
<?php

   $hook_version = 1;
   $hook_array = Array();

   $hook_array['before_fetch_query'] = Array();
   $hook_array['before_fetch_query'][] = Array(
      //Processing index. For sorting the array.
      1,

      //Label. A string value to identify the hook.
      'before_fetch_query example',

      //The PHP file where your class is located.
      'custom/modules/<module>/before_fetch_query_class.php',

      //The class the method is in.
      'logic_hooks_class',

      //The method to call.
      'before_fetch_query_method'
   );

?>
```

./custom/modules/<module>/before_fetch_query_class.php

```
<?php
```

```
   if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry P
oint');

   class before_fetch_query_class
   {
      function before_fetch_query_method($bean, $event, $arguments)
      {
         //logic
      }
   }

?>
```

before_relationship_add

Overview

The before_relationship_add logic hook executes before a relationship has been added between two records.

Definition

```
function before_relationship_add($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object
event	String	The current event
arguments	Array	Additional information related to the event
arguments.id	String	Module ID
arguments.module	String	Module name
arguments.related_id	String	Related module ID
arguments.related_module	String	Related module name
arguments.link	String	Link field name
arguments.relationship	String	Relationship name

Considerations

- This hook will be executed for each side of the relationship. For example, if you add an association between an account and a contact, the hook will run for both records.
- The arguments parameter will have additional information regarding the records being modified. The $bean variable will not contain this information.

Change Log

Version	Note
6.4.5	Added before_relationship_add hook

Examples

Creating a Logic Hook using the Extension Framework

./custom/Extension/modules/<module>/Ext/LogicHooks/<file>.php

```
<?php

    $hook_array['before_relationship_add'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'before_relationship_add example',

        //The PHP file where your class is located.
        'custom/modules/<module>/before_relationship_add_class.php',

        //The class the method is in.
        'before_relationship_add_class',

        //The method to call.
        'before_relationship_add_method'
    );

?>
```

./custom/modules/<module>/before_relationship_add_class.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class before_relationship_add_class
    {
        function before_relationship_add($bean, $event, $arguments)
        {
            //logic
        }
    }

?>
```

before_relationship_delete

Overview

The before_relationship_delete logic hook executes before a relationship between two records is deleted.

Definition

```
function before_relationship_delete($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object
event	String	The current event
arguments	Array	Additional information related to the event
arguments.id	String	Module id
arguments.module	String	Module name
arguments.related_id	String	Related module id
arguments.related_module	String	Related module name
arguments.link	String	Link field name

Name	Type	Description
arguments.relationship	String	Relationship name

Considerations

- This hook will be executed for each side of the relationship. For example, if you delete an association between an account and a contact, the hook will run for both records.
- The arguments parameter will have additional information regarding the records being modified. The $bean variable will not contain this information.

Change Log

Version	Note
6.4.5	Added before_relationship_delete hook.

Examples

Creating a Logic Hook using the Extension Framework

./custom/Extension/modules/<module>/Ext/LogicHooks/<file>.php

```
<?php

    $hook_array['before_relationship_delete'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'before_relationship_delete example',

        //The PHP file where your class is located.
        'custom/modules/<module>/before_relationship_delete_class.php'
,

        //The class the method is in.
        'before_relationship_delete_class',

        //The method to call.
        'before_relationship_delete_method'
    );

?>
```

./custom/modules/<module>/before_relationship_delete_class.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class before_relationship_delete_class
    {
        function before_relationship_delete_method($bean, $event, $arg
uments)
        {
            //logic
        }
    }

?>
```

before_restore

Overview

The before_restore logic hook executes before a record gets undeleted (i.e. the deleted field's value changes from 1 to 0).

Definition

```
function before_restore($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object
event	String	The current event
arguments	Array	Additional information related to the event (typically empty)

Examples

Creating a Logic Hook using the Extension Framework

./custom/Extension/modules/<module>/Ext/LogicHooks/<file>.php

```
<?php

        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'before_restore example',

        //The PHP file where your class is located.
        'custom/modules/<module>/before_restore_class.php',

        //The class the method is in.
        'before_restore_class',

        //The method to call.
        'before_restore_method'
    );

?>
```

./custom/modules/<module>/before_restore_class.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class before_restore_class
    {
        function before_restore_method($bean, $event, $arguments)
        {
            //logic
        }
    }

?>
```

Creating a Core Logic Hook

Prior to Sugar 6.3.x, logic hooks could only be created using the following method. Please note that this approach is still valid but is not recommended when building plugins as it may conflict with existing customizations.

./custom/modules/<module>/logic_hooks.php

```
<?php

$hook_version = 1;
$hook_array = Array();

$hook_array['before_restore'] = Array();
$hook_array['before_restore'][] = Array(
//Processing index. For sorting the array.
1,

//Label. A string value to identify the hook.
'before_restore example',

//The PHP file where your class is located.
'custom/modules/<module>/before_restore_class.php',

//The class the method is in.
'before_restore_class',

//The method to call.
'before_restore_method'
);

?>
```

./custom/modules/<module>/before_restore_class.php

```
<?php

if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry Poin
t');

class before_restore_class
{
```

```
function before_restore_method($bean, $event, $arguments)
{
//logic
}
}

?>
```

before_save

Overview

The before_save logic hook executes before a record is saved.

Definition

```
function before_save($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object
event	String	The current event
arguments	Array	Additional information related to the event
arguments.check_notify	Boolean	Whether or not to send notifications
arguments.isUpdate	Boolean	Whether or not the record is newly created • true = this is an update to an existing record • false = a newly created record

Considerations

- For modules that contain a user-friendly record ID (e.g. the case_number field for the Cases module), the value of that field is not available for a before_save call. This is because this business logic has yet to be executed.
- Calling save on the bean in this hook will cause an infinite loop if not handled correctly. (i.e: $bean->save())

Examples

Creating a Logic Hook using Extension Framework

./custom/Extension/modules/<module>/Ext/LogicHooks/<file>.php

```
  <?php

    $hook_array['before_save'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'before_save example',

        //The PHP file where your class is located.
        'custom/modules/<module>/before_save_class.php',

        //The class the method is in.
        'before_save_class',

        //The method to call.
        'before_save_method'
    );

?>
```

./custom/modules/<module>/before_save_class.php

```
  <?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class before_save_class
    {
        function before_save_method($bean, $event, $arguments)
```

```
        {
            //logic
        }
    }

?>
```

Creating a Core Logic Hook

Prior to Sugar 6.3.x, logic hooks could only be created using the following method. Please note that this approach is still valid but is not recommended when building plugins as it may conflict with existing customizations.

./custom/modules/<module>/logic_hooks.php

```
<?php

   $hook_version = 1;
   $hook_array = Array();

   $hook_array['before_save'] = Array();
   $hook_array['before_save'][] = Array(
      //Processing index. For sorting the array.
      1,

      //Label. A string value to identify the hook.
      'before_save example',

      //The PHP file where your class is located.
      'custom/modules/<module>/before_save_class.php',

      //The class the method is in.
      'before_save_class',

      //The method to call.
      'before_save_method'
   );

?>
```

./custom/modules/<module>/before_save_class.php

```
<?php

   if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry P
oint');

   class before_save_class
   {
      function before_save_method($bean, $event, $arguments)
      {
         //logic
      }
   }

?>
```

process_record

Overview

The process_record logic hook executes when the record is being processed as a part of the ListView or subpanel list.

Definition

```
function process_record($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object
event	String	The current event
arguments	Array	Additional information related to the event (typically empty)

Considerations

- This can be used to set values in a record's fields prior to display in the ListView or in the subpanel of a DetailView.

- This event is not fired in the EditView.
- You should not call save on the referenced bean. The bean is only populated for list fields and will result in a loss of information.

Change Log

Version	Note
5.0.0a	Added process_record hook

Examples

Creating a Logic Hook using the Extension Framework

./custom/Extension/modules/<module>/Ext/LogicHooks/<file>.php

```
<?php

    $hook_array['process_record'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'process_record example',

        //The PHP file where your class is located.
        'custom/modules/<module>/process_record_class.php',

        //The class the method is in.
        'process_record_class',

        //The method to call.
        'process_record_method'
    );

?>
```

./custom/modules/<module>/process_record_class.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');
```

```
    class process_record_class
    {
        function process_record_method($bean, $event, $arguments)
        {
            //logic
        }
    }

?>
```

Creating a Core Logic Hook

Prior to Sugar 6.3.x, logic hooks could only be created using the following method. Please note that this approach is still valid but is not recommended when building plugins as it may conflict with existing customizations.

./custom/modules/<module>/logic_hooks.php

```
<?php

   $hook_version = 1;
   $hook_array = Array();

   $hook_array['process_record'] = Array();
   $hook_array['process_record'][] = Array(
      //Processing index. For sorting the array.
      1,

      //Label. A string value to identify the hook.
      'process_record example',

      //The PHP file where your class is located.
      'custom/modules/<module>/process_record_class.php',

      //The class the method is in.
      'process_record_class',

      //The method to call.
      'process_record_method'
   );

?>
```

./custom/modules/<module>/process_record_class.php

```
<?php

   if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry P
oint');

   class process_record_class
   {
      function process_record_method($bean, $event, $arguments)
      {
         //logic
      }
   }

?>
```

User Hooks

User hooks execute logic around user actions such as logging in and logging out.

after_login

Overview

The after_login hook executes after a user logs into the system.

Definition

```
function after_login($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object
event	String	The current event
arguments	Array	Additional information related to the event

Name	Type	Description
		(typically empty)

Change Log

Version	Note
5.0.0a	Added after_login hook

Example

./custom/modules/Users/logic_hooks.php

```
<?php

    $hook_version = 1;
    $hook_array = Array();

    $hook_array['after_login'] = Array();
    $hook_array['after_login'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'after_login example',

        //The PHP file where your class is located.
        'custom/modules/Users/logic_hooks_class.php',

        //The class the method is in.
        'logic_hooks_class',

        //The method to call.
        'after_login_method'
    );

?>
```

./custom/modules/Users/logic_hooks_class.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
```

```
Point');

    class logic_hooks_class
    {
        function after_login_method($bean, $event, $arguments)
        {
            //logic
        }
    }

?>
```

after_logout

Overview

The after_logout hook executes after a user logs out of the system.

Definition

```
function after_logout($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object
event	String	The current event
arguments	Array	Additional information related to the event (typically empty)

Change Log

Version	Note
5.0.0a	Added after_logout hook

Example

./custom/modules/Users/logic_hooks.php

```
<?php

    $hook_version = 1;
    $hook_array = Array();

    $hook_array['after_logout'] = Array();
    $hook_array['after_logout'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'after_logout example',

        //The PHP file where your class is located.
        'custom/modules/Users/logic_hooks_class.php',

        //The class the method is in.
        'logic_hooks_class',

        //The method to call.
        'after_logout_method'
    );

?>
```

./custom/modules/Users/logic_hooks_class.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class logic_hooks_class
    {
        function after_logout_method($bean, $event, $arguments)
        {
            //logic
        }
    }

?>
```

before_logout

Overview

The before_logout hook executes before a user logs out of the system.

Definition

```
function before_logout($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object
event	String	The current event
arguments	Array	Additional information related to the event (typically empty)

Change Log

Version	Note
5.0.0a	Added before_logout hook

Example

./custom/modules/Users/logic_hooks.php

```
<?php

    $hook_version = 1;
    $hook_array = Array();

    $hook_array['before_logout'] = Array();
    $hook_array['before_logout'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'before_logout example',
```

```
        //The PHP file where your class is located.
        'custom/modules/Users/logic_hooks_class.php',

        //The class the method is in.
        'logic_hooks_class',

        //The method to call.
        'before_logout_method'
    );

?>
```

./custom/modules/Users/logic_hooks_class.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class logic_hooks_class
    {
        function before_logout_method($bean, $event, $arguments)
        {
            //logic
        }
    }

?>
```

login_failed

Overview

The login_failed hook executes on a failed login attempt.

Definition

```
function login_failed($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object
event	String	The current event
arguments	Array	Additional information related to the event (typically empty)

Change Log

Version	Note
5.0.0a	Added login_failed hook

Example

./custom/modules/Users/logic_hooks.php

```
<?php

    $hook_version = 1;
    $hook_array = Array();

    $hook_array['login_failed'] = Array();
    $hook_array['login_failed'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'login_failed example',

        //The PHP file where your class is located.
        'custom/modules/Users/logic_hooks_class.php',

        //The class the method is in.
        'logic_hooks_class',

        //The method to call.
        'login_failed_method'
    );

?>
```

./custom/modules/Users/logic_hooks_class.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class logic_hooks_class
    {
        function login_failed_method($bean, $event, $arguments)
        {
            //logic
        }
    }

?>
```

Job Queue Hooks

Job Queue hooks execute logic when working with the Job Queue module.

job_failure

Overview

The job_failure hook executes when a job's final failure occurs.

Definition

```
function job_failure($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The SchedulersJob object
event	String	The current event
arguments	Array	Additional information

Name	Type	Description
		related to the event (typically empty)

Change Log

Version	Note
6.5.0RC1	Added job_failure hook

Example

./custom/modules/SchedulersJobs/logic_hooks.php

```
<?php

    $hook_version = 1;
    $hook_array = Array();

    $hook_array['job_failure'] = Array();
    $hook_array['job_failure'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'job_failure example',

        //The PHP file where your class is located.
        'custom/modules/SchedulersJobs/logic_hooks_class.php',

        //The class the method is in.
        'logic_hooks_class',

        //The method to call.
        'job_failure_method'
    );

?>
```

./custom/modules/SchedulersJobs/logic_hooks_class.php

```
<?php
```

```
    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class logic_hooks_class
    {
        function job_failure_method($bean, $event, $arguments)
        {
            //logic
        }
    }

?>
```

---

job_failure_retry

Overview

The job_failure_retry hook executes each time a job fails before its final failure. If you only want action on the final failure, use the job_failure logic hook.

Definition

```
function job_failure_retry($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The SchedulersJob object
event	String	The current event
arguments	Array	Additional information related to the event (typically empty)

Change Log

Version	Note
6.5.0RC1	Added job_failure_retry hook

Example

./custom/modules/SchedulersJobs/logic_hooks.php

```
<?php

    $hook_version = 1;
    $hook_array = Array();

    $hook_array['job_failure_retry'] = Array();
    $hook_array['job_failure_retry'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'job_failure_retry example',

        //The PHP file where your class is located.
        'custom/modules/SchedulersJobs/logic_hooks_class.php',

        //The class the method is in.
        'logic_hooks_class',

        //The method to call.
        'job_failure_retry_method'
    );
?>
```

./custom/modules/SchedulersJobs/logic_hooks_class.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class logic_hooks_class
    {
        function job_failure_retry_method($bean, $event, $arguments)
        {
            //logic
        }
    }

?>
```

SNIP Hooks

SNIP hooks execute logic when working with the SNIP email-archiving service.

after_email_import

Overview

The after_email_import hook executes after a SNIP email has been imported.

Definition

```
function after_email_import($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The Email object
event	String	The current event
arguments	Array	Additional information related to the event (typically empty)

Considerations

- This is a global logic hook where the logic hook reference must be placed in ./custom/modules/logic_hooks.php.

Change Log

Version	Note
6.5.0RC1	Added after_email_import hook

Example

./custom/modules/logic_hooks.php

```
<?php

    $hook_version = 1;
    $hook_array = Array();

    $hook_array['after_email_import'] = Array();
    $hook_array['after_email_import'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'after_email_import example',

        //The PHP file where your class is located.
        'custom/modules/SNIP/logic_hooks_class.php',

        //The class the method is in.
        'logic_hooks_class',

        //The method to call.
        'after_email_import_method'
    );

?>
```

./custom/modules/SNIP/logic_hooks_class.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class logic_hooks_class
    {
        function after_email_import_method($bean, $event, $arguments)
        {
            //logic
        }
    }

?>
```

before_email_import

Overview

The before_email_import hook executes before a SNIP email has been imported.

Definition

```
function before_email_import($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The Email object
event	String	The current event
arguments	Array	Additional information related to the event (typically empty)

Considerations

- This is a global logic hook where the logic hook reference must be placed in ./custom/modules/logic_hooks.php.

Change Log

Version	Note
6.5.0RC1	Added before_email_import hook

Example

./custom/modules/logic_hooks.php

```
<?php

    $hook_version = 1;
    $hook_array = Array();
```

```
    $hook_array['before_email_import'] = Array();
    $hook_array['before_email_import'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'before_email_import example',

        //The PHP file where your class is located.
        'custom/modules/SNIP/logic_hooks_class.php',

        //The class the method is in.
        'logic_hooks_class',

        //The method to call.
        'before_email_import_method'
    );

?>
```

./custom/modules/SNIP/logic_hooks_class.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class logic_hooks_class
    {
        function before_email_import_method($bean, $event, $arguments)
        {
            //logic
        }
    }

?>
```

API Hooks

API hooks execute logic when working with the REST API and routing.

---

after_routing

Overview

The after_routing hook executes when the v10+ REST Service has found the route for the request.

Definition

```
function after_routing($event, $arguments){}
```

Arguments

Name	Type	Description
event	String	The name of the logic hook event
arguments	Array	Additional information related to the event
arguments.api	Object	The RestService Object
arguments.request	Object	The RestResponse Object

Considerations

- This is a global logic hook where the logic hook reference must be placed in ./custom/modules/logic_hooks.php.
- This hook can change request object parameters that influence routing.
- This hook should not be used for any type of display output.

Change Log

Version	Note
7.0.0RC1	Added after_routing hook

Example

./custom/modules/logic_hooks.php

```
<?php

    $hook_version = 1;
    $hook_array = Array();

    $hook_array['after_routing'] = Array();
    $hook_array['after_routing'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'after_routing example',

        //The PHP file where your class is located.
        'custom/modules/logic_hooks_class.php',

        //The class the method is in.
        'logic_hooks_class',

        //The method to call.
        'after_routing_method'
    );

?>
```

./custom/modules/logic_hooks_class.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class logic_hooks_class
    {
        function after_routing_method($event, $arguments)
        {
            //logic
        }
    }

?>
```

before_api_call

Overview

The before_api_call hook executes when the v10+ REST Service is about to call the API implementation.

Definition

```
function before_api_call($event, $arguments){}
```

Arguments

Name	Type	Description
event	String	The name of the logic hook event
arguments	Array	Additional information related to the event
arguments.api	Object	The RestService Object
arguments.request	Object	The RestResponse Object

Considerations

- This is a global logic hook where the logic hook reference must be placed in ./custom/modules/logic_hooks.php.
- This hook can change the method being called and the parameters.
- This hook should not be used for any type of display output.

Change Log

Version	Note
7.0.0RC1	Added before_api_call hook

Example

./custom/modules/logic_hooks.php

```
<?php

  $hook_version = 1;
```

```
    $hook_array = Array();

    $hook_array['before_api_call'] = Array();
    $hook_array['before_api_call'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'before_api_call example',

        //The PHP file where your class is located.
        'custom/modules/logic_hooks_class.php',

        //The class the method is in.
        'logic_hooks_class',

        //The method to call.
        'before_api_call_method'
    );

?>
```

./custom/modules/logic_hooks_class.php

```
  <?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class logic_hooks_class
    {
        function before_api_call_method($event, $arguments)
        {
            //logic
        }
    }

?>
```

before_filter

Overview

The before_filter hook executes when the v10+ REST Service is about to route the request.

Definition

```
function before_filter($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	An empty bean object of the module being queried.
event	String	The name of the logic hook event.
arguments	Array	Additional information related to the event.
arguments[0]	Object	The SugarQuery Object
arguments[1]	Array	The query options

Considerations

- This hook can be executed for a specific module in ./custom/modules/<module>/logic_hooks.php or globally in ./custom/modules/logic_hooks.php.
- This hook can change request object parameters that influence routing.
- This hook should not be used for any type of display output.

Change Log

Version	Note
7.0.0RC1	Added before_filter hook

Example

./custom/modules/{module}/logic_hooks.php

```
<?php

    $hook_version = 1;
    $hook_array = Array();

    $hook_array['before_filter'] = Array();
    $hook_array['before_filter'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'before_filter example',

        //The PHP file where your class is located.
        'custom/modules/{module}/logic_hooks_class.php',

        //The class the method is in.
        'logic_hooks_class',

        //The method to call.
        'before_filter_method'
    );

?>
```

./custom/modules/{module}/logic_hooks_class.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class logic_hooks_class
    {
        function before_filter_method($bean, $event, $arguments)
        {
            //logic
        }
    }

?>
```

before_respond

Overview

The before_respond hook executes before the v10+ REST Service call returns data to the user.

Definition

```
function before_respond($event, $response){}
```

Arguments

Name	Type	Description
event	String	The name of the logic hook event
response	Object	The RestResponse Object

Considerations

- This is a global logic hook where the logic hook reference must be placed in ./custom/modules/logic_hooks.php.
- It is not advised to remove or unset any data. Sugar may be using this data and it can adversely affect your instance.
- This hook should not be used for any type of display output.
- This hook should be used when you want to add data to all API responses. If you are looking to modify data received from one specific endpoint, It is a much better approach to override that specific endpoint rather than to use this logic hook.

Change Log

Version	Note
7.0.0RC1	Added before_respond hook

Example

./custom/modules/logic_hooks.php

```
<?php

    $hook_version = 1;
    $hook_array = Array();

    $hook_array['before_respond'] = Array();
    $hook_array['before_respond'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'after_routing example',

        //The PHP file where your class is located.
        'custom/modules/logic_hooks_class.php',

        //The class the method is in.
        'logic_hooks_class',

        //The method to call.
        'before_respond_method'
    );

?>
```

./custom/modules/logic_hooks_class.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class logic_hooks_class
    {
        function before_respond_method($event, $response)
        {
            //logic
        }
    }

?>
```

before_routing

Overview

The before_routing hook executes when the v10+ REST Service is about to route the request.

Definition

```
function before_routing($event, $arguments){}
```

Arguments

Name	Type	Description
event	String	The name of the logic hook event
arguments	Array	Additional information related to the event
arguments.api	Object	The RestService Object
arguments.request	Object	The RestResponse Object

Considerations

- This is a global logic hook where the logic hook reference must be placed in ./custom/modules/logic_hooks.php.
- This hook can change request object parameters that influence routing.
- This hook should not be used for any type of display output.

Change Log

Version	Note
7.0.0RC1	Added before_routing hook

Example

./custom/modules/logic_hooks.php

```
<?php

    $hook_version = 1;
    $hook_array = Array();

    $hook_array['before_routing'] = Array();
    $hook_array['before_routing'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'before_routing example',

        //The PHP file where your class is located.
        'custom/modules/logic_hooks_class.php',

        //The class the method is in.
        'logic_hooks_class',

        //The method to call.
        'before_routing_method'
    );

?>
```

./custom/modules/logic_hooks_class.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class logic_hooks_class
    {
        function before_routing_method($event, $arguments)
        {
            //logic
        }
    }

?>
```

Web Logic Hooks

Overview

Web logic hooks let administrators post record and event information to a specified URL when certain sugar events take place.

The administration panel for web logic hooks can be found by navigating to Admin > Web Logic Hooks in the Sugar application. When a web logic hook is triggered, Sugar creates a record in the job queue.

Note: You must have the cron set up and running in order to process the job queue for web logic hooks. The POST of the information to the external URL will happen when the cron runs and not when the actual event occurs.

Arguments

Name	Description
URL	The URL to post JSON-encoded information
Module Name	The name of the module to which the web logic hook will apply
Trigger Event	The event that will trigger the web logic hook The following events are applicable to a web logic hook: • after_save • after_delete • after_relationship_add • after_relationship_delete • after_login • after_logout • after_login_failed
Request Method	The request method to use when sending the information The following methods may be used: • POST • GET • PUT

	• DELETE

Example

The following example shows how to receive the information posted to the web logic hook URL parameter.

http://{url}/receive.php

```
  <?php

//get the posted JSON data
$data = file_get_contents('php://input');
//decode the data
$decoded_data = json_decode(trim($data));

//use the data
$file = 'receivedData-'.time().'.txt';
file_put_contents($file, print_r($decoded_data, true));

?>
```

Result

receivedData-1380158171.txt

```
  stdClass Object
(
    [isUpdate] => 1
    [dataChanges] => Array
    (
    )

    [bean] => Account
    [data] => stdClass Object
    (
        [id] => e0623cdb-ac4b-e7ff-f681-5242e9116892
        [name] => Super Star Holdings Inc
        [date_modified] => 2013-09-25T21:16:06-04:00
        [modified_user_id] => 1
        [modified_by_name] => admin
        [created_by] => 1
        [created_by_name] => Administrator
        [description] =>
```

```
[deleted] =>
[assigned_user_id] => seed_will_id
[assigned_user_name] => Will Westin
[team_count] =>
[team_name] => Array
(
    [0] => stdClass Object
        (
            [id] => East
            [name] => East
            [name_2] =>
            [primary] => 1
        )

    [1] => stdClass Object
        (
            [id] => West
            [name] => West
            [name_2] =>
            [primary] =>
        )

)
[linkedin] =>
[facebook] =>
[twitter] =>
[googleplus] =>
[account_type] => Customer
[industry] => Energy
[annual_revenue] =>
[phone_fax] =>
[billing_address_street] => 111 Silicon Valley Road
[billing_address_city] => Los Angeles
[billing_address_state] => NY
[billing_address_postalcode] => 84028
[billing_address_country] => USA
[rating] =>
[phone_office] => (374) 791-2199
[phone_alternate] =>
[website] =>
[ownership] =>
[employees] =>
[ticker_symbol] =>
[shipping_address_street] => 111 Silicon Valley Road
[shipping_address_city] => Los Angeles
[shipping_address_state] => NY
```

```
        [shipping_address_postalcode] => 84028
        [shipping_address_country] => USA
        [email] => Array
        (
            [0] => stdClass Object
                (
                    [email_address] => example@example.tw
                    [invalid_email] =>
                    [opt_out] =>
                    [primary_address] => 1
                    [reply_to_address] =>
                )

            [1] => stdClass Object
                (
                    [email_address] => the.example@example.name
                    [invalid_email] =>
                    [opt_out] =>
                    [primary_address] =>
                    [reply_to_address] =>
                )

        )
        [email1] => example@example.tw
        [parent_id] =>
        [sic_code] =>
        [parent_name] =>
        [email_opt_out] =>
        [invalid_email] =>
        [campaign_id] =>
        [campaign_name] =>
    )
    [event] => after_save
)
```

Logic Hook Release Notes

This page documents historical changes to Sugar's logic hook libraries.

10.2.0

The following parameter was added in the 10.2.0 release:

- after_save arguments.stateChanges

7.0.0RC1

- Web Logic Hooks were introduced in this release.
- The following hooks were also added in the 7.0.0RC1 release:
    - after_routing
    - before_api_call
    - before_filter
    - before_respond
    - before_routing
- The following parameters were added in the 7.0.0RC1 release:
    - after_save arguments.isUpdate
    - after_save arguments.dataChanges
    - before_save arguments.isUpdate

6.6.0

The after_load_user hook was added in the 6.6.0 release.

6.5.0RC1

The following hooks were added in the 6.5.0RC1 release:

- after_email_import
- before_email_import
- job_failure
- job_failure_retry

6.4.5

The following hooks were added in the 6.4.5 release:

- before_relationship_add
- before_relationship_delete

6.4.4

The handle_exception hook was added in the 6.4.4 release.

6.4.3

The after_entry_point hook was added in the 6.4.3 release.

6.0.0RC1

The following hooks were added in the 6.0.0RC1 release:

- after_relationship_add
- after_relationship_delete

5.0.0a

The following hooks were added in the 5.0.0a release:

- after_login
- after_logout
- after_ui_footer
- after_ui_frame
- before_logout
- login_failed
- process_record
- server_round_trip

4.5.0c

The after_save hook was added in the 4.5.0c release.

Languages

Overview

Sugar as an application platform is internationalized and localizable. Data is stored and presented in the UTF-8 codepage, allowing for all character sets to be used. Sugar provides a language-pack framework that allows developers to build support for any language in the display of user interface labels. Each language pack has its own set of display strings which is the basis of language localization. You can add or modify languages using the information in this guide.

Please scroll to the bottom of this page for additional language topics.

Language Keys

Sugar differentiates languages with unique language keys. These keys prefix the files that correspond with particular languages. For example, the default language for the application is English (US), which is represented by the language key en_us. Any file that contains data specific to the English (US) language begins with the characters en_us. Language label keys that are not recognized will default to the English (US) version.

The following table displays the list of current languages and their corresponding keys:

Language	Language Key
Albanian (Shqip)	sq_AL
Arabic (□□□□□□□)	ar_SA
Bulgarian (Български)	bg_BG
Catalan (Català)	ca_ES
Chinese (□□□□)	zh_CN
Croatian (Hrvatski)	hr_HR
Czech (Česky)	cs_CZ
Danish (Dansk)	da_DK
Dutch (Nederlands)	nl_NL
English (UK)	en_UK
English (US)	en_us
Estonian (Eesti)	et_EE
Finnish (Suomi)	fi_FI
French (Français)	fr_FR
German (Deutsch)	de_DE
Greek (Ελληνικά)	el_EL
Hebrew (□□□□□)	he_IL
Hungarian (Magyar)	hu_HU
Italian (Italiano)	it_it
Japanese (□□□)	ja_JP
Korean (□□□)	ko_KR
Latvian (Latviešu)	lv_LV
Lithuanian (Lietuvių)	lt_LT
Norwegian (Bokmål)	nb_NO
Polish (Polski)	pl_PL
Portuguese (Português)	pt_PT
Portuguese Brazilian (Português Brasileiro)	pt_BR
Romanian (Română)	ro_RO
Russian (Русский)	ru_RU
Serbian (Српски)	sr_RS

Language	Language Key
Slovak (Slovenčina)	sk_SK
Spanish (Español)	es_ES
Spanish (Latin America) (Español (Latinoamérica))	es_LA
Swedish (Svenska)	sv_SE
Thai (□□□)	th_TH
Traditional Chinese (□□□□)	zh_TW
Turkish (Türkçe)	tr_TR
Ukrainian (Українська)	uk_UA

Change Log

The following table documents historical changes to Sugar's available languages.

Version	Change
7.8.0.0	Added ***Thai*** language pack.
7.8.0.0	Added ***Croatian*** language pack.
7.7.0.0	Added ***Traditional Chinese*** language pack.
7.6.0.0	Added ***Ukrainian*** language pack.
7.6.0.0	Added ***Arabic*** language pack.
7.2.0	Added ***Finnish*** language pack.
7.2.0	Added ***Spanish (Latin America)*** language pack.
6.6.0	Added ***Albanian*** language pack.
6.6.0	Added ***Slovak*** language pack.
6.6.0	Added ***Korean*** language pack.
6.6.0	Added ***Greek*** language pack.
6.5.1	Added ***Latvian*** language pack.
6.4.0	Added ***Serbian*** language pack.
6.4.0	Added ***Portuguese Brazilian*** language pack.
6.4.0	Added ***English (UK)*** language pack.

Version	Change
6.4.0	Added ***Catalan*** language pack.
6.2.0	Added ***Polish*** language pack.
6.2.0	Added ***Hebrew*** language pack.
6.2.0	Added ***Estonian*** language pack.
6.2.0	Added ***Czech*** language pack.
6.1.2	Added ***Turkish*** language pack.
6.1.2	Added ***Swedish*** language pack.
6.1.2	Added ***Norwegian*** language pack.
6.1.2	Added ***Lithuanian*** language pack.
6.1.0	Added ***Chinese*** language pack.
6.1.0	Added ***Russian*** language pack.
6.1.0	Added ***Romanian*** language pack.
6.1.0	Added ***Portuguese*** language pack.
6.1.0	Added ***Dutch*** language pack.
6.1.0	Added ***Japanese*** language pack.
6.1.0	Added ***Italian*** language pack.
6.1.0	Added ***Hungarian*** language pack.
6.1.0	Added ***French*** language pack.
6.1.0	Added ***Spanish*** language pack.
6.1.0	Added ***German*** language pack.
6.1.0	Added ***Danish*** language pack.
6.1.0	Added ***Bulgarian*** language pack.

Application Labels and Lists

Overview

Sugar, which is fully internationalized and localizable, differentiates languages with unique language keys. These keys prefix the files that correspond with particular languages. For example, the default language for the application is English (US), which is represented by the language key en_us. Any file that contains data specific to the English (US) language begins with the characters en_us. Language label keys that are not recognized will default to the English (US) version.

For more information on language keys, please refer to the Languages page.

Application Labels and Lists

$app_list_strings and $app_strings

The $app_list_strings array contains the various dropdown lists for the system while $app_strings contains the system application labels. The initial set of definitions can be found in ./include/language/<language key>.lang.php. As you work within the system and deploy modules and lists through Studio, any changes to these lists will be reflected in the language's extension directory: ./custom/Extension/application/Ext/Language/<language key>.<list name>.php.

Customizing Application Labels and Lists

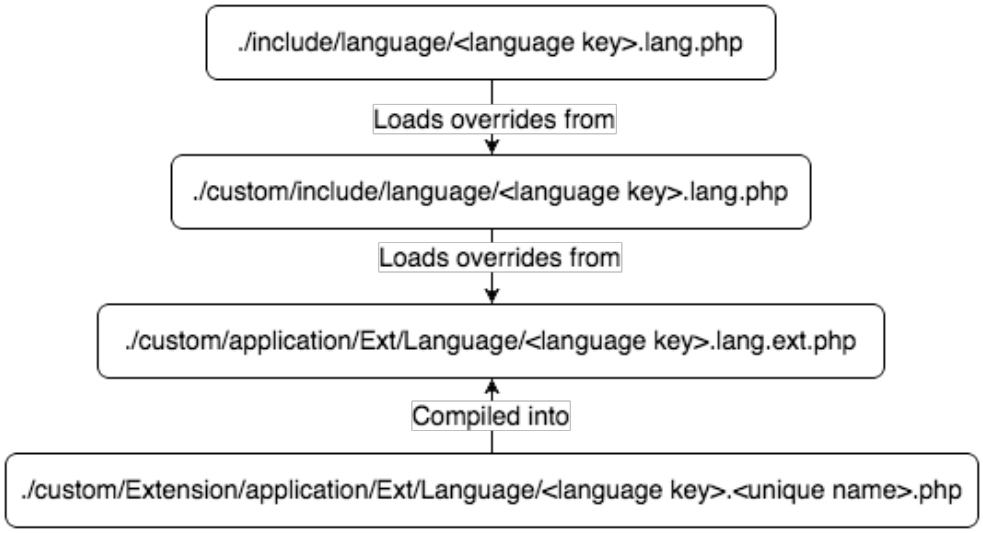
If you are developing a customization and want to be able to create or edit existing label or list values, you will need to work within the extension application directory. To do this,
create ./custom/Extension/application/Ext/Language/<language key>.<unique name>.php.

The file should contain your override values with each label index set individually. An example of this is:

```
  <?php

$app_strings['LBL_KEY'] = 'My Display Label';
$app_list_strings['LIST_NAME']['Key_Value'] = 'My Display Value';
```

Once the file is created with your adjustments, navigate to Admin > Repair > Quick Rebuild & Repair. This will compile all of the Extension files from ./custom/Extension/application/Ext/Language/ to ./custom/application/Ext/Language/<language key>.lang.ext.php.

Hierarchy Diagram



Retrieving Labels

There are two ways to retrieve a label. The first is to use the translate() function found in ./include/utils.php. This function will retrieve the label for the current user's language and can also be used to retrieve labels from $mod_strings, $app_strings, or app_list_strings.

An example of this is:

```
  require_once 'include/utils.php';
$label = translate('LBL_KEY');
```

Alternatively, you can also use the global variable $app_strings as follows:

```
  global $app_strings;

$label = '';
if (isset($app_strings['LBL_KEY']))
{
    $label = $app_strings['LBL_KEY'];
}
```

Note: If a label key is not found for the user's preferred language, the system will default to "en_us" and pull the English (US) version of the label for display.

Retrieving Lists

There are two ways to retrieve a label. The first is to use the translate() function found in include/utils.php. This function will retrieve the label for the current user's language.

An example of this is:

```
  require_once 'include/utils.php';
$list = translate('LIST_NAME');

//You can also retrieve a specific list value this way
$displayValue = translate('LIST_NAME', '', 'Key_Value');
```

Alternatively, you can also use the global variable $app_list_strings as follows:

```
  global $app_list_strings;

$list = array();
if (isset($app_list_strings['LIST_NAME']))
{
    $list = $app_list_strings['LIST_NAME'];
}
```

Note: If a list key is not found for the user's preferred language, the system will default to "en_us" and pull the English (US) version of the list for display.

Accessing Application Strings in Sidecar

All language-pack strings are accessible within the Sidecar framework.

$app_strings

To access $app_strings in Sidecar, use app.lang.getAppString:

```
  app.lang.getAppString('LBL_MODULE');
```

To access $app_strings in your browser's console, use SUGAR.App.lang.getAppString:

```
  SUGAR.App.lang.getAppString('LBL_MODULE');
```

For more information, please refer to the app.lang.getAppString section of the Languages page.

$app_list_strings

To access $app_list_strings in Sidecar, use app.lang.getAppListStrings:

```
app.lang.getAppListStrings('sales_stage_dom');
```

To access $app_list_strings in your browser's console, use SUGAR.App.lang.getAppListStrings:

```
SUGAR.App.lang.getAppListStrings('sales_stage_dom');
```

For more information, please refer to the app.lang.getAppListStrings section of the Languages page.

Module Labels

```
require_once 'include/utils.php';

$label = translate('LBL_KEY', 'Accounts');
```

Overview

Sugar, which is fully internationalized and localizable, differentiates languages with unique language keys. These keys prefix the files that correspond to particular languages. For example, the default language for the application is English (US), which is represented by the language key en_us. Any file that contains data specific to the English (US) language begins with the characters en_us. Language label keys that are not recognized will default to the English (US) version.

For more information on language keys, please refer to the Languages page.

Module Labels

$mod_strings

The module language strings are stored in $mod_strings. This section explains how the $mod_strings are compiled. All modules, whether out-of-box or custom, will have an initial set of language files in ./modules/<module>/language/<language key>.lang.php.

As you work with the system and modify labels through Studio, changes to the labels are reflected in the corresponding module's custom extension directory: ./custom/Extension/modules/<module>/Ext/Language/<language key>.lang.php.

Customizing Labels

If you are developing a customization and want to be able to create new or override existing label values, you will need to work within the extension modules directory. To do this, create ./custom/Extension/modules/<module>/Ext/Language/<language key>.<unique_name>.php.

The file should contain your override values with each label index set individually. An example of this is:

```
  <?php

$mod_strings['LBL_KEY'] = 'My Display Label';
```

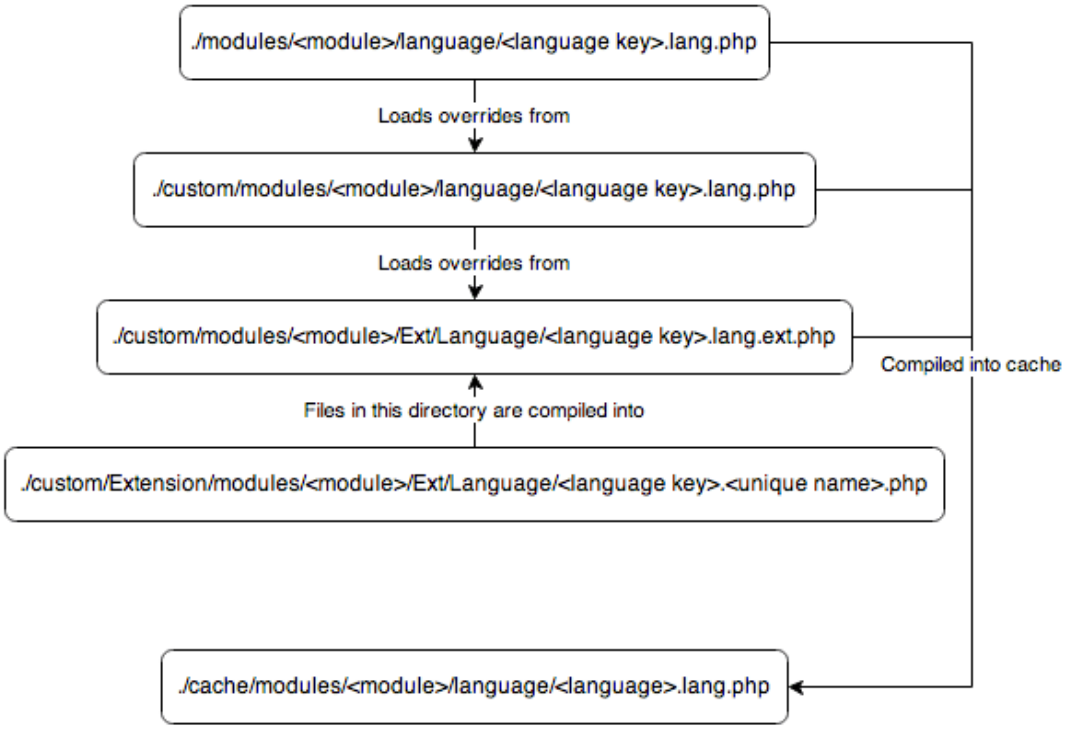
Once the file is created with your adjustments, navigate to Admin > Repair > Quick Rebuild & Repair. This will compile all of the Extension files from ./custom/Extension/modules/<module>/Ext/Language/ to ./custom/modules/<module>/Ext/Language/<language key>.lang.ext.php.

Label Cache

The file locations discussed above are compiled into the cache directory, ./cache/modules/<module>/language/<language key>.lang.php

The cached results of these files make up each module's $mod_strings definition.

Hierarchy Diagram



Retrieving Labels

There are two ways to retrieve a label. The first is to use the translate() function found in include/utils.php. This function will retrieve the label for the current user's language and can also be used to retrieve labels from $mod_strings, $app_strings, or app_list_strings.

An example of this is:

```
  require_once 'include/utils.php';

$label = translate('LBL_KEY', 'Accounts');
```

Alternatively, you can use the global variable $mod_strings as follows:

```
  global $mod_strings;

$label = '';
if (isset($mod_strings['LBL_KEY']))
{
    $label = $mod_strings['LBL_KEY'];
}
```

Accessing Module Strings in Sidecar

All language-pack strings are accessible within the Sidecar framework.

$mod_strings

To access the $mod_strings in Sidecar, use app.lang.get():

```
app.lang.get('LBL_NAME', 'Accounts');
```

To access the $mod_strings in your browser's console, use SUGAR.App.lang.get():

```
SUGAR.App.lang.get('LBL_NAME', 'Accounts');
```

For more information, please refer to the app.lang.get section of the Languages page.

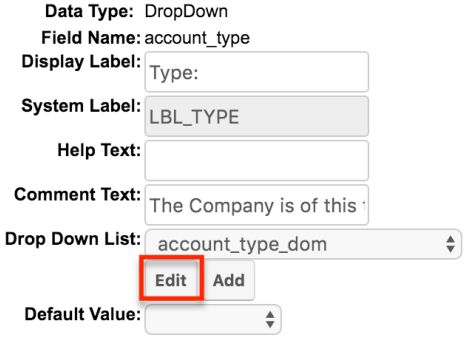
Managing Lists

Overview

There are three ways to manage lists in Sugar: by using Studio in the application, by directly modifying the list's language strings, and via the code-level Dropdown Helper. This page explains all three methods.

Managing Lists With Studio

If you know the name of the list you would like to edit, you can access the application's dropdown editor by navigating to Admin > Dropdown Editor. Alternatively, navigate to a field that uses the list (Admin > Studio > {Module} > Fields > {field_name}) and click "Edit" under the Dropdown List field:



For information on using the dropdown editor, please refer to the Developer Tools documentation in the Administration Guide.

Directly Modifying Lists

There are two ways to directly modify the language strings. The first way is to modify the custom language file, located at ./custom/include/language/<language key>.lang.php.

If you are developing a customization to be distributed and you want to be able to create new or override existing list values, you will need to work within the extension application directory. To do this you will create the following file: ./custom/Extension/application/Ext/Language/<language key>.<unique name>.php.

The file will contain your override values. Please note that within this file you will set each label index individually. An example of this is:

```
  <?php

$app_list_strings['LIST_NAME']['Key_Value'] = 'My Display Value';
```

Once the file is created with your adjustments, navigate to Admin > Repair > Quick Rebuild & Repair. This will compile all of the Extension files from ./custom/Extension/application/Ext/Language/ to ./custom/application/Ext/Language/<language key>.lang.ext.php.

Managing Lists With Dropdown Helper

You can use the dropdown helper to manage lists at the code level. This example demonstrates how to add and update values for a specific dropdown list:

```
  require_once 'modules/Studio/DropDowns/DropDownHelper.php';

$dropdownHelper = new DropDownHelper();

$parameters = array();
$parameters['dropdown_name'] = 'example_list';

$listValues = array(
    'Key_Value_1' => 'Display Value 1',
    'Key_Value_2' => 'Display Value 2',
    'Key_Value_3' => 'Display Value 3'
```

```
);

$count = 0;
foreach ($listValues as $key=>$value) {
    $parameters['slot_'. $count] = $count;
    $parameters['key_'. $count] = $key;
    $parameters['value_'. $count] = $value;
    //set 'use_push' to true to update/add values while keeping old va
lues
    $parameters['use_push'] = true;
    $count++;
}

$dropdownHelper->saveDropDown($parameters);
```

Language Packs

Overview

Language packs are module-loadable packages that add support for new, localized languages to Sugar.

Creating a Language Pack

To create a language pack, choose a unique language key. This key is specific to your language definitions and should be unique from other language keys in the same instance to avoid conflicts. It is also important that your language key follows the xx_xx format. For demonstrative purposes, we will create a Lorem Ipsum language pack with the language key Lo_Ip.

Application and Module Strings

In the module and application language definitions, there is a combination of $app_list_strings, $mod_strings, and $mod_process_order_strings variables. Your custom language needs to have a definition created to reflect each language key in a standard definition. To do this, duplicate an existing language and simply change the language values within the document and replace the language key in the name of the file with your new key. Be sure to only change the array values and leave the array keys as they are inside of each file.

Note: Should you miss an index, it will default to English.

The first step is to identify all of the application and module language files. While language files may vary from version to version due to new

features, the stock language paths are generally as follows:

- ./include/language/xx_xx.lang.php
- ./include/SugarObjects/implements/assignable/language/xx_xx.lang.php
- ./include/SugarObjects/implements/email_address/language/xx_xx.lang.php
- ./include/SugarObjects/implements/team_security/language/xx_xx.lang.php
- ./include/SugarObjects/templates/basic/language/xx_xx.lang.php
- ./include/SugarObjects/templates/company/language/xx_xx.lang.php
- ./include/SugarObjects/templates/company/language/application/xx_xx.lang.php
- ./include/SugarObjects/templates/file/language/xx_xx.lang.php
- ./include/SugarObjects/templates/file/language/application/xx_xx.lang.php
- ./include/SugarObjects/templates/issue/language/xx_xx.lang.php
- ./include/SugarObjects/templates/issue/language/application/xx_xx.lang.php
- ./include/SugarObjects/templates/person/language/xx_xx.lang.php
- ./include/SugarObjects/templates/sale/language/xx_xx.lang.php
- ./include/SugarObjects/templates/sale/language/application/xx_xx.lang.php
- ./install/language/xx_xx.lang.php
- ./modules/Accounts/language/xx_xx.lang.php
- ./modules/ACL/language/xx_xx.lang.php
- ./modules/ACLActions/language/xx_xx.lang.php
- ./modules/ACLFields/language/xx_xx.lang.php
- ./modules/ACLRoles/language/xx_xx.lang.php
- ./modules/Activities/language/xx_xx.lang.php
- ./modules/ActivityStream/Activities/language/xx_xx.lang.php
- ./modules/Administration/language/xx_xx.lang.php
- ./modules/Audit/language/xx_xx.lang.php
- ./modules/Bugs/language/xx_xx.lang.php
- ./modules/Calendar/language/xx_xx.lang.php
- ./modules/Calls/language/xx_xx.lang.php
- ./modules/CampaignLog/language/xx_xx.lang.php
- ./modules/Campaigns/language/xx_xx.lang.php
- ./modules/CampaignTrackers/language/xx_xx.lang.php
- ./modules/Cases/language/xx_xx.lang.php
- ./modules/Charts/language/xx_xx.lang.php
- ./modules/Configurator/language/xx_xx.lang.php
- ./modules/Connectors/language/xx_xx.lang.php
- ./modules/Contacts/language/xx_xx.lang.php
- ./modules/Contracts/language/xx_xx.lang.php
- ./modules/ContractTypes/language/xx_xx.lang.php
- ./modules/Currencies/language/xx_xx.lang.php
- ./modules/CustomQueries/language/xx_xx.lang.php
- ./modules/DataSets/language/xx_xx.lang.php
- ./modules/DocumentRevisions/language/xx_xx.lang.php
- ./modules/Documents/language/xx_xx.lang.php
- ./modules/DynamicFields/language/xx_xx.lang.php

- ./modules/EAPM/language/xx_xx.lang.php
- ./modules/EmailAddresses/language/xx_xx.lang.php
- ./modules/EmailMan/language/xx_xx.lang.php
- ./modules/EmailMarketing/language/xx_xx.lang.php
- ./modules/Emails/language/xx_xx.lang.php
- ./modules/EmailTemplates/language/xx_xx.lang.php
- ./modules/Employees/language/xx_xx.lang.php
- ./modules/ExpressionEngine/language/xx_xx.lang.php
- ./modules/Expressions/language/xx_xx.lang.php
- ./modules/Feedbacks/language/xx_xx.lang.php
- ./modules/Filters/language/xx_xx.lang.php
- ./modules/ForecastManagerWorksheets/language/xx_xx.lang.php
- ./modules/Forecasts/language/xx_xx.lang.php
- ./modules/ForecastWorksheets/language/xx_xx.lang.php
- ./modules/Groups/language/xx_xx.lang.php
- ./modules/Help/language/xx_xx.lang.php
- ./modules/History/language/xx_xx.lang.php
- ./modules/Holidays/language/xx_xx.lang.php
- ./modules/Home/language/xx_xx.lang.php
- ./modules/Import/language/xx_xx.lang.php
- ./modules/InboundEmail/language/xx_xx.lang.php
- ./modules/KBDocuments/language/xx_xx.lang.php
- ./modules/KBTags/language/xx_xx.lang.php
- ./modules/LabelEditor/language/xx_xx.lang.php
- ./modules/Leads/language/xx_xx.lang.php
- ./modules/MailMerge/language/xx_xx.lang.php
- ./modules/Manufacturers/language/xx_xx.lang.php
- ./modules/Meetings/language/xx_xx.lang.php
- ./modules/MergeRecords/language/xx_xx.lang.php
- ./modules/ModuleBuilder/language/xx_xx.lang.php
- ./modules/Notes/language/xx_xx.lang.php
- ./modules/Notifications/language/xx_xx.lang.php
- ./modules/OAuthKeys/language/xx_xx.lang.php
- ./modules/OAuthTokens/language/xx_xx.lang.php
- ./modules/Opportunities/language/xx_xx.lang.php
- ./modules/OptimisticLock/language/xx_xx.lang.php
- ./modules/PdfManager/language/xx_xx.lang.php
- ./modules/pmse_Business_Rules/language/xx_xx.lang.php
- ./modules/pmse_Emails_Templates/language/xx_xx.lang.php
- ./modules/pmse_Inbox/language/xx_xx.lang.php
- ./modules/pmse_Project/language/xx_xx.lang.php
- ./modules/ProductBundleNotes/language/xx_xx.lang.php
- ./modules/ProductBundles/language/xx_xx.lang.php
- ./modules/ProductCategories/language/xx_xx.lang.php
- ./modules/Products/language/xx_xx.lang.php
- ./modules/ProductTemplates/language/xx_xx.lang.php

- ./modules/ProductTypes/language/xx_xx.lang.php
- ./modules/Project/language/xx_xx.lang.php
- ./modules/ProjectTask/language/xx_xx.lang.php
- ./modules/ProspectLists/language/xx_xx.lang.php
- ./modules/Prospects/language/xx_xx.lang.php
- ./modules/Quotas/language/xx_xx.lang.php
- ./modules/Quotes/language/xx_xx.lang.php
- ./modules/Relationships/language/xx_xx.lang.php
- ./modules/Releases/language/xx_xx.lang.php
- ./modules/ReportMaker/language/xx_xx.lang.php
- ./modules/Reports/language/xx_xx.lang.php
- ./modules/RevenueLineItems/language/xx_xx.lang.php
- ./modules/Roles/language/xx_xx.lang.php
- ./modules/SavedSearch/language/xx_xx.lang.php
- ./modules/Schedulers/language/xx_xx.lang.php
- ./modules/SchedulersJobs/language/xx_xx.lang.php
- ./modules/Shippers/language/xx_xx.lang.php
- ./modules/SNIP/language/xx_xx.lang.php
- ./modules/Studio/language/xx_xx.lang.php
- ./modules/Styleguide/language/xx_xx.lang.php
- ./modules/SugarFavorites/language/xx_xx.lang.php
- ./modules/Sync/language/xx_xx.lang.php
- ./modules/Tasks/language/xx_xx.lang.php
- ./modules/TaxRates/language/xx_xx.lang.php
- ./modules/TeamNotices/language/xx_xx.lang.php
- ./modules/Teams/language/xx_xx.lang.php
- ./modules/TimePeriods/language/xx_xx.lang.php
- ./modules/Trackers/language/xx_xx.lang.php
- ./modules/UpgradeWizard/language/xx_xx.lang.php
- ./modules/Users/language/xx_xx.lang.php
- ./modules/UserSignatures/language/xx_xx.lang.php
- ./modules/WebLogicHooks/language/xx_xx.lang.php
- ./modules/WorkFlow/language/xx_xx.lang.php
- ./modules/WorkFlowActions/language/xx_xx.lang.php
- ./modules/WorkFlowActionShells/language/xx_xx.lang.php
- ./modules/WorkFlowAlerts/language/xx_xx.lang.php
- ./modules/WorkFlowAlertShells/language/xx_xx.lang.php
- ./modules/WorkFlowTriggerShells/language/xx_xx.lang.php

Dashlet Strings

Dashlet strings are mostly specific to BWC dashboards. Within each dashlet language definition is a $dashletStrings variable. Create a definition to mimic each dashlet file to reflect the new language. To do this, duplicate an existing language of your choice and change the language key in the path. Be sure to only change the array values and leave the array keys as they are inside of each file.

The dashlet paths are listed below:

- ./modules/Calendar/Dashlets/CalendarDashlet/CalendarDashlet.xx_xx.lang.php
- ./modules/Charts/Dashlets/CampaignROIChartDashlet/CampaignROIChartDashlet.xx_xx.lang.php
- ./modules/Charts/Dashlets/MyModulesUsedChartDashlet/MyModulesUsedChartDashlet.xx_xx.lang.php
- ./modules/Charts/Dashlets/MyOpportunitiesGaugeDashlet/MyOpportunitiesGaugeDashlet.xx_xx.lang.php
- ./modules/Charts/Dashlets/MyPipelineBySalesStageDashlet/MyPipelineBySalesStageDashlet.xx_xx.lang.php
- ./modules/Charts/Dashlets/MyTeamModulesUsedChartDashlet/MyTeamModulesUsedChartDashlet.xx_xx.lang.php
- ./modules/Charts/Dashlets/OpportunitiesByLeadSourceByOutcomeDashlet/OpportunitiesByLeadSourceByOutcomeDashlet.xx_xx.lang.php
- ./modules/Charts/Dashlets/OpportunitiesByLeadSourceDashlet/OpportunitiesByLeadSourceDashlet.xx_xx.lang.php
- ./modules/Charts/Dashlets/OutcomeByMonthDashlet/OutcomeByMonthDashlet.xx_xx.lang.php
- ./modules/Charts/Dashlets/PipelineBySalesStageDashlet/PipelineBySalesStageDashlet.xx_xx.lang.php
- ./modules/Home/Dashlets/ChartsDashlet/ChartsDashlet.xx_xx.lang.php
- ./modules/Home/Dashlets/InvadersDashlet/InvadersDashlet.xx_xx.lang.php
- ./modules/Home/Dashlets/JotPadDashlet/JotPadDashlet.xx_xx.lang.php
- ./modules/Home/Dashlets/RSSDashlet/RSSDashlet.xx_xx.lang.php
- ./modules/SugarFavorites/Dashlets/SugarFavoritesDashlet/SugarFavoritesDashlet.xx_xx.lang.php
- ./modules/TeamNotices/Dashlets/TeamNoticesDashlet/TeamNoticesDashlet.xx_xx.lang.php
- ./modules/Trackers/Dashlets/TrackerDashlet/TrackerDashlet.xx_xx.lang.php

Templates

Sugar also contains templates that are used when emailing users. Duplicate an existing language template and change the language text as needed. This time, you will need to leave the language keys exactly as they are in the file.

The template paths are listed below:

- ./include/language/xx_xx.notify_template.html

Configuration

Once you have your language definitions ready, you will need to tell Sugar a new

language should be listed for users. For development purposes, you can add $sugar_config['languages']['Lo_Ip'] = 'Lorem Ipsum'; to your config_override.php. It is important to note that this language configuration will automatically be set for you during the installation of a module loadable language package.

Module Loadable Packages

Once you have the new language files ready, create an installer package so the files can be installed to a new instance. To do this, create an empty directory and move the files into it, mimicking the folder structures shown above. Once that is completed, create a manifest.php in the root of the new directory with a $manifest['type'] of "langpack". An example manifest file is shown below this section. For more information on building manifests, please visit the introduction to the manifest page.

Note: Sugar Sell Essentials customers do not have the ability to upload custom file packages to Sugar using Module Loader.

Example Manifest File

```
  <?php

$manifest = array (
  'acceptable_sugar_versions' =>
  array (
    'regex_matches' =>
    array (
      '12.0.*',
    ),
  ),
  'acceptable_sugar_flavors' =>
  array (
    'PRO',
    'ENT',
    'ULT',
  ),
  'readme' => '',
  'key' => 1454474352,
  'author' => 'SugarCRM',
  'description' => 'Installs an example Lorem Ipsum language pack',
  'icon' => '',
  'is_uninstallable' => true,
  'name' => 'Lorem Ipsum Language Pack',
  'published_date' => '2018-02-03 00:00:00',
  'type' => 'langpack',
```

```
  'version' => 1454474352,
  'remove_tables' => '',
);

?>
```

Once your manifest is completed, you will need to zip up the contents of your new folder. An example of a language pack installer can be downloaded here. When your language pack is ready, it can be installed through the module loader. The installation will automatically add your new language to the $sugar_config['languages'] array in your config.php. After installation. it is highly recommended to navigate to the Sugar Administration page, click on "Repairs", and execute the following repair tools:

- Quick Repair and Rebuild
- Rebuild Javascript Languages

The new language should now be available for use in your Sugar instance. If you do not see the language listed, please clear your browser cache and refresh the page.

Extensions

Overview

The extension framework, defined in ./ModuleInstall/extensions.php, provides the capability to modify Sugar metadata such as vardefs and layouts in a safe way that supports installing, uninstalling, enabling, and disabling without interfering with other customizations.

Application extensions are stored under ./custom/Extension/application/Ext/ and module extensions are under ./custom/Extension/modules/<module>/Ext/. The files in each of these directories are aggregated into a single file with a predefined name for the system to use. An example of this is the vardefs extension. The vardef extension directory for Accounts is located in ./custom/Extension/modules/Accounts/Ext/Vardefs/. When a module is installed, uninstalled, enabled, or disabled, the files contained in this directory are merged into ./custom/modules/Accounts/Ext/Vardefs/vardefs.ext.php. A Quick Repair & Rebuild will also cause the files to merge.

The core extension mappings are listed in the Topics section at the bottom of this page.

Extensions Properties

Each extension contains the following properties:

Property	Description
Extension Scope	• **All** : Extension can be applied to the ./custom/application/ or ./custom/<module>/ directory • **Application** : Extension can only be applied to the ./custom/application/ directory • **Module** : Extension can only be applied to the ./custom/<module>/ directory
Definition Variable	The variable that Sugar for utilizing the extension definition. If defined, this variable must be set with the appropriate definition properties.
Extension Directory	The directory that the extension compiles files from • If the customization is for the application, the extension file should be placed in ./custom/Extension/application/Ext/<extension_directory>/ • If the customization is for a module, the extension file should be placed in ./custom/Extension/modules/<module>/Ext/<extension_directory>/
Compiled Extension File	The name of the compiled extension file • If the extension is for the application, the compiled file will be located in ./custom/application/Ext/<extension>/<extension>.ext.php • If the extension is for a module, the compiled file will be located in ./custom/modules/<module>/Ext/<extension>/<extension>.ext.php

Manifest Installdef	The index of the $installdef in the manifest file for module loadable packages.

ActionFileMap

Overview

The ActionFileMap extension maps actions to files, which helps you map a file to a view outside of ./custom/modules/<module>/views/view.<name>.php. This page is only applicable to modules running in backward compatibility mode.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	**Value**
Extension Scope	Module
Sugar Variable	$action_file_map
Extension Directory	./custom/Extension/modules/<module>/Ext/ActionFileMap/
Compiled Extension File	./custom/<module>/Ext/ActionFileMap/action_file_map.ext.php
Manifest Installdef	$installdefs['action_file_map']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in ./custom/Extension/modules/<module>/Ext/ActionFileMap/ to map a new action in the system. The following example will create a new action called "example" in a module:

./custom/Extension/modules/<module>/Ext/ActionFileMap/<file>.php

```
<?php
```

```
$action_file_map['new_action'] = 'custom/modules/<module>/new_action.p
hp';
```

Next, create your action file:

./custom/modules/<module>/new_action.php

```
<?php

//Encoded as JSON for AJAX layouts
echo '{"content":"Example View"}';

?>
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and compile your customization into ./custom/modules/<module>/Ext/ActionFileMap/action_file_map.ext.php

Module Loadable Package

When building a module-loadable package, you can use the $installdefs['action_file_map'] index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The basepath of the file to be installed.
to_module	String	The key of the module the file is to be installed to.

The example below demonstrates the proper install definition that should be used in the ./manifest.php file in order to add the Action File Map file to a specific module. You should note that when using this approach, you still need to use the $installdefs['copy'] index for the Action file, but Sugar will automatically execute Rebuild Extensions to reflect the new Action in the system.

./manifest.php

```
<?php
```

```
$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'ActionRemap_Example',
    'action_file_map' => array(
        array(
            'from' => '<basepath>/Files/custom/Extension/modules/<modu
le>/Ext/ActionFileMap/<file>.php',
            'to_module' => '<module>',
        )
    ),
    'copy' => array(
        array(
            'from' => '<basepath>/Files/custom/example.php',
            'to' => 'custom/example.php'
        )
    )
);
```

Alternatively, you may use the $installdefs['copy'] index for the Action File Map Extension file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the $installdefs['copy'] index and module-loadable packages, please refer to the Introduction to the Manifest page.

ActionReMap

Overview

The ActionReMap extension maps new actions to existing actions. This extension is only applicable to modules running in backward compatibility mode.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value

Extension Scope	Module
Sugar Variable	$action_remap
Extension Directory	./custom/Extension/modules/<module>/Ext/ActionReMap/
Compiled Extension File	./custom/<module>/Ext/ActionReMap/action_remap.ext.php
Manifest Installdef	$installdefs['action_remap']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the file system, you can create a file in ./custom/Extension/modules/<module>/Ext/ActionReMap/ to map an action to another defined action. The following example will map the action 'example' to 'detailview':

./custom/Extension/modules/<module>/Ext/ActionReMap/<file>.php

```
<?php

$action_remap['example'] = 'detailview';
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and compile your customization into ./custom/modules/<module>/Ext/ActionReMap/action_remap.ext.php.

Module Loadable Package

When building a module loadable package, you can use the $installdefs['action_remap'] index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The basepath of the file to be installed
to_module	String	The key for the module

		where the file will be installed

The example below demonstrates the proper install definition that should be used in the ./manifest.php file in order to add the Action Remap file to a specific module. You should note that when using this approach, Sugar will automatically execute Rebuild Extensions to reflect your changes in the system.

./manifest.php

```
<?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'ActionRemap_Example',
    'action_remap' => array(
        array(
            'from' => '<basepath>/Files/custom/Extension/modules/<modu
le>/Ext/ActionReMap/<file>.php',
            'to_module' => '<module>',
        )
    )
);
```

Alternatively, you may use the $installdefs['copy'] index. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the $installdefs['copy'] index and module-loadable packages, please refer to the Introduction to the Manifest page.

ActionViewMap

Overview

The ActionViewMap extension maps additional actions for a module.

Note: Actions that apply to modules running in backward compatibility mode are mapped in ./custom/modules/<module>/controller.php.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Module
Sugar Variable	$action_view_map
Extension Directory	./custom/Extension/modules/<module>/Ext/ActionViewMap/
Compiled Extension File	./custom/<module>/Ext/ActionViewMap/action_view_map.ext.php
Manifest Installdef	$installdefs['action_view_map']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in ./custom/Extension/modules/<module>/Ext/ActionViewMap/ to map a new view in the system. The following example will map a new action called 'example' to the 'example' view:

./custom/Extension/modules/<module>/Ext/ActionViewMap/<file>.php

```
<?php

$action_view_map['example'] = 'example';
```

./custom/modules/<module>/views/view.example.php

```
<?php

if(!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry Point
');

require_once('include/MVC/View/views/view.detail.php');
```

```
class <module>ViewExample extends ViewDetail
{
    function <module>ViewExample()
    {
        parent::ViewDetail();
    }
    function display()
    {
        echo 'Example View';
    }
}
?>
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and compile your customization into ./custom/modules/<module>/Ext/ActionViewMap/action_view_map.ext.php.

Module Loadable Package

When building a module-loadable package, use the $installdefs['action_view_map'] index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The basepath of the file
to_module	String	The key for the module where the file will be installed

The example below demonstrates the proper install definition for the ./manifest.php file in order to add the Action View Map file to a specific module. You should note that when using this approach, you still need to use the $installdefs['copy'] index for the View file, however, Sugar will automatically execute Rebuild Extensions to reflect the new Action View in the system.

./manifest.php

```
<?php

$manifest = array(
    ...
);
```

```
$installdefs = array(
    'id' => 'actionView_example',
    'action_view_map' => array(
        array(
            'from' => '<basepath>/Files/custom/Extension/modules/<modu
le>/Ext/ActionViewMap/<file>.php',
            'to_module' => '<module>',
        )
    ),
    'copy' => array(
        array(
            'from' => '<basepath>/Files/custom/modules/<module>/views/
view.example.php',
            'to' => 'custom/modules/<module>/views/view.example.php',
        ),
    )
);
```

Alternatively, you may use the $installdefs['copy'] index for the Action View Map Extension file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild.

For more information on module-loadable packages, please refer to the Introduction to the Manifest page .

Administration

Overview

The Administration extension adds new panels to Sugar's Administration page.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Module: Administration
Sugar Variable	$admin_group_header

Extension Directory	./custom/Extension/modules/Administration/Ext/Administration/
Compiled Extension File	./custom/modules/Administration/Ext/Administration/administration.ext.php
Manifest Installdef	$installdefs['administration']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in ./custom/Extension/modules/Administration/Ext/Administration/<file>.php to add new Administration Links in the system. The following example will add a new panel to the Administration page called "Example Admin Panel", which will contain one link called "Example Link":

The following example will create a new admin panel:

./custom/Extension/modules/Administration/Ext/Administration/<file>.php

```
    <?php

    $admin_option_defs = array();
    $admin_option_defs['Administration']['<section key>'] = array(
        //Icon name. Available icons are located in ./themes/default/i
mages
        'Administration',

        //Link name label
        'LBL_LINK_NAME',

        //Link description label
        'LBL_LINK_DESCRIPTION',

        //Link URL - For Sidecar modules
        'javascript:void(parent.SUGAR.App.router.navigate("<module>/<p
ath>", {trigger: true}));',

        //Alternatively, if you are linking to BWC modules
        //'./index.php?module=<module>&action=<action>',
    );
```

```
$admin_group_header[] = array(
    //Section header label
    'LBL_SECTION_HEADER',

    //$other_text parameter for get_form_header()
    '',

    //$show_help parameter for get_form_header()
    false,

    //Section links
    $admin_option_defs,

    //Section description label
    'LBL_SECTION_DESCRIPTION'
);
```

Next, we will populate the panel label values:

./custom/Extension/modules/Administration/Ext/Language/en_us.<name>.php

```
<?php

$mod_strings['LBL_LINK_NAME'] = 'Example Link';
$mod_strings['LBL_LINK_DESCRIPTION'] = 'Link Description';
$mod_strings['LBL_SECTION_HEADER'] = 'Example Admin Panel';
$mod_strings['LBL_SECTION_DESCRIPTION'] = 'Section Description';
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions, compiling your customization into ./custom/modules/Administration/Ext/Administration/administration.ext.php, and the new panel will appear in the Administration section.

Module Loadable Package

When building a module loadable package, use the $installdefs['administration'] index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The base path of the file to be installed

The example below demonstrates the proper install definition that should be used in the ./manifest.php file in order to add the Administration file to the system. If you are utilizing a Language file, as recommended above, you must use the $installdefs['language'] index to install the Language definition. Using the $installdefs['administration'] index will automatically execute Rebuild Extensions to reflect the new Administration Links in the system.

```
    <?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'administration_example',
    'administration' => array(
        array(
           'from' => '<basepath>/custom/Extension/modules/Administrati
on/Ext/Administration/<file>.php'
        )
    ),
    'language' => array(
        array(
            'from' => '<basepath>/custom/Extensions/modules/Administra
tion/Ext/Language/en_us.<file>.php',
            'to_module' => 'Administration',
            'language' =>'en_us'
        )
    )
);
```

Alternatively, you may also choose to use the $installdefs['copy'] index for the Administration Link Extension file. When using this approach, you may need to manually run a repair action such as a Quick Repair and Rebuild.

For more information on the $installdefs['copy'] index and module-loadable packages, please refer to the Introduction to the Manifest page.

Application Schedulers

Overview

Application Scheduler extensions add custom functions that can be used by scheduler jobs.

Note: This Extension is a duplicate of the ScheduledTasks extension, which typically places Scheduled Tasks in the Schedulers directory.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Application
Sugar Variable	$job_strings
Extension Directory	./custom/Extension/application/Ext/ScheduledTasks/
Compiled Extension File	./custom/application/Ext/ScheduledTasks/scheduledtasks.ext.php
Manifest Installdef	$installdefs['appscheduledefs']

Implementation

The following sections illustrates the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in ./custom/Extension/application/Ext/ScheduledTasks/ to add a new Scheduler Task to the system. The following example will create a new Scheduler Task 'example_job':

./custom/Extension/application/Ext/ScheduledTasks/<file>.php

```
<?php

$job_strings[] = 'exampleJob';
```

```
function exampleJob()
{
    //logic here

    //return true for completed
    return true;
}
```

Next create the Language file for the Scheduler, so that the Job properly displays in Admin > Schedulers section:

./custom/Extension/modules/Schedulers/Ext/Language/<language>.<file>.php

```
<?php
//Label will be LBL_[upper case function name]
$mod_strings['LBL_EXAMPLEJOB'] = 'Example Job';
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and the customizations will be compiled into ./custom/application/Ext/ScheduledTasks/scheduledtasks.ext.php

Module Loadable Package

When building a module loadable package, you can use the $installdefs['appscheduledefs'] index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The base path of the file to be installed.

The example below demonstrates the proper install definition that should be used in the ./manifest.php file in order to add the custom Scheduler definition file to the system. You should note that, when using this approach, Sugar will automatically execute Rebuild Extensions to reflect the new scheduler in the system.

./manifest.php

```
<?php
```

```
$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'appScheduledTasks_example',
    'appscheduledefs' => array(
        array(
            'from' => '<basepath>/Files/custom/Extension/application/E
xt/ScheduledTasks/<file>.php',
        )
    ),
    'language' => array(
        array(
            'from' =>'<basepath>/Files/custom/Extension/modules/Schedu
lers/Ext/Language/<file>.php',
            'to_module' => 'Schedulers',
            'language' => 'en_us'
        )
    )
);
```

Alternatively, you may use the $installdefs['copy'] index for the Scheduled Task Extension file. When using this approach, you may need to manually run a repair action such as a Quick Repair and Rebuild.

For more information about the $installdefs['copy'] index and module-loadable packages, please refer to the Introduction to the Manifest page.

Console

Overview

The Console extension adds custom CLI commands to Sugar. More information on creating custom commands can be found in the CLI documentation.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Application
Extension Directory	./custom/Extension/application/Ext/Console/
Compiled Extension File	./custom/application/Ext/Console/console.ext.php
Manifest Installdef	$installdefs['console']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

To create a Sugar console extension, please refer to our CLI documentation.

Module Loadable Package

When building a module loadable package, you can use the $installdefs['console'] index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The basepath of the file to be installed.

The example below demonstrates the proper install definition that should be used in the ./manifest.php file in order to add the console command.

./manifest.php

```
<?php

$manifest = array(
    ...
);

$installdefs = array (
  'id' => 'console_Example',
  'console' => array(
      array(
```

```
          'from' => '<basepath>/Files/custom/Extension/application/Ext
/Console/RegisterHelloWorldCommand.php'
      )
  ),
  'copy' => array (
    0 => array (
      'from' => '<basepath>/Files/custom/src/Console/Command/HelloWorl
dCommand.php',
      'to' => 'custom/src/Console/Command/HelloWorldCommand.php',
    ),
  ),
);
```

Alternatively, you may use the $installdefs['copy'] index to copy the file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the $installdefs['copy'] index and module-loadable packages, please refer to the Introduction to the Manifest page.

Download the module loadable example package here.

Dependencies

Overview

Dependencies create dependent actions for fields and forms that can leverage more complicated logic.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Module
Sugar Variable	$dependencies
Extension Directory	./custom/Extension/modules/<module>/Ext/Dependencies/
Compiled Extension File	./custom/modules/<module>/Ext/Depen

	dencies/deps.ext.php
Manifest Installdef	$installdefs['dependencies']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in ./custom/Extension/modules/<module>/Ext/Dependencies/<file>.php to map a new dependency in the system. The following example will create a new required field dependency on a module that is evaluated upon editing a record:

./custom/Extension/modules/<module>/Ext/Dependencies/<file>.php

```
  <?php

$dependencies['<module>']['<unique name>'] = array(
    'hooks' => array("edit"),
    //Trigger formula for the dependency. Defaults to 'true'.
    'trigger' => 'true',
    'triggerFields' => array('<trigger field>'),
    'onload' => true,
    //Actions is a list of actions to fire when the trigger is true
    'actions' => array(
        array(
            'name' => 'SetRequired', //Action type
            //The parameters passed in depend on the action type
            'params' => array(
                'target' => '<field>',
                'label'  => '<field label>', //normally <field>_label
                'value' => 'equal($<trigger field>, "Closed")', //Form
ula
            ),
        ),
    ),
);
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions, compiling your customization into./custom/modules/<module>/Ext/Dependencies/deps.ext.php, and the

dependency will be in place.

Module Loadable Package

When building a module loadable package, you can use the $installdefs['dependencies'] index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The base path of the file
to_module	String	The key for the module where the file will be installed

The example below demonstrates the proper install definition that should be used in the ./manifest.php file in order to add the Dependency file to a specific module. You should note that when using this approach, Sugar will automatically execute Rebuild Extensions to reflect the new Dependency in the system.

./manifest.php

```
  <?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'dependency_example',
    'dependencies' => array(
        array(
            'from' => '<basepath>/Files/custom/Extension/modules/<modu
le>/Ext/Dependencies/<file>.php',
            'to_module' => '<module>',
        )
    )
);
```

Alternatively, you may use the $installdefs['copy'] index for the Dependency Extension file. When using this approach, you may need to manually run a repair action such as a Quick Repair and Rebuild.

For more information on the $installdefs['copy'] index and module-loadable packages, please refer to the Introduction to the Manifest page.

EntryPointRegistry

Overview

The EntryPointRegistry extension maps additional entry points to the system. Please note that entry points will soon be deprecated. Developers should move custom logic to endpoints. For more information, please refer to the Entry Points page.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Application
Sugar Variable	$entry_point_registry
Extension Directory	./custom/Extension/application/Ext/EntryPointRegistry/
Compiled Extension File	./custom/application/Ext/EntryPointRegistry/entry_point_registry.ext.php
Manifest Installdef	$installdefs['entrypoints']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in ./custom/Extension/application/Ext/EntryPointRegistry/ to map a new entry point in the system. The following example will create a new entry point called exampleEntryPoint that will display the text, "Hello World":

./custom/Extension/application/Ext/EntryPointRegistry/<file>.php

```
<?php
```

```
$entry_point_registry['exampleEntryPoint'] = array(
    'file' => 'custom/exampleEntryPoint.php',
    'auth' => true
);
```

Next, create the file that will contain the entry point logic. This file can be located anywhere you choose, but we recommend putting it in the custom directory. More information on custom entry points can be found on the Creating Custom Entry Points page.

./custom/exampleEntryPoint.php

```
<?php

if(!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry Point
');

echo "Hello World!";
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild.The system will then rebuild the extensions and compile your customization into ./custom/application/Ext/EntryPointRegistry/entry_point_registry.ext.php

Module Loadable Package

When building a module loadable package, you can use the $installdefs['entrypoints'] index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The base path of the file to be installed.

The example below demonstrates the proper install definition that should be used in the ./manifest.php file, in order to add the Entry Point Extension file to the system. You should note that when using this approach, you still need to use the $installdefs['copy'] index for the entry point's logic file, however Sugar will automatically execute Rebuild Extensions to reflect the new Entry Point in the system.

./manifest.php

```
<?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'entryPoint_Example',
    'entrypoints' => array(
        array(
            'from' => '<basepath>/Files/custom/Extension/application/E
xt/EntryPointRegistry/<file>.php'
        )
    ),
    'copy' => array(
        array(
            'from' => '<basepath>/Files/custom/exampleEntryPoint.php',
            'to' => 'custom/exampleEntryPoint.php'
        )
    )
);
```

Alternatively, you may use the $installdefs['copy'] index for the Entry Point Extension file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the $installdefs['copy'] index and module loadable packages, please refer to the Introduction to the Manifest page.

Extensions

Overview

This extension allows for developers to create custom extensions within the framework. Custom extensions are used alongside the extensions found in ./ModuleInstaller/extensions.php.

Properties

The following extension properties are available. For more information, please

refer to the Extension Property documentation.

Property	Value
Extension Scope	Application
Sugar Variable	$extensions
Extension Directory	./custom/Extension/application/Ext/Extensions/
Compiled Extension File	./custom/application/Ext/Extensions/extensions.ext.php
Manifest Installdef	$installdefs['extensions']

Parameters

- **section** : Section name in the manifest file
- **extdir** : The directory containing the extension files
- **file** : The name of the file where extension files are compiled into
- **module** : Determines how the framework will interpret the extension (optional)
    - **<Empty>** : Will enable the extension for all modules
        - **Ext** : ./custom/Extension/modules/<module>/Ext/<Custom Extension>/
        - **Ext File** : ./custom/modules/<module>/Ext/<Extension Name>.ext.php
    - **<Specific Module>** : Will enable the extension for the specified module
        - **Ext Directory** : ./custom/Extension/modules/<Specific Module>/Ext/<Custom Extension>/
        - **Ext File** : ./custom/modules/<Specific Module>/Ext/<Extension Name>.ext.php
    - **Application** : enables the extension for application only
        - **Ext Directory** : ./custom/Extension/application/Ext/<Custom Extension>/
        - **Ext File** : ./custom/application/Ext/<Custom Extension>/<Extension Name>.ext.php

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in

./custom/Extension/application/Ext/Extensions/ to map a new extension in the system. The following example will create a new extension called "example", which is only enabled for "application":

./custom/Extension/application/Ext/Extensions/<file>.php

```
<?php

$extensions['example'] = array(
    'section' => 'example',
    'extdir' => 'Example',
    'file' => 'example.ext.php',
    'module' => 'application' //optional paramater
);
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will rebuild the extensions and compile your customization into ./custom/application/Ext/Extensions/extensions.ext.php. Then you will be able to add your own extension files in ./custom/Extension/application/Ext/Example/.

Module Loadable Package

When building a module loadable package, you can use the $installdefs['extensions'] index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The base path of the file to be installed

The example below will demonstrate the proper install definition that should be used in the ./manifest.php file in order to add the Extension file to the system. You should note that when using this approach, Sugar will automatically execute Rebuild Extensions to reflect the new Extension in the system.

./manifest.php

```
<?php

$manifest = array(
    ...
);
```

```
$installdefs = array(
    'id' => 'customExtension_Example',
    'extensions' => array(
        array(
            'from' => '<basepath>/Files/custom/Extension/application/E
xt/Extensions/<file>.php'
        )
    )
);
```

Alternatively, you may use the $installdefs['copy'] index for the Extension file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the $installdefs['copy'] index and module-loadable packages, please refer to the Introduction to the Manifest page.

FileAccessControlMap

Overview

The FileAccessControlMap extension restricts specific view actions from users of the system.

Note: This is only applicable to modules running in backward compatibility mode.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	**Value**
Extension Scope	Module
Sugar Variable	$file_access_control_map
Extension Directory	./custom/Extension/modules/<module>/Ext/FileAccessControlMap/
Compiled Extension File	./custom/modules/<module>/Ext/FileAccessControlMap/file_access_control_map.ext.php

Manifest Installdef	$installdefs['file_access']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in ./custom/Extension/modules/<module>/Ext/FileAccessControlMap/ to restrict a view of a module. The following example will create a new restriction for the detail view:

./custom/Extension/modules/<module>/Ext/FileAccessControlMap/<file>.php

```
<?php

$file_access_control_map['modules']['<lowercase module>']['actions'] =
 array(
    'detailview',
);
```

Navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and compile your customization into ./custom/modules/<module>/Ext/FileAccessControlMap/file_access_control_map.ext.php

Module Loadable Package

When building a module loadable package, you can use the $installdefs['file_access'] index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The base path of the file to be installed
to_module	String	The key of the module the file is to be installed to

The example below demonstrates the proper install definition that should be used in the ./manifest.php file, in order to add the File Access Control Map file to a

specific module. You should note that when using this approach, Sugar will automatically execute Rebuild Extensions to reflect the new Action in the system.

./manifest.php

```
<?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'ActionRemap_Example',
    'file_access' => array(
        array(
            'from' => '<basepath>/Files/custom/Extension/modules/<modu
le>/Ext/FileAccessControlMap/<file>.php',
            'to_module' => '<module>',
        )
    )
);
```

Alternatively, you may use the $installdefs['copy'] index for the File Access Control Map Extension file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the $installdefs['copy'] index and module-loadable packages, please refer to the Introduction to the Manifest page.

Include

Overview

The Modules extension maps additional modules in the system, typically when Module Builder deploys a module.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value

Extension Scope	Application
Sugar Variable	$beanList, $beanFiles, $moduleList
Extension Directory	./custom/Extension/application/Ext/Include/
Compiled Extension File	./custom/application/Ext/Include/modules.ext.php
Manifest Installdef	$installdefs['beans']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in ./custom/Extension/application/Ext/Include/ to register a new module in the system. This extension is normally used when deploying custom modules. The example below shows what this file will look like after a module is deployed:

./custom/Extension/application/Ext/Include/<file>.php

```
  <?php

$beanList['cust_module'] = 'cust_module';
$beanFiles['cust_module'] = 'modules/cust_module/cust_module.php';
$moduleList[] = 'cust_module';
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and compile your customization into ./custom/application/Ext/Include/modules.ext.php.

Module Loadable Package

When building a module loadable package, you can use the $installdefs['beans'] index to install the extension file.

Installdef Properties

Name	Type	Description
module	String	The key of the module to

		be installed
class	String	The class name of the module to be installed
path	String	The path to the module's class
tab	Boolean	Whether or not the module will have a navigation tab (defaults to false)

The example below demonstrates the proper install definition that should be used in the ./manifest.php file, in order to add a custom module to the system. When using this approach, you still need to use the $installdefs['copy'] index for Module directory, however, Sugar will automatically execute the database table creation process, relationship table creation process, as well as Rebuild Extensions to reflect the new Module in the system.

./manifest.php

```
  <?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'Beans_Example',
    'beans' => array(
        array(
            'module' => 'cust_Module',
            'class' => 'cust_Module',
            'path' => 'modules/cust_Module/cust_Module.php',
            'tab' => true
        )
    ),
    'copy' => array(
        array(
            'from' => '<basepath>/Files/modules/cust_Module',
            'to' => 'modules/cust_Module'
        )
    )
);
```

Alternatively, you may use the $installdefs['copy'] index for copying the Modules definition file into the ./custom/Extension/application/Ext/Include/ directory. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the $installdefs['copy'] index and module loadable packages, please refer to the Introduction to the Manifest page.

JSGroupings

Overview

The JSGroupings extension allows for additional JavaScript grouping files to be created or added to existing groupings within the system.

JSGroupings are file packages containing one or more minified JavaScript libraries. The groupings enhance system performance by reducing the number of JavaScript files that need to be downloaded for a given page. Some examples of JSGroupings in Sugar are sugar_sidecar.min.js, which contains the Sidecar JavaScript files, and sugar_grp1.js, which contains the base JavaScript files.

You can find all of the groups listed in ./jssource/JSGroupings.php. Each group is loaded only when needed by a direct call (e.g., from a TPL file). For example, sugar_grp1.js is loaded for almost all Sugar functions, while sugar_grp_yui_widgets.js will usually be loaded for just record views.

To load a JSGroupings file for a custom module, simply add a new JSGrouping and then include the JavaScript file for your custom handlebars template. You can also append to an existing grouping, such as ./include/javascript/sugar_grp7.min.js, to include the code globally.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Application
Sugar Variable	$js_groupings
Extension Directory	./custom/Extension/application/Ext/JSGroupings/
Compiled Extension File	./custom/application/Ext/JSGroupings/jsgroups.ext.php
Manifest Installdef	$installdefs['jsgroups']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

Creating New JSGroupings

When working directly with the filesystem, you can create a file in ./custom/Extension/application/Ext/JSGroupings/ to add or append Javascript to JSGroupings in the system. The following example will add a new JSGrouping file to the system:

./custom/Extension/application/Ext/JSGroupings/<file>.php

```
<?php

//creates the file cache/include/javascript/newGrouping.js
$js_groupings[] = $newGrouping = array(
    'custom/file1.js' => 'include/javascript/newGrouping.js',
    'custom/file2.js' => 'include/javascript/newGrouping.js',
);
```

Next, create the Javascript files that will be grouped as specified in the JSGrouping definition above:

./custom/file1.js

```
function one(){
    //logic
}
```

./custom/file2.js

```
function two(){
    //logic
}
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and compile your customization into ./custom/application/Ext/JSGroupings/jsgroups.ext.php.

Finally, navigate to Admin > Repair > Rebuild JS Grouping Files. This will create the grouping file in the cache directory as specified:

./cache/include/javascript/newGrouping.js

```
function one(){}/* End of File custom/file1.js */function two(){}/* En
d of File custom/file2.js */
```

Appending to Existing JSGroupings

In some situations, you may find that you need to append your own JavaScript to a core Sugar JSGrouping. Similarly to creating a new JSGrouping, to append to a core JSGrouping you can create a new PHP file in ./custom/Extension/application/Ext/JSGroupings/. The example below demonstrates how to add the file ./custom/file1.js to ./cache/include/javascript/sugar_grp7.min.js.

./custom/Extension/application/Ext/JSGroupings/<file>.php

```
<?php

//Loop through the groupings to find grouping file you want to append
to

foreach ($js_groupings as $key => $groupings)
{
    foreach  ($groupings as $file => $target)
    {
     //if the target grouping is found
        if ($target == 'include/javascript/sugar_grp7.min.js')
        {
            //append the custom JavaScript file
            $js_groupings[$key]['custom/file1.js'] = 'include/javascri
pt/sugar_grp7.min.js';
        }
        break;
    }
}
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and compile your customization into ./custom/application/Ext/JSGroupings/jsgroups.ext.php.

Module Loadable Package

When building a module loadable package, you can use the $installdefs['jsgroups'] index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The basepath of the file to be installed
to_module	String	The key for the module where the file will be installed

The example below demonstrates the proper install definition that should be used in the ./manifest.php file to add the JSGrouping Extension file to the system. When using this approach, you still need to use the $installdefs['copy'] index for the custom JavaScript files you are adding to JSGroupings. Sugar will automatically execute Rebuild Extensions to reflect the new JSGrouping, however, you will still need to navigate to Admin > Repair > Rebuild JS Grouping Files, to create the grouping file in the cache directory.

./manifest.php

```
<?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'jsGroupings_Example',
    'jsgroups' => array(
        array(
            'from' => '<basepath>/Files/custom/Extension/application/E
xt/JSGroupings/<file>.php',
            'to_module' => 'application',
        )
    ),
    'copy' => array(
        array(
```

```
            'from' => '<basepath>/Files/custom/file1.js',
            'to' => 'custom/file1.js'
        ),
        array(
            'from' => '<basepath>/Files/custom/file2.js',
            'to' => 'custom/file2.js'
        )
    )
);
```

Alternatively, you may use the $installdefs['copy'] index for the JSGrouping Extension file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the $installdefs['copy'] index and module-loadable packages, please refer to the Introduction to the Manifest page.

Considerations

- The grouping path you specify will be created in the cache directory.
- If you wish to add a grouping that contains a file that is part of another group already, add a '.' after <file>.js to make the element key unique.

Language

Overview

The Language extension adds or overrides language strings.

This extension is applicable to both the application and module framework. For more information, please refer to the Language Framework page.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	All - Application & Module
Sugar Variables	• If adding language files to

	application: $app_strings / $app_list_strings • If adding language files to <module>: $mod_strings
Extension Directory	• Application: ./custom/Extension/application/Ext/Language/ • Module: ./custom/Extension/modules/<module>/Ext/Language/
Compiled Extension File	• Application: ./custom/application/Ext/Language/<language_key>.lang.ext.php • Module: ./custom/modules/<module>/Ext/Language/<language_key>.lang.ext.php
Manifest Installdef	$installdefs['language']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

Creating New Application Label

When working directly with the filesystem, you can create a file in ./custom/Extension/application/Ext/Language/ to add Labels for languages to the system. The following example will add a new Label 'LBL_EXAMPLE_LABEL' to the system:

./custom/Extension/application/Ext/Language/<language>.<file>.php

```
  <?php

$app_strings['LBL_EXAMPLE_LABEL'] = 'Example Application Label';
```

Navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and compile your customization into./custom/application/Ext/Language/<language>.lang.ext.php

Creating New Module Label

When working directly with the filesystem, you can create a file in ./custom/Extension/modules/<module>/Ext/Language/ to add Labels for languages to a particular module. The following example will add a new Label 'LBL_EXAMPLE_MODULE_LABEL' to a module:

./custom/Extension/modules/<module>/Ext/Language/<language>.<file>.php

```
  <?php

$mod_strings['LBL_EXAMPLE_MODULE_LABEL'] = 'Example Module Label';
```

Navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and compile your customization into ./custom/modules/<module>/Ext/Language/<language>.lang.ext.php

Module Loadable Package

When building a module loadable package, you can use the $installdefs['language'] index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The basepath of the file to be installed.
to_module	String	The key of the module the file is to be installed to.
language	String	The key of the language the file is to be installed to.

The example below will demonstrate the proper install definition that should be used in the ./manifest.php file, in order to add the Language Extension file to the system. You should note that when using this approach Sugar will automatically execute Rebuild Extensions to reflect the new Labels in the system.

./manifest.php

```
  <?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'language_Example',
    'language' => array(
        array(
            'from' => '<basepath>/Files/custom/Extension/application/E
xt/Language/<language>.lang.ext.php',
            'to_module' => 'application',
            'language' => '<language>'
        )
    )
);
```

Alternatively, you may use the $installdefs['copy'] index to copy the file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the $installdefs['copy'] index and module-loadable packages, please refer to the Introduction to the Manifest page.

Layoutdefs

Overview

The Layoutdefs extension adds or overrides subpanel definitions.

Note: This extension is only applicable to modules running in backward compatibility mode.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Module
Sugar Variable	$layout_defs

Extension Directory	./custom/Extension/modules/<module>/Ext/Layoutdefs/
Compiled Extension File	./custom/<module>/Ext/Layoutdefs/layoutdefs.ext.php
Manifest Installdef	$installdefs['layoutdefs']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in ./custom/Extension/modules/<module>/Ext/Layoutdefs/ to add Layout Definition files to the system. The following example will add a subpanel definition for a module relationship:

./custom/Extension/modules/<module>/Ext/Layoutdefs/<file>.php

```
  <?php

$layout_defs["<module>"]["subpanel_setup"]['<subpanel key>'] = array (
    'order' => 100,
    'module' => '<related module>',
    'subpanel_name' => 'default',
    'sort_order' => 'asc',
    'sort_by' => 'id',
    'title_key' => 'LBL_SUBPANEL_TITLE',
    'get_subpanel_data' => '<subpanel key>',
    'top_buttons' => array (
        array (
          'widget_class' => 'SubPanelTopButtonQuickCreate',
        ),
        array (
          'widget_class' => 'SubPanelTopSelectButton',
          'mode' => 'MultiSelect',
        ),
    ),
);
```

Please note that, if you are attempting to override parts of an existing subpanel definition, you should specify the exact index rather than redefining the entire

array. An example of overriding the subpanel top_buttons index is shown below:

```
  <?php

$layout_defs["<module>"]["subpanel_setup"]["<subpanel key>"]["top_butt
ons"] = array(
    array(
        'widget_class' => 'SubPanelTopButtonQuickCreate',
    ),
);
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and compile your customizations to ./custom/modules/<module>/Ext/Layoutdefs/layoutdefs.ext.php .

Module Loadable Package

When building a module loadable package, you can use the $installdefs['layoutdefs'] index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The base path of the file to be installed
to_module	String	The key for the module where the file will be installed

The example below demonstrates the proper install definition that should be used in the ./manifest.php file in order to add the Layout Definition Extension file to the system. When using this approach, Sugar will automatically execute Rebuild Extensions to reflect the customizations added with the Layout definition.

./manifest.php

```
  <?php

$manifest = array(
    ...
);

$installdefs = array(
```

```
    'id' => 'layoutdefs_Example',
    'layoutdefs' => array(
        array(
            'from' => '<basepath>/Files/custom/Extension/modules/<modu
le>/Ext/Layoutdefs/<file>.php',
            'to_module' => '<module>',
        )
    )
);
```

Alternatively, you may use the $installdefs['copy'] index to copy the file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the $installdefs['copy'] index and module-loadable packages, please refer to the Introduction to the Manifest page.

LogicHooks

Overview

The LogicHooks extension adds actions to specific events such as, for example, before saving a bean. For more information on logic hooks in Sugar, please refer to the Logic Hooks documentation.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	All - Application & Module
Sugar Variable	$hook_array
Extension Directory	Application - ./custom/Extension/application/Ext/LogicHooks/ Module - ./custom/Extension/modules/<module>/Ext/LogicHooks/
Compiled Extension File	Application - ./custom/application/Ext/LogicHooks/logichooks.ext.php Module - ./custom/modules/<module>/E

	xt/LogicHooks/logichooks.ext.php
Manifest Installdef	$installdefs['hookdefs']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in ./custom/Extension/application/Ext/LogicHooks/ to add a new logic hook to the application. The following example will create a new before_save logic hook that executes for all modules:

./custom/Extension/application/Ext/LogicHooks/<file>.php

```
  <?php

    $hook_array['before_save'][] = Array(
        1,
        'Custom Logic',
        'custom/application_hook.php',
        'ApplicationHookConsumer',
        'before_method'
    );
```

Next, create the hook class:

./custom/application_hook.php

```
  <?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class ApplicationHookConsumer
    {
        function before_method($bean, $event, $arguments)
        {
            //logic
        }
    }
```

```
?>
```

Finally, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and the customizations will be compiled into ./custom/application/Ext/LogicHooks/logichooks.ext.php. Your logic hook will run before saving records in any module.

Module Loadable Package

When building a module loadable package, you can use the $installdefs['hookdefs'] index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The basepath of the file to be installed.
to_module	String	The key of the module the file is to be installed to.

The example below demonstrates the proper install definition that should be used in the ./manifest.php file in order to add the Action View Map file to a specific module. You should note that when using this approach, you still need to use the $installdefs['copy'] index for the hook class file, however Sugar will automatically execute Rebuild Extensions to reflect the new logic hook in the system.

./manifest.php

```
  <?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'actionView_example',
    'hookdefs' => array(
        array(
            'from' => '<basepath>/Files/custom/Extension/application/E
xt/LogicHooks/<file>.php',
            'to_module' => 'application',
        )
```

```
    ),
    'copy' => array(
        array(
            'from' => '<basepath>/Files/custom/application_hook.php',
            'to' => 'custom/application_hook.php',
        ),
    )
);
```

Alternatively, you may use the $installdefs['copy'] index to copy the file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the $installdefs['copy'] index and module-loadable packages, please refer to the Introduction to the Manifest page.

Alternative Installdef

Although not recommended, you could utilize the $installdefs['logic_hooks'] index to deploy a logic hook to the system. Please note that there are a couple caveats to this installation method:

- The $installdefs['logic_hooks'] index method only works for module-based logic hooks.
- The $installdefs['logic_hooks'] index method installs to ./custom/modules/<module>/logic_hooks.php, which is not part of the Extension framework.

Properties

Name	Type	Description
module	String	The key of the module for the logic hook to be installed to.
hook	String	The type of logic hook to be installed.
order	Integer	The number in which the logic hook should run.
description	String	A description of the logic hook to be installed.
file	String	The file path which contains the logic hook class.
class	String	The class which houses

		the logic hook functionality.
function	String	The function the logic hook will execute.

The example below will demonstrate the $installdefs['logic_hooks'] index in the ./manifest.php file, in order to add an after save logic hook to a specific module. You should note that when using this approach, you still need to use the $installdefs['copy'] index for the hook class file.

./manifest.php

```
  <?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'actionView_example',
    'logic_hooks' => array(
        array(
            'module' => '<module>',
            'hook' => 'after_save',
            'order' => 1,
            'description' => 'Example After Save LogicHook',
            'file' => 'custom/modules/<module>/module_hook.php',
            'class' => '<module>HookConsumer'
            'function' => 'after'
        )
    ),
    'copy' => array(
        array(
            'from' => '<basepath>/Files/custom/modules/<module>/module
_hook.php',
            'to' => 'custom/modules/<module>/module_hook.php',
        ),
    )
);
```

Modules

Overview

The Modules extension maps additional modules in the system, typically when Module Builder deploys a module.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Application
Sugar Variable	$beanList, $beanFiles, $moduleList
Extension Directory	./custom/Extension/application/Ext/Include/
Compiled Extension File	./custom/application/Ext/Include/modules.ext.php
Manifest Installdef	$installdefs['beans']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in ./custom/Extension/application/Ext/Include/ to register a new module in the system. This extension is normally used when deploying custom modules. The example below shows what this file will look like after a module is deployed:

./custom/Extension/application/Ext/Include/<file>.php

```
  <?php

$beanList['cust_module'] = 'cust_module';
$beanFiles['cust_module'] = 'modules/cust_module/cust_module.php';
$moduleList[] = 'cust_module';
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and compile your customization into ./custom/application/Ext/Include/modules.ext.php.

Module Loadable Package

When building a module loadable package, you can use the $installdefs['beans'] index to install the extension file.

Installdef Properties

Name	Type	Description
module	String	The key of the module to be installed
class	String	The class name of the module to be installed
path	String	The path to the module's class
tab	Boolean	Whether or not the module will have a navigation tab (defaults to false)

The example below demonstrates the proper install definition that should be used in the ./manifest.php file, in order to add a custom module to the system. When using this approach, you still need to use the $installdefs['copy'] index for Module directory, however, Sugar will automatically execute the database table creation process, relationship table creation process, as well as Rebuild Extensions to reflect the new Module in the system.

./manifest.php

```
  <?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'Beans_Example',
    'beans' => array(
        array(
            'module' => 'cust_Module',
            'class' => 'cust_Module',
            'path' => 'modules/cust_Module/cust_Module.php',
            'tab' => true
        )
    ),
```

```
    'copy' => array(
        array(
            'from' => '<basepath>/Files/modules/cust_Module',
            'to' => 'modules/cust_Module'
        )
    )
);
```

Alternatively, you may use the $installdefs['copy'] index for copying the Modules definition file into the ./custom/Extension/application/Ext/Include/ directory. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the $installdefs['copy'] index and module loadable packages, please refer to the Introduction to the Manifest page.

Platforms

Overview

The Platforms extension adds allowed REST API platforms when restricting custom platforms through the use of the disable_unknown_platforms configuration setting.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Application
Extension Directory	./custom/Extension/application/Ext/Platforms/
Compiled Extension File	./custom/application/Ext/Platforms/platforms.ext.php
Manifest Installdef	$installdefs['platforms']

Implementation

The following sections illustrate the various ways to implement a new platform type to a Sugar instance.

File System

When working directly with the filesystem, enable the disable_unknown_platforms configuration by setting $sugar_config['disable_unknown_platforms'] = true in your ./config_override.php. This will prevent the system from allowing unknown platform types from accessing the rest endpoints. Next, create a file in ./custom/Extension/application/Ext/Platforms/ to map a new platform in the system. The following example adds a new platform called 'integration' that can be used throughout the system:

./custom/Extension/application/Ext/Platforms/<file>.php

```
<?php

$platforms[] = 'integration';
```

Finally, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and your customizations will be compiled into ./custom/application/Ext/Platforms/platforms.ext.php.

Alternatively, platforms can also be added by creating ./custom/clients/platforms.php and appending new platform types to the $platforms variable. This method of creating platforms is still compatible but is not recommended from a best practices standpoint.

Once installed, developers can take advantage of the new platform type when authenticating with the REST API oauth2/token endpoint. An example is shown below:

```
{
  "grant_type": "password",
  "username": "admin",
  "password": "password",
  "client_id": "sugar",
  "client_secret": "",
  "platform": "integration"
}
```

Module Loadable Package

When building a module loadable package, you can use the $installdefs['platforms'] index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The base path of the file to be installed

The example below demonstrates the proper install definition that should be used in the ./manifest.php file in order to add the utils to the system. You should note that when using this approach, Sugar will automatically execute Rebuild Extensions to reflect the new utils in the system.

./manifest.php

```
<?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'Platforms_Example',
    'platforms' => array(
        array(
            'from' => '<basepath>/Files/custom/Extension/application/E
xt/Platforms/<file>.php',
        )
    )
);
```

Alternatively, you may use the $installdefs['copy'] index to copy the file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the $installdefs['copy'] index and module-loadable packages, please refer to the Introduction to the Manifest page.

ScheduledTasks

Overview

The ScheduledTasks extension adds custom functions that can be used by scheduler jobs. For more information about schedulers in Sugar, please refer to the Schedulers documentation.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Module: Schedulers
Sugar Variable	$job_strings
Extension Directory	./custom/Extension/modules/Schedulers/Ext/ScheduledTasks/
Compiled Extension File	./custom/Schedulers/Ext/ScheduledTasks/scheduledtasks.ext.php
Manifest Installdef	$installdefs['scheduledefs']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in ./custom/Extension/modules/Schedulers/Ext/ScheduledTasks/ to add a new Scheduler Task to the system. The following example will create a new Scheduler Task 'example_job':

./custom/Extension/modules/Schedulers/Ext/ScheduledTasks/<file>.php

```
  <?php

$job_strings[] = 'exampleJob';

function exampleJob()
{
    //logic here

    //return true for completed
    return true;
}
```

Next, create the Language file for the scheduler so that the job properly displays in

Admin > Schedulers:

./custom/Extension/modules/Schedulers/Ext/Language/<language>.<file>.php

```
  <?php

//Label will be LBL_[upper case function name]
$mod_strings['LBL_EXAMPLEJOB'] = 'Example Job';
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and the customizations will be compiled into ./custom/modules/Schedulers/Ext/ScheduledTasks/scheduledtasks.ext.php

Module Loadable Package

When building a module loadable package, you can use the $installdefs['scheduledefs'] index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The basepath of the file to be installed

The example below demonstrates the proper install definition that should be used in the ./manifest.php file in order to add the custom Scheduler definition file to the system. When using this approach, Sugar will automatically execute Rebuild Extensions to reflect the new Scheduler in the system.

./manifest.php

```
  <?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'actionView_example',
    'scheduledefs' => array(
        array(
            'from' => '<basepath>/Files/custom/Extension/modules/Sched
uler/Ext/ScheduledTasks/<file>.php',
```

```
        )
    ),
    'language' => array(
        array(
            'from' =>'<basepath>/Files/custom/Extension/modules/Schedu
lers/Ext/Language/<file>.php',
            'to_module' => 'Schedulers',
            'language' => 'en_us'
        )
    )
);
```

Alternatively, you may use the $installdefs['copy'] index to copy the file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the $installdefs['copy'] index and module-loadable packages, please refer to the Introduction to the Manifest page.

Sidecar

Overview

The Sidecar extension installs metadata files to their appropriate directories.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Module
Sugar Variable	$viewdefs
Extension Directory	./custom/Extension/modules/<module>/Ext/clients/<client>/<type>/<subtype>/
Compiled Extension File	./custom/<module>/Ext/clients/<client>/<type>/<subtype>/<subtype>.ext.php
Manifest Installdef	$installdefs['sidecar']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in ./custom/Extension/modules/<module>/Ext/clients/<client>/<type>/<subtype>/ to append the metadata extensions. The example below demonstrates how to add a new subpanel to a specific module:

./custom/Extension/modules/<module>/Ext/clients/base/layouts/subpanels/<file>.php

```
<?php

$viewdefs['<module>']['base']['layout']['subpanels']['components'][] =
 array(
    'layout' => 'subpanel',
    'label' => 'LBL_RELATIONSHIP_TITLE',
    'context' => array(
        'link' => '<link_name>',
    )
);
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and compile your customization into ./custom/modules/<module>/Ext/clients/base/layouts/subpanels/subpanels.ext.php

Module Loadable Package

When building a module loadable package, you can use the $installdefs['sidecar'] index to install the metadata file.

Installdef Properties

Name	Type	Description
from	String	The base path of the file to be installed **Note**: When adding the file to a module loadable package, its 'from' path must be formatted as clien

		ts/<client>/<type>/<subtype>/<file>.php for Sugar to recognize the installation location.
to_module	String	• The key for the module where the file will be installed • If not populated, 'application' is used

The example below demonstrates the proper install definition that should be used in the ./manifest.php file in order to add the metadata file to a specific module. When using this approach, Sugar will automatically execute Rebuild Extensions and Metadata Rebuild to reflect your changes in the system.

./manifest.php

```
<?php

$manifest = array(
    ...
);

$installdefs = array (
    'id' => 'sidecar_example',
    'sidecar' => array(
        array(
            'from' => '<basepath>/Files/custom/<module>/clients/base/l
ayouts/subpanels/<file>.php',
            'to_module' => '<module>',
        ),
    ),
);
```

Alternatively, you may use the $installdefs['copy'] index to copy the file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the $installdefs['copy'] index and module-loadable packages, please refer to the Introduction to the Manifest page.

TinyMCE

Overview

The TinyMCE extension affects the TinyMCE WYSIWYG editor's configuration for backward compatible modules such as PDF Manager and Campaign Email Templates.

To review the default configuration for TinyMCE, please refer to the code in ./include/SugarTinyMCE.php. Sidecar's TinyMCE configuration can be edited using the Sidecar Framework and editing the htmleditable_tinymce field.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Application
Sugar Variable	$defaultConfig, $buttonConfigs, $pluginsConfig
Extension Directory	./custom/Extension/application/Ext/TinyMCE/
Compiled Extension File	./custom/application/Ext/TinyMCE/tinymce.ext.php
Manifest Installdef	$installdefs['tinymce']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in ./custom/Extension/application/Ext/TinyMCE/ to customize the TinyMCE configuration. The following example will increase the height of the TinyMCE window, remove buttons, and remove plugins from the WYSIWYG Editor:

./custom/Extension/application/Ext/TinyMCE/<file>.php

```
<?php

$defaultConfig['height'] = '1000';
```

```
$buttonConfigs['default'] = array(
    'buttonConfig' => "bold,italic,underline,strikethrough,separator,b
ullist,numlist",
    'buttonConfig2' => "justifyleft,justifycenter,justifyright,justify
full",
    'buttonConfig3' => "fontselect,fontsizeselect",
);

$pluginsConfig['default'] = 'advhr,preview,paste,directionality';
```

Navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and compile your customization into ./custom/application/Ext/TinyMCE/tinymce.ext.php

Module Loadable Package

When building a module loadable package, you can use the $installdefs['tinymce'] index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The base path of the file to be installed

The example below demonstrates the proper install definition that should be used in the ./manifest.php file, in order to add the UserPage file to the system. You should note that when using this approach Sugar will automatically execute Rebuild Extensions to reflect the changes to TinyMCE in the system.

./manifest.php

```
<?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'tinyMCE_Example',
    'tinymce' => array(
        array(
```

```
            'from' => '<basepath>/Files/custom/Extension/application/Ext/TinyMCE/<file>.php',
            'to_module' => 'application'
        )
    )
);
```

Alternatively, you may use the $installdefs['copy'] index to copy the file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the $installdefs['copy'] index and module loadable packages, please refer to the Introduction to the Manifest page.

UserPage

Overview

The UserPage extension adds sections to the User Management view.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Module: Users
Extension Directory	./custom/Extension/modules/Users/Ext/UserPage/
Compiled Extension File	./custom/Users/Ext/UserPage/userpage.ext.php
Manifest Installdef	$installdefs['user_page']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in

./custom/Extension/modules/Users/Ext/UserPage/ to add custom elements to the User page. The following example will add a custom table to the Users module's detail view:

./custom/Extension/modules/Users/Ext/UserPage/<file>.php

```
<?php

$HTML=<<<HTML

    <table cellpadding="0" cellspacing="0" width="100%" border="0" cla
ss="list view">
        <tbody>
            <tr height="20">
                <th scope="col" width="15%">
                    <slot>Header</slot>
                </th>
            </tr>
            <tr height="20" class="oddListRowS1">
                <td scope="row" valign="top">
                    Content
                </td>
            </tr>
        </tbody>
    </table>
HTML;

    echo $HTML;
```

Navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and compile your customization into ./custom/modules/Users/Ext/UserPage/userpage.ext.php

Module Loadable Package

When building a module loadable package, you can use the $installdefs['user_page'] index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The base path of the file to be installed

The example below demonstrates the proper install definition that should be used in the ./manifest.php file in order to add the UserPage file to the system. When using this approach, Sugar will automatically execute Rebuild Extensions to reflect the changes to the User view in the system.

./manifest.php

```
<?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'userPage_Example',
    'user_page' => array(
        array(
            'from' => '<basepath>/Files/custom/Extension/modules/Users
/Ext/UserPage/<file>.php',
        )
    )
);
```

Alternatively, you may use the $installdefs['copy'] index to copy the file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the $installdefs['copy'] index and module-loadable packages, please refer to the Introduction to the Manifest page.

Utils

Overview

The Utils extension adds functions to the global utility function list.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value

Extension Scope	Application
Extension Directory	./custom/Extension/application/Ext/Utils/
Compiled Extension File	./custom/application/Ext/Utils/custom_utils.ext.php
Manifest Installdef	$installdefs['utils']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in ./custom/Extension/application/Ext/Utils/ to map a new action in the system. The following example will create a new function called 'exampleUtilFunction' that can be used throughout the system:

./custom/Extension/application/Ext/Utils/<file>.php

```
<?php

function exampleUtilFunction()    '
{
    //logic
}
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and your customizations will be compiled into ./custom/application/Ext/Utils/custom_utils.ext.php .

Alternatively, functions can also be added by creating ./custom/include/custom_utils.php. This method of creating utils is still compatible but is not recommended from a best practices standpoint.

Module Loadable Package

When building a module loadable package, you can use the $installdefs['utils'] index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The base path of the file to be installed

The example below demonstrates the proper install definition that should be used in the ./manifest.php file in order to add the utils to the system. You should note that when using this approach, Sugar will automatically execute Rebuild Extensions to reflect the new utils in the system.

./manifest.php

```
<?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'utils_Example',
    'utils' => array(
        array(
            'from' => '<basepath>/Files/custom/Extension/application/E
xt/Utils/<file>.php',
        )
    )
);
```

Alternatively, you may use the $installdefs['copy'] index to copy the file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the $installdefs['copy'] index and module-loadable packages, please refer to the Introduction to the Manifest page.

Vardefs

Overview

The Vardefs extension adds or overrides system vardefs, which provide the Sugar application with information about SugarBeans.

For more information on vardefs in Sugar, please refer to the Vardefs documentation .

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Module
Sugar Variable	$dictionary
Extension Directory	./custom/Extension/modules/<module>/Ext/Vardefs/
Compiled Extension File	./custom/<module>/Ext/Vardefs/vardefs.ext.php
Manifest Installdef	$installdefs['vardefs']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in ./custom/Extension/modules/<module>/Ext/Vardefs/ to edit or add vardefs to a module in the system.

The most common use of the Vardef extension is to alter the attributes of an existing vardef. To do this,avoid redefining the entire vardef and instead update the specific index you want to change. The following example updates the Required property on the Name field in a module:

./custom/Extension/modules/<module>/Ext/Vardefs/<file>.php

```
$dictionary['<module>']['fields']['name']['required'] = false;
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and your customizations will be compiled into ./custom/modules/<module>/Ext/Vardefs/vardefs.ext.php .

Notice Never specify vardefs for a module under another module's extension path. For example, do not specify **$dictionary['Accounts']['fields']['name']['required'] = false** under

./custom/Extension/modules/Contacts/Ext/Vardefs/. Doing so will result in unexpected behavior within the system.

Module Loadable Package

When building a module loadable package, you can use the $installdefs['vardefs'] index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The base path of the file to be installed
to_module	String	The key for the module where the file will be installed

The example below demonstrates the proper install definition that should be used in the ./manifest.php file in order to add the Vardefs file to a specific module. You should note that when using this approach Sugar will automatically execute Rebuild Extensions to reflect the vardef changes in the system.

./manifest.php

```
<?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'vardefs_Example',
    'vardefs' => array(
        array(
            'from' => '<basepath>/Files/custom/Extension/modules/<modu
le>/Ext/Vardefs/<file>.php',
            'to_module' => '<module>',
        )
    )
);
```

Alternatively, you may use the $installdefs['copy'] index to copy the file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the $installdefs['copy'] index and module-loadable packages, please refer to the Introduction to the Manifest page.

Creating Custom Fields

If your goal is to manually create a custom field on an instance, you should be using the Module Installer to create the field. This can be used both for an installer package and programmatically. An example of creating a field from a module loadable package can be found under Package Examples in the article, Creating an Installable Package that Creates New Fields. An example of programmatically creating a field can be found in the Manually Creating Custom Fields section of the Module Vardefs documentation.

WirelessLayoutdefs

Overview

The WirelessLayoutdefs extension adds additional subpanels to wireless views. This extension is only applicable to modules running in backward compatibility mode.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Module
Sugar Variable	$layout_defs
Extension Directory	./custom/Extension/modules/<module>/Ext/WirelessLayoutdefs/
Compiled Extension File	./custom/<module>/Ext/WirelessLayoutdefs/wireless.subpaneldefs.ext.php
Manifest Installdef	$installdefs['wireless_subpanels']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in ./custom/Extension/modules/<module>/Ext/WirelessLayoutdefs/ to add a subpanel to a module in the system. The following example will add a new subpanel to a specified module:

./custom/Extension/modules/<module>/Ext/WirelessLayoutdefs/<file>.php

```
<?php

$layout_defs['<module>']['subpanel_setup']['<subpanel module>'] = arra
y(
    'order' => 10,
    'module' => '<subpanel module>',
    'get_subpanel_data' => '<subpanel name>',
    'title_key' => 'LBL_SUBPANEL_TITLE',
);
```

Navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and compile your customization into ./custom/modules/<mo dule>/Ext/WirelessLayoutdefs/wireless.subpaneldefs.ext.php.

Module Loadable Package

When building a module loadable package, you can use the $installdefs['wireless_subpanels'] index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The base path of the file to be installed
to_module	String	The key for the module where the file will be installed

The example below will demonstrate the proper install definition that should be used in the ./manifest.php file in order to add the subpanel file to a specific module. You should note that when using this approach Sugar will automatically execute Rebuild Extensions to reflect the subpanel in the system.

./manifest.php

```
<?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'wirelessLayoutdefs_Example',
    'wireless_subpanels' => array(
        array(
            'from' => '<basepath>/Files/custom/Extension/modules/<modu
le>/Ext/WirelessLayoutdefs/<file>.php',
            'to_module' => '<module>',
        )
    )
);
```

Alternatively, you may use the $installdefs['copy'] index to copy the file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the $installdefs['copy'] index and module-loadable packages, please refer to the Introduction to the Manifest page.

WirelessModuleRegistry

Overview

The WirelessModuleRegistry extension adds modules to the available modules for mobile.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Application
Sugar Variable	$wireless_module_registry
Extension Directory	./custom/Extension/application/Ext/WirelessModuleRegistry/

Compiled Extension File	./custom/application/Ext/WirelessModuleRegistry/wireless_module_registry.ext.php
Manifest Installdef	$installdefs['wireless_modules']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in ./custom/Extension/application/Ext/WirelessModuleRegistry/ to add modules to the list of available modules for mobile. The following example will add a new module, 'cust_module', to the list of available modules for mobile:

./custom/Extension/application/Ext/WirelessModuleRegistry/<file>.php

```
<?php

$wireless_module_registry['cust_module'] = array(
    //enables/disables record creation
    'disable_create' => false,
);
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and compile your customization into ./custom/application/Ext/WirelessModuleRegistry/wireless_module_registry.ext.php.

Module Loadable Package

When building a module loadable package, you can use the $installdefs['wireless_modules'] index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The base path of the file to be installed.
to_module	String	The key of the module the

		file is to be installed to.

The example below demonstrates the proper install definition that should be used in the ./manifest.php file in order to add the Wireless Module Registry file to the system. You should note that when using this approach, Sugar will automatically execute Rebuild Extensions to reflect the new module in the mobile application.

./manifest.php

```
<?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'wirelessModules_Example',
    'wireless_modules' => array(
        array(
            'from' => '<basepath>/Files/custom/Extension/application/E
xt/WirelessModuleRegistry/<file>.php',
            'to_module' => 'application',
        )
    )
);
```

Alternatively, you may use the $installdefs['copy'] index to copy the file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the $installdefs['copy'] index and module-loadable packages, please refer to the Introduction to the Manifest page.

Filters

Overview

Filters are a way to predefine searches on views that render a list of records such as list views, pop-up searches, and lookups. This page explains how to implement the various types of filters for record list views.

Filters contain the following properties:

Property	Type	Description
id	String	A unique identifier for the filter
name	String	The label key for the display label of the filter
filter_definition	Array	The filter definition to apply to the results
editable	Boolean	Determines whether the user can edit the filter. **Note**: If you are creating a predefined filter for a user, this should be set to false
is_template	Boolean	Used with initial pop-up filters to determine if the filter is available as a template

Note: If a filter contains custom fields, those fields must be search-enabled in Studio > {Module Name} > Layouts > Search.

Operators

Operators, defined in ./clients/base/filters/operators/operators.php, are expressions used by a filter to represent query operators. They are constructed in the filter_definition to help generate the appropriate query syntax that is sent to the database for a specific field type. Operators can be defined on a global or module level. The accepted paths are listed below:

- ./clients/base/filters/operators/operators.php
- ./custom/clients/base/filters/operators/operators.php
- ./modules/<module>/clients/base/filters/operators.php
- ./custom/modules/<module>/clients/base/filters/operators.php

The list of stock operators is shown below:

Operator	Label / Key	Description
$contains	• is any of / LBL_OPERATOR_CONTAINS ◦ Used by multienum	Matches anything that contains the value.
$empty	• is empty /	Matches on empty values.

	LBL_OPERATOR_EMPTY ◦ Used by enum and tag	
$not_contains	• is not any of / LBL_OPERATOR_NOT_CONTAINS ◦ Used by multienum	Matches anything that does not contain the specified value.
$not_empty	• is not empty / LBL_OPERATOR_NOT_EMPTY ◦ Used by enum and tag	Matches on non-empty values.
$in	• is any of / LBL_OPERATOR_CONTAINS ◦ Used by enum, int, relate, teamset, and tag	Finds anything where field matches one of the values as specified as an array.
$not_in	• is not any of / LBL_OPERATOR_NOT_CONTAINS ◦ Used by enum, relate, teamset, and tag	Finds anything where the field does not match any of the values in the specified array of values.
$equals	• exactly matches / LBL_OPERATOR_MATCHES ◦ Used by varchar, name, email, text, and textarea • is equal to / LBL_OPERATOR_EQUALS ◦ Used by currency, int, double, float, decimal,	Performs an exact match on that field.

	and date • is / LBL_OPERATOR_IS ◦ Used by bool, phone, radioenum, and parent	
$starts	• starts with / LBL_OPERATOR_STARTS_WITH ◦ Used by varchar, name, email, text, textarea, and phone • is equal to / LBL_OPERATOR_EQUALS ◦ Used by datetime and dateti mecombo	Matches on anything that starts with the value.
$not_equals	• is not equal to / LBL_OPERATOR_NOT_EQUALS ◦ Used by currency, int, double, float, and decimal • is not / LBL_OPERATOR_IS_NOT ◦ Used by radioenum	Matches on non-matching values.
$gt	• is greater than / LBL_OPERATOR_GREATER_THAN ◦ Used by currency, int, double, float, and decimal • after / LBL_OPERATOR_AFTER ◦ Used by date	Matches when the field is greater than the value.

$lt	• is less than / LBL_OPERATOR_LESS_THAN ◦ Used by currency, int, double, float, and decimal • before / LBL_OPERATOR_BEFORE ◦ Used by date	Matches when the field is less than the value.
$gte	• is greater than or equal to / LBL_OPERATOR_GREATER_THAN_OR_EQUALS ◦ Used by currency, int, double, float, and decimal • after / LBL_OPERATOR_AFTER ◦ Used by datetime and dateti mecombo	Matches when the field is greater than or equal to the value
$lte	• is less than or equal to / LBL_OPERATOR_LESS_THAN_OR_EQUALS ◦ Used by currency, int, double, float, and decimal • before / LBL_OPERATOR_BEFORE ◦ Used by datetime and dateti mecombo	Matches when the field is less than or equal to the value.
$between	• is between / LBL_OPERATOR_BETWEEN ◦ Used by currency,	Matches when a numerical value is between two other numerical values.

	int, double, float, and decimal	
last_7_days	• last 7 days / LBL_OPERATOR_LAST_7_DAYS ◦ Used by date, datetime, and dateti mecombo	Matches date in the last 7 days relative to the current date.
next_7_days	• next 7 days / LBL_OPERATOR_NEXT_7_DAYS ◦ Used by date, datetime, and dateti mecombo	Matches dates in the next 7 days relative to the current date.
last_30_days	• last 30 days / LBL_OPERATOR_LAST_30_DAYS ◦ Used by date, datetime, and dateti mecombo	Matches dates in the last 30 days relative to the current date.
next_30_days	• next 30 days / LBL_OPERATOR_NEXT_30_DAYS ◦ Used by date, datetime, and dateti mecombo	Matches dates in the next 30 days relative to the current date.
last_month	• last month / LBL_OPERATOR_LAST_MONTH ◦ Used by date, datetime, and dateti mecombo	Matches dates in the previous month relative to the current month.
this_month	• this month / LBL_OPERATOR_THIS_MONTH	Matches dates in the current month.

	◦ Used by date, datetime, and dateti mecombo	
next_month	• next month / LBL_OPERATOR_NEXT_MONTH ◦ Used by date, datetime, and dateti mecombo	Matches dates in the next month relative to the current month.
last_year	• last year / LBL_OPERATOR_LAST_YEAR ◦ Used by date, datetime, and dateti mecombo	Matches dates in the last year relative to the current year.
this_year	• this year / LBL_OPERATOR_THIS_YEAR ◦ Used by date, datetime, and dateti mecombo	Matches dates in the current year.
next_year	• next year / LBL_OPERATOR_NEXT_YEAR ◦ Used by date, datetime, and dateti mecombo	Matches dates in the next year relative to the current year.
$dateBetween	• is between / LBL_OPERATOR_BETWEEN ◦ Used by date, datetime, and dateti mecombo	Matches dates between two given dates.
yesterday	• yesterday / LBL_OPERATOR_YESTERD	Matches dates on

	AY ◦ Used by date, datetime, and dateti mecombo	yesterday relative to the current date.
today	• today / LBL_OPERATOR_TODAY ◦ Used by date, datetime, and dateti mecombo	Matches dates in the current date.
tomorrow	• tomorrow / LBL_OPERATOR_TOMORROW ◦ Used by date, datetime, and dateti mecombo	Matches dates on tomorrow relative to the current date.

Example

The example below defines a filter where the type field must contain the value Customer and the name field must start with the letter A.

```
$filters = array(
    array(
        'type' => array(
            '$in' => array(
                'Customer',
            ),
        ),
    ),
    array(
        'name' => array(
            '$starts' => 'A',
        ),
    ),
);
```

Sub-Expressions

Sub-expressions group filter expressions into groupings. By default, all expressions are bound by an $and expression grouping.

Sub-Expression	Description
$and	Joins the filters in an "and" expression
$or	Joins the filters in an "or" expression

Note: Sub-Expressions are only applicable to predefined filters and cannot be used for initial filters.

The example below defines a filter where the name field must begin with the letters A or C.

```
$filters = array(
    '$or' => array (
        array(
            'name' => array(
                '$starts' => 'A',
            ),
        ),
        array(
            'name' => array(
                '$starts' => 'C',
            ),
        ),
    )
);
```

Module Expressions

Module expressions operate on modules instead of specific fields. The current module can be specified by either using the module name _this or by leaving the module name as a blank string.

Module Expression	Description
$favorite	Filters the records by the current users favorited items.
$owner	Filters the records by the assigned user.

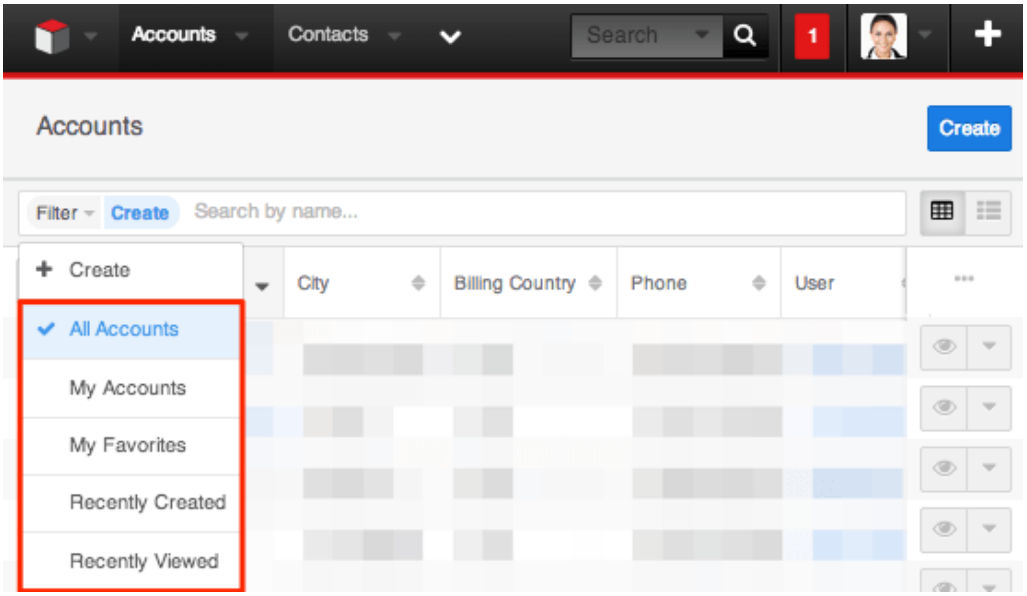
The example below defines a filter where records must be favorited items.

```
$filters = array(
    array(
        '$favorite' => '_this'
    ),
);
```

Filter Examples

Adding Predefined Filters to the List View Filter List

To add a predefined filter to the module's list view, create a new filter definition extension, which will append the filter to the module's viewdefs.



The following example will demonstrate how to add a predefined filter on the Accounts module to return all records with an account type of "Customer" and industry of "Other".

To create a predefined filter, create a display label extension in ./custom/Extension/modules/<module>/Ext/Language/. For this example, we will create:

./custom/Extension/modules/Accounts/Ext/Language/en_us.filterAccountByTypeAndIndustry.php

```
<?php
```

```
$mod_strings['LBL_FILTER_ACCOUNT_BY_TYPE_AND_INDUSTRY'] = 'Customer/Ot
her Accounts';
```

Next, create a custom filter extension in ./custom/Extension/modules/<module>/Ext/clients/base/filters/basic/.

For this example, we will create:

./custom/Extension/modules/Accounts/Ext/clients/base/filters/basic/filterAccountBy TypeAndIndustry.php

```
<?php

$viewdefs['Accounts']['base']['filter']['basic']['filters'][] = array(
    'id' => 'filterAccountByTypeAndIndustry',
    'name' => 'LBL_FILTER_ACCOUNT_BY_TYPE_AND_INDUSTRY',
    'filter_definition' => array(
        array(
            'account_type' => array(
                '$in' => array(
                    'Customer',
                ),
            ),
        ),
        array(
            'industry' => array(
                '$in' => array(
                    'Other',
                ),
            ),
        ),
    ),
    'editable' => false,
    'is_template' => false,
);
```

You should notice that the editable and is_template options have been set to "false". If editable is not set to "false", the filter will not be displayed in the list view filter's list.

Finally, navigate to Admin > Repair and click "Quick Repair and Rebuild" to

rebuild the extensions and make the predefined filter available for users.

Adding Initial Filters to Lookup Searches

To add initial filters to record lookups and type-ahead searches, define a filter template. This will allow you to filter results for users when looking up a parent related record. The following example will demonstrate how to add an initial filter for the Account lookup on the Contacts module. This initial filter will limit records to having an account type of "Customer" and a dynamically assigned user value determined by the contact's assigned user.

To add an initial filter to the Contacts record view, create a display label for the filter in ./custom/Extension/modules/<module>/Ext/Language/. For this example , we will create:

./custom/Extension/modules/Accounts/Ext/Language/en_us.filterAccountTemplate.php

```
<?php

$mod_strings['LBL_FILTER_ACCOUNT_TEMPLATE'] = 'Customer Accounts By A Dynamic User';
```

Next, create a custom template filter extension in ./custom/Extension/modules/<module>/Ext/clients/base/filters/basic/. For this example, create:

./custom/Extension/modules/Accounts/Ext/clients/base/filters/basic/filterAccountTemplate.php

```
<?php

$viewdefs['Accounts']['base']['filter']['basic']['filters'][] = array(
    'id' => 'filterAccountTemplate',
    'name' => 'LBL_FILTER_ACCOUNT_TEMPLATE',
    'filter_definition' => array(
        array(
            'account_type' => array(
                '$in' => array(),
            ),
        ),
        array(
            'assigned_user_id' => ''
        )
```

```
    ),
    'editable' => true,
    'is_template' => true,
);
```

As you can see, the filter_definition contains arrays for account_type and assigned_user_id. These filter definitions will receive their values from the contact record view's metadata. You should also note that this filter has is_template and editable set to "true". This is required for initial filters.

Once the filter template is in place, modify the contact record view's metadata. To accomplish this, edit ./custom/modules/Contacts/clients/base/views/record/record.php to adjust the account_name field. If this file does not exist in your local Sugar installation, navigate to Admin > Studio > Contacts > Layouts > Record View and click "Save & Deploy" to generate it. In this file, identify the panel_body array as shown below:

```
1 =>
array (
    'name' => 'panel_body',
    'label' => 'LBL_RECORD_BODY',
    'columns' => 2,
    'labelsOnTop' => true,
    'placeholders' => true,
    'newTab' => false,
    'panelDefault' => 'expanded',
    'fields' =>
    array (
        0 => 'title',
        1 => 'phone_mobile',
        2 => 'department',
        3 => 'do_not_call',
        4 => 'account_name',
        5 => 'email',
    ),
),
```

Next, modify the account_name field to contain the initial filter parameters.

```
1 =>
array (
    'name' => 'panel_body',
```

```
    'label' => 'LBL_RECORD_BODY',
    'columns' => 2,
    'labelsOnTop' => true,
    'placeholders' => true,
    'newTab' => false,
    'panelDefault' => 'expanded',
    'fields' =>
    array (
        0 => 'title',
        1 => 'phone_mobile',
        2 => 'department',
        3 => 'do_not_call',
        4 => array (
            //field name
            'name' => 'account_name',

            //the name of the filter template
            'initial_filter' => 'filterAccountTemplate',

            //the display label for users
            'initial_filter_label' => 'LBL_FILTER_ACCOUNT_TEMPLATE',

            //the hardcoded filters to pass to the templates filter de
finition
            'filter_populate' => array(
                'account_type' => array('Customer')
            ),

            //the dynamic filters to pass to the templates filter defi
nition
            //please note the index of the array will be for the field
 the data is being pulled from
            'filter_relate' => array(
                //'field_to_pull_data_from' => 'field_to_populate_data
_to'
                'assigned_user_id' => 'assigned_user_id',
            )
        ),
        5 => 'email',
    ),
),
```

Finally, navigate to Admin > Repair and click "Quick Repair and Rebuild". This will rebuild the extensions and make the initial filter available for users when selecting

a parent account for a contact.

Adding Initial Filters to Drawers from a Controller

When creating your own views, you may need to filter a drawer called from within your custom controller. Using an initial filter, as described in the Adding Initial Filters to Lookup Searches section, we can filter a drawer with predefined values by creating a filter object and populating the config.filter_populate property as shown below:

```
//create filter
var filterOptions = new app.utils.FilterOptions()
    .config({
        'initial_filter': 'filterAccountTemplate',
        'initial_filter_label': 'LBL_FILTER_ACCOUNT_TEMPLATE',
        'filter_populate': {
            'account_type': ['Customer'],
            'assigned_user_id': 'seed_sally_id'
        }
    })
    .format();

//open drawer
app.drawer.open({
    layout: 'selection-list',
    context: {
        module: 'Accounts',
        filterOptions: filterOptions,
        parent: this.context
    }
});
```

To create a filtered drawer with dynamic values, create a filter object and populate the config.filter_relate property using the populateRelate method as shown below:

```
//record to filter related fields by
var contact = app.data.createBean('Contacts', {
    'first_name': 'John',
    'last_name': 'Smith',
    'assigned_user_id': 'seed_sally_id'
});

//create filter
var filterOptions = new app.utils.FilterOptions()
```

```
    .config({
        'initial_filter': 'filterAccountTemplate',
        'initial_filter_label': 'LBL_FILTER_ACCOUNT_TEMPLATE',
        'filter_populate': {
            'account_type': ['Customer'],
        },
        'filter_relate': {
            'assigned_user_id': 'assigned_user_id'
        }
    })
    .populateRelate(contact)
    .format();

//open drawer
app.drawer.open({
    layout: 'selection-list',
    context: {
        module: 'Accounts',
        filterOptions: filterOptions,
        parent: this.context
    }
});
```

Duplicate Check

Overview

The duplicate-check framework provides the capability to alter how the system searches for duplicate records in the database when creating records. For information on duplicate checking during imports, please refer to the index documentation.

Default Strategy

The default duplicate-check strategy, located in ./data/duplicatecheck/FilterDuplicateCheck.php, is referred to as FilterDuplicateCheck. This strategy utilizes the Filter API and a defined filter in the vardefs of the module to search for duplicates and rank them based on matching data.

Custom Filter

To alter a defined filter for a stock a module, you only need to update the Vardefs using the Extensions framework. If you are working with a custom module, you can

modify the vardefs in ./modules/<module>/vardefs.php directly. The FilterDuplicateCheck Strategy accepts two properties in its metadata:

Name	Type	Description
filter_template	array	An array containing the Filter Definition for which fields to search for duplicates on. Please consult the Filter API for further information on the filter syntax
ranking_fields	array	A list of arrays with the following properties. The order in which you list the fields for ranking will determine the ranking score. The first ranks higher, than those after it. **in_field_name**: Name of field in vardefs **dupe_field_name**: Name of field returned by filter

Example

The following example will demonstrate how to manipulate the stock Accounts duplicate check to not filter on the shipping address city. The default Account duplicate_check defintion, located in ./modules/Accounts/vardefs.php, is shown below.

```
  ...
'duplicate_check' => array(
    'enabled' => true,
    'FilterDuplicateCheck' => array(
        'filter_template' => array(
            array(
                '$or' => array(
                    array('name' => array('$equals' => '$name')),
                    array('duns_num' => array('$equals' => '$duns_num'
)),
                    array(
                        '$and' => array(
                            array('name' => array('$starts' => '$name'
```

```
)),
                                array(
                                    '$or' => array(
                                        array('billing_address_city' => ar
ray('$starts' => '$billing_address_city')),
                                        array('shipping_address_city' => a
rray('$starts' => '$shipping_address_city')),
                                    )
                                ),
                            )
                        ),
                    )
                ),
            ),
            'ranking_fields' => array(
                array('in_field_name' => 'name', 'dupe_field_name' => 'nam
e'),
                array('in_field_name' => 'billing_address_city', 'dupe_fie
ld_name' => 'billing_address_city'),
                array('in_field_name' => 'shipping_address_city', 'dupe_fi
eld_name' => 'shipping_address_city'),
            )
        )
),
...
```

To add or remove fields from the check, you will need to manipulate $dictionary['<module>']['duplicate_check']['FilterDuplicateCheck']['filter_template'] and $dictionary['<module>']['duplicate_check']['FilterDuplicateCheck']['ranking_fields'] . For our example, we will simply remove the references to shipping_address_city. It is important to familiarize yourself with filter operators before making any changes.

./custom/Extension/modules/Accounts/Ext/Vardefs/newFilterDuplicateCheck.php

```
  <?php

$dictionary['Account']['duplicate_check']['FilterDuplicateCheck'] = ar
ray(
    'filter_template' => array(
        array(
            '$or' => array(
                array('name' => array('$equals' => '$name')),
                array('duns_num' => array('$equals' => '$duns_num')),
```

```
                array(
                    '$and' => array(
                        array('name' => array('$starts' => '$name')),
                        array(
                            '$or' => array(
                                array('billing_address_city' => array(
'$starts' => '$billing_address_city')),
                            )
                        ),
                    )
                ),
            )
        ),
    ),
    'ranking_fields' => array(
        array('in_field_name' => 'name', 'dupe_field_name' => 'name'),
        array('in_field_name' => 'billing_address_city', 'dupe_field_n
ame' => 'billing_address_city'),
    )
);
```

Finally, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then enable the custom duplicate-check class.

If you want to disable the duplicate check entirely, you can set $dictionary['<module>']['duplicate_check']['enabled'] to false in your vardefs and run a Quick Repair and Rebuild.

```
  $dictionary['Account']['duplicate_check']['enabled'] = false;
```

Custom Strategies

Custom duplicate-check class files are stored under ./custom/data/duplicatecheck/. The files in this directory can be enabled on a module's duplicate_check property located in ./custom/Extension/modules/<module>/Ext/Vardefs/. Only one duplicate-check class can be enabled on a module at a given time.

Duplicate Check Class

To create a custom duplicate-check strategy, you need to create a custom duplicate-check class that extends the base DuplicateCheckStrategy,

./data/duplicatecheck/DuplicateCheckStrategy.php. To work, this custom class requires the implementation of two methods:

Method Name	Description
setMetadata	Sets up properties for duplicate-check logic based on the passed-in metadata
findDuplicates	Finds possible duplicate records for a given set of field data

Example

The following example will create a new duplicate-check class called "OneFieldDuplicateCheck" that will query the database based on the configured field for records that contain data similar to that one field:

./custom/data/duplicatecheck/OneFieldDuplicateCheck.php

```
  <?php

if(!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry Point
');

class OneFieldDuplicateCheck extends DuplicateCheckStrategy
{

    protected $field;

    public function setMetadata($metadata)
    {
        if (isset($metadata['field'])) {
            $this->field = $metadata['field'];
        }
    }

    public function findDuplicates()
    {
        if (empty($this->field)){
            return null;
        }

        $Query = new SugarQuery();
        $Query->from($this->bean);
        $Query->where()->ends($this->field,$this->bean->{$this->field}
);
        $Query->limit(10);
```

```
        //Filter out the same Bean during Edits
        if (!empty($this->bean->id)) {
            $Query->where()->notEquals('id',$this->bean->id);
        }
        $results = $Query->execute();
        return array(
            'records' => $results
        );
    }
}
```

Vardef Settings

The duplicate-check vardef settings are configured on each module and can be altered using the Extension framework.

Name	Type	Description
enabled	boolean	Whether or not duplicate-check framework is enabled on the module
<class_name>	array	The class name that will provide duplicate checking, set to an array of metadata that gets passed to the duplicate-check class

If you want to enable the OneFieldDuplicateCheck strategy (shown above) for the Accounts module, you must create the following file:

./custom/Extension/modules/Accounts/Ext/Vardefs/newDuplicateCheck.php

```
  <?php

$dictionary['Account']['duplicate_check'] = array(
 'enabled' => true,
 'OneFieldDuplicateCheck' => array(
            'field' => 'name'
        )
);
```

Finally, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then enable the custom duplicate-check class.

Programmatic Usage

You can also use the duplicate-check framework in code such as in a logic hook or a scheduler. The following example shows how to use the module's defined duplicate-check strategy when utilizing a bean object by simply calling the findDuplicates method on the bean object:

```
  $account = BeanFactory::newBean('Accounts');
$account->name = 'Test';
$duplicates = $account->findDuplicates();

if (count($duplicates['records'])>0){
    $GLOBALS['log']->fatal("Duplicate records found for Account");
}
```

Elastic Search

Overview

The focus of this document is to guide developers on how Sugar integrates with Elastic Search.

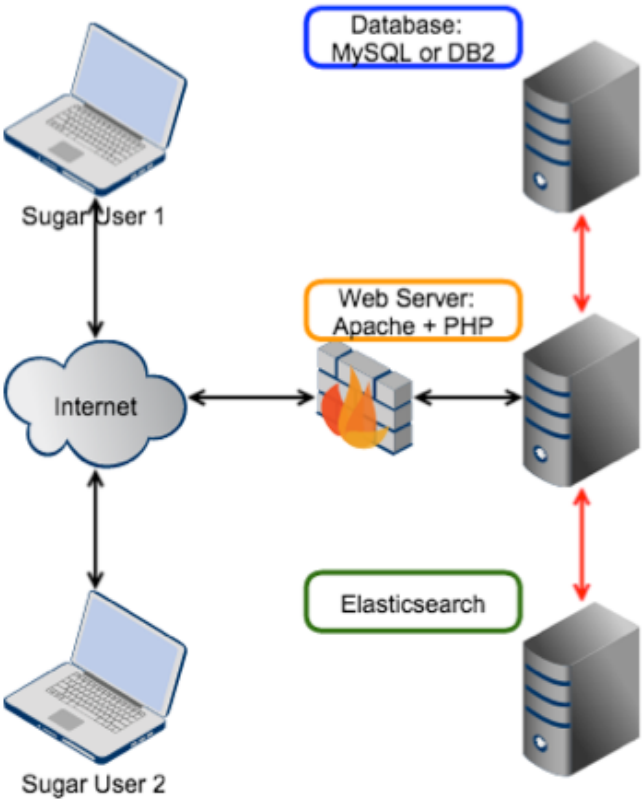
Resources

We recommend the following resources to get started with Elasticsearch:

- Elasticsearch: The Definitive Guide
- Elasticsearch Reference

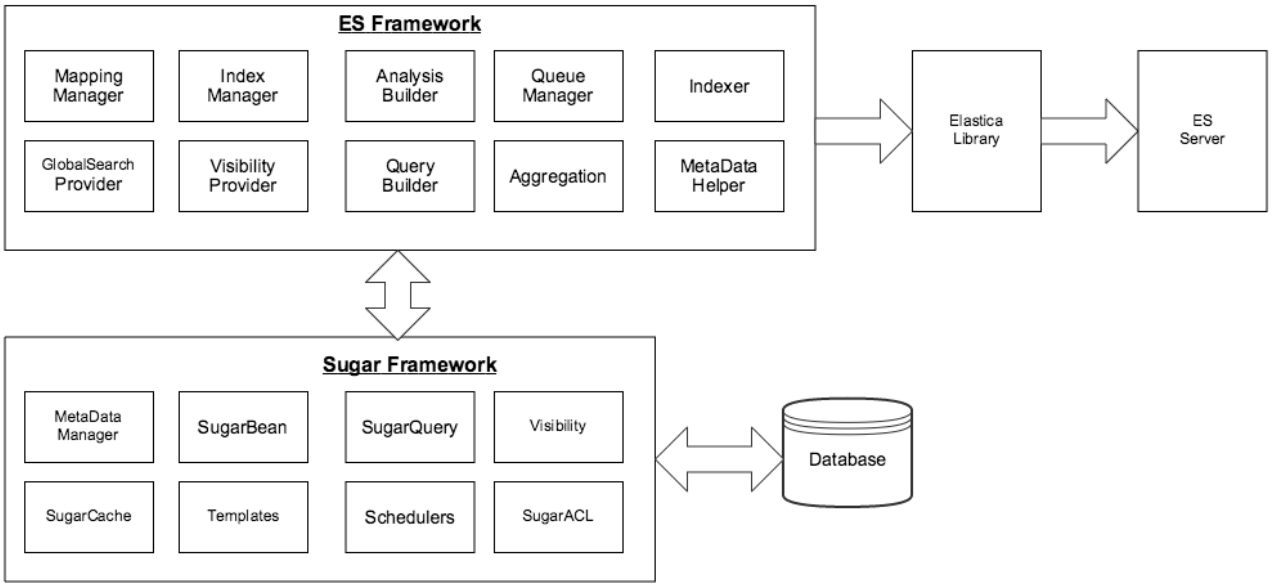
Deployment

The following figure shows a typical topology of Elasticsearch with a Sugar system. All the communication with Elasticsearch goes through Sugar via REST APIs.



Framework

The following figure shows the overall architecture with main components in the Elasticsearch framework and related components in Sugar framework.



Main Components

This section describes how to use each of the main components in detail.

Mapping Manager

This component builds the mappings for providers in the system. The mappings are used for Elastic to interpret data stored there, which is similar to a database schema.

1. Define the full_text_search property in a module's vardef file for a given field:

- **enabled** : sets the field to be enabled for global search;
- **searchable** : sets the field to be searchable or not;
- **boost** : sets the boost value so that the field can be ranked differently at search time.
- **type** : overrides the sugar type originally defined;
- **aggregations** : sets the properties specific related to aggregations;

2. Define what analyzers to use for a given sugar type:

- The analyzers for both indexing and searching are specified. They may be different for the same field. The index analyzer is used at indexing time, while the search analyzer is used at query time.
- Each of the providers generates its own mapping.

3. Generates mappings for each field and sends it over to Elasticsearch.

Example Mapping

This section outlines how the mapping is created for Account module's name field. The Account module's name field is defined as "searchable" in the "full_text_search" index of the vardefs. The Sugar type "name" uses the regular and ngram string analyzers for indexing and search.

./modules/Accounts/vardefs.php

```
  ...
'name' => array(
    'name' => 'name',
    'type' => 'name',
    'dbType' => 'varchar',
    'vname' => 'LBL_NAME',
    'len' => 150,
    'comment' => 'Name of the Company',
    'unified_search' => true,
    'full_text_search' => array(
        'enabled' => true,
        'searchable' => true,
        'boost' => 1.9099999999999999,
    ),
    'audited' => true,
    'required' => true,
    'importable' => 'required',
    'duplicate_on_record_copy' => 'always',
    'merge_filter' => 'selected',
),
...
```

The field handler, located in ./src/Elasticsearch/Provider/GlobalSearch/Handler/Implement/MultiFieldHandler.php, defines the mappings used by Elasticsearch. The class property $typesMultiField in the MultiFieldHandler class defines the relationship between types and mappings. For the "name" type, we use the gs_string and gs_string_wildcard definitions to define our mappings and analyzers.

./src/Elasticsearch/Provider/GlobalSearch/Handler/Implement/MultiFieldHandler.php

```
  ...
protected $typesMultiField = array(
    ...
    'name' => array(
        'gs_string',
        'gs_string_wildcard',
    ),
    ...
);
...
```

The class property $multiFieldDefs in the MultiFieldHandler class defines the actual Elasticsearch mapping to be used.

```
  ...
protected $multiFieldDefs = [
    ...
    /*
     * Default string analyzer with full word matching base ond
     * the standard analyzer. This will generate hits on the full
     * words tokenized by the standard analyzer.
     */
    'gs_string' => [
        'type' => 'text',
        'index' => true,
        'analyzer' => 'gs_analyzer_string',
        'store' => true,
    ],

    /*
     * String analyzer using ngrams for wildcard matching. The
     * weighting of the hits on this mapping are less than full
     * matches using the default string mapping.
     */
    'gs_string_wildcard' => [
        'type' => 'text',
        'index' => true,
        'analyzer' => 'gs_analyzer_string_ngram',
        'search_analyzer' => 'gs_analyzer_string',
        'store' => true,
    ],
```

```
    ...
];
...
```

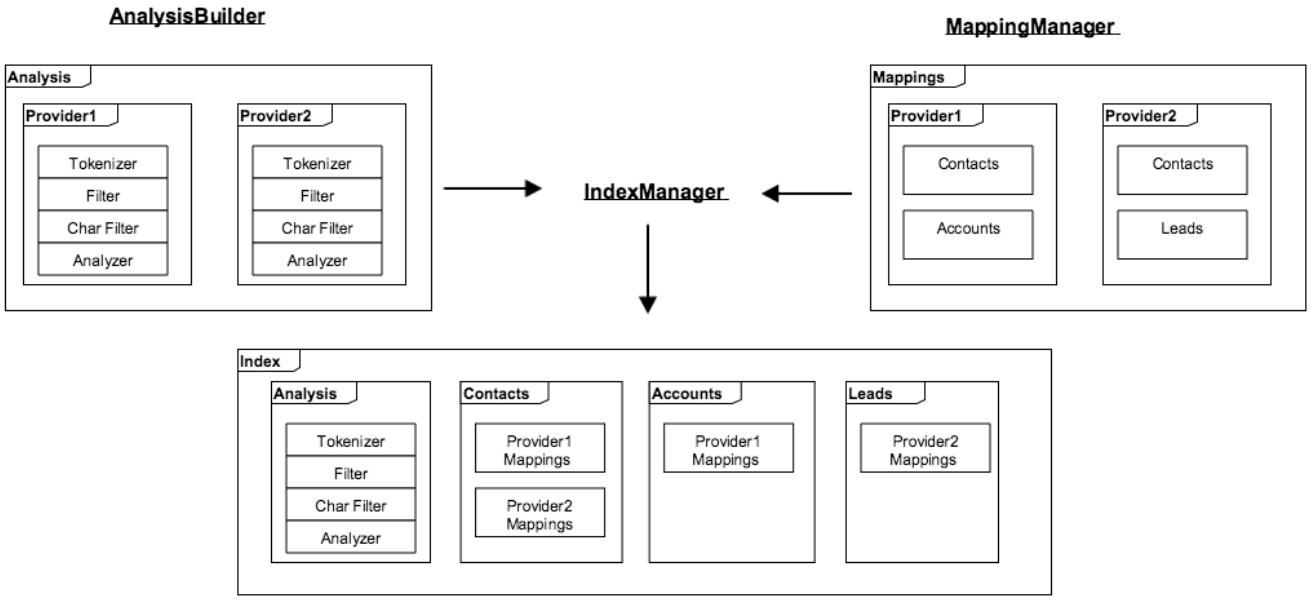
The mapping is created from the definitions in Elasticsearch. A sample mapping is shown below:

```
  "Accounts__name": {
 "include_in_all": false,
 "index": "not_analyzed",
 "type": "string",
 "fields":{
  "gs_string_wildcard":{
   "index_analyzer": "gs_analyzer_string_ngram",
   "store": true,
   "search_analyzer": "gs_analyzer_string",
   "type": "string"
  },
  "gs_string":{
   "store": true,
   "analyzer": "gs_analyzer_string",
   "type": "string"
  }
 }
}
```

Index Manager

An index includes types, similar to a database including tables. In our system, a type is based on a module consisting of mappings.

As shown in the following figure, the Index Manager combines the analysis built from Analysis Builder and the mappings from Mapping Manager from different providers and organizes them by modules.

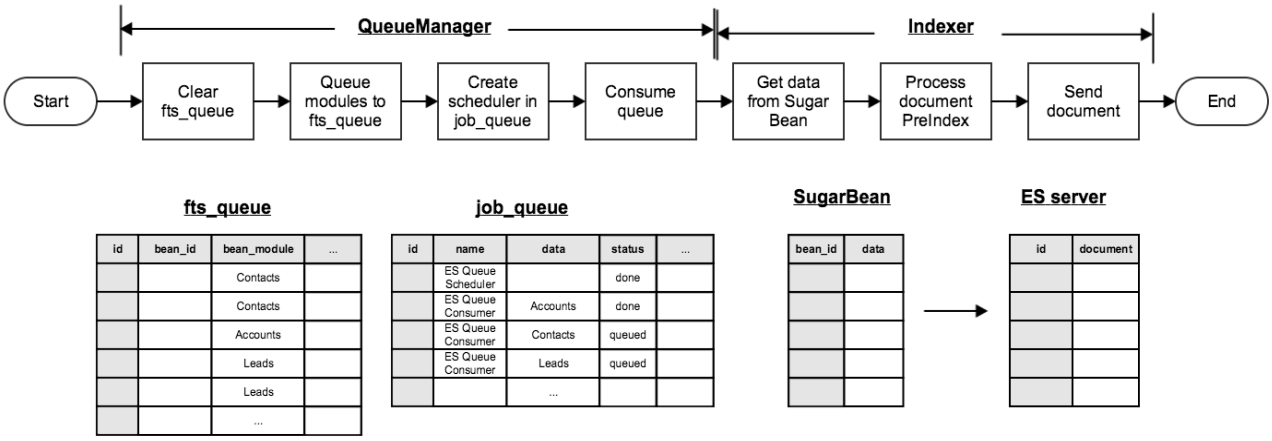


Indexer & Queue Manager

After creating the index, data needs to be moved from the Sugar system to Elasticsearch. The queue manager uses the fts_queue and job_queue tables to queue the information. The indexer then processes data from Sugar bean into the format of Elasticsearch documents.

The whole process is as follows:

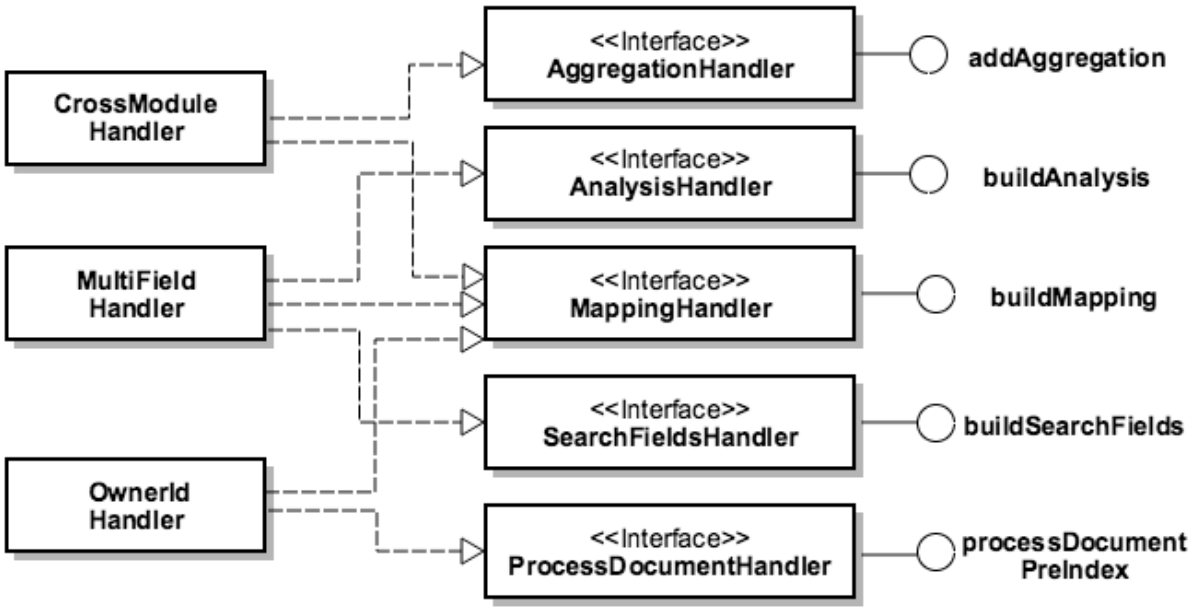
1. Each Sugar bean is added to fts_queue table.
2. Elasticsearch queue scheduler is added to job_queue table.
3. When the job starts, the scheduler generates an Elasticsearch queue consumer for each module.
4. Each consumer queries the fts_queue table fo find the corresponding sugar beans.
5. The indexer gets fields from each bean and processes them into individual documents.
6. The indexer uses the bulk APIs to send documents to Elasticsearch in batches.



Global Search Provider

The provider supplies information to build analysis, mapping, query, etc. It is done by using handlers. Currently, there are two providers in the system: global search and visibility. To extend the functionalities, new providers and handlers may be added.

The following figure shows the existing handler interfaces and some of the handlers implementing them.



Query Builder

The Query builder composes the query string for search. A structure of a typical multi-match query is shown as follows:

```
  {
    "query": {
        "filtered": {
            "query": {
                "bool": {
                    "must": [{
                        "bool": {
                            "should": [{
                                "multi_match": {
                                    "type": "cross_fields",
                                    "query": "test",
                                    "fields": [
                                        "Cases__name.gs_string^1.53",
                                        "Cases__name.gs_string_wildcar
d^0.69",
                                        "Cases__description.gs_string^
0.66",
                                        "Cases__description.gs_text_wi
ldcard^0.23",
                                        "Bugs__name.gs_string^1.51",
                                        "Bugs__name.gs_string_wildcard
^0.68",
```

```
                                        "Bugs__description.gs_string^0
.68",
                                        "Bugs__description.gs_text_wil
dcard^0.24"
                                    ],
                                    "tie_breaker": 1
                                }
                            }]
                        }
                    },
                    {
                        "bool": {
                            "should": [{
                                "multi_match": {
                                    "type": "cross_fields",
                                    "query": "query",
                                    "fields": [
                                        "Cases__name.gs_string^1.53",
                                        "Cases__name.gs_string_wildcar
d^0.69",
                                        "Cases__description.gs_string^
0.66",
                                        "Cases__description.gs_text_wi
ldcard^0.23",
                                        "Bugs__name.gs_string^1.51",
                                        "Bugs__name.gs_string_wildcard
^0.68",
                                        "Bugs__description.gs_string^0
.68",
                                        "Bugs__description.gs_text_wil
dcard^0.24"
                                    ],
                                    "tie_breaker": 1
                                }
                            }]
                        }
                    }]
                }
            },
            "filter": {
                "bool": {
                    "must": [{
                        "bool": {
                            "should": [{
                                "bool": {
                                    "must": [{
```

```
                                    "term": {
                                        "_type": "Bugs"
                                    }
                                }]
                            }
                        },
                        {
                            "bool": {
                                "must": [{
                                    "term": {
                                        "_type": "Cases"
                                    }
                                }]
                            }
                        }]
                    }
                }]
            }
        }
    },
    "highlight": {
        "pre_tags": [
            "<strong>"
        ],
        "post_tags": [
            "<\/strong>"
        ],
        "require_field_match": 1,
        "number_of_fragments": 3,
        "fragment_size": 255,
        "encoder": "html",
        "order": "score",
        "fields": {
            "*.gs_string": {
                "type": "plain",
                "force_source": false
            },
            "*.gs_string_exact": {
                "type": "plain",
                "force_source": false
            },
            "*.gs_string_html": {
                "type": "plain",
                "force_source": false
            },
```

```
            "*.gs_string_wildcard": {
                "type": "plain",
                "force_source": false
            },
            "*.gs_text_wildcard": {
                "type": "plain",
                "force_source": false
            },
            "*.gs_phone_wildcard": {
                "type": "plain",
                "force_source": false
            },
            "*.gs_email": {
                "type": "plain",
                "force_source": false,
                "number_of_fragments": 0
            },
            "*.gs_email_wildcard": {
                "type": "plain",
                "force_source": false,
                "number_of_fragments": 0
            }
        }
    }
}
```

Figure 7: Multi-match query with aggregations.

A query may include multiple filters, post filters, queries, and settings, which can get complicated sometimes. To add new queries, we recommend adding new classes implementing *QueryInterface*, instead of modifying Query Builder or even calling Elastica APIs directly.

ACL

ACL control is done in the Query Builder. The search fields are filtered based on their ACL access levels when being added to the query string in a specific query. For instance, MultiMatchQuery used for global search may include the following filterings:

1. If a field is owner read (i.e., SugarACL::ACL_OWNER_READ_WRITE), it will be added to a sub-query with a filter of ownerId.
2. If a field is not owner read, its access level must not equal to SugarACL::ACL_NO_ACCESS, in order to be added as one of the search

fields in the query.

The corresponding function is MultiMatchQuery.php::isFieldAccessible(). Potentially this function can be shared by different queries that require ACL control.

Visibility

The visibility model is applied when building the query too. It is done by adding filters built by individual visibility strategies, defined in ./data/visibility/ directory. These strategies implement StrategyInterface including:

1. building analysis,
2. building mapping,
3. processing document before being indexed,
4. getting bean index fields,
5. adding visibility filters.

The functions are similar to the global search provider's handler interfaces. The query builder only uses adding the visibility filters, while others provide information to Analysis builder, Mapping Manager, Indexer, etc. Hence the visibility provider is separated as an independent provider in the framework.

Global Search

Overview

How to customize the global search results.

Configuring Search Results

The Search-List view, located in ./clients/base/views/search-list/, is responsible for the global search results page. By default, it contains the rowactions that are available for each result. The following sections will outline how to configure the primary and secondary fields.

Primary and Secondary Fields

Each row returned from the global search is split into two lines. The first line contains the primary fields and avatar. A primary field is a field considered to be important to the identification of a record and is usually composed of the records name value and a link to the record. The avatar is an icon specific to the module the results are coming from.

The second line contains the highlighted matches and any secondary fields specified. A highlight is a hit returned by elastic search. Highlights are driven from elastic and no definition is needed in the metadata to define them. Both primary and secondary fields can be highlighted. A secondary field is a field that has regularly accessed data. It is important to note that the initial type-ahead drop-down will not show specified secondary fields unless the user hits enter and is taken to the search results page. Different modules have different secondary fields. For example, an account's secondary fields are email and phone number, while cases' secondary fields are case ID number and related account.

To leverage the metadata manager, the global search view reuses the same metadata format as other views. You can add any indexed field in elastic search to the search list view. An example of the stock metadata is shown below.

./modules/{module}/clients/base/views/search-list/search-list.php

```
<?php

$viewdefs[$moduleName]['base']['view']['search-list'] = array(
    'panels' => array(
        array(
            'name' => 'primary', // This is mandatory and the word `pr
imary` can not be changed.
            'fields' => array(
                /*
                 * Put here the list of primary fields.
                 * You can either define a field as a string and the m
etadata
                 * manager will load the field vardefs, or you can def
ine as an
                 * array if you want to add properties or override var
defs
                 * properties just for the search results view.
                 *
                 * You most likely want to show the module icon on the
 left of the
                 * view. For this, you need to add the following pictu
re field as
                 * a primary field.
                 */
                array(
                    'name' => 'picture',
                    'type' => 'avatar',
                    'size' => 'medium',
                    'css_class' => 'pull-left',
                ),
```

```
                array(
                    'name' => 'name',
                    'type' => 'name',
                    'link' => true,
                    'label' => 'LBL_SUBJECT',
                ),
            ),
        ),
        /*
         * If you don't want secondary fields, you can remove the foll
owing array.
         */
        array(
            'name' => 'secondary', // This is mandatory and the word `
secondary` can not be changed.
            'fields' => array(
                /*
                 * Put here the list of secondary fields
                 */
                'status', // This field is defined as a string for exa
mple.
                array(
                    'name' => 'description',
                    'label' => 'LBL_SEARCH_DESCRIPTION',
                ),
            ),
        ),
    ),
);
```

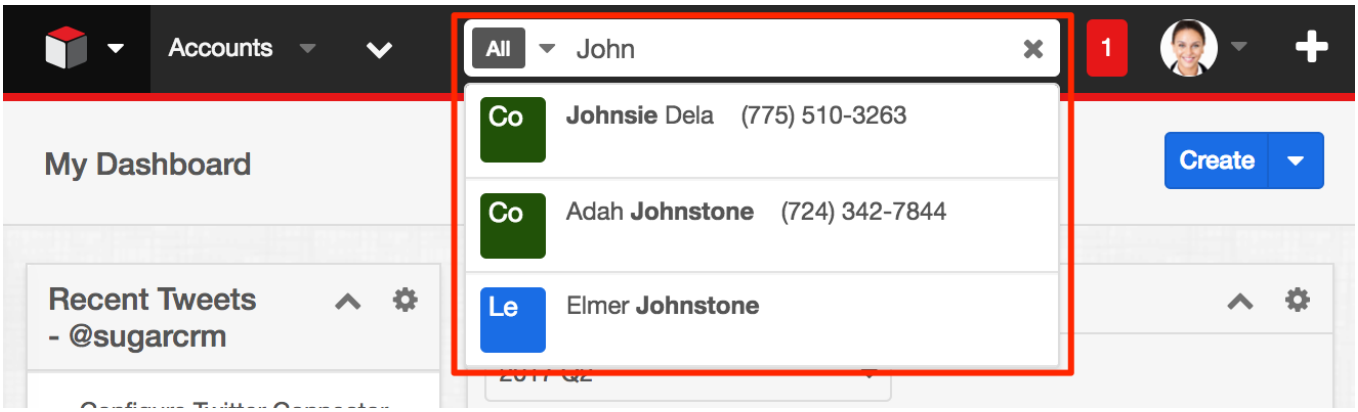
Note: You can technically override the row actions for the results of a module, however, the view currently only supports the preview action so it is not recommended to customizes these actions.

Examples

The following examples demonstrate how to modify the primary and secondary rows of the global search.

Adding Fields to the Primary Row

This example will demonstrate how to add "Mobile" into the search-list view's primary row for Contact module results.



To accomplish this, you can duplicate ./modules/Contacts/clients/base/views/search-list/search-list.php to ./custom/modules/Contacts/clients/base/views/search-list/search-list.php if it doesn't exist. Once in place, add your phone_mobile definition to the $viewdefs['Contacts']['base']['view']['search-list']['panels'] array definition with the name attribute equal to "primary'. An example is shown below:

./custom/modules/Contacts/clients/base/views/search-list/search-list.php

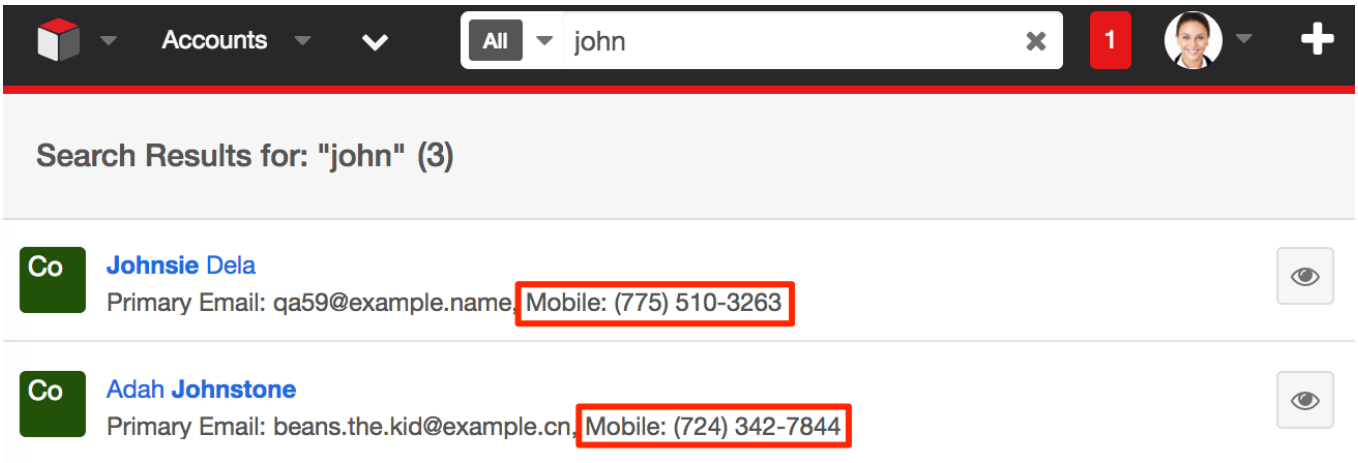
```
<?php

$viewdefs['Contacts']['base']['view']['search-list'] = array(
    'panels' => array(
        array(
            'name' => 'primary',
            'fields' => array(
                array(
                    'name' => 'picture',
                    'type' => 'avatar',
                    'size' => 'medium',
                    'readonly' => true,
                    'css_class' => 'pull-left',
                ),
                array(
                    'name' => 'name',
                    'type' => 'name',
                    'link' => true,
                    'label' => 'LBL_SUBJECT',
                ),
                // Adding the mobile into first row of the results
                'phone_mobile',
            ),
```

```
        ),
        array(
            'name' => 'secondary',
            'fields' => array(
                array(
                    'name' => 'email',
                    'label' => 'LBL_PRIMARY_EMAIL',
                ),
            ),
        ),
    ),
);
```

Once in place, navigate to Admin > Repair > Quick Repair and Rebuild. Your changes will then be reflected in the system.

Adding Fields to the Secondary Row

This example will demonstrate how to add "Mobile" into the search-list view's secondary row for Contact module results. It is important to note that the initial type-ahead drop-down will not show specified secondary fields unless the user hits enter and is taken to the search results page.



In this example, we are going to add Portal Name into search-list for Contact module results.

To accomplish this, you can duplicate ./modules/Contacts/clients/base/views/search-list/search-list.php to ./custom/modules/Contacts/clients/base/views/search-

list/search-list.php if it doesn't exist. Once in place, add your phone_mobile definition to the $viewdefs['Contacts']['base']['view']['search-list']['panels'] array definition with the name attribute equal to "secondary'. An example is shown below:

```
<?php

$viewdefs['Contacts']['base']['view']['search-list'] = array(
    'panels' => array(
        array(
            'name' => 'primary',
            'fields' => array(
                array(
                    'name' => 'picture',
                    'type' => 'avatar',
                    'size' => 'medium',
                    'readonly' => true,
                    'css_class' => 'pull-left',
                ),
                array(
                    'name' => 'name',
                    'type' => 'name',
                    'link' => true,
                    'label' => 'LBL_SUBJECT',
                ),
            ),
        ),
        array(
            'name' => 'secondary',
            'fields' => array(
                array(
                    'name' => 'email',
                    'label' => 'LBL_PRIMARY_EMAIL',
                ),
                // Adding mobile to second row for search results in g
lobalsearch
                'phone_mobile'
            ),
        ),
    ),
);
```

Once in place, navigate to Admin > Repair > Quick Repair and Rebuild. Your changes will then be reflected in the system.

Sugar Logic

Overview

Sugar Logic, a feature in all Sugar products, enables custom business logic that is easy to create, manage, and reuse on both the server and client.

Sugar Logic is made up of multiple components that build on each other and is extensible at every step. The base component is the Sugar Formula Engine, which parses and evaluates human-readable formulas. Dependencies are units made up of triggers and actions that can express custom business logic. Each dependency defines a list of actions to be performed depending on the outcome of a trigger formula.

Terminology

- **Formula** : An expression that conforms to the Formula Engine syntax, consisting of nested functions and field variables
- **Function** : A method that can be called in a formula
- **Trigger** : A formula that evaluates to either true or false when a field in the equation is updated or when a record is retrieved/saved
- **Action** : A method that modifies the current record or layout in some way
- **Dependency** : A complete logical unit that includes a trigger and one or more actions

Sugar Formula Engine

Formulas

The fundamental object is called a formula. A formula can be evaluated for a given record using the Sugar Logic parser.

Some example formulas are:

Basic addition:

```
add(1, 2)
```

Boolean values:

```
not(equal($billing_state, "CA"))
```

Calculation:

```
multiply(number($employees), $seat_cost, 0.0833)
```

Types

Sugar Logic has several fundamental types: number, string, boolean, and enum (lists). Functions may take in any of these types or combinations thereof and return as output one of these types. Fields may also often have their value set to only a certain type.

Number Type

Number types essentially represent any real number (which includes positive, negative, and decimal numbers). They can be plainly typed as input to any function. For example, the operation (10 + 10 + (15 - 5)) can be performed as follows:

```
add(10, 10, subtract(15, 5))
```

String Type

A string type is very much like a string in most programming languages. However, it can only be declared within double quotes. For example, consider this function, which returns the length of the input string:

```
strlen("Hello World")
```

The function would appropriately return the value 11.

Boolean Type

A boolean type is simple. It can be one of two values: "true" or "false". This type mainly acts as a flag that indicates whether a condition has been met or not. For example, this function contains takes two strings as input and returns "true" if the

first string contains the second string, or "false" otherwise:

```
and(contains("Hello World", "llo Wo"), true)
```

The function would appropriately return the value "true".

Enum Type (list)

An enum type is a collection of items. The items do not need to all be of the same type, they can be varied. An enum can be declared using the enum function as follows:

```
enum("hello world!", false, add(10, 15))
```

Alternatively, the createList() function (an alias to enum) can be used to create enums in a formula.

```
createList("hello world!", false, add(10, 15))
```

This function would appropriately return an enum type containing "hello world!", false, and 25 as its elements.

Link Type (relationship)

A link represents one side of a relationship and is used to access related records. For example, the accounts link field of Contacts is used to access the account_type field of the account related to a contact:

```
related($accounts, "account_type")
```

For most of the out-of-the-box relationships, links are named with the name of the related module, in lower case.

Follow these steps to find the name of the link fields for relationships that do not follow the convention above:

1. Open the vardef file for the module you are working on: ./cache/modules/{module}/{module}vardefs.php

2. Find the link field that matches the relationship you are looking for.

Functions

Functions are methods to be used in formulas. Each function has a function name, a parameter count, a parameter type requirement, and returns a value. Some functions such as add() can take any number of parameters. For example: add(1), add(1, 2), and add(1, 2, 3) are all valid formulas. Functions are designed to produce the same result whether executed on the server or client.

Triggers

A trigger is an object that listens for changes in field values and, after a change occurs, triggers the associated actions in a dependency.

Actions

Actions are functions that modify a target in some way and are fired when the trigger is TRUE. Most Actions require at least two parameters: a target and a formula. For example, a style action will change the style of a field based on a passed-in string formula. A value action will update a value of a field by evaluating a passed in formula.

notActions

notActions are functions that modify a target in some way and are fired when the trigger is FALSE. notActions are optional and if they are not defined then nothing will fire when the trigger is FALSE. Most notActions require at least two parameters: a target and a formula. For example, a style action will change the style of a field based on a passed-in string formula. A value action will update a value of a field by evaluating a passed in formula.

Dependencies

A Dependency describes a set of rules based on a trigger and a set of actions. Examples include a field whose properties must be updated dynamically or a panel which must be hidden when a drop down value is not selected. When a Dependency is triggered it will appropriately carry out the action it is designed to. A basic Dependency is when a field's value is dependent on the result of evaluating a Expression. For example, consider a page with five fields with It's "a", "b", "c", "d", and "sum". A generic Dependency can be created between "sum" and the other four fields by using an Expression that links them together, in this case an add Expression. So we can define the Expression in this manner: 'add($a, $b, $c, $d)' where each field id is prefixed with a dollar ($) sign so that the value of the field is dynamically replaced at the time of the execution of the Expression.

An example of a more customized Dependency is when the field's style must be somehow updated to a certain value. For example, the DIV with id "temp" must be colored blue. In this case we need to change the background-color property of "temp". So we define a StyleAction in this case and pass it the field id and the style change that needs to be performed and when the StyleAction is triggered, it will change the style of the object as we have specified.

Sugar Logic Based Features

Calculated Fields

Fields with calculated values can now be created from within Studio and Module Builder. The values are calculated based on Sugar Logic formulas. These formulas are used to create a new dependency that are executed on the client side in edit views and the server side on save. The formulas are saved in the varies or vardef extensions and can be created and edited directly. For example, the metadata for a simple calculated commission field in opportunities might look like:

```
  'commission_c' => array(
    'name' => 'commission_c',
    'type' => 'currency',
    'calculated' => true,
    'formula' => 'multiply($amount, 0.1)',
    //enforced causes the field to appear read-only on the layout
    'enforced' => true
),
```

Dependent fields

A dependent field will only appear on a layout when the associated formula evaluates to the Boolean value true. Currently these cannot be created through Studio and must be enabled manually with a custom vardef or vardef extension. The "dependency" property contains the expression that defines when this field should be visible. An example field that only appears when an account has an annual revenue greater than one million.

```
  'dep_field'=> array(
    'name' => 'dep_field',
    'type' => 'varchar',
    'dependency' => 'greaterThan($annual_revenue, 1000000)',
),
```

Dependent dropdowns

Dependent dropdowns are dropdowns for which options change when the selected value in a trigger dropdown changes. The metadata is defined in the vardefs and contains two major components, a "trigger" id which is the name of the trigger dropdown field and a 'visibility grid' that defines the set of options available for each key in the trigger dropdown. For example, you could define a sub-industry field in accounts whose available values depend on the industry field.

```
  'sub_industry_c' => array(
    'name' => 'sub_industry_c',
    'type' => 'enum',
    'options' => 'sub_industry_dom',
    'visibility_grid'=> array(
        'trigger' => 'industry',
        'values' => array(
            'Education' => array(
                'primary',
                'secondary',
                'college'
            ),
            'Engineering' => array(
                'mechanical',
                'electrical',
                'software'
            ),
        ),
    ),
),
```

Clearing the Sugar Logic Cache

The ./updatecache.php script traverses the Expression directory for every file that ends with "Expression.php" (ignoring a small list of excepted files) and constructs both a PHP and a JavaScript functions cache file which resides in ./cache/Expressions/functions_cache.js and ./cache/Expressions/functionmap.php. If you create your custom expressions, you will need to rebuild the sugar logic functions by navigating to:

```
Admin > Repair > Rebuild Sugar Logic Functions
```

Dependency Actions

Overview

The following sections outline the available SugarLogic dependency actions.

Dependency Parameters

Dependency definitions will generally contain most, if not all, of the parameters displayed in the table below. The following sections will outline each dependency action as well as dependency specific parameters.

Parameter	Type	Description
hooks	Array	The views to execute the trigger on. Possible values are: "edit", "view", "save" and "all". If you include 'save' or 'all' then SugarCRM will try to save the calculated value to the database. So if your dependency is display only then only include the views that it will show up on.
trigger	String	Optional. The trigger for the dependency. Defaults to 'true'.
triggerFields	Array	The list of fields to watch for change events. When changed, the trigger expressions will be recalculated.
onload	Boolean	Whether or not to trigger the dependencies when the page is loaded.
actions	Array	The list of dependencies to execute when the trigger expression is true.
notActions	Array	The list of dependencies to execute when the trigger expression is false.

ReadOnly

Overview

The SugarLogic ReadOnly action, located in ./include/Expressions/Actions/ReadOnlyAction.php, is used to determine if a field is editable or not based on a formula.

Implementation

While the dependency metadata for your module can be defined in ./modules/<module>/metadata/dependencydefs.php and ./custom/modules/<module>/metadata/dependencydef.php, it is recommended to use the extension framework when customizing stock modules to prevent third-party plugins from conflicting with your customizations. The following section will demonstrate how to implement a read-only dependency.

ReadOnly Parameters

Parameter	Type	Description
target	String	The name of the field to make read only.
value	String	This parameter can accept a boolean formula or true and false values. Normally you would put a SugarLogic formula here to set the boolean.

For more information on the various parameters in the dependency definitions, please refer to the dependency actions documentation.

Example

For our example, we will create a dependency on the Accounts module that makes the name field read-only when the lock_record_c field has been checked. The first step is to create the lock_record_c checkbox field in Studio and add it to your Record View layout. When this checkbox is checked, we will make the name field read-only. Our example extension definition is shown below:

./custom/Extension/modules/<module>/Ext/Dependencies/custom_name_read_only.php

```
  <?php

$dependencies['Accounts']['readonly_fields'] = array(
    'hooks' => array("edit"),
    'trigger' => 'true',
    //Optional, the trigger for the dependency. Defaults to 'true'.
    'triggerFields' => array('lock_record_c'),
    'onload' => true,
    //Actions is a list of actions to fire when the trigger is true
    // You could list multiple fields here each in their own array und
er 'actions'
    'actions' => array(
        array(
            'name' => 'ReadOnly',
            //The parameters passed in will depend on the action type
set in 'name'
            'params' => array(
                'target' => 'name',
                'value' => 'equal($lock_record_c,true)',
            ),
        ),
    ),
);
```

Once you have all the files in place you will need to navigate to Admin > Repairs > and run a Quick Repair and Rebuild.

Note: It is important that the module name is plural ('Accounts' vs. 'Account') and that the name of the dependency, "readonly_fields" in this example, is unique.

Considerations

1. In some scenarios, you may want a specific field to always be read-only. To accomplish this, you can modify the 'value' attribute to always be "true". Given the above example, you would modify:

   ```
   'value' => 'equal($lock_record_c,true)',
   ```

   to be:

   ```
   'value' => 'true',
   ```

SetOptions

Overview

The SugarLogic SetOptions action, located in ./include/Expressions/Actions/SetOptionsAction.php, is used to set the options list of a dropdown field based on a formula.

Implementation

While the dependency metadata for your module can be defined in ./modules/<module>/metadata/dependencydefs.php and ./custom/modules/<module>/metadata/dependencydef.php, it is recommended to use the extension framework when customizing stock modules to prevent third party plugins from conflicting with your customizations. The following section will demonstrate how to implement a read-only dependency.

Setoptions Parameters

Parameter	Type	Description
target	String	The name of the dropdown field that you want to change the option list for
keys	String	A formula used to get the option list keys for the target field or a list name from which keys will be extracted.
labels	String	A formula used to get the option list labels for the target field or a list name from which labels will be extracted.

For more information on the various parameters in the dependency definitions, please refer to the dependency actions documentation.

Examples

The following sections outline the various ways this dependency can be implemented.

Using an Existing DropDown List

You can also set the options list to any current options list already in the system For example if you wanted to have the industry dropdown in Accounts show the 'bug_type_dom' list from Bugs you could do this

```
  <?php

$dependencies['Leads']['setoptions_industry'] = array(
   'hooks' => array("edit","save"),
   'trigger' => 'true',
   'triggerFields' => array('industry'),
   'onload' => true,
   'actions' => array(
     array(
       'name' => 'SetOptions',
       'params' => array(
         'target' => 'industry',
         'keys' => 'getDropdownKeySet("bug_type_dom")',
         'labels' => 'getDropdownValueSet("bug_type_dom")'
       ),
     ),
   ),
);
```

This would grab the keys and label from the 'bug_type_dom' using the getDropdownKeySet() and getDropdownValueSet() JavaScript functions and display them instead of the normal list.

Once you have the file in place, you will need to navigate to Admin > Repairs > and run a Quick Repair and Rebuild.

Note: It is important that the module name is plural ('Accounts' vs. 'Account') and that the name of the dependency, "setoptions_industry" in this example, is unique.

Complex Dynamic Lists

For our first example, we will change a dropdown called fruits_c to include only fruits that are in season. This could be done with a dependent dropdown but that would require the user to pick the proper season. With this, we can have a function that returns only fruit that is in season right now. I added the dropdown fruits_c to Leads and created a new list for it that looks like this

```
  $app_list_strings['fruits_list']=array (
 'Apples' => 'Apples',
 'Strawberries' => 'Strawberries',
 'Mangos' => 'Mangos',
 'Pineapples' => 'Pineapples',
 'Blackberries' => 'BlackBerries',
 'BlueBerries' => 'BlueBerries',
);
```

To keep it simple I made the labels and the keys the same.

Then I extended SugarLogic as outlined in Extending SugarLogic and made a new function called fruitInSeason() that returns a string reflecting what fruit is in season right now. To work for the createList() function it would return a list like **"Apples","Mangos","BlueBerries"**.

./custom/Extension/modules/<module>/Ext/Dependencies/custom_fruit_in_season.php

```
  <?php

$dependencies['Leads']['setoptions_fruit'] = array(
   'hooks' => array("edit","save"),
   'trigger' => 'true',
   'triggerFields' => array('fruits_c'),
   'onload' => true,
   'actions' => array(
      array(
        'name' => 'SetOptions',
        'params' => array(
           'target' => 'fruits_c',
           'keys' => "createList(fruitInSeason())",
           'labels' => "createList(fruitInSeason())"
        ),
      ),
    ),
);
```

The createList() function is a JavaScript function from Sidecar. It requires a comma delimited quote enclosed list of options.

We only want this to affect EditViews and Saves, not normal record views since

they need to be able to display all fruits and not a truncated list of them. So we make the 'hooks' array

```
'hooks' => array("edit","save"),
```

Once you have the file in place, you will need to navigate to Admin > Repairs > and run a Quick Repair and Rebuild.

Note: It is important that the module name is plural ('Leads' vs. 'Lead') and that the name of the dependency, "setoptions_fruit" in this example, is unique.

SetPanelVisibility

Overview

The SugarLogic SetPanelVisibility action, defined in ./include/Expressions/Actions/PanelVisibilityAction.php, is used to determine the visibility of a record view panel based on a formula.

Implementation

While the dependency metadata for your module can be defined in ./modules/<module>/metadata/dependencydefs.php and ./custom/modules/<module>/metadata/dependencydef.php, it is recommended to use the extension framework when customizing stock modules to prevent third-party plugins from conflicting with your customizations. The following section will demonstrate how to implement a read-only dependency.

SetPanelVisibility Parameters

Parameter	Type	Description
target	String	The id of the panel to hide
value	String	Formula used to determine if the panel should be visible.

For more information on the various parameters in the dependency definitions, please refer to the dependency actions documentation.

Example

For our example, we will create a dependency on the Cases module that will hide a specific panel if the status field on a case is set to "Closed". Our example extension definition is shown below:

./custom/Extension/modules/<module>/Ext/Dependencies/hide_panel_2_dep.php

```
  <?php

$dependencies['Cases']['panel_2_visibility'] = array(
    'hooks' => array("edit","view"),
    'trigger' => 'equal($status, "Closed")',
    'triggerFields' => array('status'),
    'onload' => true,
    //Actions is a list of actions to fire when the trigger is true
    'actions' => array(
        array(
            'name' => 'SetPanelVisibility',
            'params' => array(
                'target' => 'detailpanel_2',
                'value' => 'true',
            ),
        )
    ),
    //notActions is a list of actions to fire when the trigger is fals
e
    'notActions' => array(
        array(
            'name' => 'SetPanelVisibility',
            'params' => array(
                'target' => 'detailpanel_2',
                'value' => 'false',
            ),
        ),
    ),
);
```

Once you have the file in place, you will need to navigate to Admin > Repairs > and run a Quick Repair and Rebuild.

Note: It is important that the module name is plural ('Cases' vs. 'Case') and that the name of the dependency, "panel_2_visibility" in this example, is unique.

SetRequired

Overview

The SugarLogic SetRequired action, located in ./include/Expressions/Actions/SetRequiredAction.php, is used to determine if a field is required.

Implementation

While the dependency metadata for your module can be defined in ./modules/<module>/metadata/dependencydefs.php and ./custom/modules/<module>/metadata/dependencydef.php, it is recommended to use the extension framework when customizing stock modules to prevent third-party plugins from conflicting with your customizations. The following section will demonstrate how to implement a read-only dependency.

SetRequired Parameters

Parameter	Type	Description
target	String	The name of the field to make required.
label	String	id of label element for this field
value	String	Formula used to determine if the field should be required.

For more information on the various parameters in the dependency definitions, please refer to the dependency actions documentation.

Example

For our example, we will create a dependency on the Cases module that will mark the resolution field as required when the status field is set to "Closed". Our example extension definition is shown below:

./custom/Extension/modules/<module>/Ext/Dependencies/required_resolution_dep.php

```
  <?php

$dependencies['Cases']['required_resolution_dep'] = array(
```

```
    'hooks' => array("edit"),
    'trigger' => 'true',
    'triggerFields' => array('status'),
    'onload' => true,
    //Actions is a list of actions to fire when the trigger is true
    'actions' => array(
        array(
            'name' => 'SetRequired',
            //The parameters passed in will depend on the action type
set in 'name'
            'params' => array(
                'target' => 'resolution',
                'label' => 'resolution_label',
                'value' => 'equal($status, "Closed")',
            ),
        ),
    ),
);
```

Once you have the file in place, you will need to navigate to Admin > Repairs > and run a Quick Repair and Rebuild.

Note: It is important that the module name is plural ('Cases' vs. 'Case') and that the name of the dependency, "required_resolution_dep" in this example, is unique.

SetValue

Overview

The SugarLogic SetValue action, located in ./include/Expressions/Actions/SetValueAction.php, is used to set the value of a field based on a formula.

Implementation

While the dependency metadata for your module can be defined in ./modules/<module>/metadata/dependencydefs.php and ./custom/modules/<module>/metadata/dependencydef.php, it is recommended to use the extension framework when customizing stock modules to prevent third party plugins from conflicting with your customizations. The following section will demonstrate how to implement a read-only dependency.

SetValue Parameters

Parameter	Type	Description
target	String	The name of the field to target for visibility.
value	String	SugarLogic formula used to get the value for the target field.

For more information on the various parameters in the dependency definitions, please refer to the dependency actions documentation.

Examples

The follow sections outline the various ways this dependency can be implemented.

Dependency Extensions

For our example, we will create a dependency on the Leads module that will display the number of activities related to a Lead. Activities are composed of calls, meetings, tasks, notes, and emails. An example extension definition is shown below:

./custom/Extension/modules/<module>/Ext/Dependencies/custom_phone_alternate.php

```
  <?php

$dependencies['Leads']['activity_count_dep'] = array(
    'hooks' => array("edit", "view"), //not including save so that the
 value isn't stored in the DB
    'trigger' => 'true', //Optional, the trigger for the dependency. D
efaults to 'true'.
    'onload' => true, //Whether or not to trigger the dependencies whe
n the page is loaded
    'actions' => array(
        array(
            'name' => 'SetValue',
            'params' => array(
                'target' => 'activity_count_c',
                'value' => 'add(
                    count($notes),
                    count($calls),
                    count($emails),
```

```
                    count($meetings),
                    count($tasks)
                )'
            )
        )
    )
);
```

Once you have the file in place, you will need to navigate to Admin > Repairs > and run a Quick Repair and Rebuild.

Note: It is important that the module name is plural ('Cases' vs. 'Case') and that the name of the dependency, "activity_count_dep" in this example, is unique.

Chaining Dependencies

You can also add as many actions as you need to the array. In the example below, we want to display our count value but prevent users from being able to edit the value. An example extension definition is shown below:

```
  <?php

$dependencies['Leads']['number_of_cases_dep'] = array(
    'hooks' => array("edit", "view"), //not including save so that the
 value isn't stored in the DB
    'trigger' => 'true', //Optional, the trigger for the dependency. D
efaults to 'true'.
    //'triggerFields' => array('status'), //unneeded for this example
as its not field triggered
    'onload' => true,
    'actions' => array(
        array(
            'name' => 'SetValue',
            'params' => array(
                'target' => 'activity_count_c',
                'value' => 'add(
                    count($notes),
                    count($calls),
                    count($emails),
                    count($meetings),
                    count($tasks)
                )'
            )
        ),
```

```
        array(
            'name' => 'ReadOnly',
            'params' => array(
                'target' => 'activity_count_c',
                'value' => 'true',  //Set to true instead of a formula
 because its always read-only
            ),
        )
    )
);
```

Once you have the file in place, you will need to navigate to Admin > Repairs > and run a Quick Repair and Rebuild.

Note: It is important that the module name is plural ('Cases' vs. 'Case') and that the name of the dependency, "number_of_cases_dep" in this example, is unique.

Dependencies in Field Definitions

Unlike several of the other dependencies, SetValue is built into Studio. So this dependency can be set as a custom vardef value or in the vardefs file of a custom module. If you wanted to add this dependency to an existing field then you can create a vardef extension such as ./custom/Extension/modules/<module>/Ext/Vardefs/. An example extension definition is shown below:

./custom/Extension/modules/Accounts/Ext/Vardefs/activity_count_c.php

```
  <?php

$dictionary['Lead']['fields']['activity_count_c']['options'] = 'numeri
c_range_search_dom';
$dictionary['Lead']['fields']['activity_count_2_c']['calculated'] = 't
rue';
$dictionary['Lead']['fields']['activity_count_2_c']['formula'] = 'add(
    count($calls),
    count($emails),
    count($meetings),
    count($notes),
    count($tasks)
)';
$dictionary['Lead']['fields']['activity_count_2_c']['enforced'] = 'tru
e';
$dictionary['Lead']['fields']['activity_count_2_c']['enable_range_sear
```

```
ch'] = '1';
```

Once you have the file in place, you will need to navigate to Admin > Repairs > and run a Quick Repair and Rebuild.

SetVisibility

Overview

The SugarLogic SetVisibility action, located in ./include/Expressions/Actions/VisibilityAction.php , is used to determine the visibility logic of a field based on a formula.

Implementation

While the dependency metadata for your module can be defined in ./modules/<module>/metadata/dependencydefs.php and ./custom/modules/<module>/metadata/dependencydef.php, it is recommended to use the extension framework when customizing stock modules to prevent third party plugins from conflicting with your customizations. The following section will demonstrate how to implement a read-only dependency.

SetVisibility Parameters

Parameter	Type	Description
target	String	The name of the field to target for visibility.
value	String	This parameter can accept a boolean formula or true and false values. Normally you would put a SugarLogic formula here to set the boolean.

For more information on the various parameters in the dependency definitions, please refer to the dependency actions documentation.

Examples

The follow sections outline the various ways this dependency can be implemented.

SetVisibility Dependency Extensions

For our example, we will create a dependency on the Accounts module that shows the phone_alternate field when the phone_office field has been populated. An example is shown below.

./custom/Extension/modules/<module>/Ext/Dependencies/custom_phone_alternate.php

```
<?php

$dependencies['Accounts']['phone_alternate_hide'] = array(
        'hooks' => array("edit"),
        'triggerFields' => array('phone_office'),
        'onload' => true,
        //Actions is a list of actions to fire when the trigger is tru
e
        'actions' => array(
            array(
                'name' => 'SetVisibility',
                'params' => array(
                    'target' => 'phone_alternate',
                    'value' => 'not(equal($phone_office,""))',
                ),
            ),
        ),
    );
```

Once you have the file in place, you will need to navigate to Admin > Repairs > and run a Quick Repair and Rebuild.

Note: It is important that the module name is plural ('Accounts' vs. 'Account') and that the name of the dependency, "phone_alternate_hide" in this example, is unique.

Visibility Dependencies in Field Definitions

Unlike several of the other dependencies, SetVisibility is built into Studio. So this dependency can be set as a custom vardef value or in the varefs file for a new module. If you wanted to add this dependency to an existing field then you could add a file to ./custom/Extension/modules/<module>/Ext/Vardefs/. An example is

shown below.

To accomplish this, we will create an extension in ./custom/Extension/modules/Accounts/Ext/Vardefs/.

./custom/Extension/modules/Accounts/Ext/Vardefs/phone_alternate.php

```
<?php

$dictionary['Account']['fields']['phone_alternate']['dependency']='not
(equal($phone_office,""))
```

Next, you will need to navigate to Admin > Repairs > and run a Quick Repair and Rebuild. Once that is done, you can enter a value into phone_office and the phone_alternate field will show up once you tab out of the phone_office field. If you were coding a custom module with new fields, then you would just include it in the modules vardefs.php file as shown below

```
<?php

$dictionary['myModule'] = array(
    ...
    'fields' => array(
        ...
        'phone_alternate' => array(
     'name' => 'phone_alternate',
     'vname' => 'LBL_PHONE_ALTERNATE',
     'type' => 'varchar',
     'len' => 10,
     'dependency'=> 'not(equal($phone_office,""))',
     'comment' => 'Other Phone Number',
     'merge_filter' => 'enabled',
        ),
        ...
    )
    ...
);
```

Extending Sugar Logic

Overview

How to write custom Sugar Logic functions.

Writing a Custom Formula Function

The most important feature of Sugar Logic is that it is simply and easily extendable. Both custom formula functions and custom actions can be added in an upgrade-safe manner to allow almost any custom logic to be added to Sugar. Custom functions will be stored in ./custom/include/Expressions/Expression/{Type}/{Function_Name}.php.

The first step in writing a custom function is to decide what category the function falls under. Take, for example, a function for calculating the factorial of a number. In this case, we will be returning a number so we will create a file in ./custom/include/Expressions/Expression/Numeric/ called FactorialExpression.php. In the new PHP file we just created, we will define a class called FactorialExpression that will extend NumericExpression. All formula functions must follow the format {functionName}Expression.php and the class name must match the file name.

Next, we need to decide what parameters the function will accept. In this case, we need take in a single parameter, the number to return the factorial of. Since this class will be a sub-class of NumericExpression, it by default accepts only numeric types and we do not need to worry about specifying the type requirement.

Next, we must define the logic behind evaluating this expression. So we must override the abstract evaluate() function. The parameters can be accessed by calling an internal function getParameters() which returns the parameters passed into this object. So, with all this information, we can go ahead and write the code for the function.

Note: For the custom function to appear in Studio after completing these steps, you must compile your code by running the Rebuild Sugar Logic Functions job in Admin > Repair and then clear your browser cache.

```
  <?php

require_once 'include/Expressions/Expression/Numeric/NumericExpression
.php';

class FactorialExpression extends NumericExpression
{
    function evaluate()
    {
        $params = $this->getParameters();
        // params is an Expression object, so evaluate it
```

```
        // to get its numerical value
        $number = $params->evaluate();
        // exception handling
        if ( ! is_int( $number ) )
        {
            throw new Exception("factorial: Only accepts integers");
        }

        if ( $number < 0 )
        {
            throw new Exception("factorial: The number must be positiv
e");
        }

        // special case 0! = 1
        if ( $number == 0 ) return 1;

        // calculate the factorial
        $factorial = 1;
        for ( $i = 2 ; $i <= $number ; $i ++ )
        {
            $factorial = $factorial * $i;
        }

        return $factorial;
    }

    // Define the javascript version of the function
    static function getJSEvaluate()
    {
        return <<<EOQ
        var params = this.getParameters();
        var number = params.evaluate();
        // reg-exp integer test
        if ( ! /^\d*$/.test(number) )
        throw "factorial: Only accepts integers";

        if ( number < 0 )
        throw "factorial: The number must be postitive";
        // special case, 0! = 1
        if ( number == 0 ) return 1;
        // compute factorial
        var factorial = 1;

        for ( var i = 2 ; i <= number ; i ++ )
        factorial = factorial * i;
```

```
        return factorial;
EOQ;
        }

        function getParameterCount()
        {
            return 1; // we only accept a single parameter
        }

        static function getOperationName()
        {
            return "factorial";
            // our function can be called by 'factorial'
        }
    }

?>
```

One of the key features of Sugar Logic is that the functions should be defined in both PHP and JavaScript and have the same functionality under both circumstances. As you can see above, the getJSEvaluate() method should return the JavaScript equivalent of your evaluate() method. The JavaScript code is compiled and assembled for you after you run the "Rebuild Sugar Logic Functions" script through Admin > Repair.

Writing a Custom Action

Using custom actions, you can easily create reusable custom logic or integrations that can include user-editable logic using the Sugar Formula Engine. Custom actions will be stored in ./custom/include/Expressions/Actions/{ActionName}.php. Actions files must end in Action.php and the class defined in the file must match the file name and extend the AbstractAction class. The basic functions that must be defined are fire, getDefinition, getActionName, getJavascriptClass, and getJavscriptFire. Unlike functions, there is no requirement that an action works exactly the same for both server and client side as this is not always sensible or feasible.

A simple action could be "WarningAction" that shows an alert warning the user that something may be wrong and logs a message to the ./sugarcrm.log file if triggered on the server side. It will take in a message as a formula so that the message can be customized at runtime. We would do this by creating a PHP file in ./custom/include/Expressions/Actions/WarningAction.php as shown below:

./custom/include/Expressions/Actions/WarningAction.php

```
  <?php

require_once("include/Expressions/Actions/AbstractAction.php");

class WarningAction extends AbstractAction    {

    protected $messageExp = "";

    /**
    * Returns the javascript class equavalent to this php class
    * @return string javascript.
    */
    static function getJavascriptClass()
    {
        return <<<EOQ

        SUGAR.forms.WarningAction = function(message)
        {
            this.messageExp = message;
        };

        //Use the sugar extend function to extend AbstractAction
        SUGAR.util.extend(SUGAR.forms.WarningAction, SUGAR.forms.Abstr
actAction,
        {
            //javascript exection code
            exec : function()
            {
            //assume the message is a formula
            var msg = SUGAR.forms.evalVariableExpression(this.messageE
xp);
            alert(msg.evaluate());
            }
        });
    EOQ;
    }

    /**
    * Returns the javascript code to generate this actions equivalent.
    * @return string javascript.
    */

    function getJavascriptFire()
    {
```

```
        return "new SUGAR.forms.WarningAction('{$this->messageExp}')";
    }

    /**
    * Applies the Action to the target.
    * @param SugarBean $target
    */
    function fire(&$target)
    {
        //Parse the message formula and log it to fatal.
        $expr = Parser::replaceVariables($this->messageExp, $target);
        $result = Parser::evaluate($expr)->evaluate();
        $GLOBALS['log']->warn($result);
    }

    /**
    * Returns the definition of this action in array format.
    */
    function getDefinition()
    {
        return array("message" => $this->messageExp);
    }

    /**
    * Returns the short name used when defining dependencies that use
this action.
    */
    static function getActionName()
    {
        return "Warn";
    }
}

?>
```

Using Sugar Logic Directly

How to use Sugar Logic

Accessing an External API with a Sugar Logic Action

Let us say we were building a new Action called "SetZipCodeAction" that uses the Yahoo geocoding API to get the zip code for a given street + city + state address.

Since the Yahoo geocoding API requires JSON requests and returns XML data, we will have to write both PHP and JavaScript code to make and interpret the requests. Because accessing external APIs in a browser is considered cross-site scripting, a local proxy will have to be used. We will also allow the street, city, state parameters to be passed in as formulas so the action could be used in more complex Dependencies.

First, we should add a new action that acts as the proxy for retrieving data from the Yahoo API. The easiest place to add that would be a custom action in the "Home" module. The file that will act as the proxy will be .custom/modules/Home/geocode.php. It will take in the parameters via a REST call, make the call to the Yahoo API, and return the result in JSON format.

geocode.php contents:

```
  <?php

function getZipCode($street, $city, $state)
{
    $appID = "6ruuUKjV34Fydi4TE.ca.I02rWh.9LTMPqQnSQo4QsCnjF5wIvyYRSXP
IzqlDbI.jfE-";
    $street = urlencode($street);
    $city = urlencode($city);
    $state = urlencode($state);
    $base_url = "http://local.yahooapis.com/MapsService/V1/geocode?";
    $params = "appid={$appID}&street={$street}&city={$city}&state={$st
ate}";

    //use file_get_contents to easily make the request
    $response = file_get_contents($base_url . $params);

    //The PHP XML parser is going to be overkill in this case, so just
 pull the zipcode with a regex.
    preg_match('/\([\d-]*)\/', $response, $matches);

    return $matches[1];
}


if (!empty($_REQUEST['execute']))
{
 if (empty($_REQUEST['street']) || empty($_REQUEST['city']) || empty($
_REQUEST['state']))
```

```
 {
  echo("Bad Request");
 }
 else
 {
  echo json_encode(array('zip' => getZipCode($_REQUEST['street'], $_RE
QUEST['city'], $_REQUEST['state'])));
 }
}

?>
```

Next, we will need to map the geocode action to the geocode.php file. This is done by adding an action map to the Home Module. Create the file ./custom/modules/Home/action_file_map.php and add the following line of code:

```
  <?php

$action_file_map['geocode'] = 'custom/modules/Home/geocode.php';
```

We are now ready to write our Action. Start by creating the file ./custom/include/Expressions/Actions/SetZipCodeAction.php. This file will use the proxy function directly from the PHP side and make an asynchronous call on the javascript side to the proxy.

SetZipCodeAction.php contents:

```
  <?php

require_once "include/Expressions/Actions/AbstractAction.php";

class SetZipCodeAction extends AbstractAction
{
    protected $target = "";
    protected $streetExp = "";
    protected $cityExp = "";
    protected $stateExp = "";

    function SetZipCodeAction($params)
    {
        $this->target = empty($params['target']) ? " " : $params['targ
et'];
```

```
        $this->streetExp = empty($params['street']) ? " " : $params['s
treet'];
        $this->cityExp = empty($params['city']) ? " " : $params['city'
];
        $this->stateExp = empty($params['state']) ? " " : $params['sta
te'];
    }

    static function getJavascriptClass()
    {
        return "'($this->target}', '{$this->streetExp}', '{$this->city
Exp}', '{$this->stateExp}')";
    }

    function fire(&$bean)
    {
        require_once("custom/modules/Home/geocode.php");
        $vars = array(
            'street' => 'streetExp',
            'city' => 'cityExp',
            'state' => 'stateExp'
        );

        foreach($vars as $var => $exp)
        {
            $toEval = Parser::replaceVariables($this->$exp, $bean);
            $var = Parser::evaluate($toEval)->evaluate();
        }

        $target = $this->target;
        $bean->$target = getZipCode($street, $city, $state);
    }

    function getDefinition()
    {
        return array(
            "action" => $this->getActionName(),
            "target" => $this->target,
        );
    }

    static function getActionName()
    {
        return "SetZipCode";
    }
```

```
}

?>
```

Once you have the action written, you need to call it somewhere in the code. Currently, this must be done as shown above using custom views, logic hooks, or custom modules.

Creating a Custom Dependency for a View

Dependencies can also be created and executed outside of the built in features. For example, if you wanted to have the description field of the Calls module become required when the subject contains a specific value, you could extend the calls edit view to include that dependency.

./custom/modules/Calls/views/view.edit.php

```
  <?php

require_once "include/MVC/View/views/view.edit.php";
require_once "include/Expressions/Dependency.php";
require_once "include/Expressions/Trigger.php";
require_once "include/Expressions/Expression/Parser/Parser.php";
require_once "include/Expressions/Actions/ActionFactory.php";

class CallsViewEdit extends ViewEdit
{

    function CallsViewEdit()
    {
        parent::ViewEdit();
    }

    function display()
    {
        parent::display();
        $dep = new Dependency("description_required_dep");
        $triggerExp = 'contains($name, "important")';
        //will be array('name')

        $triggerFields = Parser::getFieldsFromExpression($triggerExp);
        $dep->setTrigger(new Trigger($triggerExp, $triggerFields));
```

```
        //Set the description field to be required if "important" is i
n the call subject
        $dep->addAction(ActionFactory::getNewAction('SetRequired', arr
ay(
            'target' => 'description',
            'label' => 'description_label',
            'value' => 'true'
        )));

        //Set the description field to NOT be required if "important"
is NOT in the call subject
        $dep->addFalseAction(ActionFactory::getNewAction('SetRequired'
, array(
            'target' => 'description',
            'label' => 'description_label',
            'value' => 'false'
        )));

        //Evaluate the trigger immediatly when the page loads
        $dep->setFireOnLoad(true);
        $javascript = $dep->getJavascript();
        echo

        SUGAR.forms.AssignmentHandler.registerView('EditView');

        {$javascript}


EOQ;
    }
}

?>
```

The above code creates a new Dependency object with a trigger based on the 'name' (Subject) field in of the Calls module. It then adds two actions. The first will set the description field to be required when the trigger formula evaluates to true (when the subject contains "important"). The second will fire when the trigger is false and removes the required property on the description field. Finally, the javascript version of the Dependency is generated and echoed onto the page.

Creating a Custom Dependency Using Metadata

The files should be located in ./custom/Extension/modules/{module}/Ext/Dependencies/{dependency name}.php and be rebuilt with a quick repair after modification.

Dependencies can have the following properties:

- **hooks** : Defines when the dependency should be evaluated. Possible values are edit (on edit/quickCreate views), view (Detail/Wireless views), save (during a save action), and all (any time the record is accessed/saved).
- **trigger** : A boolean formula that defines if the actions should be fired. (optional, defaults to 'true')
- **triggerFields** : An array of field names that when when modified should trigger re-evaluation of this dependency on edit views. (optional, defaults to the set of fields found in the trigger formula)
- **onload** : If true, the dependency will be fired when an edit view loads (optional, defaults to true)
- **actions** : An array of action definitions to fire when the trigger formula is true.
- **notActions** : An array of action definitions to fire when the trigger formula is false. (optional)

The actions are defined as an array with the name of the action and a set of parameters to pass to that action. Each action takes different parameters, so you will have to check each actions class constructor to check what parameters it expects.

The following example dependency will set the resolution field of cases to be required when the status is Closed:

```
  <?php

$dependencies['Cases']['required_resolution_dep'] = array(
    'hooks' => array("edit"),
    //Optional, the trigger for the dependency. Defaults to 'true'.
    'trigger' => 'true',
    'triggerFields' => array('status'),
    'onload' => true,
    //Actions is a list of actions to fire when the trigger is true
    'actions' => array(
        array(
            'name' => 'SetRequired',
            //The parameters passed in will depend on the action type
set in 'name'
```

```
            'params' => array(
                'target' => 'resolution',
                //id of the label to add the required symbol to
                'label' => 'resolution_label',
                //Set required if the status is closed
                'value' => 'equal($status, "Closed")'
            )
        ),
    ),
    //Actions fire if the trigger is false. Optional.
    'notActions' => array(),
);
```

Using Dependencies in Logic Hooks

Overview

Dependencies can not only be executed on the server side but can be useful entirely on the server. For example, you could have a dependency that sets a rating based on a formula defined in a language file.

```
  <?php

require_once "include/Expressions/Dependency.php";
require_once "include/Expressions/Trigger.php";
require_once "include/Expressions/Expression/Parser/Parser.php";
require_once "include/Expressions/Actions/ActionFactory.php";

class Update_Account_Hook
{
    function updateAccount($bean, $event, $args)
    {
        $formula = translate('RATING_FORMULA', 'Accounts');
        $triggerFields = Parser::getFieldsFromExpression($formula);
        $dep = new Dependency('updateRating');
        $dep->setTrigger(new Trigger('true', $triggerFields));
        $dep->addAction(ActionFactory::getNewAction('SetValue', array(
            'target' => 'rating',
            'value' => $formula
        )));
```

```
        $dep->fire($bean);
    }
}
```

Administration

Overview

The Administration class is used to manage settings stored in the database config table.

The Administration Class

The Administration class is located in ./modules/Administration/Administration.php. Settings modified using this class are written to the config table.

Creating / Updating Settings

To create or update a specific setting, you can specify the new value using the saveSetting() function as shown here:

```
  require_once 'modules/Administration/Administration.php';

$administrationObj = new Administration();

//save the setting
$administrationObj->saveSetting("MyCategory", "MySetting", 'MySettings
Value');
```

Retrieving Settings

You can access the config settings by using the retrieveSettings() function. You can filter the settings by category by passing in a filter parameter. If no value is passed to retrieveSettings(), all settings will be returned. An example is shown here:

```
  require_once 'modules/Administration/Administration.php';
```

```
$administrationObj = new Administration();

//Retrieve all settings in the category of 'MyCategory'.
//This parameter can be left empty to retrieve all settings.
$administrationObj->retrieveSettings('MyCategory');

//Use a specific setting
$MySetting = $administrationObj->settings['MyCategory_MySetting'];
```

Considerations

The Administration class will store the settings in the config table, and the config table is cached using SugarCache as the admin_settings_cache. This class should be used for storing dynamic settings for a module, connector or the system as a whole, such as the last run date (for a scheduler) or an expiring token (for a connector). You should not use the Administration class to store User-based settings, settings that are frequently changing or being written to via the API, or excessively large values, as this can degrade the performance of the instance since all settings are loaded into the admin_settings_cache, which is used throughout the system.

Alternatively, you can use the Configurator class, located in ./modules/Configurator/Configurator.php, to store the settings in the ./config_override.php file if the settings are not dynamic or not changing programmatically. It is important to note that settings stored using the configurator will cause a metadata hash refresh that may lead to users being logged out of the system.

Configurator

Overview

The Configurator class, located in ./modules/Configurator/Configurator.php, handles the config settings found in ./config.php and ./config_override.php.

Retrieving Settings

You can access the Sugar config settings by using the global variable $GLOBALS['sugar_config'] as shown below:

```
  global $sugar_config;
```

```
//Use a specific setting
$MySetting = $sugar_config['MySetting'];
```

If you should need to reload the config settings, this is an example of how to retrieve a specific setting using the configurator:

```
  require_once 'modules/Configurator/Configurator.php';

$configuratorObj = new Configurator();

//Load config
$configuratorObj->loadConfig();

//Use a specific setting
$MySetting = $configuratorObj->config['MySetting'];
```

Creating / Updating Settings

To create or update a specific setting, you can specify the new value using the configurator as shown below:

```
  require_once 'modules/Configurator/Configurator.php';

$configuratorObj = new Configurator();

//Load config
$configuratorObj->loadConfig();

//Update a specific setting
$configuratorObj->config['MySetting'] = "MySettingsValue";

//Save the new setting
$configuratorObj->saveConfig();
```

Considerations

When looking to store custom settings, the Configurator class will store the settings in the ./config_override.php file. This class should only be used for long-term configuration options as Configurator->saveConfig() will trigger a metadata

refresh that may log users out of the system.

Alternatively, you can use the Administration class, located in ./modules/Administration/Administration.php, to store settings in the config table in the database. This Administration class should be used for storing dynamic settings such as a last run date or an expiring token.

Core Settings

activitystreamcleaner

Description	Array that defines the parameters for the Activity Stream purger.
Type	Array
Versions	9.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['activitystreamcleaner'] = array();`

activitystreamcleaner.keep_all_relationships_activities

Description	This value is used by the ActivityStreamPurger scheduler job to determine whether the link type activities are to be removed. By default, these records are removed along with other activity types such as create, update, etc. This can be overridden by setting this value equal to true.
Type	Boolean
Range of values	true or false
Versions	9.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['activitystreamcleaner']['keep_all_relationships_activ`

	`ities'] = true;`

activitystreamcleaner.limit_scheduler_run

Description	Sets the number of activity stream records to delete per batch when run by the scheduled job that purges activity stream records.
Type	Integer
Range of values	Integers, values below 0 become 0
Versions	9.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	25000
Override Example	`$sugar_config['activitystreamcleaner']['limit_scheduler_run'] = 1000;`

activitystreamcleaner.months_to_keep

Description	This value is used by the ActivityStreamPurger scheduler job. When this job runs, it uses this value to determine which Activity Stream records to prune from the activities table. Activities with a date older than the current date minus this number of months will be removed from the table.
Type	Integer
Range of values	Any integer greater than or equal to 0
Versions	9.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	6
Override Example	`$sugar_config['activitystreamcleaner']['months_to_keep'] = 12;`

activity_streams

Description	Array that defines parameters for processing Data Privacy erasure requests for activity streams.
Type	Array
Versions	8.0.1+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['activity_streams'] = array();`

activity_streams.erasure_job_delay

Description	Defines the number of minutes between Activity Stream Erasure jobs. When multiple jobs are queued, this is the number of minutes that will separate their scheduled execution to allow Data Privacy erasures to occur in batches and minimize concurrent execution. These jobs can take some time to run and can tax the overall server performance if the number of Activity Stream records is very large. This setting is related to activity_streams.erasure_job_limit. These settings should be considered together to optimize throughput of Activity Erasure jobs.
Type	Integer
Versions	8.0.1+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['activity_streams']['erasure_job_delay'] = 5;`

activity_streams.erasure_job_limit

Description	Defines the maximum number of Data Privacy Records that can be processed by a single running instance of the Activity Stream Erasure job. This setting is related to activity_streams.erasure_job_delay.

	These settings should be considered together to optimize throughput of Activity Erasure jobs. For example, the longer that an activity erasure job processes, the more delay will be needed to prevent too many jobs executing in parallel.
Type	Integer
Versions	8.0.1+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	5
Override Example	`$sugar_config['activity_streams'][ 'erasure_job_limit'] = 8;`

additional_js_config

Description	Configuration values for when the ./cache/config.js file is generated. It is important to note that after changing this setting or any of its subsettings in your configuration, you must navigate to Admin > Repairs > Quick Repair & Rebuild.
Type	Array
Versions	7.5.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['additional_js_confi g'] = array();`

additional_js_config.alertAutoCloseDelay

Description	Defines the default auto close delay for system alerts. It is important to note that after changing this setting in your configuration, you must navigate to Admin > Repairs > Quick Repair & Rebuild.
Type	Integer

Versions	7.8.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	9000
Override Example	`$sugar_config['additional_js_config']['alertAutoCloseDelay'] = 3000;`

additional_js_config.authStore

Description	Defines the implementation of authentication data storage. The default storage, 'cache', is persistent and uses the localStorage API. Alternatively, 'cookie' storage may be used. This will store the authentication data in your browser window until the browser itself is closed. It is important to note that this behavior will differ between browsers. It is important to note that after changing this setting in your configuration, you must navigate to Admin > Repairs > Quick Repair & Rebuild. **Note**: This setting is not respected for instances that use SugarIdentity.
Type	String
Range of values	'cache' and 'cookie'
Versions	7.5.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	cache
Override Example	`$sugar_config['additional_js_config']['authStore'] = 'cookie';`

additional_js_config.disableOmnibarTypeahead

Description	Disables the typeahead feature in the listview Omnibar and force users to hit Enter when filtering. From a technical perspective, this will reduce the number

	of hits to the server.
Type	Boolean
Range of values	true and false
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['additional_js_config']['disableOmnibarTypeahead'] = true;`

additional_js_config.logger.write_to_server

Description	Enables the front end messages to be logged. The logger level must be tuned accordingly under Administration settings. Developers can set the client-side flag by running App.config.logger.writeToServer = true; in their browser's console. To simulate logging actions through the console, developers can use: App.logger.trace('message'); App.logger.debug('message'); App.logger.info('message'); App.logger.warn('message'); App.logger.fatal('message');
Type	Boolean
Range of values	true and false
Versions	7.5.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['additional_js_config']['logger']['write_to_server'] = true;`

additional_js_config.sidecarCompatMode

Description	By default, the Sidecar framework will prevent customizations from calling private Sidecar methods. If after upgrading you find this to be an issue, the configuration can be set to true as a temporary workaround. All JavaScript customizations are expected to use public Sidecar APIs.
Type	Boolean
Range of values	true and false
Versions	7.10.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['additional_js_config']['sidecarCompatMode'] = true;`

additional_js_config.skipTutorial

Description	Disables the tutorial system shown when users log in for the first time.
Type	Boolean
Range of values	true and false
Versions	7.0.0+
Products	Professional, Enterprise, Ultimate
Default Value	false
Override Example	`$sugar_config['additional_js_config']['skipTutorial'] = true;`

admin_access_control

Description	Removes Sugar Updates, Upgrade Wizard, Backups and Module Builder from the Admin menu. For developers, these restrictions can be found in ./include/MVC/Controller/file_access_control_map.php and overriden by creating ./custom/include/MVC/Controller/file_access_control_map.php.

Type	Boolean
Range of values	true and false
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['admin_access_control'] = true;`

admin_export_only

Description	Allow only users with administrative privileges to export data.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['admin_export_only'] = true;`

allowFreezeFirstColumn

Description	Global admin config that turns the frozen first columns on/off. This setting can be toggled on and off via Admin > System Settings in Sugar or via code in the config.php file. When this setting is turned off, the ability to freeze the first column of data on a list view is disabled. If the setting is enabled, users will have an option to either freeze or unfreeze the first column on a per-module basis, similar to how you can toggle whether or not a field is included as a column in your list view.
Type	Boolean
Versions	12.0.0+

Products	Enterprise, Serve, Sell
Default Value	true
Override Example	`$sugar_config['allowFreezeFirstColumn'] = false;`

allow_oauth_via_get

Description	As of 7.8, Sugar does not support auth tokens being passed in from GET query string parameters by default. While allowing this functionality is not a recommended practice from a security standpoint, administrators may enable the setting at their own risk.
Type	Boolean
Range of values	true and false
Versions	7.8.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['allow_oauth_via_get'] = true;`

allow_pop_inbound

Description	Inbound email accounts are setup to work with IMAP protocols by default. If your email provider required POP3 access instead of IMAP, you can enables POP3 as an available inbound email protocol. Please note that mailboxes configured with a POP3 connection are not supported by SugarCRM and may cause unintended consequences. IMAP is the recommended protocol to use for inbound email accounts.
Type	Boolean
Range of values	true and false
Versions	5.5.1+

Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['allow_pop_inbound'] = true;`

allow_sendmail_outbound

Description	Enables the option of choosing sendmail as an SMTP server in Admin > Email Settings. Sendmail must be enabled on the server for this option to work. Please note that mailboxes configured with sendmail are not supported and may cause unintended consequences.
Type	Boolean
Range of values	true and false
Versions	5.5.1+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['allow_sendmail_outbound'] = true;`

analytics

Description	An array defining properties for an analytics connector.
Type	Array
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['analytics'] = array();`

analytics.connector

Description	The name of the connector to use for gathering analytics.
Type	String

Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['analytics']['connector'] = 'Pendo';`

analytics.enabled

Description	Determines if the analytics are enabled.
Type	Boolean
Range of values	true and false
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['analytics']['enabled'] = true;`

analytics.id

Description	The tracking id for the analytics connector.
Type	String
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['analytics']['id'] = 'UA-XXXXXXX-X';`

api

Description	API specific configurations.
Type	Integer
Versions	7.5.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['api']=array();`

api.timeout

Description	The timeout in seconds when uploading files through the REST API.
Type	Integer
Versions	7.5.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	180
Override Example	`$sugar_config['api']['timeout'] = 240;`

authenticationClass

Description	The class to be used for login authentication.
Type	String
Range of values	SAMLAuthenticate
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	SugarAuthenticate
Override Example	`$sugar_config['authenticationClass'] = 'SAMLAuthenticate';`

aws

Description	This configuration's functionality has been deprecated and will be removed in an upcoming release.
Type	Array
Versions	6.7.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['aws'] = array();`

aws.aws_key

Description	This configuration's functionality has been deprecated and will be removed in an upcoming release.
Type	String
Versions	6.7.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['aws']['aws_key'] = 'key';`

aws.aws_secret

Description	This configuration's functionality has been deprecated and will be removed in an upcoming release.
Type	String
Versions	6.7.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['aws']['aws_secret'] = 'secret';`

aws.upload_bucket

Description	This configuration's functionality has been deprecated and will be removed in an upcoming release.
Type	String
Versions	6.7.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['aws']['upload_bucket'] = 'bucket';`

cache

Description	Array that defines additional properties

	for SugarCache if it's enabled (see external_cache).
Type	Array
Versions	8.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['cache'] = array();`

cache.backend

Description	Defines the SugarCache backend to use via a fully qualified class name (FQCN).
Type	String
Range of values	'Sugarcrm\Sugarcrm\Cache\Backend\Redis' , '\Sugarcrm\Sugarcrm\Cache\Backend\APCu' , '\Sugarcrm\Sugarcrm\Cache\Backend\Memcached' , '\Sugarcrm\Sugarcrm\Cache\Backend\InMemory' , ''
Versions	8.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	Sugarcrm\Sugarcrm\Cache\Backend\BackwardCompatible
Override Example	`$sugar_config['cache']['backend'] = 'Sugarcrm\Sugarcrm\Cache\Backend\Redis';`

cache.encryption_key

Description	Used to store encryption key for SugarCache is in multi-tenant mode. By default, Sugar will generate a random UUID string for the encryption key as needed. It is not recommended to change or override this value since it will be regenerated whenever the cache is cleared. Not applicable without cache.multi_tenant.
Type	String
Range of values	N/A as it is generated automatically

Versions	8.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	Random value (generated on cache initialization)
Override Example	`$sugar_config['cache']['encryption_key'] = 'Sugar-generated encryption key';`

cache.multi_tenant

Description	Defines whether the SugarCache backend will be used by multiple Sugar instances as part of a multi-tenant deployment. This will modify Sugar's caching behavior to maintain isolation between Sugar instances.
Type	Boolean
Range of values	true and false
Versions	8.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['cache']['multi_tenant'] = true;`

cache_dir

Description	This is the directory SugarCRM will store all cached files. Can be relative to Sugar root directory.
Type	String
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	cache/
Override Example	`$sugar_config['cache_dir'] = 'cache/';`

cache_expire_timeout

Description	The length of time cached items should be expired after.
Type	Integer
Versions	6.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	300
Override Example	`$sugar_config['cache_expire_timeout'] = 400;`

calendar

Description	An array that defines all of the various settings for the Calendar module.
Type	Array
Versions	6.4.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['calendar'] = array();`

calendar.day_timestep

Description	Sets the default day time step.
Type	Integer
Range of values	15, 30 and 60
Versions	6.4.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	15
Override Example	`$sugar_config['calendar']['day_timestep'] = 15;`

calendar.default_view

Description	Changes the default view in the

	calendar module.
Type	String
Range of values	'day', 'week', 'month' and 'share'
Versions	6.4.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['calendar']['default_view'] = 'week';`

calendar.items_draggable

Description	Enable/Disable drag-and-drop feature to move calendar items.
Type	Boolean
Range of values	true and false
Versions	6.4.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	true
Override Example	`$sugar_config['calendar']['items_draggable'] = true;`

calendar.items_resizable

Description	Sets whether items on the calendar can be resized via clicking and dragging.
Type	Boolean
Range of values	true and false
Versions	6.5.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	true
Override Example	`$sugar_config['calendar']['items_resizable'] = true;`

calendar.show_calls_by_default

Description	Display/Hide calls by default.

Type	Boolean
Range of values	true and false
Versions	6.4.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	true
Override Example	`$sugar_config['calendar']['show_calls_by_default'] = true;`

calendar.week_timestep

Description	The default week step size when viewing the calendar.
Type	Integer
Range of values	15, 30, and 60
Versions	6.4.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['calendar']['week_timestep'] = 30;`

check_query

Description	Validates queries when adding limits.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	true
Override Example	`$sugar_config['check_query'] = true;`

check_query_cost

Description	Sets the maximum cost limit of a query.
Type	Integer

Versions	5.2.0+
Products	Enterprise, Ultimate, Serve, Sell
Default Value	10
Override Example	`$sugar_config['check_query_cost'] = 10;`

clear_resolved_date

Description	In Sugar Serve version 9.3 and higher, the Resolved Date field on Cases is automatically cleared when the case's status changes from "Closed", "Rejected", or "Duplicate" to any other value. Setting this parameter to false prevents Sugar from automatically clearing the Resolved Date field.
Type	Boolean
Range of values	true and false
Versions	9.3.0+
Products	Serve
Default Value	true
Override Example	`$sugar_config['clear_resolved_date'] = false;`

cloud_insight

Description	Array that defines the properties for SugarCloud Insights.
Type	Array
Versions	9.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['cloud_insight'] = array();`

cloud_insight.enabled

Description	Determines if the SugarCloud Insights

	service is enabled.
Type	Boolean
Range of values	true and false
Versions	9.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	true
Override Example	`$sugar_config['cloud_insight']['enabled'] = false;`

cloud_insight.key

Description	Specifies the unique key for the SugarCloud Insights service.
Type	String
Versions	9.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['cloud_insight']['key'] = '';`

cloud_insight.url

Description	The current URL for SugarCloud Insights service.
Type	String
Versions	9.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['cloud_insight']['url'] = 'https://sugarcloud-insights.service.sugarcrm.com';`

collapse_subpanels

Description	Pertains to Sidecar modules only. By default, all subpanels are in a collapsed state. If a user expands a subpanel,

	Sugar caches this preference so that it remains expanded on future visits to the same view. Set this value to 'true' to force a collapsed state for subpanels regardless of user preference, which may improve page-load performance by not querying for data until a user explicitly chooses to expand a subpanel.
Type	Boolean
Range of values	true and false
Versions	7.6.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	true
Override Example	`$sugar_config['collapse_subpanels'] = true;`

cron

Description	Array that defines all of the cron parameters.
Type	Array
Versions	6.5.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['cron'] = array();`

cron.enforce_runtime

Description	Determines if cron.max_cron_runtime is enforced during the cron run.
Type	Boolean
Range of values	true and false
Versions	7.2.2.2+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['cron']['enforce_run`

	`time'] = true;`

cron.max_cron_jobs

Description	Maximum jobs per cron run. Default is 10. If you are using a version prior to 6.5.14, you will need to also populate max_jobs to set this value due to bug #62936 (https://web.sugarcrm.com/support/issues/62936).
Type	Integer
Versions	6.5.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	6.5.x: 10 7.6.x: 25
Override Example	`$sugar_config['cron']['max_cron_jobs'] = 10;`

cron.max_cron_runtime

Description	Determines the maximum time in seconds that a single job should be allowed to run. If a single job exceeds this limit, cron.php is aborted with the long-running job marked as in progress in the job queue. The next time cron runs, it will skip the job that overran the limit and start on the next job in the queue. This limit is only enforced when cron.enforce_runtime is set to true.
Type	Integer
Versions	6.5.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	180
Override Example	`$sugar_config['cron']['max_cron_runtime'] = 60;`

cron.min_cron_interval

Description	Minimum time between cron runs. Setting this to 0 will disable throttling completely.
Type	Integer
Versions	6.5.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	30
Override Example	`$sugar_config['cron']['min_cron_interval'] = 30;`

csrf

Description	An array defining attributes for CSRF token protection.
Type	Array
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['csrf'] = array();`

csrf.opt_in

Description	CSRF tokens are injected in BWC modules on forms which allow "modify" actions as an additional protection layer against CSRF attacks. This functionality complements SugarCRM's referrer validation for "modify" actions. When forms are posted which trigger a "modify" action without having the proper CSRF token set, the end user will be served a CSRF token mismatch message and requested action will be denied. Additionally, a fatal warning message will be logged ("CSRF: attack vector detected, invalid form token detected"). When using this feature make sure any custom BWC module

	implement the proper CSRF token injection. CSRF token protection is an opt-in feature of Sugar 7.7 and removed in 7.8+.
Type	Boolean
Range of values	true and false
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate
Default Value	false
Override Example	`$sugar_config['csrf']['opt_in'] = false;`

csrf.soft_fail_form

Description	When opted in for CSRF form authentication, any failures will result in a CSRF message to the user indicating a potential CSRF attack and an aborted action. If you are unsure whether the CSRF tokens are properly implemented in your backward compatible modules, this configuration parameter can be set to true. Setting this to true will avoid any exceptions being thrown to the end user. By default, any failures will result in a fatal log message indicating the issue. If you are running in "soft failure", a second fatal message will be logged such as "CSRF: attack vector NOT mitigated, soft failure mode enabled". Be careful enabling this configuration as it will only log CSRF authentication failures and will not protect your system.
Type	Boolean
Range of values	true and false
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['csrf']['soft_fail_f`

	`orm'] = true;`

csrf.token_size

Description	The size in bytes of the CSRF token to generate.
Type	Integer
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	32
Override Example	`$sugar_config['csrf']['token_size'] = 16;`

customPortalPlatforms

Description	This configuration allows a platform with custom authentication to be excluded from an IDM integration.
Type	Array
Versions	9.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['customPortalPlatforms'] = array('platform1', 'platform2');`

custom_help_base_url

Description	Allows an instance to specify a custom help url for their user
Type	String
Versions	6.4.3+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	http://www.sugarcrm.com/crm/product_doc.php
Override Example	`$sugar_config['custom_help_base_ur`

	`l'] = 'http://www.custom_url/index.php';`

custom_help_url

Description	Designate the URL used to redirect the user to help documentation.
Type	String
Versions	6.4.3+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	http://www.sugarcrm.com/crm/product_doc.php
Override Example	`$sugar_config['custom_help_url'] = 'http://www.sugarcrm.com/crm/product_doc.php';`

dbconfig

Description	Defines all of the connection parameters for the database server.
Type	Array
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['dbconfig'] = array();`

dbconfig.db_host_instance

Description	Defines the host instance for MSSQL connections.
Type	String
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	empty
Override Example	`$sugar_config['dbconfig']['db_host`

	`_instance'] = 'SQLEXPRESS';`

dbconfig.db_host_name

Description	Part of the 'dbconfig' array. Defines the host name of the database server.
Type	String
Range of values	host name of database server
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	empty
Override Example	`$sugar_config['dbconfig']['db_host_name'] = 'localhost';`

dbconfig.db_manager

Description	Part of the 'dbconfig' array. Defines the specific library used to connect with your database.
Type	String
Range of values	The class name of the database driver, Possible values are: 'MysqlManager', 'MysqliManager', 'FreeTDSManager', 'MssqlManager', and 'SqlsrvManager'.
Versions	6.4.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	Determined by install: 'MysqlManager', 'MysqliManager', 'FreeTDSManager', 'MssqlManager', or 'SqlsrvManager'
Override Example	`$sugar_config['dbconfig']['db_manager'] = 'MysqliManager';`

dbconfig.db_name

Description	Part of the 'dbconfig' array. Defines the database name to connect to on the database server.

Type	String
Range of values	database name
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	empty
Override Example	`$sugar_config['dbconfig']['db_name'] = 'sugar_db';`

dbconfig.db_password

Description	Part of the 'dbconfig' array. Defines the password that correlates to the db_user_name parameter.
Type	String
Range of values	password of the user defined in db_user_name
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	empty
Override Example	`$sugar_config['dbconfig']['db_password'] = 'sql_password';`

dbconfig.db_port

Description	Part of the 'dbconfig' array. Defines the port number on the server to connect to for authentication and transactions.
Type	String
Range of values	port number to connect to on the database server
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	3306
Override Example	`$sugar_config['dbconfig']['db_port'] = '3306';`

dbconfig.db_type

Description	Defines the type of database being used with Sugar. It is important to note that db2 and oracle are only applicable to Sugar Enterprise and Ultimate.
Type	String
Range of values	'mysql', 'mssql', 'db2', and 'oci8'
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['dbconfig']['db_type'] = 'mysql';`

dbconfig.db_user_name

Description	Defines the user to connect to the Sugar database as.
Type	String
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['dbconfig']['db_user_name'] = 'sql_user';`

dbconfigoption.autofree

Description	Automatically frees the database reference when it closes the reference.
Type	Boolean
Range of values	true and false
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['dbconfigoption']['autofree'] = true;`

dbconfigoption.collation

Description	The set of rules to be used for comparing characters in a character set.
Type	String
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	utf8_general_ci
Override Example	`$sugar_config['dbconfigoption']['collation'] = 'utf8_general_ci';`

dbconfigoption.debug

Description	Enables debugging on the database connection.
Type	Boolean
Range of values	true and false
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['dbconfigoption']['debug'] = true;`

dbconfigoption.persistent

Description	Determines whether Sugar should use persistent connection when possible to connecting to the database.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['dbconfigoption']['p`

	`ersistent'] = true;`

dbconfigoption.ssl

Description	Enables SSL on the database connection.
Type	Boolean
Range of values	true and false
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['dbconfigoption']['ssl'] = true;`

dbconfigoption.ssl_options

Description	An array detailing the SSL database connection options.
Type	Array
Range of values	true and false
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	empty
Override Example	`$sugar_config['dbconfigoption']['ssl_options'] = array();`

dbconfigoption.ssl_options.ssl_capath

Description	The path the trusted SSL certificate authority file in PEM format for SSL connection to the database.
Type	String
Range of values	true and false
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell

Default Value	empty
Override Example	`$sugar_config['dbconfigoption']['ssl_options']['ssl_capath'] = 'path/to/ca-cert';`

dbconfigoption.ssl_options.ssl_cert

Description	The path the SSL certificate file for SSL connection to the database.
Type	String
Range of values	true and false
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	empty
Override Example	`$sugar_config['dbconfigoption']['ssl_options']['ssl_cert'] = 'path/to/cert';`

dbconfigoption.ssl_options.ssl_cipher

Description	The SSL cipher to be used for encryption.
Type	String
Range of values	true and false
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	empty
Override Example	`$sugar_config['dbconfigoption']['ssl_options']['ssl_cipher'] = 'cipher';`

dbconfigoption.ssl_options.ssl_key

Description	The path the SSL key for SSL connection to the database.
Type	String
Range of values	true and false

Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	empty
Override Example	`$sugar_config['dbconfigoption']['ssl_options']['ssl_key'] = 'path/to/key';`

default_currency_significant_digits

Description	Changes the number of significant digits in currency by default.
Type	Integer
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	2
Override Example	`$sugar_config['default_currency_significant_digits'] = 2;`

default_date_format

Description	Modifies the default date format for all users.
Type	String
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	m/d/Y
Override Example	`$sugar_config['default_date_format'] = 'm/d/Y';`

default_decimal_seperator

Description	Sets the character used as a decimal separator for numbers.
Type	String

Range of values	Any character
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	.
Override Example	`$sugar_config['default_decimal_separator'] = '.';`

default_email_client

Description	Sets the default email client that opens when users send emails.
Type	String
Range of values	'sugar', 'external'
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	sugar
Override Example	`$sugar_config['default_email_client'] = 'sugar';`

default_language

Description	Sets each user's default language. Possible values include any language offered by Sugar, such as: 'ar_SA', 'bg_BG', 'ca_ES', 'cs_CZ', 'da_DK', 'de_DE', 'el_EL', 'en_UK', 'en_us', 'es_ES', 'es_LA', 'et_EE', 'fi_FI', 'fr_FR', 'he_IL', 'hu_HU', 'it_it', 'ja_JP', 'ko_KR', 'lt_LT', 'lv_LV', 'nb_NO', 'nl_NL', 'pl_PL', 'pt_BR', 'pt_PT', 'ro_RO', 'ru_RU', 'sk_SK', 'sq_AL', 'sr_RS', 'sv_SE', 'tr_TR', 'uk_UA', 'zh_CN'
Type	String
Range of values	Any available language
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell

Default Value	en_us
Override Example	`$sugar_config['default_language'] = 'en_us';`

default_number_grouping_seperator

Description	Sets the character used as the 1000s separator for numbers.
Type	String
Range of values	Any character
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	,
Override Example	`$sugar_config['default_number_grouping_seperator'] = ',';`

default_permissions

Description	Array that defines the ownership and permissions for directories and files created naturally by the application.
Type	Array
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['default_permissions'] = array();`

default_permissions.dir_mode

Description	Part of the 'default_permissions' array. Used in UNIX-based systems only to define the permissions on newly created directories. The value is stored in decimal notation while UNIX file permissions are octal. For example, an octal value of 1528 equates to the permissions 2770.

Type	Integer
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['default_permissions']['dir_mode'] = 1528;`

default_permissions.file_mode

Description	Part of the 'default_permissions' array. Used in UNIX-based systems only to define the permissions on newly created files. The value is stored in decimal notation while UNIX file permissions are octal. For example, an octal value of 432 in equates to the permissions 660.
Type	Integer
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['default_permissions']['file_mode'] = 432;`

default_permissions.group

Description	Used in UNIX-based systems only to define the group membership of any newly created directories and files. This value should be a group that the Apache user is a member of to help ensure proper functionality.
Type	String
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['default_permissions']['group'] = 'apache';`

default_permissions.user

Description	Part of the 'default_permissions' array. Used in UNIX-based systems only to define the ownership of any newly created directories and files. This value should be the Apache user.
Type	String
Range of values	Apache user
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['default_permissions']['user'] = 'apache';`

default_user_is_admin

Description	Allows for determining whether a user is a system administrator by default.
Type	Boolean
Range of values	true, false
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['default_user_is_admin'] = true;`

deny_license_update

Description	Determines whether the license key in Admin > License Management is editable in the user interface.
Type	Boolean
Range of values	true and false
Versions	9.3.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['deny_license_update`

	`'] = true;`

developerMode

Description	Rebuilds various cached files when a page is accessed. Can be set by an admin in Admin > System Settings.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['developerMode'] = true;`

developer_mode_visible

Description	Determines whether the Developer Mode flag is visible to administrators in Admin > System Settings.
Type	Boolean
Range of values	true or false
Versions	11.3.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	true
Override Example	`$sugar_config['developer_mode_visible'] = false;`

diagnostic_file_max_lifetime

Description	The interval in seconds of when to expire and remove diagnostic files. It is important to note that the "Remove diagnostic files" scheduler job must be enabled to remove the diagnostic files.
Type	Integer

Versions	7.6.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	604800
Override Example	`$sugar_config['diagnostic_file_max_lifetime'] = 604800;`

disabled_languages

Description	Allows an admin to select languages that are disabled for the instance.
Type	String
Versions	6.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	empty
Override Example	`$sugar_config['disabled_languages'] = 'bg_BG,da_DK,de_DE';`

disable_count_query

Description	Removes the count totals from listviews. This is commonly used to prevent performing expensive count queries on the database when loading listviews and subpanels. It is important to note that in 7.x, this parameter will only affect modules running in Backward Compatibility mode.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['disable_count_query'] = true;`

disable_export

Description	Prevents exports of data into .csv files. Normally set in the UI via Admin > Locale.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['disable_export'] = true;`

disable_related_calc_fields

Description	When a calculated field in Sugar uses the related function in the Sugar Logic, this will cause the calculated field to be executed when the related module is updated. This can cause a cascading effect through the system to update related calculated fields. When this happens you may receive a 502 Gateway Error. Please note that this is a global setting that will affect all modules. If you have a calculated field in Accounts that sums up all Opportunities for the account, setting this value to true will no longer update the opportunity account sum in Accounts until the account record itself is modified. However, if this setting is left disabled, the sum would update any time a related opportunity or the account is modified.
Type	Boolean
Range of values	true and false
Versions	6.3.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell

Override Example	`$sugar_config['disable_related_calc_fields'] = true;`

disable_team_access_check

Description	Prevents the system from checking to see if the creating/editing user has access to the record being saved. In normal circumstances, if a user creates a record and assigns it to a team they are not part of - their private team will be added. Setting this to true will prevent this.
Type	Boolean
Range of values	true and false
Versions	5.5.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['disable_team_access_check'] = true;`

disable_unknown_platforms

Description	Controls whether or not unregistered platforms are allowed to be used when logging in using REST API. Custom platforms can be registered via Admin > Configure API Platforms or by using the Platform extension. Used to prevent excessive metadata generation when invalid or unrecognized platform types are specified in an API call.
Type	Boolean
Range of values	true and false
Versions	7.6.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	7.6 - 7.10: false 7.11 and higher: true
Override Example	`$sugar_config['disable_unknown_pla`

	`tforms'] = true;`

disable_vcr

Description	Disables record paging in the detailview (VCR controls). Increases performance by not loading all records from a listview into memory when accessing the record detailview. In 7.x versions, this setting is only applicable to modules running in Backward Compatibility Mode.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['disable_vcr'] = true;`

dump_slow_queries

Description	Logs slow queries to the sugar log file.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['dump_slow_queries'] = true;`

email_default_client

Description	Sets the default email client for all users.
Type	String

Range of values	sugar, external
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	sugar
Override Example	`$sugar_config['email_default_client'] = 'sugar';`

email_default_delete_attachments

Description	When deleting an email, this setting will mark all related notes as deleted, and attempt to delete files that are related to those notes.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	true
Override Example	`$sugar_config['email_default_delete_attachments'] = false;`

email_default_editor

Description	Allows configuring the default editor type for email. 'plain' sets the editor to only use plain text. 'html' allows the editor to be html enabled.
Type	String
Range of values	'plain' and 'html'
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	html
Override Example	`$sugar_config['email_default_editor'] = 'plain';`

email_mailer_timeout

Description	The connection timeout period when sending an email. **Note**: The default value for this configuration is 2 seconds for Sugar versions 12.2.0 and higher. The default value for Sugar versions 7.8.0.0 to 12.1.0 is 10 seconds.
Type	Integer
Versions	7.8.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	2
Override Example	`$sugar_config['email_mailer_timeout'] = 30;`

enable_inline_reports_edit (Deprecated in future release)

Description	Allows a user to edit specific field types (e.g., dropdowns, text fields) in a rows and columns report without having to navigate directly to the record.
Type	Boolean
Range of values	true and false
Versions	6.3.0–12.0.0
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['enable_inline_reports_edit'] = true;`

enable_link_to_drawer

Description	When true, the link-to-drawer feature known as "Focus Drawers" will be enabled for links to Sidecar module records instance-wide. Focus Drawers

	are exclusive to Sugar Sell and Sugar Serve.
Type	Boolean
Range of values	true and false
Versions	10.3.0+
Products	Serve, Sell
Default Value	true
Override Example	`$sugar_config['enable_link_to_drawer'] = false;`

enable_long_text_search

Description	When true, Elasticsearch will enable long-text search for the following field types: 'longtext', 'htmleditable_tinymce'.
Type	Boolean
Range of values	true and false
Versions	10.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['enable_long_text_search'] = true;`

enable_mobile_redirect

Description	Flag indicating whether smartphone users are automatically redirected to the mobile view when navigating to a Sugar instance.
Type	Boolean
Range of values	true and false
Versions	7.1.5+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	true
Override Example	`$sugar_config['enable_mobile_redir`

	`ect'] = false;`

enable_one_index

Description	If you are running Elasticsearch 6 or higher, Elasticsearch creates a separate index for each full-text-search-enabled module. However, performance advantages are gained from instead using a single index for the entire instance. The 'enable_one-index' config enables you to move from 1-index-per-module to 1-index-per-instance. This setting can be enabled by an administrator via Admin > Search > Re-Index, and the re-index will automatically switch the instance over to 1 index and change this config to true.
Type	Boolean
Range of values	true or false
Versions	11.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['enable_one_index'] = true;`

exclude_notifications

Description	This setting controls the modules that are excluded from assignment notifications.
Type	Array
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['exclude_notifications'] = array();`

exclude_notifications.module

Description	Allows an administrator to explicitly disable modules from sending assignment notifications to users.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['exclude_notifications'][''] = true;`

external_cache

Description	SugarCache basic configuration.
Type	Array
Versions	6.2.0+
Products	Enterprise, Serve, Sell
Override Example	`$sugar_config['external_cache'] = array();`

external_cache.memcache

Description	This setting controls the memcache properties.
Type	Array
Versions	6.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['external_cache']['memcache'] = array();`

external_cache.memcache.host

Description	The host url for memchache.
Type	String

Versions	6.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	127.0.0.1
Override Example	`$sugar_config['external_cache']['memcache']['host'] = '192.168.1.1';`

external_cache.memcache.port

Description	The host port for memcache.
Type	Integer
Versions	6.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	11211
Override Example	`$sugar_config['external_cache']['memcache']['port'] = 11212;`

external_cache.redis

Description	This setting controls the redis properties.
Type	Array
Versions	9.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['external_cache']['redis'] = array();`

external_cache.redis.host

Description	The IP address or the hostname of the Redis host.
Type	String
Range of values	'127.0.0.1', '192.168.0.2', 'redis.serveraddress.com', 'local.redis.server.local'
Versions	9.2.0+
Products	Professional, Enterprise, Ultimate,

	Serve, Sell
Default Value	127.0.0.1
Override Example	`$sugar_config['external_cache'] ['redis'] ['host'] = '127.0.0.1';`

external_cache.redis.persistent

Description	The type of connection to the Redis server: persistent or not.
Type	Boolean
Range of values	true and false
Versions	9.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	true
Override Example	`$sugar_config['external_cache'] ['redis'] ['persistent'] = false;`

external_cache.redis.port

Description	The Port number configured in the Redis host.
Type	Integer
Range of values	Any TCP port number
Versions	9.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	6379
Override Example	`$sugar_config['external_cache'] ['redis'] ['port'] = 6377;`

external_cache.redis.timeout

Description	Connection timeout to the Redis server. 0 means unlimited or OS defined.
Type	Integer
Range of values	Any integer greater than or equal to 0
Versions	9.2.0+

Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['external_cache'] ['redis'] ['timeout'] = false;`

external_cache_db_gc_probability

Description	Probability factor to determine when garbage collection on stale keys from the DB backend will happen.
Type	Integer
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	0.0001
Override Example	`$sugar_config['external_cache_db_gc_probability'] = 0.0005;`

external_cache_db_gc_threshold

Description	The threshold in milliseconds to flag garbage collection queries as [SLOW] in sugarcrm.log.
Type	Integer
Range of values	true and false
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	200
Override Example	`$sugar_config['external_cache_db_gc_threshold'] = 500;`

external_cache_disabled

Description	Disables all external caching in Sugar. This is normally set to true to determine if there is a conflict with PHP caching.
Type	Boolean

Range of values	true and false
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['external_cache_disabled'] = true;`

external_cache_disabled_memcached

Description	Disables Memcached caching from working with Sugar. Recommended setting is false and is normally set to true to determine if there is a conflict with PHP caching.
Type	Boolean
Range of values	true and false
Versions	6.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['external_cache_disabled_memcached'] = false;`

external_cache_disabled_wincache

Description	Disables WinCache caching from working with Sugar. Recommended setting is false and is normally set to true to determine if there is a conflict with PHP caching.
Type	Boolean
Range of values	true and false
Versions	6.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['external_cache_disa`

	`bled_wincache'] = false;`

external_cache_force_backend

Description	Force given external key/value cache backend. Make sure that the requirements are met and any other cache backend specific configuration is applied.
Type	String
Range of values	apc, db, file, memcache, memcached, memory, redis, smash, wincache, and zend
Versions	6.2.0+
Products	Professional, Enterprise
Default Value	empty
Override Example	`$sugar_config['external_cache_force_backend'] = 'db';`

forms

Description	An array defining form requirements.
Type	Array
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['forms'] = array();`

forms.requireFirst

Description	Presents all required fields grouped together in the first panel on the EditView form.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell

Default Value	false
Override Example	`$sugar_config['forms']['requireFirst'] = true;`

freeze_list_headers

Description	Global admin config that turns the frozen headers on/off. By default, most list views now have frozen headers and a static pagination element at the bottom of the screen. When the setting is turned off (set to false), this will revert to non-frozen headers and a user having to scroll down to see the pagination buttons. This is enforced in almost every list view in Sugar with the exception of some unique list views that do not have pagination controls.
Type	Boolean
Versions	12.0.0+
Products	Enterprise, Serve, Sell
Default Value	true
Override Example	`$sugar_config['freeze_list_headers'] = false;`

gs_use_shortcut_operator

Description	The default value of true permits use of shortcut operators in global search. These operators are: '&' for AND, '|' for OR, and '-' for NOT. Setting this to false will disable the shortcut operators in global search and cause these characters to be interpreted as literals.
Type	Boolean
Range of values	true and false
Versions	8.3.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	true

Override Example	`$sugar_config['gs_use_shortcut_operator'] = false`

hide_admin_licensing

Description	Hides the License settings subpanel in the administrative panel.
Type	Boolean
Range of values	true and false
Versions	6.5.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['hide_admin_licensing'] = true;`

hide_full_text_engine_config

Description	Determines if the FTS settings are present in the admin search page in Admin > Search.
Type	Boolean
Range of values	true and false
Versions	6.5.15+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['hide_full_text_engine_config'] = true;`

hide_subpanels

Description	This setting only applies to modules running in Backward Compatibility Mode. When a DetailView is loaded, all subpanels are collapsed. Collapsing subpanels on load increases performance by not querying for data

	until a user explicitly expands a subpanel.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['hide_subpanels'] = true;`

hide_subpanels_on_login

Description	This setting only applies to modules running in backward compatibility mode. Collapses subpanels per session. When a DetailView is initially loaded during a session, all subpanels are collapsed. Once explanded, it will remain expanded until the user logs out. Collapsing subpanels on load increases performance by not querying for data until a user explicitly expands a subpanel.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['hide_subpanels_on_login'] = true;`

hint

Description	The array of configurations related to Hint.
Type	Array
Versions	10.3.0+

Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['hint'] = array();`

hint.hint_install_target_geo

Description	This optional property allows you to, before installation of Hint, explicitly set which region you would like Hint to connect to, overriding the default behavior of the Hint installer. The default behavior is for Hint to connect to services hosted in the region nearest to that of the Sugar instance on which it is being installed. See the Hint Administration Guide for more details on Hint service regions: https://support.sugarcrm.com/Documentation/Installable_Connectors/Hint/Hint_Installation_Guide/#Choosing_the_Hint_Services_Region **Note**: Once configured, it is not possible to modify the region Hint connects to. Switching regions requires completely uninstalling and reinstalling Hint, which will cause you to lose all existing system and end-user settings. **Note**: 'APSE' is only available for 12.0 and higher.
Type	String
Range of values	'US', 'EU', 'APSE'
Versions	10.3.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['hint']['hint_install_target_geo'] = 'US';`

history_max_viewed

Description	The number of history items from the

	tracker to display for a user.
Type	Integer
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	50
Override Example	`$GLOBALS['sugar_config']['history_max_viewed'] = 25;`

host_name

Description	Sets the host name of the instance, such as the website address or location where the instance is hosted and accessed.
Type	String
Versions	6.5.10+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	localhost
Override Example	`$sugar_config['host_name'] = 'localhost';`

import_max_records_per_file

Description	The number of records to process per batch during an import using the Import Wizard.
Type	Integer
Range of values	Any integer
Versions	7.5.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	100
Override Example	`$sugar_config['import_max_records_per_file'] = 50;`

installer_locked

Description	Sets whether the installer is locked or not. When false, it is possible to access Sugar's initial configuration page to reinstall the instance.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	true
Override Example	`$sugar_config['installer_locked'] = false;`

jobs

Description	Job Queue configurations.
Type	Array
Versions	6.5.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['jobs'] = array();`

jobs.hard_lifetime

Description	Hard deletes all jobs that are older than the hard cutoff. Default is 21 days.
Type	Integer
Versions	6.5.1+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	21
Override Example	`$sugar_config['jobs']['hard_lifetime'] = 21;`

jobs.max_retries

Description	Maximum number of failures for job. Default is 5.
Type	Integer
Versions	6.5.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	5
Override Example	`$sugar_config['jobs']['max_retries'] = 5;`

jobs.min_retry_interval

Description	Minimal interval between job reruns. Default is 30 seconds.
Type	Integer
Versions	6.5.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	30
Override Example	`$sugar_config['jobs']['min_retry_interval'] = 30;`

jobs.soft_lifetime

Description	Soft deletes all jobs that are older than cutoff. Default is 21 days.
Type	Integer
Versions	6.5.1+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	7
Override Example	`$sugar_config['jobs']['soft_lifetime'] = 7;`

jobs.timeout

Description	If a job is running longer than the limit, the job is failed by force. Specified in

	seconds. Default is 3600 seconds (1 hour).
Type	Integer
Versions	6.5.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	3600
Override Example	`$sugar_config['jobs']['timeout'] = 86400;`

languages

Description	The list of languages available in the system. An administrator can limit the languages available by removing them from this array. When modifying this array, you will need to make sure that the default_language config setting matches a language available in the list.
Type	Array
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	All available languages
Override Example	`$sugar_config['languages'] = array ('en_us' => 'English (US)');`

list_max_entries_per_page

Description	Listview items per page.
Type	String
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	20
Override Example	`$sugar_config['list_max_entries_per_page'] = '20';`

list_report_max_per_page

Description	Sets the maximum number of reports that are listed on each page in the Reports module.
Type	Integer
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	100
Override Example	`$sugar_config['list_report_max_per_page'] = 100;`

logger

Description	An array that defines all of the logging settings.
Type	Array
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['logger'] = array();`

logger.channels

Description	This setting controls the PSR-3 Logger channels. There are stock channels included in Sugar that can be utilized for various troubleshooting, and custom channels can easily be utilized in custom code. Please consult the PSR-3 Logger Documentation for further information on modifying channels and utilizing them.
Type	Array
Versions	7.9.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['logger']['channels'`

	`] = array();`

logger.channels.authentication

Description	This setting controls the PSR-3 Logger Channel configuration for the authentication channel.
Type	Array
Versions	7.10.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['logger']['channels'][authentication'] = array();`

logger.channels.authentication.handlers

Description	This setting controls the PSR-3 Logger Handler utilized by the authentication logging channel.
Type	Array
Versions	7.10.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['logger']['channels'][authentication']['handlers'][] = 'File';`

logger.channels.authentication.level

Description	This setting controls the authentication PSR-3 Logger channel's log level. If you need to troubleshoot authentication issues in Sugar, and do not want to enable debug logging on the entire system, you can utilize this channel for more robust logging around the Authentication Controller.
Type	String
Range of values	'emergency', 'alert', 'critical', 'error', 'warning', 'notice', 'info', 'debug', 'off'
Versions	7.10.0.0+

Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['logger']['channels']['authentication']['level'] = 'debug';`

logger.channels.authentication.processors

Description	This setting controls the authentication PSR-3 Logger channels processors.
Type	Array
Range of values	'request', 'backtrace'
Versions	7.10.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['logger']['authentication']['authentication']['processors'] = array('request', 'backtrace');`

logger.channels.channel

Description	This setting controls the PSR-3 Logger channels. There are stock channels included in Sugar that can be utilized for various troubleshooting, and custom channels can easily be utilized in custom code. The channel configuration typically consists of three properties, the level, handlers, and processors, however not all need to be defined as they do inherit their properties from the default Logger implementation. Please consult the PSR-3 Logger Documentation for further information on adding custom channels and utilizing them.
Type	Array
Versions	7.9.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['logger']['channels'`

	`][''] = array();`

logger.channels.channel.handlers

Description	This setting controls the PSR-3 Logger Handler utilized by the defined logging channel. By default Sugar only comes with the 'File' handler, however custom handlers can easily be added as outlined in our PSR-3 Logger Documentation.
Type	Array
Versions	7.9.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['logger']['channels'][''][handlers'][] = 'File';`

logger.channels.channel.level

Description	This setting controls the PSR-3 Log Level for the specified log channel.
Type	String
Range of values	'emergency', 'alert', 'critical', 'error', 'warning', 'notice', 'info', 'debug', 'off'
Versions	7.9.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['logger']['channels'][''][level'] = 'alert';`

logger.channels.channel.processors

Description	This setting controls a specific PSR-3 Logger channels processors. To more easily debug issues, you can enable the provided backtrace or request Log Processors to provide more insight into a particular log. For more information on adding custom processors and using them on logging channels consult the PSR-3 Logger Documentation.
Type	Array

Range of values	'request', 'backtrace'
Versions	7.9.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['logger']['channels']['']['processors'] = array('request', 'backtrace');`

logger.channels.db

Description	This setting controls the PSR-3 Logger Channel configuration for the db channel.
Type	Array
Versions	8.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['logger']['channels']['db'] = array();`

logger.channels.db.handlers

Description	This setting controls the PSR-3 Logger Handler utilized by the db logging channel.
Type	Array
Versions	8.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['logger']['channels']['db']['handlers'][] = 'File';`

logger.channels.db.level

Description	This setting controls the Log Level for the db channel of the PSR-3 Logger.
Type	String
Range of values	'emergency', 'alert', 'critical', 'error', 'warning', 'notice', 'info', 'debug', 'off'
Versions	8.1.0+

Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['logger']['channels']['db']['level'] = 'debug';`

logger.channels.db.processors

Description	This setting controls the db PSR-3 Logger channels processors.
Type	Array
Range of values	'request', 'backtrace'
Versions	8.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['logger']['channels']['db']['processors'] = array('request', 'backtrace');`

logger.channels.deprecation

Description	This setting controls the PSR-3 Logger Channel configuration for the deprecation channel. If you need to clean up your customizations from usage of deprecated functionality of Sugar or some underlying libraries which implement runtime deprecation logging and you do not want to enable warning logging on the entire system, you can utilize this channel.
Type	Array
Versions	11.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	empty
Override Example	`$sugar_config['logger']['channels']['deprecation'] = [];`

logger.channels.deprecation.handlers

Description	This setting controls the PSR-3 Logger Handler utilized by the deprecation

	logging channel. By default, Sugar only comes with the File handler, however custom handlers can be added as outlined in our PSR-3 Logger documentation.
Type	Array
Versions	11.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	empty
Override Example	`$sugar_config['logger']['channels']['deprecation']['handlers'][] = [ 'type' => 'File', 'name' => 'symfony_deprecations', ];`

logger.channels.deprecation.level

Description	This setting controls the deprecation PSR-3 Logger channel's log level.
Type	String
Range of values	emergency, alert, critical, error, warning, notice, info, debug, off
Versions	11.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	alert
Override Example	`$sugar_config['logger']['channels']['deprecation']['level'] = 'warning';`

logger.channels.deprecation.processors

Description	This setting controls the deprecation PSR-3 Logger channels processors. To more easily debug issues, you can enable the provided backtrace or request Log Processors to provide more insight into a particular log. For more information on adding custom processors and using them on logging channels, please refer to the PSR-3 Logger documentation.

Type	Array
Range of values	request, backtrace
Versions	11.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	empty
Override Example	`$sugar_config['logger']['deprecation']['processors'] = ['request', 'backtrace'];`

logger.channels.input_validation

Description	This setting controls the PSR-3 Logger Channel configuration for the input_validation channel.
Type	Array
Versions	7.9.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['logger']['channels']['input_validation'] = array();`

logger.channels.input_validation.handlers

Description	This setting controls the PSR-3 Logger Handler utilized by the input_validation logging channel.
Type	Array
Versions	7.9.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['logger']['channels']['input_validation']['handlers'][] = 'File';`

logger.channels.input_validation.level

Description	This setting controls the logging level for input validation failure messages when validation.soft_fail is enabled.
Type	String

Range of values	'emergency', 'alert', 'critical', 'error', 'warning', 'notice', 'info', 'debug', 'off'
Versions	7.9.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['logger']['channels'][input_validation'][level'] = 'warning';`

logger.channels.input_validation.processors

Description	This setting controls the input_validation PSR-3 Logger channels processors.
Type	Array
Range of values	'request', 'backtrace'
Versions	7.9.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['logger']['channels'][input_validation']['processors'] = array('request', 'backtrace');`

logger.channels.metadata

Description	This setting controls the PSR-3 Logger Channel configuration for the metadata channel.
Type	Array
Versions	7.9.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['logger']['channels'][metadata'] = array();`

logger.channels.metadata.handlers

Description	This setting controls the PSR-3 Logger Handler utilized by the metadata logging channel.
Type	Array

Versions	7.9.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['logger']['channels']['metadata']['handlers'][] = 'File';`

logger.channels.metadata.level

Description	This logging channel can be used to track and debug metadata refresh issues. Overly frequent metadata refreshes will cause performance issues for affected Sugar instances. This metadata logging channel can help determine the frequency and causes of metadata refreshes.
Type	String
Range of values	'emergency', 'alert', 'critical', 'error', 'warning', 'notice', 'info', 'debug', 'off'
Versions	7.9.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['logger']['channels']['metadata']['level'] = 'debug';`

logger.channels.metadata.processors

Description	This setting controls the metadata PSR-3 Logger channels processors.
Type	Array
Range of values	'request', 'backtrace'
Versions	7.9.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['logger']['channels']['metadata']['processors'] = array('request', 'backtrace');`

logger.channels.rest

Description	This setting controls the PSR-3 Logger Channel configuration for the rest channel.
Type	Array
Versions	7.10.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['logger']['channels']['rest'] = array();`

logger.channels.rest.handlers

Description	This setting controls the PSR-3 Logger Handler utilized by the rest logging channel.
Type	Array
Versions	7.10.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['logger']['channels']['rest']['handlers'][] = 'File';`

logger.channels.rest.level

Description	This setting controls the Log Level for the rest channel of the PSR-3 Logger.
Type	String
Range of values	'emergency', 'alert', 'critical', 'error', 'warning', 'notice', 'info', 'debug', 'off'
Versions	7.10.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['logger']['channels']['rest']['level'] = 'debug';`

logger.channels.rest.processors

Description	This setting controls the rest PSR-3 Logger channels processors.
Type	Array
Range of values	'request', 'backtrace'

Versions	7.10.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['logger']['channels']['rest']['processors'] = array('request', 'backtrace');`

logger.file.dateFormat

Description	The date format for the log file is any value that is acceptable to the PHP strftime() function. The default is '%c'. For a complete list of available date formats, please see the strftime() PHP documentation at http://php.net/manual/en/function.strftime.php.
Type	String
Range of values	Pattern for date format
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	%c
Override Example	`$sugar_config['logger']['file']['dateFormat'] = '%c';`

logger.file.ext

Description	The extension of the log file. The default value is '.log'.
Type	String
Range of values	Extension for the log
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	.log
Override Example	`$sugar_config['logger']['file']['ext'] = '.log';`

logger.file.maxLogs

Description	When the log file grows to the logger.file.maxSize value, the system will automatically roll the log file. The logger.file.maxLogs value controls the max number of logs that will be saved before it deletes the oldest. The default value is 10.
Type	Integer
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	10
Override Example	`$sugar_config['logger']['file']['maxLogs'] = 10;`

logger.file.maxSize

Description	This value controls the max file size of a log before the system will roll the log file. It must be set in the format '10MB' where 10 is number of MB to store. Always use MB as no other value is currently accepted. To disable log rolling set the value to false. The default value is '10MB'.
Type	String
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	10MB
Override Example	`$sugar_config['logger']['file']['maxSize'] = '10MB';`

logger.file.name

Description	The name of the log file to be written to.
Type	String
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell

Default Value	sugarcrm
Override Example	`$sugar_config['logger']['file']['name'] = 'sugarcrm';`

logger.file.suffix

Description	The suffix to the file name to track logs chronologically. For instance, if you wanted to append the month and year to a file name, you can change this setting to '%m_%Y'. For a complete list of available date formats, please see the strftime() PHP documentation at http://php.net/manual/en/function.strftime.php.
Type	String
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	empty
Override Example	`$sugar_config['logger']['file']['suffix'] = '%m_%Y';`

logger.level

Description	Determines the logging level of the system. The recommended setting is 'fatal'.
Type	String
Range of values	'debug', 'info', 'warn', 'deprecated', 'error', 'fatal', 'security', 'off'
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	fatal
Override Example	`$sugar_config['logger']['level'] = 'fatal';`

logger_visible

Description	Determines whether the Logger

	Settings panel is visible to administrators in Admin > System Settings.
Type	Boolean
Range of values	true and false
Versions	6.7.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	true
Override Example	`$sugar_config['logger_visible'] = false;`

log_dir

Description	Sets the location in the file system where the Sugar log file will be stored. By default, it is set to '.' meaning that it is stored in the root instance directory.
Type	String
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	.
Override Example	`$sugar_config['log_dir'] = '.';`

log_file

Description	Designates the file name where the instance's logs will be stored.
Type	String
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	sugarcrm.log
Override Example	`$sugar_config['log_file'] = 'new_sugarcrm.log'`

log_memory_usage

Description	Logs the memory usage.
Type	Boolean
Range of values	true and false
Versions	5.5.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['log_memory_usage'] = true;`

maintenanceMode

Description	Allows the instance to be placed in a maintenance mode where non-admin users cannot access the instance.
Type	Boolean
Range of values	true and false
Versions	7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['maintenanceMode'] = false;`

marketing_extras_enabled

Description	This configuration disables the marketing content on the Sugar login screen.
Type	Boolean
Range of values	true and false
Versions	8.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	true

Override Example	`$sugar_config['marketing_extras_enabled'] = false;`

mark_emails_seen

Description	Determines whether to mark an email as read before importing the email to Sugar during the inbound email import. This is not recommended as an import failure will cause the email to be marked as read which will be skipped during the next inbound email import.
Type	Boolean
Range of values	true and false
Versions	6.5.17+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['mark_emails_seen'] = true;`

mass_actions

Description	Array that defines mass action behaviors.
Type	Array
Versions	7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['mass_actions'] = array();`

mass_actions.mass_delete_chunk_size

Description	Number of records per chunk while performing a mass delete.
Type	Integer
Versions	7.0.0+

Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	20
Override Example	`$sugar_config['mass_delete_chunk_size'] = 20;`

mass_actions.mass_link_chunk_size

Description	Number of records per chunk while performing mass linking updates.
Type	Integer
Versions	7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	20
Override Example	`$sugar_config['mass_actions']['mass_link_chunk_size'] = 20;`

mass_actions.mass_update_chunk_size

Description	Number of records per chunk while performing a mass update.
Type	Integer
Versions	7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	500
Override Example	`$sugar_config['mass_actions']['mass_update_chunk_size'] = 500;`

mass_actions.max_records_to_merge

Description	Number of records per chunk while performing a mass merge.
Type	Integer
Versions	7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell

Default Value	5
Override Example	`$sugar_config['mass_actions']['max_records_to_merge'] = 20;`

max_aggregate_email_attachments_bytes

Description	The maximum allowed size of all uploaded attachments added together for a single email message. Users may upload files as email attachments within to the lowest of the PHP upload_max_filesize, post_max_size, and system upload_maxsize size limits, but users cannot upload more files to a single message than the max_aggregate_email_attachments_bytes configuration permits.
Type	Integer
Versions	7.10.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	10000000
Override Example	`$sugar_config['max_aggregate_email_attachments_bytes'] = 20000000;`

max_session_time

Description	Determines the maximum lock time in seconds between session requests. When a session request is locked for long periods of time, other requests are blocked until it is released. A null value will not implement a max session time.
Type	Integer
Versions	6.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	null
Override Example	`$sugar_config['max_session_time'] = 1;`

metrics_enabled

Description	Whether or not system metrics have been enabled.
Type	Boolean
Range of values	true and false
Versions	6.7.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['metrics_enabled'] = true;`

metric_providers

Description	A Name/Path array of metric providers for measuring system metrics
Type	Array
Versions	6.7.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['metric_providers'] = array ('provider' => 'path/to/provider');`

metric_settings

Description	The individual settings required by a metric provider. And values provided will be passed when instantiating the metric providers object.
Type	Array
Versions	6.7.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['metric_settings'] = array ('provider' => array('setti`

	`ng' => 'value'));`

minify_resources

Description	Determines whether minification and compression are applied to javascript resources
Type	Boolean
Range of values	true and false
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	true
Override Example	`$sugar_config['minify_resources'] = false;`

moduleInstaller.disableActions

Description	Part of the moduleInstaller array. When packageScan is set to 'true', Sugar does not restrict any specific actions that can be completed during the installation process. The disableActions parameter allows you to define any actions you wish to restrict from executing.
Type	Array
Range of values	Specific actions to restrict in module packages
Versions	5.2.0j+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	array()
Override Example	`$sugar_config['moduleInstaller']['disableActions'] = array('pre_execute', 'post_execute');`

moduleInstaller.disableFileScan

Description	When packageScan is set to 'true',

	Sugar scans all files in an installable package to ensure that the file extensions are acceptable and that the files do not contain denylisted class or function calls. Setting the disableFileScan parameter to 'true' avoids this scan from occurring while still enforcing other parameters set.
Type	Boolean
Range of values	true and false
Versions	5.2.0j+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['moduleInstaller']['disableFileScan'] = true;`

moduleInstaller.packageScan

Description	Enables package scanning on any modules uploaded through Module Loader prior to the installation. If the package is found to violate any restrictions of the packageScan, the installation will not proceed and an error report will be generated to the user attempting the install.
Type	Boolean
Range of values	true and false
Versions	5.5.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['moduleInstaller']['packageScan'] = true;`

moduleInstaller.validExt

Description	Part of the moduleInstaller array. When moduleInstaller.packageScan is set to true, Sugar will not allow certain file extensions to be present in an

	installable package. By default, Sugar allows the following extensions: 'png', 'gif', 'jpg', 'css', 'js', 'php', 'txt', 'html', 'htm', 'tpl', 'pdf', 'md5', 'xml', 'hbs', 'less', and 'wsdl'. This parameter allows you to define **additional** extensions deemed safe to install on your instance of Sugar.
Type	Array
Range of values	File extensions to allow
Versions	6.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	array()
Override Example	`$sugar_config['moduleInstaller']['validExt'] = array('swf', 'log');`

mso_fixup_paragraph_tags

Description	Determines whether email HTML is scrubbed for empty paragraph tags when displayed in the application. Setting this value to true will enable the HTML scrubbing. Enabling this setting does not affect the HTML stored in the database from the initial import. Created as a result of bug 66022 (https://web.sugarcrm.com/support/issues/66022).
Type	Boolean
Range of values	true and false
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['mso_fixup_paragraph_tags'] = true;`

new_email_addresses_opted_out

Description	If true, then newly created EmailAddress records in Sugar will be opted-out by default. This setting can also be controlled using the "Opt-out new email addresses by default" setting in Admin > System Email Settings.
Type	Boolean
Range of values	true and false
Versions	8.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['new_email_addresses_opted_out'] = true;`

noPrivateTeamUpdate

Description	Prevents name changes to a users private team. This setting can be modified by changing in Admin > System Settings > Advanced > Prevent name changes by users to update their Private Team Name
Type	Boolean
Range of values	true and false
Versions	7.6.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['noPrivateTeamUpdate'] = true;`

oauth2

Description	Configutations for the oauth2 server.
Type	Array
Versions	7.0.0+
Products	Professional, Enterprise, Ultimate,

	Serve, Sell
Override Example	`$sugar_config['oauth2'] = array();`

oauth2.access_token_lifetime

Description	The lifetime of the access token in seconds. This setting controls how often Sugar will check to see if the user's token has expired. It is recommended for this value not to surpass more than half of the oauth2.refresh_token_lifetime setting. **Note**: This setting cannot exceed the maximum PHP session timeout, which is configured by PHP setting session.gc_maxlifetime. In SugarCloud the maximum session timeout is set to 7200s (2 hours). **Note**: This setting is not respected for instances that use SugarIdentity.
Type	Integer
Versions	7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	3600
Override Example	`$sugar_config['oauth2']['access_token_lifetime'] = 3600;`

oauth2.refresh_token_lifetime

Description	The lifetime of refresh token in seconds. We recommend the oauth2.refresh_token_lifetime remains at 1209600 seconds or less for security. Should you increase this number, do not exceed 2147220 seconds. **Note**: This setting is not respected for instances that use SugarIdentity.

Type	Integer
Versions	7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	1209600
Override Example	`$sugar_config['oauth2']['refresh_token_lifetime'] = 1409600;`

oauth_token_expiry

Description	Sets whether OAuth tokens will expire.
Type	String
Range of values	true and false
Versions	7.5.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	False
Override Example	`$sugar_config['oauth_token_expiry'] = '0';`

oauth_token_life

Description	Sets the length (in seconds) of the life of an OAuth token.
Type	Integer
Versions	7.5.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	86400
Override Example	`$sugar_config['oauth_token_life'] = '86400';`

oracle_enable_ci

Description	By default, Oracle searching is case sensitive in Sugar, which can be a problem if you are not sure of the case

	of the data you are searching for. Using this settings, you can turn on insensitive search for Oracle 10g and 11g.
Type	Boolean
Range of values	true and false
Versions	6.1.3+
Products	Enterprise, Ultimate
Default Value	false
Override Example	`$sugar_config['oracle_enable_ci'] = true;`

passwordHash

Description	Array that defines the password hashing behaviors. **Note**: This configuration's functionality has been deprecated in Sugar versions 13.0 and higher and will be removed in a future release.
Type	Array
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['passwordHash'] = array();`

passwordHash.algo

Description	The specific algorithm to be used by the hashing backend. The available values depend on the selected passwordHash.backend. See http://php.net/manual/en/password.constants.php for more information when using the native backend. **Note**: This configuration's functionality has been deprecated in Sugar versions 13.0 and higher and will be removed in

	a future release.
Type	String
Range of values	For native backend: "PASSWORD_DEFAULT" and "PASSWORD_BCRYPT". For sha2 backend: "CRYPT_SHA256" and "CRYPT_SHA512"
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	"PASSWORD_DEFAULT" for native. "CRYPT_SHA256" for sha2 backend.
Override Example	`$sugar_config['passwordHash']['algo'] = 'PASSWORD_BCRYPT';`

passwordHash.allowLegacy

Description	Allow logins of users who have their password stored using the insecure legacy MD5 hash. During the transition period for the 7.7 series, this will be allowed out of the box. Versions past 7.7 will no longer allow by default authentication against insecure hashes. This configuration parameter can be used to change the default behavior. **Note**: This configuration's functionality has been deprecated in Sugar versions 13.0 and higher and will be removed in a future release.
Type	Boolean
Range of values	true and false
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	7.7.x: true 7.8.x+:false
Override Example	`$sugar_config['passwordHash']['all`

	`owLegacy'] = false;`

passwordHash.backend

Description	The password hash backend class to use. By default, the "native" backend is used which uses Blowfish to hash the passwords in the database. An alternative is using the "sha2" backend which makes use of SHA-2 hashing instead. Depending on the backend, different configuration options are available for passwordHash.algo and passwordHash.options. **Note**: This configuration's functionality has been deprecated in Sugar versions 13.0 and higher and will be removed in a future release.
Type	String
Range of values	'sha2' and 'native'
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	native
Override Example	`$sugar_config['passwordHash']['backend'] = 'sha2';`

passwordHash.options

Description	The available configuration values depend on the selected passwordHash.backend. See http://php.net/manual/en/function.password-hash.php when using the native backend and http://php.net/manual/en/function.crypt.php for the sha2 backend. Note that only the following specified options are allowed. For native backend: passwordHash.options.cost. For sha2 backend: passwordHash.options.rounds.

	Note: This configuration's functionality has been deprecated in Sugar versions 13.0 and higher and will be removed in a future release.
Type	Array
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['passwordHash']['options'] = array();`

passwordHash.options.cost

Description	An available option when the passwordHash.backend configuration is set to "native". More information on the native backend can be found at http://php.net/manual/en/function.password-hash.php **Note**: This configuration's functionality has been deprecated in Sugar versions 13.0 and higher and will be removed in a future release.
Type	Integer
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	10
Override Example	`$sugar_config['passwordHash']['options']['cost'] = 15;`

passwordHash.options.rounds

Description	An available option when the passwordHash.backend configuration is set to "sha2". More information on the sha2 backend can be found at http://php.net/manual/en/function.crypt.php

	Note: This configuration's functionality has been deprecated in Sugar versions 13.0 and higher and will be removed in a future release.
Type	Integer
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	5000
Override Example	`$sugar_config['passwordHash']['options']['rounds'] = 4000;`

passwordHash.rehash

Description	When enabled, the system will automatically rehash the user's password when successfully authenticated. This allows adoption to the newly configured password hash backend without resetting user's passwords. **Note**: This configuration's functionality has been deprecated in Sugar versions 13.0 and higher and will be removed in a future release.
Type	Boolean
Range of values	true and false
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	true
Override Example	`$sugar_config['passwordHash']['rehash'] = false;`

passwordsetting

Description	Defines all of the password requirements for the instance.
Type	Array
Versions	5.5.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['passwordsetting'] = array();`

passwordsetting.forgotpasswordON

Description	Enables the Forgot Password features. When enabled, users will have the ability to reset their own passwords at the Login page. Set in UI via Admin->Password Management.
Type	String
Range of values	0, 1
Versions	5.5.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	1
Override Example	`$sugar_config['passwordsetting']['forgotpasswordON'] = '0';`

passwordsetting.linkexpiration

Description	Determines whether the password reset link expires.
Type	String
Range of values	0, 1
Versions	6.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['passwordsetting']['linkexpiration'] = '0';`

passwordsetting.onelower

Description	Configures whether at least one lower-case letter is required in users' passwords.
Type	String
Range of values	0, 1
Versions	6.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['passwordsetting']['onespecial'] = '0';`

passwordsetting.onenumber

Description	Configures whether at least one number is required in users' passwords.
Type	String
Range of values	0, 1
Versions	6.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	1
Override Example	`$sugar_config['passwordsetting']['onenumber'] = '1';`

passwordsetting.oneupper

Description	Configures whether at least one upper-case letter is required in users' passwords.
Type	Boolean
Range of values	0, 1
Versions	6.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['passwordsetting']['oneupper'] = 1;`

passwordsetting.systexpirationtype

Description	Specifies the unit of measurement for passwordsetting.systexpiratiotime. The available options are: Days (1), Weeks (7) and Months (30). This value can be set in the UI via Admin > Password Management.
Type	Integer
Range of values	1, 7, and 30
Versions	5.5.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	1
Override Example	`$sugar_config['passwordsetting']['systexpirationtype'] = '7';`

pdf_file_max_lifetime

Description	The interval in seconds of when to expire and remove generated PDF files. It is important to note that the "Remove temporary PDF files" scheduler job must be enabled to remove the PDF files.
Type	Integer
Versions	7.6.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	86400
Override Example	`$sugar_config['pdf_file_max_lifetime'] = 86400;`

perfProfile

Description	Allows for an admin to tweak aspects of the system for performance enhancements when fetching records. As every system is different, your mileage will vary when using the perfProfile settings. Increases or decreases in performance will depend on a combination of primarily the

	amount of users, amount of unique team sets, and data size per module. The perfProfile parameters are available to be able to fine tune the team security performance no your platform. It is not recommended to change any setting directly in a production environment without testing and understanding the impact.
Type	Array
Versions	7.2.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['perfProfile'] = array();`

perfProfile.TeamSecurity

Description	Determines whether the team security filtering is applied.
Type	Array
Versions	7.2.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['perfProfile']['TeamSecurity'] = array();`

perfProfile.TeamSecurity.default

Description	This setting is used to enable or disable Team Security denormalization feature by default across the application.
Type	Array
Range of values	true and false
Versions	7.7.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['perfProfile']['TeamSecurity']['default'] = array();`

perfProfile.TeamSecurity.default.disable_subquery_optimizer_hint

Description	As of Sugar 11.0.2, the default behavior of TeamSecurity on MySQL no longer uses an INNER JOIN to a 'team_sets' subquery; instead, Sugar 11.0.2 and higher use WHERE team_set_id IN [team_sets subquery] with a MySQL optimizer hint. To disable this approach and switch back to the INNER JOIN implementation, set this config to 'true'.
Type	Boolean
Range of values	true and false
Versions	11.0.2+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['perfProfile']['TeamSecurity']['default']['disable_subquery_optimizer_hint'] = true;`

perfProfile.TeamSecurity.default.teamset_prefetch

Description	Fetches the list of team sets that the current user is a member of in a separate query and use those ID's directly in the team security clause to avoid adding a subquery. Under certain conditions, this may improve team security performance. It is important to note that 'default' applies this applied to all modules. To set specific settings for a specific module, you will need to use: $sugar_config['perfProfile']['TeamSecurity']['{module}']['teamset_prefetch'] = true;
Type	Boolean
Range of values	true and false
Versions	7.2.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false

Override Example	`$sugar_config['perfProfile']['TeamSecurity']['default']['teamset_prefetch'] = true;`

perfProfile.TeamSecurity.default.teamset_prefetch_max

Description	The maximum amount of team set ID's to include in the team security clause. If the current user is a member of more unique teams sets than this value, the system will fall back to using a regular subquery instead. Under certain conditions, this may improve team security performance. It is important to note that 'default' applies this applied to all modules. To set specific settings for a specific module, you will need to use: $sugar_config['perfProfile']['TeamSecurity']['{module}']['teamset_prefetch_max'] = true;
Type	Integer
Versions	7.2.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	empty
Override Example	`$sugar_config['perfProfile']['TeamSecurity']['default']['teamset_prefetch_max'] = 500;`

perfProfile.TeamSecurity.default.use_denorm

Description	This setting is used to enable or disable Team Security denormalization feature. Team Security denormalization can significantly improve SQL query performance when there are a large number of Teams in a Sugar instance. The cost is that this feature relies on a separate Team security denormalization table that needs to be updated as records change to ensure Sugar's Team security visibility rules are applied properly. It is important to note that 'default' applies this to all modules. To

	set specific settings for a specific module, you will need to use: $sugar_config['perfProfile']['TeamSecurity']['{module}']['use_denorm'] = true;
Type	Boolean
Range of values	true and false
Versions	8.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['perfProfile']['TeamSecurity']['default']['use_denorm'] = true;`

perfProfile.TeamSecurity.default.where_condition

Description	Determines whether the team security filtering is applied in the WHERE clause instead of SELECT clause. Under certain conditions, this may improve team security performance. It is important to note that 'default' applies this applied to all modules. To set specific settings for a specific module, you will need to use: $sugar_config['perfProfile']['TeamSecurity']['{module}']['where_condition'] = true;
Type	Boolean
Range of values	true and false
Versions	7.2.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['perfProfile']['TeamSecurity']['default']['where_condition'] = true;`

perfProfile.TeamSecurity.gs_use_normalized_teams

Description	Determines whether the team set IDs are cached; this may improve

	Elasticsearch indexing performance, especially for those instances with a large number of team sets per user. Admin must schedule re-indexing Elasticsearch server if this value has been changed.
Type	Boolean
Range of values	true and false
Versions	11.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['perfProfile']['TeamSecurity']['gs_use_normalized_teams'] = true;`

perfProfile.TeamSecurity.inline_update

Description	When denormalization is in use (perfProfile.TeamSecurity.default.use_denorm config set to true), this setting will allow the denormalization table to be updated in real time as records are updated in the system. If this setting is disabled, the denormalization table must be scheduled by the Sugar administrator via Schedulers > Rebuild Denormalized Team Security Data.
Type	Boolean
Range of values	true and false
Versions	8.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['perfProfile']['TeamSecurity']['inline_update'] = true;`

pmse_settings_default

Description	Settings for troubleshooting and debugging SugarBPM
Type	Array
Versions	7.6.0.0+
Products	Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['pmse_settings_default'] = array();`

pmse_settings_default.error_number_of_cycles

Description	Number of cycles before triggering an error in SugarBPM
Type	String
Versions	7.6.0.0+
Products	Enterprise, Ultimate, Serve, Sell
Default Value	10
Override Example	`$sugar_config['pmse_settings_default']['error_number_of_cycles'] = 15;`

pmse_settings_default.error_timeout

Description	Time in seconds of timeout before triggering an error in SugarBPM
Type	String
Versions	7.6.0.0+
Products	Enterprise, Ultimate, Serve, Sell
Default Value	40
Override Example	`$sugar_config['pmse_settings_default']['error_timeout'] = 45;`

pmse_settings_default.logger_level

Description	The default logger level for SugarBPM
Type	String
Range of values	emergency, alert, critical, error, warning, notice, info, debug
Versions	7.6.0.0+

Products	Enterprise, Ultimate, Serve, Sell
Default Value	critical
Override Example	`$sugar_config['pmse_settings_default']['logger_level'] = 'emergency';`

portal_enableSelfSignUp

Description	This portal configuration option controls whether external portal sign-ups are permitted and is also controlled in the user interface via the "Allow new users to sign up" Sugar Portal setting. See the Sugar Portal page in the Administration Guide for details on this setting: https://support.sugarcrm.com/SmartLinks/Administration_Guide/Developer_Tools/Sugar_Portal/hdr_Configure_Portal This option is disabled by default for new instances. **Note**: Upon upgrade to 11.1.0 or higher from version 11.0.x or lower, the default value will be 'true' for instances that had their portal enabled before upgrade and at least one contact configured for portal.
Type	Boolean
Range of values	true and false
Versions	11.1.0+
Products	Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['portal_enableSelfSignUp'] = true;`

processes_auto_save_interval

Description	Defines the interval (in milliseconds) at which auto-saving of process definitions will occur while using the SugarBPM process designer. A value of 0 will disable auto-saving.
Type	Integer

Versions	8.3.0+
Products	Enterprise, Ultimate, Serve, Sell
Default Value	30000
Override Example	`$sugar_config['processes_auto_save_interval'] = 120000`

processes_auto_validate_on_autosave

Description	Defines whether validation of process definitions will automatically occur when a process definition is auto-saved. To enable this setting, 'processes_auto_save_interval' must be greater than 0.
Type	Boolean
Range of values	true and false
Versions	8.3.0+
Products	Enterprise, Ultimate, Serve, Sell
Default Value	true
Override Example	`$sugar_config['processes_auto_validate_on_autosave'] = false`

processes_auto_validate_on_import

Description	Determines whether validation of process definitions will automatically occur immediately after a process definition is imported.
Type	Boolean
Range of values	true and false
Versions	8.3.0+
Products	Enterprise, Ultimate, Serve, Sell
Default Value	true
Override Example	`$sugar_config['processes_auto_validate_on_import'] = false`

proxy_visible

Description	Determines whether the Proxy Settings panel is visible to administrators in Admin > System Settings. Instances running on Sugar's cloud environment will have this setting enforced as false.
Type	Boolean
Range of values	true or false
Versions	11.3.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	true
Override Example	`$sugar_config['proxy_visible'] = false;`

push_notification

Description	The array of configurations related to push notifications.
Type	Array
Versions	11.0.0+
Products	Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['push_notification'] = array();`

push_notification.enabled

Description	For Sugar instances that are eligible for push notifications, use this setting to turn push notifications on or off. See the Android and iOS User Guides for the requirements to be eligible: https://support.sugarcrm.com/Documentation/Mobile_Solutions/SugarCRM_Mobile/SugarCRM_Mobile_for_iOS_User_Guide/#Eligible_Sugar_Instances https://support.sugarcrm.com/Documentation/Mobile_Solutions/SugarCRM_Mobile/SugarCRM_Mobile_for_Android_User_Guide/#Eligible_Sugar_Instances
Type	Boolean

Range of values	true and false
Versions	11.0.0+
Products	Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['push_notification']['enabled'] = true;`

push_notification.service_provider

Description	For Sugar instances that are eligible for push notifications, this setting specifies the provider. Currently, SugarPush is the only available provider. See the Android and iOS User Guides for the requirements to be eligible: https://support.sugarcrm.com/Documentation/Mobile_Solutions/SugarCRM_Mobile/SugarCRM_Mobile_for_iOS_User_Guide/#Eligible_Sugar_Instances https://support.sugarcrm.com/Documentation/Mobile_Solutions/SugarCRM_Mobile/SugarCRM_Mobile_for_Android_User_Guide/#Eligible_Sugar_Instances
Type	String
Range of values	'SugarPush'
Versions	11.0.0+
Products	Enterprise, Ultimate, Serve, Sell
Default Value	'SugarPush'
Override Example	`$sugar_config['push_notification']['service_provider'] = 'SugarPush';`

require_accounts

Description	Determines whether an account is required for record creation within the system.
Type	Boolean
Range of values	true and false

Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	true
Override Example	`$sugar_config['require_accounts'] = false;`

rest_response_etag_cache_age

Description	Controls how long the browser should cache rest responses.
Type	Integer
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	10
Override Example	`$sugar_config['rest_response_etag_cache_age'] = 15;`

roleBasedViews

Description	Enables role based views and dropdowns.
Type	Boolean
Range of values	true and false
Versions	7.6.0.0+
Products	Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['roleBasedViews'] = true;`

SAML_provisionUser

Description	Determines whether or not a new user is auto-provisioned when authenticating through SAML. Setting this setting to false will prevent a new user from being created.

Type	Boolean
Range of values	true and false
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	true
Override Example	`$sugar_config['SAML_provisionUser'] = false;`

SAML_X509Cert

Description	The SAML Certificate Key.
Type	String
Versions	6.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['SAML_X509Cert'] = '-----BEGIN CERTIFICATE-----CERTIFICATE KEY-----END CERTIFICATE-----';`

schedule_report_with_chart

Description	By default, reports distributed via the Reports Schedule module in 8.1 and later will not include a chart in the emailed PDF even if one has been configured for display in Sugar. Because the report charts are not re-built when the scheduler runs a scheduled report, the chart may not accurately represent the data presented in the PDF. Instead, the chart will represent the data that was pulled the last time a user triggered a "Run Report" action on the report, which may or may not match the data available when the report is generated for the scheduled report. Consider this potential conflict carefully before using this config to override the default behavior.

Type	Boolean
Range of values	true and false
Versions	8.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['schedule_report_with_chart'] = true;`

search_engine

Description	Array that defines the search enginer behaviors.
Type	Array
Versions	7.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['search_engine'] = array();`

search_engine.max_bulk_delete_threshold

Description	The maximun number of records that can be deleted at a time.
Type	Integer
Versions	7.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	3000
Override Example	`$sugar_config['search_engine']['max_bulk_delete_threshold'] = 3000;`

search_engine.max_bulk_query_threshold

Description	The maximum number of records to process before starting to bulk insert. Prevents memory issues.
Type	Integer

Versions	7.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	15000
Override Example	`$sugar_config['search_engine']['max_bulk_query_threshold'] = 20000;`

search_engine.max_bulk_threshold

Description	The maximum number of records to process before starting to bulk insert. Prevents memory issues.
Type	Integer
Versions	7.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	5000
Override Example	`$sugar_config['search_engine']['search_engine.max_bulk_threshold'] = 10000;`

search_engine.postpone_job_time

Description	Amount of time to postpone a job by so that it's not executed twice during the same request.
Type	Integer
Versions	7.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	120
Override Example	`$sugar_config['search_engine']['search_engine.postpone_job_time'] = 70;`

search_wildcard_infront

Description	In Sugar 7.x+, this setting is only valid for modules running in BWC mode.

	When enabled, automatically adds a wildcard in front of any searches performed in the application. This setting is not recommended to be enabled as preceding wildcards results in database indices not being utilized and performance decreasing.
Type	Boolean
Range of values	true and false
Versions	6.4.3+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['search_wildcard_infront'] = true;`

session_dir

Description	Directory on the server to store Sugar session data. If left empty, the PHP session settings will be inherited.
Type	String
Range of values	Directory path
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	empty
Override Example	`$sugar_config['session_dir'] = '/tmp/SugarSession/';`

showThemePicker

Description	Removes the theme selection drop down.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate,

	Serve, Sell
Default Value	true
Override Example	`$sugar_config['showThemePicker'] = false;`

show_download_tab

Description	Used to determine whether the download tab will appear in the User settings. The download tab provides users with access to Sugar plug-ins and other available downloads.
Type	Boolean
Range of values	true and false
Versions	6.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['show_download_tab'] = true;`

site_url

Description	Current URL of your Sugar instance. This value is critical in its accuracy for multiple points of functionality in the instance.
Type	String
Range of values	Current URL of Sugar
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['site_url'] = 'http://my.sugarinstance.com';`

slow_query_time_msec

Description	Slow query time threshold.
Type	String

Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	5000
Override Example	`$sugar_config['slow_query_time_msec'] = '1000';`

smtp_mailer_debug

Description	The smtp_mailer_debug is used for increasing the debugging output for PHPMailer on individual instances so that more information can be seen when there are bugs or escalations. The default is 0 and produces no additional output for errors. 1-4 log the additional error information to sugarcrm.log as specified by PHPMailer at https://github.com/PHPMailer/PHPMailer/wiki/SMTP-Debugging#debug-levels .
Type	Integer
Range of values	0, 1, 2, 3, 4
Versions	7.9.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['smtp_mailer_debug'] = 4;`

stack_trace_errors

Description	Displays stack trace of errors.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['stack_trace_errors'] = true;`

studio_max_history

Description	When layout changes are made in Studio, a history of those changes are recorded with each save & deploy under ./custom/history/modules/. The studio_max_history parameter controls how many files Sugar keeps for a particular module. If this parameter is undefined, a default of 50 is observed and to keep all historical Studio actions, set this parameter to 0.
Type	Integer
Range of values	Any integer
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	50
Override Example	`$sugar_config['studio_max_history'] = 100;`

sugar_version

Description	The current version of Sugar that is being used. This value should not be modified as it is updated in the config when Sugar is upgraded.
Type	String
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['sugar_version'] = '6.7.3';`

time_aware_job_max_batch_size

Description	The number of records processed per batch by the "Process Time-Aware

	Schedules" job scheduler. It is important to note that the scheduler job must be enabled.
Type	Integer
Versions	10.3.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	100
Override Example	`$sugar_config['time_aware_job_max_batch_size'] = 200;`

tmp_dir

Description	The directory path for temporary XML files used by charts and diagnostics.
Type	String
Range of values	Filesystem path
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	cache/xml/
Override Example	`$sugar_config['tmp_dir'] = 'cache/xml/';`

tmp_file_max_lifetime

Description	The interval in seconds of when to expire and remove temporary upload files. It is important to note that the "Remove temporary files" scheduler job must be enabled to remove the temporary files.
Type	Integer
Versions	7.6.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	86400
Override Example	`$sugar_config['tmp_file_max_lifeti`

	`me'] = 86400;`

tracker_max_display_length

Description	The number of records that will be shown per record in the "Last Viewed" section located under each module tab.
Type	Integer
Versions	6.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['tracker_max_display_length'] = 45;`

tracker_module_config

Description	Array that defines the parameters that control which modules are tracked when Tracker Actions are enabled.
Type	Array
Versions	11.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['tracker_module_config'] = array();`

tracker_module_config.disable

Description	The list of modules that are ignored by Tracker Actions. If tracker_module_config.enable_only is defined, then tracker_module_config.disable will be ignored. If neither config is defined, then all modules are enabled. By default, neither config is defined and all actions for all modules across all platforms will be enabled when Tracker Actions is enabled.
Type	Array

Range of values	Module keys
Versions	11.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['tracker_module_config']['disable'] = ['Reports', 'ActivityStream'];`

tracker_module_config.enable_only

Description	A list of module names for which records are saved in the tracker table when Tracker Actions are enabled; all other modules will be disabled. If tracker_module_config.enable_only is empty or null or undefined, tracker_module_config.disable will apply. If neither config is defined, then all modules are enabled. By default, neither config is defined and all actions for all modules across all platforms will be enabled when Tracker Actions is enabled.
Type	Array
Range of values	Module keys
Versions	11.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['tracker_module_config']['enable_only'] = ['Reports', 'ActivityStream'];`

uninstall_timeout

Description	Controls the timeout time in seconds for an uninstall process.
Type	Integer
Versions	10.2.1+
Products	Professional, Enterprise, Ultimate, Serve, Sell

Default Value	600
Override Example	`$sugar_config['uninstall_timeout'] = 300;`

unique_key

Description	Specifies the unique identifier for the instance. This value is used in features such as PHP caching, FTS indexing, and email archiving. It is extremely important that this string is unique from any other instances deployed even if they are only for development purposes only.
Type	String
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['unique_key'] = 'c0b5475f3f5b26ddb2976edc8865b5f6';`

upload_badext

Description	An array of extensions that cannot be uploaded in their native file format. Sugar will append a .txt extension to the end of any files with an invalid extension to avoid security issues with running unauthorized scripts on an instance.
Type	Array
Range of values	Extensions that cannot be uploaded as is
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['upload_badext'][] = 'swf';`

upload_dir

Description	The directory path where uploaded files are stored for note attachments, documents, and module loadable packages. By default, uploads are stored in the ./upload/ directory. For more information, refer to the Uploads documentation.
Type	String
Range of values	Directory path
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	upload/
Override Example	`$sugar_config['upload_dir'] = 'upload/'`

upload_maxsize

Description	The maximum individual file size that users can upload to modules that support file uploads. Restricts the upload limit from within Sugar without affecting any other applications that may be running on your server. This limit can be easily adjusted in Sugar via Admin > System Settings but is only useful if it is set to a size smaller than the php.ini limits. For more information, refer to the Uploads documentation.
Type	Integer
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['upload_maxsize'] = 40000000;`

upload_wrapper_class

Description	The name of the class used as upload wrapper.

Type	String
Versions	6.7.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	UploadStream
Override Example	`$sugar_config['upload_wrapper_class'] = 'SugarUploadS3';`

use_common_ml_dir

Description	A security control that allows you to restrict Module Loader to read modules from a specific directory on the server and disable the ability to upload new modules into the Module Loader. To specify a new directory you will need to populate the config parameter 'common_ml_dir'.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['use_common_ml_dir'] = true;`

use_php_code_json

Description	Determines if the environment has a valid version of PHP-JSON. This should be determined by the function returnPhpJsonStatus() and shouldn't be overridden.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell

Override Example	`$sugar_config['use_php_code_json'] = true;`

use_real_names

Description	Display users' full names instead of their User Names in assignment fields.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	true
Override Example	`$sugar_config['use_real_names'] = true;`

use_sprites

Description	A sprite is a two-dimensional image or animation that is integrated into a larger scene. This parameter is used to disable sprites. This is set to true by default (if you have GD libraries installed).
Type	Boolean
Range of values	true and false
Versions	6.4.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['use_sprites'] = false;`

validation

Description	An array that defines settings for user input validation behaviors.
Type	Array
Versions	7.7.1.0+

Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['validation'] = array();`

validation.compat_mode

Description	Determines compatibility mode for superglobals in the input validation framework. When enabled, setting, unsetting, and modifying $_GET, $_POST, or $_REQUEST values through PHP code is supported. Using PHP code to manipulate superglobals is discouraged. User input parameters should be considered read-only. As a transition period, this is currently allowed out-of-box, but may change in a future release. It is highly recommended for developers to verify their customizations work with this parameter set to false in preparation for the 7.9 release.
Type	Boolean
Range of values	true and false
Versions	7.7.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	7.7.x: true 7.8.x: true 7.9.x: false
Override Example	`$sugar_config['validation']['compat_mode'] = false;`

validation.soft_fail

Description	Determines whether soft failure mode for the input validation framework is enabled. When this mode is enabled, any input validation violations will only be reported as a warning in the sugarcrm.log without having any

	negative impact on the request. When disabled, violations are reported as fatal in the sugarcrm.log and actual exceptions are being thrown resulting in an HTTP 500 response. It is highly recommended for developers to verify their customizations work with this parameter set to false in preparation for the 8.0 release.
Type	Boolean
Range of values	true and false
Versions	7.7.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	7.7.x: true 7.8.x: true 7.9.x: true 8.0.x: false
Override Example	`$sugar_config['validation']['soft_fail'] = false;`

verify_client_ip

Description	Whether or not to verify the client IP. Setting this to true will enable the system checking to see if the user is accessing Sugar from the IP address of their last page load. **Note**: This default value must be set to "false" in order to use Sugar Connect and Sugar Integrate.
Type	Boolean
Range of values	true and false
Versions	10.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['verify_client_ip']`

	`= true;`

web_logic_hook_timeout

Description	Timeout for requests to web logic hooks. Any requests that run longer than the set limit will fail and trigger an error in the log.
Type	Integer
Versions	11.1.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	10
Override Example	`$sugar_config['web_logic_hook_timeout'] = 2;`

wl_list_max_entries_per_page

Description	The number of records to be shown per page on the listview of the mobile browser.
Type	Integer
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	10
Override Example	`$sugar_config['wl_list_max_entries_per_page'] = 10;`

wl_list_max_entries_per_subpanel

Description	Determines the number of records shown in the subpanels on the DetailView of the mobile browser.
Type	Integer
Versions	5.2.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell

Default Value	3
Override Example	`$sugar_config['wl_list_max_entries_per_subpanel'] = 3;`

xhprof_config

Description	Configuration settings for xhprof. More information on xhprof can be found at http://pecl.php.net/package/xhprof.
Type	Array
Versions	6.5.10+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['xhprof_config'] = array();`

xhprof_config.enable

Description	Enables the xhprof profiler.
Type	Boolean
Range of values	true and false
Versions	6.5.10+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	false
Override Example	`$sugar_config['xhprof_config']['enable'] = true;`

xhprof_config.filter_wt

Description	The wall time. Values are specified in milliseconds.
Type	Integer
Versions	7.6.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['xhprof_config']['filter_wt'] = 2;`

xhprof_config.flags

Description	The flags for xhprof profiler.
Type	String
Versions	6.5.10+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['xhprof_config']['flags'] = XHPROF_FLAGS_CPU + XHPROF_FLAGS_MEMORY;`

xhprof_config.ignored_functions

Description	An array of function names to ignore from the profile.
Type	Array
Versions	6.5.10+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['xhprof_config']['ignored_functions'] = array("function_name");`

xhprof_config.log_to

Description	The path to log the xhprof profiler output to.
Type	String
Versions	6.5.10+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Override Example	`$sugar_config['xhprof_config']['log_to'] = '{instance server path}/cache/xhprof';`

xhprof_config.manager

Description	The xhprof manager class to use. Prior

	to 7.7, specifying values xhprof_config.save_to, xhprof_config.mongodb_uri, xhprof_config.mongodb_db, xhprof_config.mongodb_collection, xhprof_config.mongodb_options, and xhprof_config.filter_wt will need to have set xhprof_config.manager set to 'SugarXHprofPerformance'. As of 7.7, setting xhprof_config.manager is not longer required. If you want to customize SugarXHprof, you can create the folder ./custom/include/SugarXHprof/ and create a file with your custom class class name. The custom class will need to extend SugarXHprof. If a custom class doesn't exist or hasn't been specified, SugarXHprof will be used.
Type	String
Versions	6.5.10+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	SugarXHprof
Override Example	`$sugar_config['xhprof_config']['manager'] = 'CustomSugarXHprof';`

xhprof_config.memory_limit

Description	The memory limit to set while saving profile data to storage.
Type	String
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	2048M
Override Example	`$sugar_config['xhprof_config']['memory_limit'] = '1024M';`

xhprof_config.mongodb_collection

Description	The name of the mongo db collections.
Type	String
Versions	7.6.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	results
Override Example	`$sugar_config['xhprof_config']['mongodb_db'] = 'results_new';`

xhprof_config.mongodb_db

Description	The name of the mongo database.
Type	String
Versions	7.6.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	xhprof
Override Example	`$sugar_config['xhprof_config']['mongodb_db'] = 'xhprof2';`

xhprof_config.mongodb_options

Description	Options for mongo db connection. The list of construct options can be found at: http://php.net/manual/en/mongoclient.construct.php#mongo.mongoclient.construct.parameters
Type	Array
Versions	7.6.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	results
Override Example	`$sugar_config['xhprof_config']['mongodb_options'] = array();`

xhprof_config.mongodb_uri

Description	The mongo server URL.

Type	String
Versions	7.6.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	mongodb://localhost:27017
Override Example	`$sugar_config['xhprof_config']['mongodb_uri'] = 'mongodb://localhost:27018';`

xhprof_config.sample_rate

Description	The sample rate of the xhprof profiler. 1/{specified value} requests are profiled. To sample all requests, set this value to 1.
Type	Integer
Versions	6.5.10+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	10
Override Example	`$sugar_config['xhprof_config']['sample_rate'] = 1;`

xhprof_config.save_to

Description	The place to save xhprof data. If set to 'mongodb', the data is stored in the mongo database.
Type	String
Range of values	'file' and 'mongodb'
Versions	7.6.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	mongodb
Override Example	`$sugar_config['xhprof_config']['save_to'] = 'mongodb';`

xhprof_config.track_elastic

Description	Whether or not to track elastic data.
Type	Boolean
Range of values	true or false
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	true
Override Example	`$sugar_config['xhprof_config']['track_elastic'] = false;`

xhprof_config.track_elastic_backtrace

Description	Whether or not to store the elastic backtrace data.
Type	Boolean
Range of values	true or false
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	true
Override Example	`$sugar_config['xhprof_config']['track_elastic_backtrace'] = false;`

xhprof_config.track_sql

Description	Whether or not to track SQL queries.
Type	Boolean
Range of values	true or false
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	true
Override Example	`$sugar_config['xhprof_config']['track_sql'] = false;`

xhprof_config.track_sql_backtrace

Description	Whether or not to store the SQL backtrace data.
Type	Boolean
Range of values	true or false
Versions	7.7.0.0+
Products	Professional, Enterprise, Ultimate, Serve, Sell
Default Value	true
Override Example	`$sugar_config['xhprof_config']['track_sql_backtrace'] = false;`

Silent Installer Settings

Overview

The Sugar silent installer facilitates and automates the installation of the Sugar application after the files have been copied onto the server. This is done by populating correct parameters in a configuration file and then making a web request to kick off the installation.

config_si.php

The ./config_si.php file is located at the root level of the Sugar application. It contains an array of name-value pairs consisting of the relevant parameters for installation of the app. The array name is $sugar_config_si. An example file looks like:

```
  <?php

$sugar_config_si = array (
    'setup_site_url' => 'http://${domainname}:${webport}/sugar',
    'setup_system_name' => '${systemname}',
    'setup_db_host_name' => 'localhost',
    'setup_site_admin_user_name' => 'admin',
    'setup_site_admin_password' => '${sugarpassword}',
    'demoData' => true,
    'setup_db_type' => 'mysql',
    'setup_db_host_name' => '{db_host_name}',
    'setup_db_port_num' => 'db_port_number',
    'setup_db_database_name' => 'sugar',
    'setup_db_admin_user_name' => 'root',
```

```
    'setup_db_admin_password' => '${rootpassword}',
    'setup_db_options' => array(
     'ssl' => true,
    ),
    'setup_db_drop_tables' => false,
    'setup_db_create_database' => true,
    'setup_license_key' => '${slkey}',
    'setup_license_key_users' => '${slkeyusers}',
    'setup_license_key_expire_date' => '${slkeyexpiredate}',
    'setup_license_key_oc_licences' => '${slkey_oc_licenses}',
    'default_currency_iso4217' => 'USD',
    'default_currency_name' => 'US Dollars',
    'default_currency_significant_digits' => '2',
    'default_currency_symbol' => '$',
    'default_date_format' => 'Y-m-d',
    'default_time_format' => 'H:i',
    'default_decimal_seperator' => '.',
    'default_export_charset' => 'ISO-8859-1',
    'default_language' => 'en_us',
    'default_locale_name_format' => 's f l',
    'default_number_grouping_seperator' => ',',
    'export_delimiter' => ',',
);
```

Essential Settings

The following are settings that must be set:

General System Settings

setup_system_name

Description	A unique system name for the application that will be displayed on the web browser
Type	String

setup_site_url

Description	The site location and URL of the Sugar application
Type	String

setup_site_admin_user_name

Description	Specifies the admin username for the Sugar application
Type	String

setup_site_admin_password

Description	Specifies the admin password for the Sugar application
Type	String

demoData

Description	Indicates whether the app will be installed with demo data
Type	Boolean

Database Settings

setup_db_type

Description	Defines the type of database being used with Sugar. It is important to note that db2 and oracle are only applicable to Sugar Ent and Ult.
Type	String: valid options include mysql, mssql, db2, oracle

setup_db_host_instance

Description	Defines the host instance for MSSQL connections.
Type	String : Database Host Instance Name

setup_db_host_name

Description	Defines the hostname of the database server.
Type	String

setup_db_port_num

Description	Defines the port number on the server to connect to for authentication and transactions.
Type	String

setup_db_database_name

Description	Defines the database name to connect to on the database server.
Type	String

setup_db_admin_user_name

Description	Specifies the database administrator's username
Type	String

setup_db_admin_password

Description	Specifies the database administrator's password
Type	String

Recommended Database Settings

The following are not necessarily required, but it is recommended that at least one is set:

setup_db_create_database

Description	Specifies whether Sugar will create a new database with the name given or use an existing database.
Type	Boolean

setup_db_drop_tables

Description	Specifies whether Sugar will drop existing tables on a database if the database already exists
Type	Boolean

Extended Database Settings

There is also an option for an additional array of db config settings. These array entries are equivalent to the Core Settings options prefixed with dbconfigoption. For example dbconfigoption.collation and dbconfigoption.ssl

setup_db_options

Description	See: Architecture/Configurator/Core_Settings/index.html#dbconfigoptionautofree
Type	Array

Full-Text Search (ElasticSearch) Settings

setup_fts_type

Description	The Full-Text Search service type
Type	String, currently only supported value Elastic

setup_fts_host

Description	The hostname of the Full-Text Search service
Type	String

setup_fts_port

Description	The port number of the Full-Text Search service
Type	String

Advanced Full-Text Search Configuration Settings

While not required, the Silent Installer offers a few FTS settings not available in the browser-based installation process. These settings are considered more advanced and should not be implemented casually. Further details on advanced ElasticSearch configuration can be found in the ElasticSearch installation guide in the section Advanced Configuration.

setup_fts_curl

Description	Additional settings for the cURL request to the Elasticsearch server, keyed by the PHP cURL constants used for curl_setopt. For example `'setup_fts_curl' => array( CURLOPT_SSL_VERIFYPEER = false,),`
Type	Array

License Settings

While not required during installation, the license settings are required for Sugar to be usable by all users. Setting the license settings during the Silent Install will save the need to enter this required data after the install.

setup_license_key

Description	Specifies the license key
Type	String

setup_license_key_users

Description	Specifies the number of license key users
Type	Integer

setup_license_key_expire_date

Description	The expiration date of the license key
Type	String: yyyy-mm-dd format

setup_site_sugarbeet_automatic_checks

Description	Specifies if License Validation checks should be set to Automatic
Type	Boolean

Additionally, Offline Client license settings can be set in Silent Installer

setup_license_key_oc_licences

Description	the number of offline client users
Type	Integer

setup_num_lic_oc

Description	the number of offline client users
Type	Integer

Making the Web Request

After the ./config_si.php file is in place, the following web request can be made to kick off the installation:

```
http://${hostname}/sugar/install.php?goto=SilentInstall&cli=true
```

where ${hostname} is the name of the host. Upon completion of the process, the installation should respond with Success, and you may navigate to the web address to find an installed version of Sugar.

Configuring a Sugar-Specific Database User during Install

By default, Sugar will use the database user set for the install process as the database user for general use after installation. There are three different Silent Install options for having Sugar use a different database user once installed. Which option the installer users is determined by the dbUSRData config setting.

The dbUSRData silent install config setting has four valid options:

same

This is the default setting used, even when dbUSRData is not explicitly set. The DB user provided for installing Sugar is used for running Sugar after installation.

auto

Sugar will create a new database user using a self-generated username and password.

In your $sugar_config_si array, the options for this to work as expected are:

```
  'dbUSRData' => 'auto',
'setup_db_create_sugarsales_user' => true,
```

provide

Sugar will use the provided database username and password after installation, and the installer assumes that the database user has already been created on the database and that the provided password is valid. Essentially this is informing Sugar to use a different existing DB user after the installation has completed.

In your $sugar_config_si array, the options for this to work as expected are:

```
  'dbUSRData' => 'provide',
'setup_db_create_sugarsales_user' => false,
'setup_db_sugarsales_user' => '{existing_db_user_name}',
'setup_db_sugarsales_password' => '{existing_db_user_password}',
'setup_db_sugarsales_password_retype' => '{existing_db_user_password}'
```

,

create

Sugar will create a new database user using the provided database username and password after installation. This is similar to 'provide', except that Sugar assumes that it will be creating the database user during installation.

In your $sugar_config_si array, the options for this to work as expected are:

```
  'dbUSRData' => 'create',
'setup_db_create_sugarsales_user' => true,
'setup_db_sugarsales_user' => '{new_db_user_name}',
'setup_db_sugarsales_password' => '{new_db_user_password}',
'setup_db_sugarsales_password_retype' => '{new_db_user_password}',
```

Note: For create, the installer will create a user with access only to the database created for this Sugar instance. It will also have limited access for which host it can connect from, versus the admin user, which usually has access from any host.

Sugar will set the user to have access to localhost and the provided hostname derived from setup_site_url. If there are issues connecting to the database during or after installation, check that the user was created correctly in the database and that the user can connect from the host that the instance is on.

Other Silent Install Config Options

Many of these options match an equivalent Sugar Config option, with further descriptions available at Core Settings.

cache_dir

Description	This is the directory Sugar will store all cached files. Can be relative to Sugar root directory.
Type	String, default "cache/"

Locale Settings

The following options are locale settings. With one exception, these can be set or updated through Administration > Locale.

export_delimiter

Description	The field delimiter (separator) used when exporting records as CSV
Type	String, default ","

default_export_charset

Description	The default character set for CSV files to be encoded in when exporting
Type	String, default "ISO-8859-1"

default_currency_iso4217

Description	The ISO-4217 code of the default currency
Type	String, default "USD"

default_currency_name

Description	The name to use for the default currency
Type	String, default "US Dollars"

default_currency_significant_digits

Description	Changes the number of significant digits in currency by default. Note that this setting can not be changed through the Admin panel after installation. It can be set for each user through the user's profile or can be updated globally by updating the config array.
Type	Integer, default 2

default_currency_symbol

Description	The symbol to use for the default currency
Type	String, default "$"

default_language

Description	Sets each user's default language.

	Possible values include any language offered by Sugar, such as: 'ar_SA', 'bg_BG', 'ca_ES', 'cs_CZ', 'da_DK', 'de_DE', 'el_EL', 'en_UK', 'en_us', 'es_ES', 'es_LA', 'zh_CN'
Type	String, default "en_US"

default_date_format

Description	Modifies the default date format for all users.
Type	String, default "m/d/Y"

default_time_format

Description	Modifies the default time format for all users.
Type	String, default "H:i"

default_decimal_seperator

Description	Sets the character used as a decimal separator for numbers.
Type	String, default "."

default_number_grouping_seperator

Description	Sets the character used as the 1000s separator for numbers.
Type	String, default ","

default_locale_name_format

Description	Sets the format for displaying full name (eg "Salutation FirstName LastName" vs "LastName, FirstName"
Type	String, default "s f l"

Module Loader

Module Loader is used when installing customizations, plugins, language packs, hotfixes, and other customizations into a Sugar instance in the form of a Module Loadable Package. This documentation covers the basics and best practices for creating module loadable packages for a Sugar installation.

Note: Sugar Sell Essentials customers do not have the ability to upload custom file packages to Sugar using Module Loader.

Introduction to the Manifest

Overview

Module loadable packages rely on a manifest.php file to define the basic properties and installation steps for the package. This documentation explains the various components that make up the manifest file.

Note: Sugar Sell Essentials customers do not have the ability to upload custom file packages to Sugar using Module Loader.

Manifest Definitions

Inside of the manifest.php file, there is a $manifest variable that defines the basic properties of the module loadable package. The various manifest properties are outlined below:

Name	Type	Displayed in Module Loader	Description
key	String	No	A unique identifier for the package `$manifest['key'] = '32837';`
name	String	Yes	The name of the package `$manifest['name'] = 'Example Package';`
description	String	Yes	The description of the package `$manifest['description'] = 'Example Package Desc`

			`ription';`
built_in_version	String	No	The version of Sugar that the package was designed for `$manifest['built_in_version'] = '14.1.0';` **Note**: Some packages designed for 6.x versions of Sugar are not compatible with 12.0 and should not be installed.
version	String	Yes	The version of the package, i.e. "1.0.0" `$manifest['version'] = '1.0.0';` **Note:** The version string should be formatting according to the Semantic Versioning 2.0 standard.
acceptable_sugar_versions	Array	No	The Sugar versions that a package can be installed to `$manifest['acceptable_sugar_versions'] = array( 'exact_matches' => array( '14.1.0',`

			), //or 'regex_matches' => array('12.0.*',),); **Note**: You can define exact versions and/or use a regex formula to define a range. Exact versions will be specified using the exact_matches index and will contain an array of exact version strings (i.e. '14.1.0'). Regex formulas will be specified using the regex_matches index and will contain an array of regular expressions designed to match a group of versions (i.e. '12.0.*').
acceptable_sugar_flavors	Array	No	The Sugar products that the package can be installed to `$manifest['acceptable_sugar_flavors'] = array(  'PRO',    'ENT',    'ULT');`
author	String	No	The author of the package (i.e. "SugarCRM") `$manifest['author'] = 'SugarCRM'`

			;
readme	String	No	The optional path to a readme document to be displayed to the user during installation `$manifest['readme'] = 'README.txt';`
icon	String	No	The optional path (within the package ZIP file) to an icon image to be displayed during installation (e.g. ./patch_directory/icon.gif and ./patch_directory/images/theme.gif) `$manifest['icon'] = '';`
is_uninstallable	Boolean	No	Whether or not the package can be uninstalled Acceptable values: • 'true' will allow a Sugar administrator to uninstall the package • 'false' will disable the uninstall feature `$manifest['is_uninstallable'] =`

			`true;`
published_date	String	No	The date the package was published `$manifest['published_date'] = '2018-04-09 00:00:00';`
remove_tables	String	No	Whether or not tables generated by the $installdefs['beans'] index should be removed from an installed module (acceptable values: empty or 'prompt') `$manifest['remove_tables'] = 'prompt';`
type	String	No	Acceptable 'type' values: • **module** : Use this type if you are installing a module or developing a plugin that should install via the Module Loader. • **langpack** : Use this type to install a language

			pack via the Module Loader. Any languages installed will automatically be added to the available languages on the Sugar login screen. For an example of a module loadable language pack, refer to the Language Framework page. • **theme** : Use this type to install a Sugar theme via the Upgrade Wizard. Themes are only supported in Sugar 6.x, and will be added to the "Theme" drop-down on the Sugar Login screen. • **patch** : Use this type to install a

			patch via the Upgrade Wizard.
dependencies	Array	No	Required dependency packages: `$manifest['dependencies'] = array( array( 'id_name' => 'PackageName', 'version' => '1.0.0' ) );`
uninstall_before_upgrade	Boolean	No	This will allow Module Loader to ensure all traces of previously installed package versions (including packages that have been upgraded multiple times) have been removed. `$manifest['uninstall_before_upgrade'] = true;`

Manifest Example

An example of a manifest is shown below:

```
$manifest = array(
'key' => 1397052912,
'name' => 'My manifest',
'description' => 'My description',
'author' => 'SugarCRM',
'version' => '1.0.0',
'is_uninstallable' => true,
'published_date' => '03/02/2024 14:15:12',
'type' => 'module',
```

```
  'acceptable_sugar_versions' =>
  array(
    'exact_matches' => array(
        '14.1.0'
    ),
    //or
    'regex_matches' => array(
        '12.0.*' //any 12.0 release
    ),
  ),
  'acceptable_sugar_flavors' =>
  array(
    'PRO',
    'ENT',
    'ULT'
  ),
  'readme' => '',
  'icon' => '',
  'remove_tables' => '',
  'uninstall_before_upgrade' => false,
);
```

Installation Definitions

The following section outlines the indexes specified in the $installdefs array contained in the ./manifest.php file. The $installdefs array indexes are used by the Module Loader to determine the actual installation steps that need to be taken to install the package.

$installdef Actions

Name	Type	Description
action_file_map	Array	ActionFileMap is part of the Extension Framework. More detail can be found in the Extension Framework documentation.
action_remap	Array	ActionReMap is part of the Extension Framework. More detail can be found in the Extension Framework

		documentation.
action_view_map	Array	ActionViewMap is part of the Extension Framework. More detail can be found in the Extension Framework documentation.
administration	Array	Administration is part of the Extension Framework. More detail can be found in the Extension Framework documentation.
appscheduledefs	Array	Application ScheduledTasks is part of the Extension Framework. More detail can be found in the Extension Framework documentation.
beans	Array	Modules is part of the Extension Framework. More detail can be found in the Extension Framework documentation.
connectors	Array	An array containing Connector definitions as outlined in the Connector documentation. `$installdefs['connectors'] = array ( array ( 'connector' => '<basepath>/example/source', 'formatter' => '<basepath>/example/formatter', 'name' => 'ext_rest_example', ),);`
copy	Array	An array detailing the files and folders to be copied to Sugar

		Required parameters for each file in the array:
		
		• **from** (string) : The location of the file in the module loadable package
		
		`$installdefs['copy'][0]['from'] = '<basepath>/Files/custom/modules/Accounts/accounts_save.php';`
		
		• **to** (string) : The destination directory relative to the Sugar root
		
		`$installdefs['copy'][0]['to'] = 'custom/modules/Accounts/accounts_save.php';`
		
		Example:
		
		`$installdefs['copy'] = array( array( 'from' => '<basepath>/Files/custom/modules/Accounts/accounts_save.php', 'to' => 'custom/modules/Accounts/accounts_save.php', ),);`

csp	Array	CSP directives needed for proper operation of the package should be represented as an

		associative array where keys are directive names and values are corresponding trusted sources. Example: `$installdefs = array ( ... 'csp' => [ 'default-src' => '*.trusted.com example.com', 'img-src' => 'http: https:' ... ] ... ) ;` This parameter takes a space-delimited string of domains - just like the admin module. Setting the value of the default-src via MLP is an append operation. The values you put into the csp parameter of your manifest will be added to the CSP list in the admin module. Note that uninstall DOES NOT remove a value.
custom_fields	Array	An array of custom fields to be installed for the new moduleRequired sub-directives for each custom field formatted as: `$installdefs['custom_fields'][0]['<attribute>'] = <value>;` • **name** (String) : The internal name

		of the custom field **Note**: The suffix "_c" is appended to all custom field names (e.g. fieldname_c). • **label** (String) : The display label for the custom field • **type** (String) : The type of custom field (e.g. varchar, text, textarea, double, float, int, date, bool, enum, relate) • **max_size** (Integer) : The custom field's maximum character storage size • **require_option** (String) : Defines whether fields are 'required' or 'optional' • **default_value** (String) : The default value for the custom field • **ext1** (String) : Used to specify a drop-down name for enum and multienum type fields • **ext2** (String) : Not used • **ext3** (String) : Not used • **audited** (Boolean) : Denotes whether or not the custom field should be audited • **module** (String) :

		The module where the custom field will be added Example: `$installdefs['custom_fields'] = array(( array(( 'name' => 'text_c', 'label' => 'LBL_TEXT_C', 'type' => 'varchar', 'max_size' => 255, 'require_option' => 'optional', 'default_value' => '', 'ext1' => '', 'ext2' => '', 'ext3' => '', 'audited' => true, 'module' => 'Accounts', ),);`
console	Array	Console is part of the Extension Framework. More detail can be found in the Extension Framework documentation.
dashlets	Array	An array containing the Dashlet definition. Required parameters for each file in the array: • **name** (string) : The name of the dashlet. Used for the folder name where the Dashlet files will be stored in Sugar file system. • **from** (string) : The location of the

		dashlet files in the module loadable package. `$installdefs['dashlets'] = array( array( 'name' => 'MyDashlet', 'from' => '<basepath>/MyDashlet' ));`
dependencies	Array	Dependencies is part of the Extension Framework. More detail can be found in the Extension Framework documentation.
entrypoints	Array	EntryPointRegistry is part of the Extension Framework. More detail can be found in the Extension Framework documentation.
extensions	Array	Extensions is part of the Extension Framework. More detail can be found in the Extension Framework documentation.
file_access	Array	FileAccessControlMap is part of the Extension Framework. More detail can be found in the Extension Framework documentation.
hookdefs	Array	LogicHooks is part of the Extension Framework. More detail can be found in the Extension Framework documentation.
id	String	A unique id for the installdef definition

		`$installdefs['id'] = 'unique_name';`
image_dir	String	The directory that contains the icons for the module `$installdefs['image_dir'] = '<basepath>/icons';`
jsgroups	Array	JSGroupings is part of the Extension Framework. More detail can be found in the Extension Framework documentation.
language	Array	Language is part of the Extension Framework. More detail can be found in the Extension Framework documentation.
layoutdefs	Array	Layoutdefs is part of the Extension Framework. More detail can be found in the Extension Framework documentation.
layoutfields	Array	An array of custom fields to be added to the edit and detail views of the target modules' existing layouts
logic_hooks	Array	An array containing full logic Hook definitions, as outlined in the LogicHook documentation. `$installdefs['logic_hooks'] = array( array( 'module' => 'Accounts', 'hook' => 'before_save', 'order'`

		=> 99, 'description' => 'Example Logic Hook - Logs account name', 'file' => 'custom/modules/Accounts/accounts_save.php', 'class' => 'Accounts_Save', 'function' => 'before_save',),); LogicHooks defined for install using the $installdefs['logic_hooks'] index, will be added to the module or application folder in the custom directory, in the logic_hooks.php file. Examples: ./custom/modules/Accounts/logic_hooks.php ./custom/application/logic_hooks.php **Note**: You will still need to utilize a $installdefs['copy'] index to move the LogicHook class file into the system.
platforms	Array	Platforms is part of the Extension Framework. More detail can be found in the Extension Framework documentation.
pre_execute	Array	Executes logic from a file (or set of files) before a package is installed

		Example: `$installdefs['pre_execute'] = array( '<basepath>/pre_execute.php',);` Where the content of <basepath>/pre_execute.php is: `<?php//pre_execute logicecho 'pre_execute script ';`
post_execute	Array	Executes logic from a file (or set of files) after a package is installed Example: `$installdefs['post_execute'] = array( '<basepath>/post_execute.php',);` Where the content of <basepath>/post_execute.php is: `<?php//post_execute logicecho 'post_execute script ';`
pre_uninstall	Array	Executes logic from a file (or set of files) before a package is uninstalled Example: `$installdefs['pre_uninstall'] = array( '<`

		basepath>/pre_uninstall.php',); Where the content of <basepath>/pre_uninstall.php is: <?php//pre_uninstall logicecho 'pre_uninstall script ';
post_uninstall	Array	Executes logic from a file (or set of files) after a package is uninstalled Example: $installdefs['post_uninstall'] = array('<basepath>/post_uninstall.php'); Where the content of <basepath>/post_uninstall.php is: <?php//post_uninstall logicecho 'post_uninstall script ';
relationships	Array	An array of relationship files used to link the new modules to existing modules **Note**: A metadata path must be defined using **meta_data** (string): $installdefs['relationships'][0]['meta_data'] = '<basepath>/SugarModules/relationships/relationships/my_module_accountsMetaData.php'

;

Where the content of my_module_accountsMetaData.php is:

```
<?php$dictionary['my_module_accounts'] = array (    'true_relationship_type' => 'many-to-many',    'relationships' =>        array((            'my_module_accounts' =>       array((            'lhs_module' => 'my_Module',                'lhs_table' => 'my_module',                'lhs_key' => 'id',             'rhs_module' => 'Accounts',                'rhs_table' => 'accounts',                'rhs_key' => 'id',            'relationship_type' => 'many-to-many',    'join_table' => 'my_module_accounts_c',                'join_key_lhs' => 'my_module_accountsmy_module_ida',   'join_key_rhs' => 'my_module_accountsaccounts_idb',    ),        ),    'table' => 'my_module_accounts_c',    'fields' =>        array((        0 =>      array((
```

```
            'name' => '
id',
  'type' => 'varchar',
                    'l
en' => 36,
    ),            1 =>
                array(
(                    '
name' => 'date_modifie
d',
 'type' => 'datetime',
                ),
        2 =>
      array((
            'name' => '
deleted',
       'type' => 'bool
',
'len' => '1',
            'default' =
> '0',
    'required' => true
,                ),
         3 =>
       array((
             'name' =>
'my_module_accountsmy_
module_ida',
          'type' => 'v
archar',
      'len' => 36,
             ),
    4 =>
  array((
       'name' => 'my_m
odule_accountsaccounts
_idb',
    'type' => 'varchar
',
'len' => 36,
      ),        ),
'indices' =>        ar
ray((            0 =>
               array((
                    'n
ame' => 'my_module_acc
```

		ountsspk', 'type' => 'pri mary', 'fields' => array ((0 => 'id',),), 1 => array(('name' => ' my_module_accounts_alt ', 'type' => 'alternate_k ey', 'fields' => array((0 => 'my_module_ accountsmy_module_ida' , 1 => 'my_module _accountsaccounts_idb' ,),) ,),);
scheduledefs	Array	ScheduledTasks is part of the Extension Framework. More detail can be found in the Extension Framework documentation.
sidecar	Array	Sidecar is part of the Extension Framework. More detail can be found in the Extension Framework documentation.
tinymce	Array	TinyMCE is part of the Extension Framework. More detail can be found in the Extension Framework documentation.

user_page	Array	UserPage is part of the Extension Framework. More detail can be found in the Extension Framework documentation.
utils	Array	Utils is part of the Extension Framework. More detail can be found in the Extension Framework documentation.
vardefs	Array	Vardefs is part of the Extension Framework. More detail can be found in the Extension Framework documentation.
wireless_modules	Array	WirelessModuleRegistery is part of the Extension Framework. More detail can be found in the Extension Framework documentation.
wireless_subpanels	Array	WirelessLayoutDefs is part of the Extension Framework. More detail can be found in the Extension Framework documentation.

Note: Anything printed to the screen in the pre_execute, post_execute, pre_uninstall, or post_uninstall scripts will be displayed when clicking on the Display Log link.

Module Loader Restrictions

Overview

SugarCRM's hosting objective is to maintain the integrity of the standard Sugar functionality when we upgrade a customer instance and limit any negative impact our upgrade has on the customer's modifications. All instances hosted on Sugar's

cloud service have package scanner enabled by default. This setting is not configurable and all packages must pass the package scan for installation on Sugar's cloud environment. This includes passing all health checks.

Note: Sugar Sell Essentials customers do not have the ability to upload custom file packages to Sugar using Module Loader.

Access Controls

The Module Loader includes a Module Scanner, which grants system administrators the control they need to determine the precise set of actions that they are willing to offer in their hosting environment. This feature is available in all Sugar products. Anyone who is hosting Sugar products can take advantage of this feature, as well.

Enabling Package Scan

Scanning is disabled in default installations of Sugar and can be enabled through a configuration setting. This setting is added to ./config.php or ./config_override.php, and is not available to Administrator users to modify through the Sugar interface. Please note that this setting can only be managed on an on-site deployment and cannot be disabled for Sugar's cloud environment.

To enable Package Scan and its associated scans, add this setting to ./config_override.php:

```
$sugar_config['moduleInstaller']['packageScan'] = true;
```

There are two categories of access control in the Package Scan:

- File Scan
- Module Loader Actions

Enabling File Scan

By enabling Package Scan, File Scan will be performed on all files in the package uploaded through Module Loader. File Scan will be performed when a Sugar administrator attempts to install the package. Please note that these settings can only be managed on an on-site deployment. These settings are not permitted to be modified when hosted on Sugar's cloud service.

File Scan performs three checks:

- File extensions must be in the approved list of valid extension types.
- Files must not contain any suspicious classes.
- Files must not contain any suspicious function calls.

Please refer to the next three sections which outline the default requirements for the File Scan checks.

Valid Extension Types

File extensions must be in the approved list of valid extension types. The following extension types are valid by default:

- css
- gif
- hbs
- htm
- html
- jpg
- js
- md5
- pdf
- php
- png
- tpl
- txt
- xml

Denylisted Classes

Files must not contain any of the following classes that are considered suspicious by File Scan.

- All variable classes (i.e., $class()) are prohibited by default.
- The following classes are denylisted by default:
    - lua
    - pclzip
    - reflection
    - reflectionclass
    - reflectionexception
    - reflectionextension
    - reflectionfunction
    - reflectionfunctionabstract
    - reflectionmethod
    - reflectionobject
    - reflectionparameter
    - reflectionproperty

    - reflectionzendextension
    - reflector
    - splfileinfo
    - splfileobject
    - ziparchive

Denylisted Function Calls

Files must not contain any of the following function calls that are considered suspicious by File Scan.

- Variable functions (i.e., $func()) are prohibited by default.
- Backticks (`) are prohibited by File Scan.
- The following PHP functions are denylisted by default:
    - addfunction
    - addserver
    - array_diff_uassoc
    - array_diff_ukey
    - array_filter
    - array_intersect_uassoc
    - array_intersect_ukey
    - array_map
    - array_reduce
    - array_udiff
    - array_udiff_assoc
    - array_udiff_uassoc
    - array_uintersect
    - array_uintersect_assoc
    - array_uintersect_uassoc
    - array_walk
    - array_walk_recursive
    - call_user_func
    - call_user_func
    - call_user_func_array
    - call_user_func_array
    - chdir
    - chgrp
    - chmod
    - chroot
    - chwown
    - clearstatcache
    - construct
    - consume
    - consumerhandler
    - copy
    - copy_recursive

- create_cache_directory
- create_custom_directory
- create_function
- dir
- disk_free_space
- disk_total_space
- diskfreespace
- eio_busy
- eio_chmod
- eio_chown
- eio_close
- eio_custom
- eio_dup2
- eio_fallocate
- eio_fchmod
- eio_fchown
- eio_fdatasync
- eio_fstat
- eio_fstatvfs
- eio_fsync
- eio_ftruncate
- eio_futime
- eio_grp
- eio_link
- eio_lstat
- eio_mkdir
- eio_mknod
- eio_nop
- eio_open
- eio_read
- eio_readahead
- eio_readdir
- eio_readlink
- eio_realpath
- eio_rename
- eio_rmdir
- eio_sendfile
- eio_stat
- eio_statvfs
- eio_symlink
- eio_sync
- eio_sync_file_range
- eio_syncfs
- eio_truncate
- eio_unlink
- eio_utime

- eio_write
- error_log
- escapeshellarg
- escapeshellcmd
- eval
- exec
- fclose
- fdf_enum_values
- feof
- fflush
- fgetc
- fgetcsv
- fgets
- fgetss
- file
- file_exists
- file_get_contents
- file_put_contents
- fileatime
- filectime
- filegroup
- fileinode
- filemtime
- fileowner
- fileperms
- filesize
- filetype
- flock
- fnmatch
- fopen
- forward_static_call
- forward_static_call_array
- fpassthru
- fputcsv
- fputs
- fread
- fscanf
- fseek
- fstat
- ftell
- ftruncate
- fwrite
- get
- getbykey
- getdelayed
- getdelayedbykey

- getfunctionvalue
- getimagesize
- glob
- header_register_callback
- ibase_set_event_handler
- ini_set
- is_callable
- is_dir
- is_executable
- is_file
- is_link
- is_readable
- is_uploaded_file
- is_writable
- is_writeable
- iterator_apply
- lchgrp
- lchown
- ldap_set_rebind_proc
- libxml_set_external_entity_loader
- link
- linkinfo
- lstat
- mailparse_msg_extract_part
- mailparse_msg_extract_part_file
- mailparse_msg_extract_whole_part_file
- mk_temp_dir
- mkdir
- mkdir_recursive
- move_uploaded_file
- newt_entry_set_filter
- newt_set_suspend_callback
- ob_start
- open
- opendir
- parse_ini_file
- parse_ini_string
- passthru
- passthru
- pathinfo
- pclose
- pcntl_signal
- popen
- preg_replace_callback
- proc_close
- proc_get_status

- proc_nice
- proc_open
- readdir
- readfile
- readline_callback_handler_install
- readline_completion_function
- readlink
- realpath
- realpath_cache_get
- realpath_cache_size
- register_shutdown_function
- register_tick_function
- rename
- rewind
- rmdir
- rmdir_recursive
- session_set_save_handler
- set_error_handler
- set_exception_handler
- set_file_buffer
- set_local_infile_handler
- set_time_limit
- setclientcallback
- setcompletecallback
- setdatacallback
- setexceptioncallback
- setfailcallback
- setserverparams
- setstatuscallback
- setwarningcallback
- setworkloadcallback
- shell_exec
- simplexml_load_file
- simplexml_load_string
- spl_autoload_register
- sqlite_create_aggregate
- sqlite_create_function
- sqlitecreateaggregate
- sqlitecreatefunction
- stat
- sugar_chgrp
- sugar_chmod
- sugar_chown
- sugar_file_put_contents
- sugar_file_put_contents_atomic
- sugar_fopen

    - sugar_mkdir
    - sugar_rename
    - sugar_touch
    - sybase_set_message_handler
    - symlink
    - system
    - tempnam
    - timestampnoncehandler
    - tmpfile
    - tokenhandler
    - touch
    - uasort
    - uksort
    - umask
    - unlink
    - unzip
    - unzip_file
    - usort
    - write_array_to_file
    - write_array_to_file_as_key_value_pair
    - write_encoded_file
    - xml_set_character_data_handler
    - xml_set_default_handler
    - xml_set_element_handler
    - xml_set_end_namespace_decl_handler
    - xml_set_external_entity_ref_handler
    - xml_set_notation_decl_handler
    - xml_set_processing_instruction_handler
    - xml_set_start_namespace_decl_handler
    - xml_set_unparsed_entity_decl_handler
- The following class functions are denylisted by default:
    - All variable functions (i.e., $func()) are prohibited by default.
    - SugarLogger::setLevel
    - SugarAutoLoader::put
    - SugarAutoLoader::unlink

Health Check

Packages must pass all health checks in order to pass through the package scanner. For more information on troubleshooting Health Check output, see the collection of help articles in Troubleshooting Health Check Output.

Disabling File Scan

Note: Disabling File Scan is prohibited for instances on Sugar's cloud service.

To disable File Scan, add the following configuration setting to config_override.php:

```
  $sugar_config['moduleInstaller']['disableFileScan'] = false;
```

Extending the List of Valid Extension Types

Note: Modifying the valid extensions list is prohibited for instances on Sugar's cloud service.

To add more file extensions to the approved list of valid extension types, add the file extensions to the validExt array. The example below adds a .log file extension and .htaccess to the valid extension type list in config_override.php:

```
  $sugar_config['moduleInstaller']['validExt'] = array(
    'log',
    'htaccess'
);
```

Denylisting Additional Function Calls

Note: Denylist modifications are prohibited for instances on Sugar's cloud service.

To add additional function calls to the denylist, add the function calls to the blackList array. The example below blocks the strlen() and strtolower() functions from being included in the package:

```
  $sugar_config['moduleInstaller']['blackList'] = array(
    'strlen',
    'strtolower'
);
```

Overriding Denylisted Function Calls

Note: Denylist modifications are prohibited for instances on Sugar's cloud service.

To override the denylist and allow a specific function to be included in packages, add the function call to the blackListExempt array. The example below removes the restriction for the file_put_contents() function, allowing it to be included in the package:

```
  $sugar_config['moduleInstaller']['blackListExempt'] = array(
    'file_put_contents'
);
```

Disabling Restricted Copy

To ensure upgrade-safe customizations, System Administrators must restrict the copy action to prevent modifying the existing Sugar source code files. New files may be added anywhere (to allow new modules to be added), but core Sugar source code files must not be overwritten. This is enabled by default when you enable Package Scan.

Note: Disabling Restricted Copy is prohibited for instances on Sugar's cloud service.

To disable Restricted Copy, use this configuration setting:

```
  $sugar_config['moduleInstaller']['disableRestrictedCopy'] = true;
```

Module Loader Actions

Module loader actions, defined in ./ModuleInstall/ModuleScanner.php, are identifiers that map to the installation definitions used in the $installdefs of a manifest.

Action	$installdef Actions	Description
install_administration	administration	Installs an administration section into the Admin page
install_connectors	connectors	Installs SugarCloud Connectors
install_copy	copy	Installs files or directories
install_dashlets	dashlets	Installs dashlets into the Sugar application
install_images	image_dir	Install images into the custom directory
install_languages	language	Installs language files
install_layoutdefs	layoutdefs	Installs layouts
install_layoutfields	layoutfields	Installs custom fields

install_logichooks	logic_hooks	Installs logic hooks
install_relationships	relationships	Installs relationships
install_userpage	user_page	Installs a section to the User record page
install_vardefs	vardefs	Installs vardefs
post_execute	post_execute	Called after a package is installed
pre_execute	pre_execute	Called before a package is installed

Disabling Module Loader Actions

Certain Module Loader actions may be considered less desirable than others by a System Administrator. A System Administrator may want to allow some Module Loader actions, but disable specific actions that could impact the upgrade-safe integrity of the Sugar instance.

Note: Disabling Module Loader actions is prohibited for instances on Sugar's cloud service.

By default, all Module Loader actions are allowed. Enabling Package Scan does not affect the Module Loader actions. To disable specific Module Loader actions, add the action to the disableActions array. The example below restricts the pre_execute and post_execute actions:

```
  $sugar_config['moduleInstaller']['disableActions'] = array(
    'pre_execute',
    'post_execute'
);
```

Disabling Upgrade Wizard

If you are hosting Sugar and wish to lock down the upgrade wizard, you can set disable_uw_upload to 'true' in the config_override. This is intended for hosting providers to prevent unwanted upgrades.

```
  $sugar_config['disable_uw_upload'] = true;
```

Module Loader Restriction Alternatives

Overview

This article provides workarounds for commonly used functions that are denylisted by Sugar for Sugar's cloud environment.

Denylisted Functions

$variable()

Variable functions are sometimes used when trying to dynamically call a function. This is commonly used to retrieve a new bean object.

Restricted use:

```
  $module = "Account";
$id = "6468238c-da75-fd9a-406b-50199fe6b5f8";

//creating a new bean
$focus = new $module()

//retrieving a specific record
$focus->retrieve($id);
```

As of 6.3.0, we have implemented newBean and getBean which can be found in the BeanFactory. Below is the recommended approach to create or fetch a bean:

```
  $module = "Accounts";
$id = "6468238c-da75-fd9a-406b-50199fe6b5f8";

//creating a new bean
$focus = BeanFactory::newBean($module);

//or creating a new bean and retrieving a specific record
$focus = BeanFactory::getBean($module, $id);
```

array_filter()

The array_filter filters elements of an array using a callback function. It is restricted from use on Sugar's cloud service due to its ability to call other restricted functions.

Restricted use:

```
  /**
 * Returns whether the input integer is odd
 * @param $var
 * @return int
 */
function odd($var) {
    return($var & 1);
}

$myArray = array(
    "a"=>1,
    "b"=>2,
    "c"=>3,
    "d"=>4,
    "e"=>5
);

$filteredArray = array_filter($myArray, "odd");
```

An alternative to using array_filter is to use a foreach loop.

```
  $filteredArray = array();

$myArray = array(
    "a"=>1,
    "b"=>2,
    "c"=>3,
    "d"=>4,
    "e"=>5
);

foreach ($myArray as $key => $value) {
    // check whether the input integer is odd
    if($value & 1) {
        $filteredArray[$key] = $value;
    }
}
```

copy()

The copy method is sometimes used by developers when duplicating files in the uploads directory.

Restricted use:

```
  $result = copy($oldFile, $newFile);
```

An alternative to using copy is the duplicate_file method found in the UploadFile class.

```
  require_once 'include/upload_file.php';

$uploadFile = new UploadFile();
$result = $uploadFile->duplicate_file($oldFileId, $newFileId);
```

file_exists()

The file_exists method is used by developers to determine if a file exists.

Restricted use:

```
  if(file_exists($file_path)) {
    require_once($file);
}
```

An alternative to using file_exists is the fileExists method found in the SugarAutoLoader class.

```
  $file = 'include/utils.php';
if (SugarAutoloader::fileExists($file)) {
    require_once($file);
}
```

file_get_contents()

The file_get_contents method is used to retrieve the contents of a file.

Restricted use:

```
  $file_contents = file_get_contents('file.txt');
```

An alternative to using file_get_contents and sugar_file_get_contents is the get_file_contents method found in the UploadFile class.

```
  require_once('include/upload_file.php');

$uploadFile = new UploadFile();

//get the file location
$uploadFile->temp_file_location = UploadFile::get_upload_path($file_id
);
$file_contents = $uploadFile->get_file_contents();
```

fwrite()

The fwrite method is a function used to write content to a file. As there isn't currently a direct alternative for this function, you may find one of the following a good solution to what you are trying to achieve.

Adding/Removing Logic Hooks

When working with logic hooks, it is very common for a developer to need to modify ./custom/modules/<module>/logic_hooks.php. When creating module loadable packages, developers will sometimes use fwrite to modify this file upon installation to include their additional hooks. As of Sugar 6.3, Logic Hook Extensions were implemented to allow a developer to append custom hooks. If you would prefer to edit the logic_hooks.php file, you will need to use the check_logic_hook_file method as described below:

```
  //Adding a logic hook
require_once("include/utils.php");

$my_hook = Array(
    999,
    'Example Logic Hook',
    'custom/modules/<module>/my_hook.php',
    'my_hook_class',
    'my_hook_function'
);

check_logic_hook_file("Accounts", "before_save", $my_hook);
```

Removing a logic hook can be done by using remove_logic_hook:

```
  //Removing a logic hook
require_once("include/utils.php");

$my_hook = Array(
    999,
    'Example Logic Hook',
    'custom/modules/<module>/my_hook.php',
    'my_hook_class',
    'my_hook_function'
);

remove_logic_hook("Accounts", "before_save", $my_hook);
```

getimagesize()

The getimagesize method is used to retrieve information about an image file.

Restricted use:

```
  $img_size = getimagesize($path);
```

If you are looking to verify an image is .png or .jpeg, you can use the verify_uploaded_image method:

```
  require_once('include/utils.php');

if (verify_uploaded_image($path)) {
    //logic
}
```

If you are looking to get the mime type of an image, you can use the get_file_mime_type method:

```
  $mime_type = get_file_mime_type($path);
```

SugarOutfitters Package Guidelines

Overview

The Sugar® platform is open and flexible. Developers can customize any feature or integrate any system with Sugar using a Sugar Module Loadable Package. This flexibility requires guidelines so that SugarOutfitters customers can rest assured that all package offerings adhere to consistent quality standards and will not interfere with existing customizations or prevent software upgrades.

This guide outlines the minimum standards we expect from any Sugar Developer writing code for the Sugar platform. From the development of Sugar packages to security, user interface, encapsulation, and performance considerations, SugarCRM takes the safety and integrity of the Sugar application and its community very seriously. **Compliance with these guidelines is a requirement for any package installed into Sugar's cloud service. Failure to follow these guidelines is grounds for having your package removed from Sugar's cloud service and de-listed from SugarOutfitters.**

SugarCRM reserves the right to change these guidelines at any time. Please send any questions or feedback on these guidelines to developers@sugarcrm.com.

Best Practices

These best practice guidelines have been curated based on years of collective experience working with Sugar Packages that get used by Sugar customers every day.

Use a consistent coding style

For all PHP code, the style standard that SugarCRM uses is PSR-2. For JavaScript code, we use the applicable PSR-2 conventions as well. For JavaScript code, we emphasize readability which means we make use of utilities like Underscore.js and avoid nested callbacks. All functions, methods, classes in our code are required to have PHPDoc and JSDoc. While not all the code in the Sugar code base currently complies with these standards, we do enforce it as the standard on new code. Using a consistent code style increases the readability and maintainability of application code for those that come after you.

Invest in unit testing during development

For all JavaScript and PHP code, we strongly recommend the creation of unit tests. For PHP, we use PHPUnit and have developed a framework (to be shared) for testing Sugar server-side code using this framework. For JavaScript code, we use

Jasmine and have also developed a framework (to be shared) there to bootstrap Sidecar metadata so that Jasmine tests can run without dependency on a Sugar server.

We recommend running unit tests frequently during the development process. We suggest developers run tests before each commit and as part of an automated continuous integration process. Running tests often means you catch failures sooner which makes them easier and cheaper to fix.

Use Sugar REST APIs

The easiest way to integrate with a Sugar instance is not to install anything into Sugar at all. For Sugar 7, we have a full client REST API that is used to drive our web interface, our mobile client, and our plug-ins. If you can do it from our user interface, you can use our REST API separately to do it as well. If the REST API doesn't do everything you need it to do, it is very easily extensible. You can easily write code that adds your custom API endpoints in a minimally invasive way.

Use Module Builder when possible

Many integrations can be accomplished with the help of Module Builder and the Sugar REST API. Module Builder allows you to quickly design new modules for Sugar that match concepts that you want to add to a Sugar instance. These custom modules can then be installed and populated via the REST API, making it a powerful integration mechanism that doesn't require writing a line of Sugar code. Using custom modules will prevent conflicts with other customizations, as well. For more information, please refer to the Module Builder documentation.

Use Extension framework and Dashlets for Sugar code customizations

The Extension framework is the way to add server-side changes to a Sugar instance in a loosely coupled way. Dashlets are the best way to add new, custom user interface components to a Sugar instance that gives users maximum flexibility in how they use your app. These mechanisms also avoid conflicts with other customizations since Extensions are additive and do not replace core files.

Security Guidelines

Protecting and controlling access to CRM data is of paramount importance. It is important that Sugar Packages are good stewards of CRM data and access control.

Use TLS/SSL for web services calls

Just as we don't recommend running Sugar in production without TLS/SSL, all web-

services calls that are initiated from a Sugar Package should also use SSL. The concern is the exposure of user credentials, OAuth/session tokens, or sensitive CRM data via plaintext transmission that would otherwise be handled securely.

Do not hardcode sensitive information

You also should not hardcode any credentials, API keys, tokens, etc, within a Sugar Package. Sugar Packages and Sugar application code is never encrypted so there is always a risk that an attacker could discover these things and abuse them. Usernames and passwords, OAuth tokens, and similar credentials for accessing 3rd party systems should be stored in the database (for example, on the config table). The Sugar platform also provides encryption utilities that allow information to be stored in an encrypted form. These settings could then be changeable by the Administrator via the Administration panel or some other end-user input.

User Interface Guidelines

Sugar packages can be used to add new user interface components or front-end customizations to Sugar. It is important to consider the impact that these changes have on the user experience and the look and feel of the Sugar application.

Use the Sugar 7 Styleguide

In Sugar 7, we have doubled down on our emphasis on creating the best possible user experience. While other applications may use different usage patterns than the ones used in Sugar 7, it is important to think about how new functionality or integrations that you build in Sugar 7 fits within the overall Sugar 7 user experience. Users do not tolerate an inconsistent experience within a given tool - it makes it harder to learn to use, which lowers adoption of Sugar as well as your application.

We want to make it easier for you to build applications within Sugar that use a consistent theme and user experience patterns, so we have included a styleguide for the Sugar application. You can find a link to the styleguide from the Sugar Administration page. There you can find all the information you need to leverage Sugar 7's styling including our CSS framework.

Don't use incompatible UI frameworks or external CSS libraries

Mixing outside user interface frameworks into the Sugar application via a Sugar Package can easily break many different parts of the application. Please only include as much CSS as you need and make sure that it is properly formed so that it doesn't affect other parts of the Sugar application. At the very least, using different frameworks or themes within your application will create a disjointed experience for the end user. While your package may be installed into Sugar, the

user experience will not be seamless.

Encapsulation Guidelines

An important aspect of the quality of a Sugar Package is how well encapsulated and loosely coupled it is. A well encapsulated and loosely coupled package will encounter the fewest issues during upgrades, fewer breakages due to core code changes or due to interactions with other installed packages, as well minimizing bugs or problems that end users encounter.

Use Extensions framework and Custom Modules as much as possible

A well-encapsulated package prefers the use of Custom Modules over customizing and repurposing existing Sugar Modules. A loosely coupled package uses Extensions framework for customizing and connecting with the core Sugar application. Only override the behavior of core Sugar application files as a last resort.

Avoid customizations to core Sugar application files

Developers are strongly discouraged from overriding core Sugar Modules or Sugar framework code. In many cases, a cleaner approach for accomplishing the same goal exists. For example, using a logic hook to extend the behavior of a SugarBean instead of overriding the SugarBean itself. Every core customization is a barrier to successful upgrades that creates recurring development costs over time. This is exacerbated in heavily customized Sugar instances as other customizations may exist on these files. Anytime there is a conflict then manual intervention by a Sugar Developer is required which is not only inconvenient but costly for everyone involved. If you do make core Sugar customizations then keeping track of such changes is very important to get in front of potential conflicts with other packages and upgrades.

Use Package Scanner to ensure your Sugar Package is ready for SugarCloud

Sugar Packages should be designed with Sugar's cloud service in mind, as many Sugar customers choose this hosting option over hosting on-site or through a Sugar partner. Code that gets loaded into Sugar's cloud service must pass the Package Scanner utility and must not adversely impact the SugarCRM infrastructure. This stipulation is outlined in the SugarCloud Policy Guide. Package Scanner can be enabled on any Sugar instance via the Sugar Administration panel which allows this to be tested easily. You should also educate yourself in some alternatives to denylisted functions.

Performance Guidelines

Sugar is a modern web application that can be used to manage millions and millions of customer records which makes performance tuning and optimization of any add-on solution essential. It is best to avoid pre-optimization and use tools to properly identify the root causes of performance issues that users could encounter.

Sugar has instrumentation built into it for New Relic APM, which can monitor browser and server performance simultaneously, as well as XHprof for Sugar PHP profiling. Slow query logging can also be enabled via the Sugar Administration panel under System Settings. For more information, refer to the System documentation. The Sugar Engineering team typically uses Chrome DevTools for JavaScript profiling.

Index for large frequently used queries

The most common performance bottleneck for Sugar is the database. Using slow query logging makes it possible to identify bottlenecks and correct them. One way to address query bottlenecks is to extend vardefs to add indices during package install to improve the performance of those queries.

Add scheduled jobs to prune large database tables

If your Package adds database tables that can tend to grow very large over time, it is a best practice to include scheduled jobs in your package that can be used to prune the size of this database over time. For Sugar, these background tasks can be created and managed using the Job Queue framework. At the very least, you'll want to create a Custom Job that Sugar Administrators can then run as needed. This will allow the package to remain in tip-top shape for your users over time. Especially if they are running on Sugar's cloud service because direct access to the underlying SQL database to do manual tuning and cleanup is not permitted.

Ensure your application does not block user interface during long running processes

If your application prevents the user from getting feedback or using the interface while it is running a long process, this will impact the perceived performance of the application. Users typically expect some UI feedback within half a second. If you have transactions that will take longer than that, it is best to use the Job Queue framework to defer them for later or move them into a scheduled Custom Job.

Module Builder

Overview

The Module Builder tool allows programmers to create custom modules without writing code and to create relationships between new and existing CRM modules. To illustrate how to use Module Builder, this article will show how to create and deploy a custom module.

For this example, a custom module to track media inquiries will be created to track public relations requests within a CRM system. This use case is an often requested enhancement for CRM systems that apply across industries.

Creating New Modules

Module Builder functionality is managed within the 'Developer Tools' section of Sugar's administration console.

Upon selecting 'Module Builder', the user has the option of creating a "New Package". Packages are a collection of custom modules, objects, and fields that can be published within the application instance or shared across instances of Sugar. Once the user selects "New Package", the usernames and describes the type of Custom Module to be created. A package key, usually based on the organization or name of the package creator is required to prevent conflicts between two packages with the same name from different authors. In this case, the package will be named "MediaTracking" to explain its purpose, and a key based on the author name will be used.

Once the new package is created and saved, the user is presented with a screen to create a Custom Module. Upon selecting the "New Module" icon, a screen appears showing six different object templates.

Understanding Object Templates

Five of the six object templates contain pre-built CRM functionality for key CRM use cases. These objects are:"basic", "company", "file", "issue", "person", and "sale". The "basic" template provides fields such as Name, Assigned to, Team, Date Created, and Description. As their title denotes, the rest of these templates contain fields and application logic to describe entities similar to "Accounts", "Documents, "Cases", "Contacts", and "Opportunities", respectively. Thus, to create a Custom Module to track a type of account, you would select the "Company" template. Similarly, to track human interactions, you would select "People".

For the media tracking use case, the user will use the object template "Issue" because inbound media requests have similarities to incoming support cases. In both examples, there is an inquiry, a recipient of the issue, assignment of the issue and resolution. The final object template is named "Basic" which is the default base

object type. This allows the administrator to create their own custom fields to define the object.

Upon naming and selecting the Custom Module template named "Issue", the user can further customize the module by changing the fields and layout of the application and creating relationships between this new module and existing standard or custom modules. This Edit functionality allows a user to construct a module that meets the specific data requirements of the Custom Module.

Editing Module Fields

Fields can be edited and created using the field editor. Fields inherited from the custom module's templates can be relabeled while new fields are fully editable. New fields are added using the Add Field button. This displays a tab where you can select the type of field to add as well as any properties that field-type requires.

Editing Module Layouts

The layout editor can be used to change the appearance of the screens within the new module, including the EditView, DetailView and ListView screens. When editing the Edit View or the Detail View, new panels and rows can be dragged from the toolbox on the left side to the layout area on the right. Fields can then be dragged between the layout area and the toolbox. Fields are removed from the layout by dragging them from the layout area to the recycling icon. Fields can be expanded or collapsed to take up one or two columns on the layout using the plus and minus icons. The List, Search, Dashlet, and Subpanel views can be edited by dragging fields between hidden/visible/available columns.

Building Relationships

Once the fields and layout of the Custom Module have been defined, the user then defines relationships between this new module and existing CRM data by clicking "View Relationships". The "Add Relationship" button allows the user to associate the new module to an existing or new custom module in the same package. In the case of the Media Tracker, the user can associate the Custom Module with the existing, standard 'Contacts' module that is available in every Sugar installation using a many-to-many relationship. By creating this relationship, end-users will see the Contacts associated with each Media Inquiry. We will also add a relationship to the activities module so that a Media Inquiry can be related to calls, meetings, tasks, and emails.

Publishing and Uploading Packages

After the user has created the appropriate fields, layouts, and relationships for the custom modules, this new CRM functionality can be deployed. Click the "Deploy"

button to deploy the package to the current instance. This is the recommended way to test your package while developing. If you wish to make further changes to your package or custom modules, you should make those changes in Module Builder, and click the Deploy button again. Clicking the Publish button generates a zip file with the Custom Module definitions. This is the mechanism for moving the package to a test environment and then ultimately to the production environment. The Export button will produce a module loadable zip file, similar to the Publish functionality, except that when the zip file is installed, it will load the custom package into Module Builder for further editing. This is a good method for storing the custom package in case you would like to make changes to it in the future on another Sugar instance. Once your module has been deployed in a production environment, we highly recommend that you do not redeploy the module in Module Builder but modify the module using Studio as outlined in our Best Practices When Building Custom Modules.

After the new package has been published, the administrator must commit the package to the Sugar system through the Module Loader. The administrator uploads the files and commits the new functionality to the live application instance.

Note: Sugar Sell Essentials customers do not have the ability to upload custom file packages to Sugar using Module Loader.

Adding Custom Logic Using Code

While the key benefit of the Module Builder is that the Administrator user is able to create entirely new modules without the need to write code, there are still some tasks that require writing PHP code. For instance, adding custom logic or making a call to an external system through a Web Service. This can be done in one of two methods.

Logic Hooks

One way is by writing PHP code that leverages the event handlers, or "logic hooks", available in Sugar. In order to accomplish this, the developer must create the custom code and then add it to the manifest file for the "Media Inquiry" package. More information on creating logic hooks can be found in the Logic Hooks section. Information on adding a hook to your installer package can be found in the Creating an Installable Package for a Logic Hook example.

Custom Bean files

Another method is to add code directly to the custom bean. This is a more complicated approach because it requires understanding the SugarBean class. However, it is a far more flexible and powerful approach.

First, you must "build" your module. This can be done by either deploying your

module or clicking the Publish button. Module Builder will then generate a folder for your package in ./custom/modulebuilder/builds/. Inside that folder is where Sugar will have placed the bean files for your new module(s). In this case, we want ./custom/modulebuilder/builds/MediaTracking/SugarModules/modules/jsche_mediarequest/

Inside you will find two files of interest. The first one is {module_name}sugar.php. This file is generated by Module Builder and may be erased by further changes in module builder or upgrades to the Sugar application. You should not make any changes to this file. The second is {module_name}.php. This is the file where you make any changes you would like to the logic of your module. To add our timestamp, we would add the following code to jsche_mediarequest.php

```
  function save($check_notify = FALSE)
{
     global $current_user;
     $this->description .= "Saved on " . date("Y-m-
d g:i a"). " by ". $current_user->user_name;
     parent::save($check_notify);
}
```

The call to the parent::save function is critical as this will call on the out of box SugarBean to handle the regular Save functionality. To finish, re-deploy or re-publish your package from Module Builder.

You can now upload this module, extended with custom code logic, into your Sugar application using the Module Loader as described earlier.

Using the New Module

After you upload the new module, the new custom module appears in the Sugar instance. In this example, the new module, named "Media" uses the object template "Issue" to track incoming media inquiries. This new module is associated with the standard "Contacts" modules to show which journalist has expressed interest. In this example, the journalist has requested a product briefing. On one page, users can see the nature of the inquiry, the journalist who requested the briefing, who the inquiry was assigned to, the status, and the description.

Best Practices

Overview

Sugar provides two tools for building and maintaining custom module configurations: Module Builder and Studio. As an administrator of Sugar, it is important to understand the strengths of both tools so that you have a sound development process as you look to build on Sugar's framework.

Goal

This guide will provide you with all the necessary steps from initial definition of your module to the configuration and customization of the module functionality. Follow these tips to ensure your development process will be sound:

Build Your Module in Module Builder

Build the initial framework for your module in Module Builder. Add all the fields you feel will be necessary for the module and construct the layouts with those fields. If possible, it is even better to create your custom modules in a development environment that mirrors your production environment.

Never Redeploy a Package

Once a module has been promoted to a production environment, we recommend only making additional changes in Studio. Redeploying a package will remove all customizations related to your module in the following directories:

- ./modules/
- ./custom/modules/
- ./custom/Extension/modules/

This includes workflows, code customizations, changes through Studio, and much more. It is imperative that this directive is followed to ensure any desired configurations remain intact. When working in Studio, you can make the following types of changes:

- adding a new field
- updating the properties of a field that has been deployed with the module
- changing field layouts
- creating, modifying and removing relationships

Every Module in Module Builder Gets Its Very Own Package

While it is possible to create multiple modules in a package, this can also cause design headaches down the road. If you end up wanting to uninstall a module and it is part of a larger package, all modules in that package would need to be uninstalled. Keeping modules isolated to their own packages allows greater flexibility in the future if a module is no longer needed.

Create Relationships in Studio After the Module Is Deployed

This part is critical for success as relationships created in Module Builder cannot be removed after the module is deployed unless the package is updated and redeployed from Module Builder. Redeploying from Module Builder is what we are trying to avoid as mentioned above. If you deploy the module and then create the relationships in Studio, you can update or remove the relationships via Studio at any future point in time.

Delete the Package from Module Builder Once It Is Deployed

Once the package is deployed, delete it from Module Builder so that it will not accidentally be redeployed. The only exception to this rule is in a development environment as you may want to continue working and testing until you are ready to move the module to your production environment. If you ever want to uninstall the module at a later date, you can do so under Admin > Module Loader.

Quotes

Overview

As of Sugar 7.9.0.0, the quotes module has been moved out of Backward Compatibility mode into Sidecar. Customizations to the Quotes interface will need to be rewritten for the new Sidecar framework as an automated migration of these customizations is not possible. This document will outline common customizations and implementation methods.

Quote Identifiers

Administrators can create custom quote identifiers that are more complex than the default auto-incrementing scheme found in an out of box installation. Utilizing this functionality, we can create numbering schemes that match any business needs.

Creating Custom Identifiers

To create a custom identifier, navigate to Admin > Studio > Quotes > Fields. Next, create a TextField type field named to your liking and check the "Calculated Value" checkbox and click "Edit Formula". Once you see the formula builder, you can choose an example below or create your own. Once created, You can add the new field to your Quote module's layouts as desired.

User Field with Pseudo-Random Values

This example creates a quote number in the format of <user last name>-##### that translates to Smith-87837. This identifier starts with the last name of the user creating it, followed by 5 pseudo-random numbers based on the creation time.

```
concat(related($created_by_link, "last_name"), "-", subStr(toString(ti
mestamp($date_entered)), 5, 5))
```

Note: when using Sugar Logic, updates to related fields will update any corresponding Sugar Logic fields. The example being that an update to the Created By last name field will update all related records with a field using this formula.

Related Field Value and Auto-Incrementing Number

This example creates a quote number in the format of <billing account name>-#### that translates to Start Over Trust-0001. This number starts with the billing account name followed by a 4 digit auto-incrementing number.

```
concat(related($billing_accounts, "name"), "-", subStr(toString(add($q
uote_num, 10000)), 1, 4))
```

Note: when using Sugar Logic, updates to related fields will update any corresponding Sugar Logic fields. The example being that an update to the Billing Account name will update all related records with a field using this formula.

Record Layout

The following sections outline various changes that can be done to modify the record layout of the Quotes module, by outlining the extra views that can be updated.

Grand Totals Header

The Grand Total Header contains the calculated totals for the quote.

Modifying Fields in the Grand Totals Header

To modify the Grand Totals Header fields, you can copy ./modules/Quotes/clients/base/views/quote-data-grand-totals-header/quote-data-grand-totals-header.php to ./custom/modules/Quotes/clients/base/views/quote-data-grand-totals-header/quote-data-grand-totals-header.php or you may create a metadata extension in ./custom/

Extension/modules/Quotes/clients/base/views/quote-data-grand-totals-header/. Next, modify the $viewdefs['Quotes']['base']['view']['quote-data-grand-totals-header']['panels'][0]['fields'] index to add or remove your preferred fields.

Example

The example below will add the "Quote Number" field (quotes.quote_num) field to the layout and removes the "Total Discount" (quotes.deal_tot) from the layout. If adding a custom field from the Quotes module to the view, you will also need to add that field to the related_fields array as outlined in the Record View documentation below.

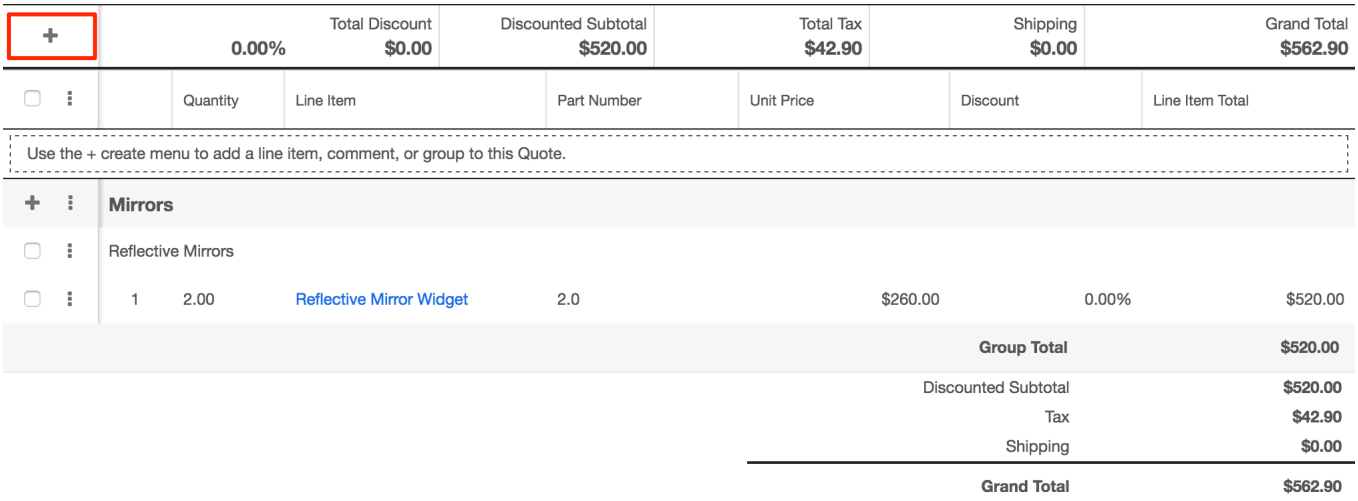
./custom/modules/Quotes/clients/base/views/quote-data-grand-totals-header/quote-data-grand-totals-header.php

```
  <?php

$viewdefs['Quotes']['base']['view']['quote-data-grand-totals-
header'] = array(
    ...
    'panels' => array(
        array(
            'name' => 'panel_quote_data_grand_totals_header',
            'label' => 'LBL_QUOTE_DATA_GRAND_TOTALS_HEADER',
            'fields' => array(
                array(
                    'name' => 'quote_num',
                    'label' => 'LBL_LIST_QUOTE_NUM',
                    'css_class' => 'quote-totals-row-item',
                ),
                array(
                    'name' => 'new_sub',
                    'css_class' => 'quote-totals-row-item',
                ),
                array(
                    'name' => 'tax',
                    'label' => 'LBL_TAX_TOTAL',
                    'css_class' => 'quote-totals-row-item',
                ),
                array(
                    'name' => 'shipping',
                    'css_class' => 'quote-totals-row-item',
                ),
                array(
                    'name' => 'total',
                    'label' => 'LBL_LIST_GRAND_TOTAL',
```

```
                    'css_class' => 'quote-totals-row-item',
                ),
            ),
        ),
    ),
);
```

Modifying Buttons in the Grand Totals Header

The Quotes List Header contains row actions to create quoted line items, comments, and groupings. The actions are identified by the plus icon in the top left of the header. Editing the 'actions' will allow you to add or remove buttons. The following section will outline how these items can be modified.



To modify the Row Actions in the Grand Total Header, you can copy ./modules/Quotes/clients/base/views/quote-data-grand-totals-header/quote-data-grand-totals-header.php to ./custom/modules/Quotes/clients/base/views/quote-data-grand-totals-header/quote-data-grand-totals-header.php or you may create a metadata extension in ./custom/Extension/modules/Quotes/clients/base/views/quote-data-grand-totals-header/. Next, modify the $viewdefs['Quotes']['base']['view']['quote-data-grand-totals-header']['buttons'] index to add or remove your preferred row actions.

Example

The example below will create a new view that extends the QuotesQuoteDataGrandTotalsHeader view and append a button to the Grand Total Header button list.

First, create your custom view type that will extend the QuotesQuoteDataGrandTotalsHeader view.

./custom/modules/Quotes/clients/base/views/quote-data-grand-totals-header/quote-data-grand-totals-header.js

```
  ({
    extendsFrom: 'QuotesQuoteDataGrandTotalsHeaderView',

    initialize: function(options) {
        this.events = _.extend({}, this.events, options.def.events, {
            'click [name="gth-custom-button"]': 'buttonClicked'
        });

        this._super('initialize', [options]);
    },

    /**
     * Click event
     */
    buttonClicked: function() {
        app.alert.show('success_alert', {
            level: 'success',
            title: 'Grand Total Header Button was clicked!'
        });
    },
})
```

This code will append a click event to our button named gth-custom-button and trigger the buttonClicked method. Once completed, add your new button array to the grand total header in the $viewdefs['Quotes']['base']['view']['quote-data-grand-totals-header']['buttons'] index.

./custom/modules/Quotes/clients/base/views/quote-data-grand-totals-header/quote-data-grand-totals-header.php

```
  <?php

$viewdefs['Quotes']['base']['view']['quote-data-grand-totals-
header'] = array(
    'buttons' => array(
        array(
            'type' => 'quote-data-actiondropdown',
            'name' => 'panel_dropdown',
```

```
            'no_default_action' => true,
            'buttons' => array(
                array(
                    'type' => 'button',
                    'icon' => 'fa-plus',
                    'name' => 'create_qli_button',
                    'label' => 'LBL_CREATE_QLI_BUTTON_LABEL',
                    'acl_action' => 'create',
                    'tooltip' => 'LBL_CREATE_QLI_BUTTON_TOOLTIP',
                ),
                array(
                    'type' => 'button',
                    'icon' => 'fa-plus',
                    'name' => 'create_comment_button',
                    'label' => 'LBL_CREATE_COMMENT_BUTTON_LABEL',
                    'acl_action' => 'create',
                    'tooltip' => 'LBL_CREATE_COMMENT_BUTTON_TOOLTIP',
                ),
                array(
                    'type' => 'button',
                    'icon' => 'fa-plus',
                    'name' => 'create_group_button',
                    'label' => 'LBL_CREATE_GROUP_BUTTON_LABEL',
                    'acl_action' => 'create',
                    'tooltip' => 'LBL_CREATE_GROUP_BUTTON_TOOLTIP',
                ),
                array(
                    'type' => 'button',
                    'icon' => 'fa-plus',
                    'name' => 'gth-custom-button',
                    'label' => 'LBL_GTH_CUSTOM_BUTTON',
                    'acl_action' => 'create',
                    'tooltip' => 'LBL_GTH_CUSTOM_BUTTON_TOOLTIP',
                ),
            ),
        ),
    ),
    ...
);
```

Finally, create labels under Quotes for the label and tooltip indexes. To accomplish this, create a language extension:

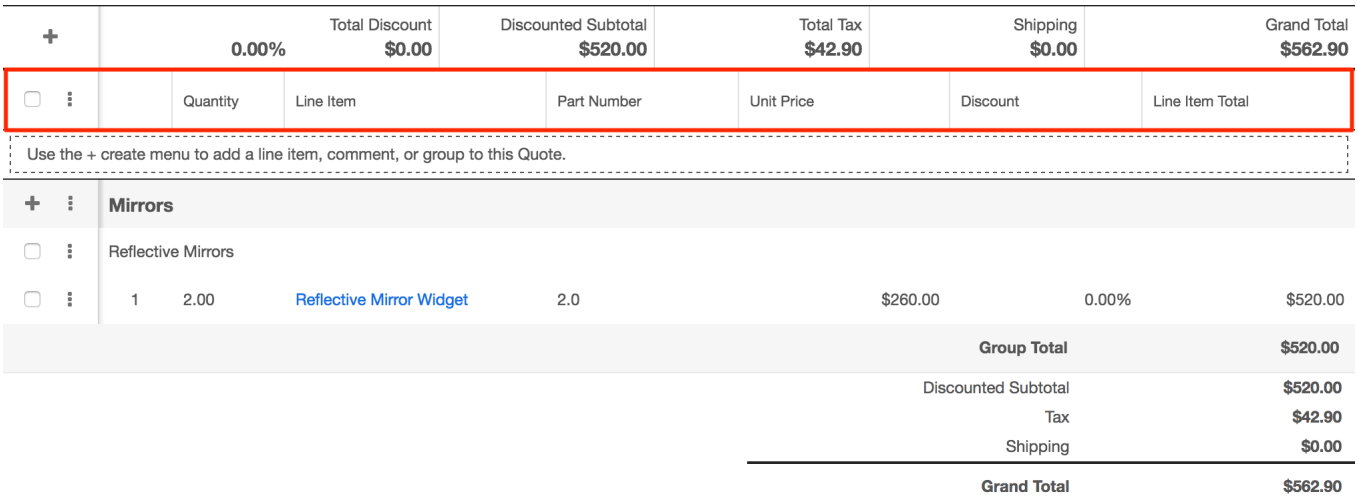
./custom/Extension/modules/Quotes/Ext/Language/en_us.custom-button.php

```
  <?php

$mod_strings['LBL_GTH_CUSTOM_BUTTON'] = 'Custom Button';
$mod_strings['LBL_GTH_CUSTOM_BUTTON_TOOLTIP'] = 'Custom Button Tooltip
';
```

Once the files are in place, navigate to Admin > Repair > Quick Repair and Rebuild. Your changes will now be reflected in the system.

List Header

The Quotes List Header is a complex view that pulls data from two different locations. The JavaScript controller is located in ./modules/Quotes/clients/base/views/quote-data-list-header/quote-data-list-header.js and pulls the metadata from ./modules/Products/clients/base/views/quote-data-group-list/quote-data-group-list.php. You should note that ProductBundleNotes combines its description field with Product fields in this list.



Modifying Fields in the List Header

To modify the List Header fields, you can copy ./modules/Products/clients/base/views/quote-data-group-list/quote-data-group-list.php to ./custom/modules/Products/clients/base/views/quote-data-group-list/quote-data-group-list.php or you may create a metadata extension in ./custom/Extension/modules/Products/Ext/clients/base/views/quote-data-group-list/.

Next, modify the $viewdefs['Products']['base']['view']['quote-data-group-list']['panels'][0]['fields'] index to add or remove your preferred fields.

Example

The example below will remove the "Part Number" (products.mft_part_num) field and replace it with the "Weight" (products.weight) field.

./custom/modules/Products/clients/base/views/quote-data-group-list/quote-data-group-list.php

```
  <?php

$viewdefs['Products']['base']['view']['quote-data-group-
list'] = array(
    'panels' => array(
        array(
            'name' => 'products_quote_data_group_list',
            'label' => 'LBL_PRODUCTS_QUOTE_DATA_LIST',
            'fields' => array(
                array(
                    'name' => 'line_num',
                    'label' => null,
                    'widthClass' => 'cell-xsmall',
                    'css_class' => 'line_num tcenter',
                    'type' => 'line-num',
                    'readonly' => true,
                ),
                array(
                    'name' => 'quantity',
                    'label' => 'LBL_QUANTITY',
                    'widthClass' => 'cell-small',
                    'css_class' => 'quantity',
                    'type' => 'float',
                ),
                array(
                    'name' => 'product_template_name',
                    'label' => 'LBL_ITEM_NAME',
                    'widthClass' => 'cell-large',
                    'type' => 'quote-data-relate',
                    'required' => true,
                ),
                array(
                    'name' => 'weight',
                    'label' => 'LBL_WEIGHT',
                    'type' => 'float',
```

```
                ),
                array(
                    'name' => 'discount_price',
                    'label' => 'LBL_DISCOUNT_PRICE',
                    'type' => 'currency',
                    'convertToBase' => true,
                    'showTransactionalAmount' => true,
                    'related_fields' => array(
                        'discount_price',
                        'currency_id',
                        'base_rate',
                    ),
                ),
                array(
                    'name' => 'discount',
                    'type' => 'fieldset',
                    'css_class' => 'quote-discount-percent',
                    'label' => 'LBL_DISCOUNT_AMOUNT',
                    'fields' => array(
                        array(
                            'name' => 'discount_amount',
                            'label' => 'LBL_DISCOUNT_AMOUNT',
                            'type' => 'discount',
                            'convertToBase' => true,
                            'showTransactionalAmount' => true,
                        ),
                        array(
                            'type' => 'discount-select',
                            'name' => 'discount_select',
                            'no_default_action' => true,
                            'buttons' => array(
                                array(
                                    'type' => 'rowaction',
                                    'name' => 'select_discount_amount_
button',
                                    'label' => 'LBL_DISCOUNT_AMOUNT',
                                    'event' => 'button:discount_select
_change:click',
                                ),
                                array(
                                    'type' => 'rowaction',
                                    'name' => 'select_discount_percent
_button',
                                    'label' => 'LBL_DISCOUNT_PERCENT',
                                    'event' => 'button:discount_select
_change:click',
```

```
                                ),
                            ),
                        ),
                    ),
                ),
                array(
                    'name' => 'total_amount',
                    'label' => 'LBL_LINE_ITEM_TOTAL',
                    'type' => 'currency',
                    'widthClass' => 'cell-medium',
                    'showTransactionalAmount' => true,
                    'related_fields' => array(
                        'total_amount',
                        'currency_id',
                        'base_rate',
                    ),
                ),
            ),
        ),
    ),
);
```

Next, create the LBL_WEIGHT label under Quotes as this is not included in the stock installation. To accomplish this, we will need to create a language extension:

./custom/Extension/modules/Quotes/Ext/Language/en_us.weight.php

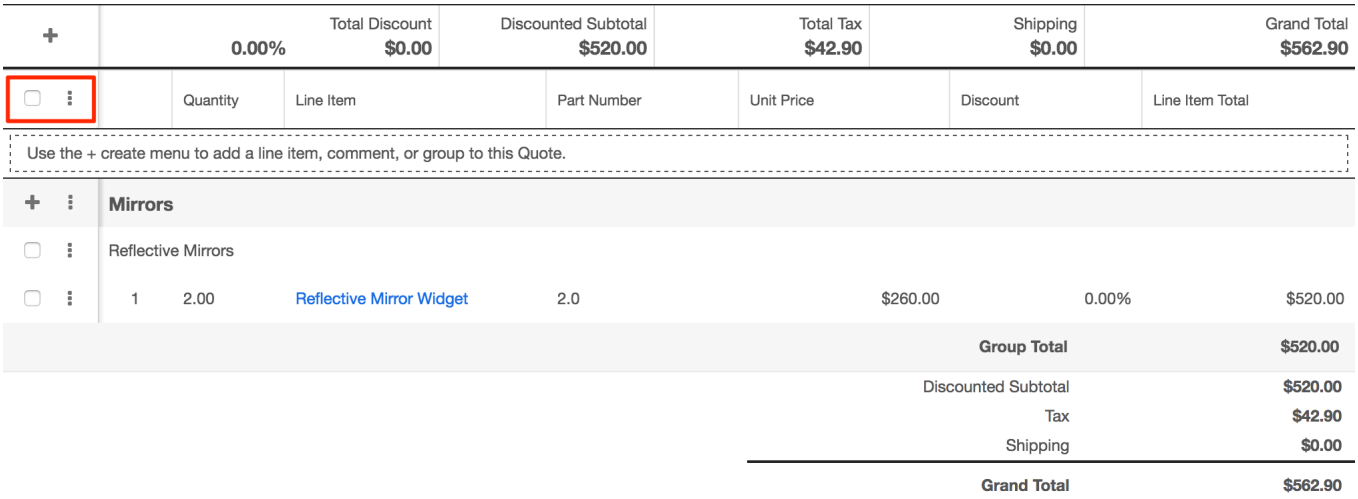
```
  <?php

$mod_strings['LBL_WEIGHT'] = 'Weight';
```

If adding a custom field from the Quotes module to the view, you will also need to add that field to the related_fields array as outlined in the Record View documentation below. Once the files are in place, navigate to Admin > Repair > Quick Repair and Rebuild. Your changes will now be reflected in the system.

Modifying Row Actions in the List Header

The Quotes List Header contains row actions to create and delete QLI groupings. The actions are identified by, the "check all" checkbox and vertical ellipsis or "hamburger" icon buttons. Editing the 'actions' will allow you to add more buttons to (or remove from) the "Group Selected" and "Delete Selected" buttons. The

following section will outline how these items can be modified.



To modify the Row Actions in the List Header, you can copy ./modules/Quotes/clients/base/views/quote-data-list-header/quote-data-list-header.php to ./custom/modules/Quotes/clients/base/views/quote-data-list-header/quote-data-list-header.php or you may create a metadata extension in ./custom/Extension/modules/Quotes/clients/base/views/quote-data-list-header/. Next, modify the $viewdefs['Quotes']['base']['view']['quote-data-list-header']['selection']['actions'] index to add or remove your preferred actions.

Example

The example below will create a new view that extends the QuotesQuoteDataListHeader view and append a row action to the List Header action list.

First, create your custom view type that will extend the QuotesQuoteDataListHeader view.

./custom/modules/Quotes/clients/base/views/quote-data-list-header/quote-data-list-header.js

```
({
  extendsFrom: 'QuotesQuoteDataListHeaderView',

  initialize: function(options) {
      this.events = _.extend({}, this.events, options.def.events, {
          'click [name="lh-custom-button"]': 'actionClicked'
      });
```

```
        this._super('initialize', [options]);
    },

    /**
     * Click event
     */
    actionClicked: function() {
        app.alert.show('success_alert', {
            level: 'success',
            title: 'List Header Row Action was clicked!'
        });
    },
})
```

This code will append a click event to our field named lh-custom-button and trigger the actionClicked method. Once completed, add your new row action to the list header in the $viewdefs['Quotes']['base']['view']['quote-data-list-header']['selection']['actions'] index.

./custom/modules/Quotes/clients/base/views/quote-data-list-header/quote-data-list-header.php

```
  <?php

$viewdefs['Quotes']['base']['view']['quote-data-list-header'] = array(
    'selection' => array(
        'type' => 'multi',
        'actions' => array(
            array(
                'name' => 'group_button',
                'type' => 'rowaction',
                'label' => 'LBL_CREATE_GROUP_SELECTED_BUTTON_LABEL',
                'tooltip' => 'LBL_CREATE_GROUP_SELECTED_BUTTON_TOOLTIP
',
                'acl_action' => 'edit',
            ),
            array(
                'name' => 'massdelete_button',
                'type' => 'rowaction',
                'label' => 'LBL_DELETE_SELECTED_LABEL',
                'tooltip' => 'LBL_DELETE_SELECTED_TOOLTIP',
                'acl_action' => 'delete',
            ),
            array(
```

```
                'name' => 'lh-custom-button',
                'type' => 'rowaction',
                'label' => 'LBL_LH_CUSTOM_ACTION',
                'tooltip' => 'LBL_LH_CUSTOM_ACTION_TOOLTIP',
                'acl_action' => 'edit',
            ),
        ),
    ),
);
```

Finally, create labels under Quotes for the label and tooltip indexes. To accomplish this, create a language extension:

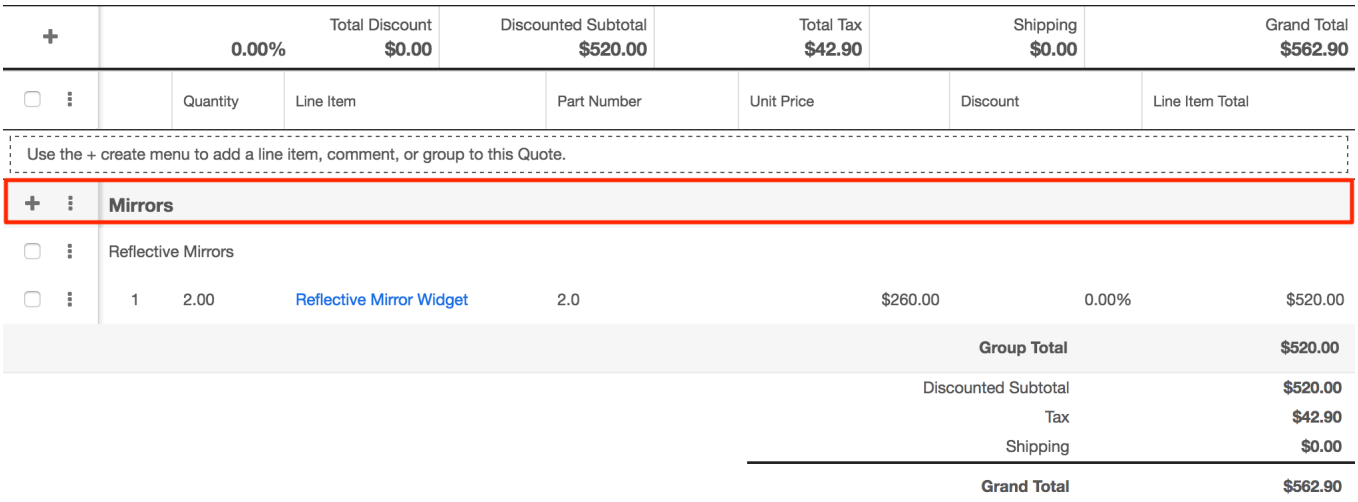
./custom/Extension/modules/Quotes/Ext/Language/en_us.lh-custom-action.php

```
  <?php

$mod_strings['LBL_LH_CUSTOM_ACTION'] = 'Custom Action';
$mod_strings['LBL_LH_CUSTOM_ACTION_TOOLTIP'] = 'Custom Action Tooltip'
;
```

Once the files are in place, navigate to Admin > Repair > Quick Repair and Rebuild. Your changes will now be reflected in the system.

Group Header

The Group Header contains the name and options for each grouping of quoted line items.



Modifying Fields in the Group Header

To modify the Group Header fields, you can copy ./modules/ProductBundles/clients/base/views/quote-data-group-header/quote-data-group-header.php to ./custom/modules/ProductBundles/clients/base/views/quote-data-group-header/quote-data-group-header.php or you may create a metadata extension in ./custom/Extension/modules/Products/clients/base/views/quote-data-group-header/. Next, modify the $viewdefs['ProductBundles']['base']['view']['quote-data-group-header'] index to add or remove your preferred fields.

Example

The example below will append the group total (product_bundles.subtotal) field to the Group Header. It's important to note that when adding additional fields, that changes to the corresponding .hbs file may be necessary to correct any formatting issues.

./custom/modules/ProductBundles/clients/base/views/quote-data-group-header/quote-data-group-header.php

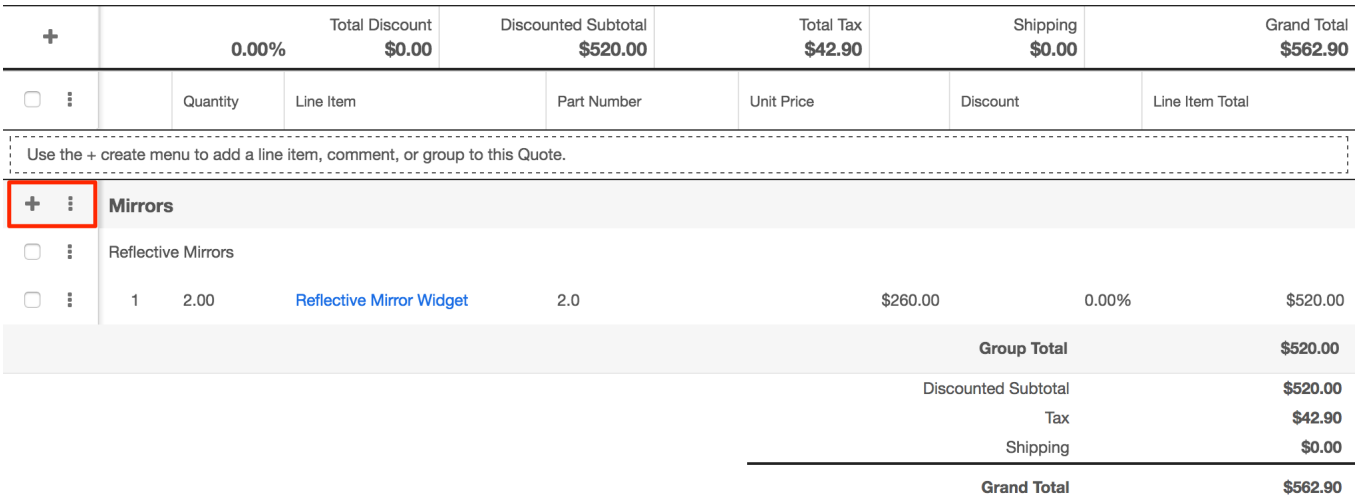
```
  <?php

$viewdefs['ProductBundles']['base']['view']['quote-data-group-
header'] = array(
    ...
```

```
    'panels' => array(
        array(
            'name' => 'panel_quote_data_group_header',
            'label' => 'LBL_QUOTE_DATA_GROUP_HEADER',
            'fields' => array(
                array(
                    'name' => 'name',
                    'type' => 'quote-group-title',
                    'css_class' => 'group-name',
                ),
                'subtotal',
            ),
        ),
    ),
);
```

If adding a custom field from the Quotes module to the view, you will also need to add that field to the related_fields array as outlined in the Record View documentation below. Once the files are in place, navigate to Admin > Repair > Quick Repair and Rebuild. Your changes will now be reflected in the system.

Modifying Row Actions in the Group Header

The Quotes Group Header contains row actions to add QLIs and comments as well as editing and deleting QLI groupings. The actions are identified by, the plus and vertical ellipsis or "hamburger" icon buttons. The following section will outline how these items can be modified.



To modify the buttons in the Group Header, you can copy ./modules/ProductBundles/clients/base/views/quote-data-group-header/quote-data-group-header.php to ./custom/modules/ProductBundles/clients/base/views/quote-data-group-header/quote-data-group-header.php or you may create a metadata extension in ./custom/Extension/modules/ProductBundles/clients/base/views/quote-data-group-header/. Next, modify the $viewdefs['ProductBundles']['base']['view']['quote-data-group-header']['buttons'] index to add or remove your preferred actions.

Example

The example below will create a new view that extends the ProductBundlesQuoteDataGroupHeader view and append a row action to the Group Header vertical elipsis action list.

First, create your custom view type that will extend the ProductBundlesQuoteDataGroupHeader view.

./custom/modules/ProductBundles/clients/base/views/quote-data-group-header/quote-data-group-header.js

```
  ({
    extendsFrom: 'ProductBundlesQuoteDataGroupHeaderView',

    initialize: function(options) {
        this.events = _.extend({}, this.events, options.def.events, {
            'click [name="gh-custom-action"]': 'actionClicked'
        });

        this._super('initialize', [options]);
    },

    /**
     * Click event
     */
    actionClicked: function() {
        app.alert.show('success_alert', {
            level: 'success',
            title: 'Group Header Button was clicked!'
        });
    },
})
```

This code will append a click event to our field named gh-custom-action and trigger the actionClicked method. Once completed, add your new row action to the

appropriate button group in $viewdefs['Quotes']['base']['view']['quote-data-group-header']['buttons']. For this example we will target a row action to the edit drop down that is identified in the arrays as having a name of "edit-dropdown". Once found, add your new array to the buttons index of that array.

./custom/modules/ProductBundles/clients/base/views/quote-data-group-header/quote-data-group-header.php

```
  <?php

$viewdefs['ProductBundles']['base']['view']['quote-data-group-
header'] = array(
    'buttons' => array(
        array(
            'type' => 'quote-data-actiondropdown',
            'name' => 'create-dropdown',
            'icon' => 'fa-plus',
            'no_default_action' => true,
            'buttons' => array(
                array(
                    'type' => 'rowaction',
                    'css_class' => 'btn-invisible',
                    'icon' => 'fa-plus',
                    'name' => 'create_qli_button',
                    'label' => 'LBL_CREATE_QLI_BUTTON_LABEL',
                    'tooltip' => 'LBL_CREATE_QLI_BUTTON_TOOLTIP',
                    'acl_action' => 'create',
                ),
                array(
                    'type' => 'rowaction',
                    'css_class' => 'btn-invisible',
                    'icon' => 'fa-plus',
                    'name' => 'create_comment_button',
                    'label' => 'LBL_CREATE_COMMENT_BUTTON_LABEL',
                    'tooltip' => 'LBL_CREATE_COMMENT_BUTTON_TOOLTIP',
                    'acl_action' => 'create',
                ),
            ),
        ),
        array(
            'type' => 'quote-data-actiondropdown',
            'name' => 'edit-dropdown',
            'icon' => 'fa-ellipsis-v',
            'no_default_action' => true,
            'buttons' => array(
                array(
```

```
                    'type' => 'rowaction',
                    'name' => 'edit_bundle_button',
                    'label' => 'LBL_EDIT_BUTTON',
                    'tooltip' => 'LBL_EDIT_BUNDLE_BUTTON_TOOLTIP',
                    'acl_action' => 'edit',
                ),
                array(
                    'type' => 'rowaction',
                    'name' => 'delete_bundle_button',
                    'label' => 'LBL_DELETE_GROUP_BUTTON',
                    'tooltip' => 'LBL_DELETE_BUNDLE_BUTTON_TOOLTIP',
                    'acl_action' => 'delete',
                ),
                array(
                    'type' => 'rowaction',
                    'name' => 'gh-custom-action',
                    'label' => 'LBL_GH_CUSTOM_ACTION',
                    'tooltip' => 'LBL_GH_CUSTOM_ACTION_TOOLTIP',
                    'acl_action' => 'edit',
                ),
            ),
        ),
    ),
    ...
);
```

Finally, create labels under Quotes for the label and tooltip indexes. To accomplish this, create a language extension:

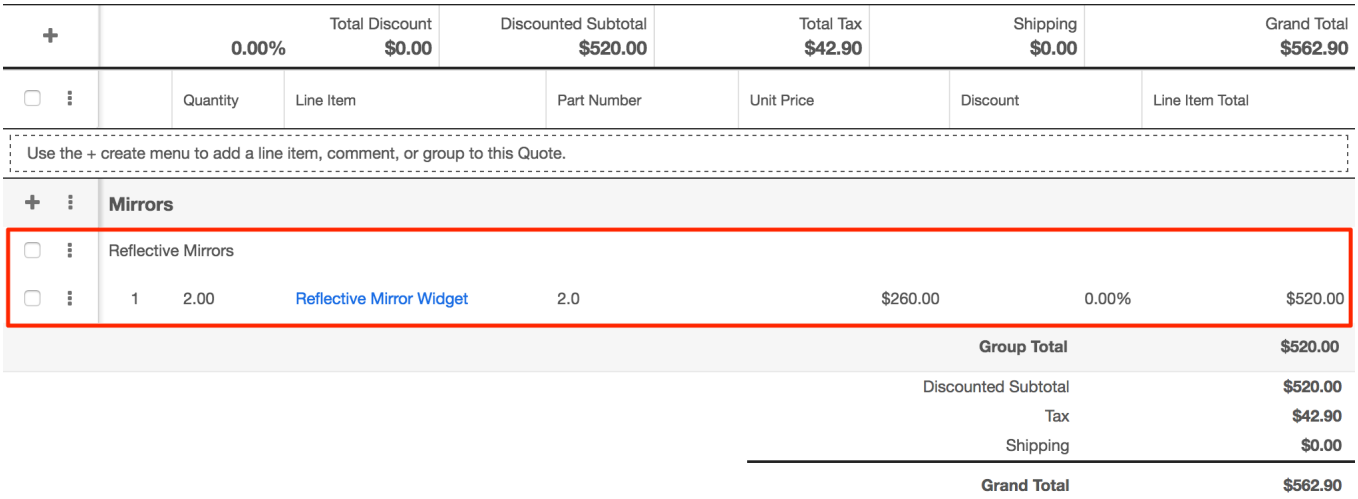
./custom/Extension/modules/Quotes/Ext/Language/en_us.gh-custom-action.php

```
  <?php

$mod_strings['LBL_GH_CUSTOM_ACTION'] = 'Custom Action';
$mod_strings['LBL_GH_CUSTOM_ACTION_TOOLTIP'] = 'Custom Action Tooltip'
;
```

Once the files are in place, navigate to Admin > Repair > Quick Repair and Rebuild. Your changes will now be reflected in the system.

Group List

The Group List contains the comments and selected quoted line items.



	Total Discount	Discounted Subtotal	Total Tax	Shipping	Grand Total
	0.00% $0.00	$520.00	$42.90	$0.00	$562.90

	Quantity	Line Item	Part Number	Unit Price	Discount	Line Item Total
Use the + create menu to add a line item, comment, or group to this Quote.						
Mirrors						
Reflective Mirrors						
1	2.00	Reflective Mirror Widget	2.0	$260.00	0.00%	$520.00

Group Total	$520.00
Discounted Subtotal	$520.00
Tax	$42.90
Shipping	$0.00
Grand Total	$562.90

Modifying Fields in the Group List

To modify the Group List fields, you can copy ./modules/Products/clients/base/views/quote-data-group-list/quote-data-group-list.php to ./custom/modules/Products/clients/base/views/quote-data-group-list/quote-data-group-list.php or you may create a metadata extension in ./custom/Extension/modules/Products/Ext/clients/base/views/quote-data-group-list/. Next, modify the $viewdefs['Products']['base']['view']['quote-data-group-list']['panels'] index to add or remove your preferred fields.

Example

The example below will remove the "Manufacturer Part Number" (products.mft_part_num) and append the "Vendor Part Number" (products.vendor_part_num) field to the Group List.

./custom/modules/Products/clients/base/views/quote-data-group-list/quote-data-group-list.php

```
  <?php

$viewdefs['Products']['base']['view']['quote-data-group-
list'] = array(
    'panels' => array(
        array(
            'name' => 'products_quote_data_group_list',
            'label' => 'LBL_PRODUCTS_QUOTE_DATA_LIST',
            'fields' => array(
```

```
        array(
            'name' => 'line_num',
            'label' => null,
            'widthClass' => 'cell-xsmall',
            'css_class' => 'line_num tcenter',
            'type' => 'line-num',
            'readonly' => true,
        ),
        array(
            'name' => 'quantity',
            'label' => 'LBL_QUANTITY',
            'widthClass' => 'cell-small',
            'css_class' => 'quantity',
            'type' => 'float',
        ),
        array(
            'name' => 'product_template_name',
            'label' => 'LBL_ITEM_NAME',
            'widthClass' => 'cell-large',
            'type' => 'quote-data-relate',
            'required' => true,
        ),
        array(
            'name' => 'vendor_part_num',
            'label' => 'LBL_VENDOR_PART_NUM',
            'type' => 'base',
        ),
        array(
            'name' => 'discount_price',
            'label' => 'LBL_DISCOUNT_PRICE',
            'type' => 'currency',
            'convertToBase' => true,
            'showTransactionalAmount' => true,
            'related_fields' => array(
                'discount_price',
                'currency_id',
                'base_rate',
            ),
        ),
        array(
            'name' => 'discount',
            'type' => 'fieldset',
            'css_class' => 'quote-discount-percent',
            'label' => 'LBL_DISCOUNT_AMOUNT',
            'fields' => array(
                array(
```

```
                            'name' => 'discount_amount',
                            'label' => 'LBL_DISCOUNT_AMOUNT',
                            'type' => 'discount',
                            'convertToBase' => true,
                            'showTransactionalAmount' => true,
                        ),
                        array(
                            'type' => 'discount-select',
                            'name' => 'discount_select',
                            'no_default_action' => true,
                            'buttons' => array(
                                array(
                                    'type' => 'rowaction',
                                    'name' => 'select_discount_amount_
button',
                                    'label' => 'LBL_DISCOUNT_AMOUNT',
                                    'event' => 'button:discount_select
_change:click',
                                ),
                                array(
                                    'type' => 'rowaction',
                                    'name' => 'select_discount_percent
_button',
                                    'label' => 'LBL_DISCOUNT_PERCENT',
                                    'event' => 'button:discount_select
_change:click',
                                ),
                            ),
                        ),
                    ),
                ),
                array(
                    'name' => 'total_amount',
                    'label' => 'LBL_LINE_ITEM_TOTAL',
                    'type' => 'currency',
                    'widthClass' => 'cell-medium',
                    'showTransactionalAmount' => true,
                    'related_fields' => array(
                        'total_amount',
                        'currency_id',
                        'base_rate',
                    ),
                ),
            ),
        ),
    ),
```

```
);
```

Next, create the LBL_VENDOR_PART_NUM label under Quotes as this is not included in the stock installation. To accomplish this, we will need to create a language extension:

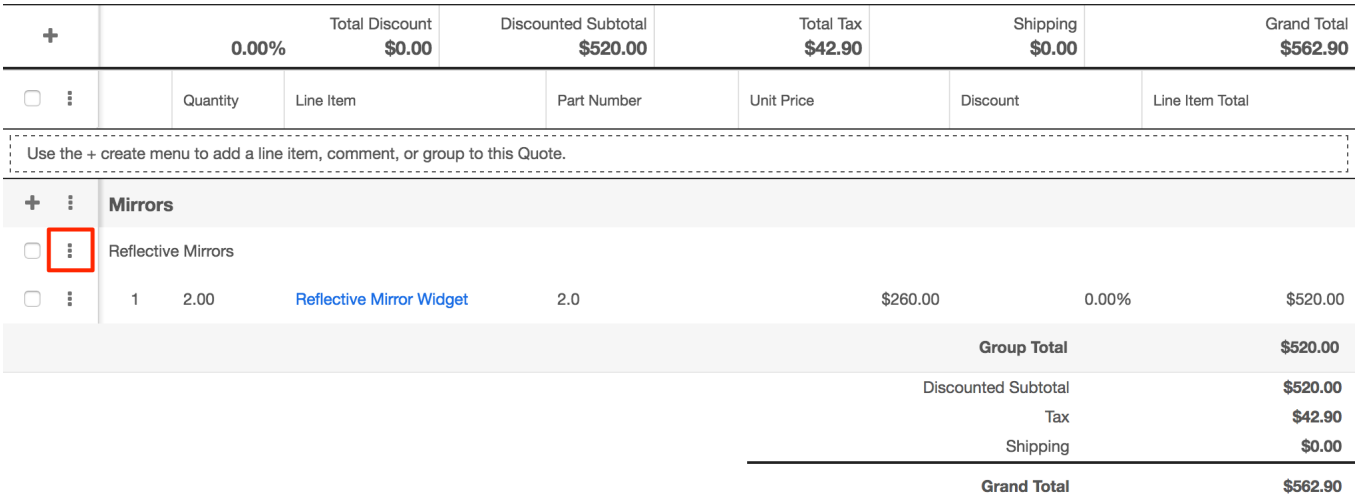
./custom/Extension/modules/Quotes/Ext/Language/en_us.vendor.php

```
  <?php

$mod_strings['LBL_VENDOR_PART_NUM'] = 'Vendor Part Number';
```

If adding a custom field from the Quotes module to the view, you will also need to add that field to the related_fields array as outlined in the Record View documentation below. Once the files are in place, navigate to Admin > Repair > Quick Repair and Rebuild. Your changes will now be reflected in the system.

Modifying Row Actions in the Group List

The Quotes Group List contains row actions to add/remove QLIs and comments. The actions are identified by, the vertical ellipsis icon buttons. The following section will outline how these items can be modified.



To modify the buttons in the Group List , you can copy ./modules/ProductBundles/clients/base/views/quote-data-group-list/quote-data-group-list.php to ./custom/modul

es/ProductBundles/clients/base/views/quote-data-group-list/quote-data-group-list.php or you may create a metadata extension in ./custom/Extension/modules/ProductBundles/clients/base/views/quote-data-group-list/. Next, modify the $viewdefs['ProductBundles']['base']['view']['quote-data-group-list']['selection'] index to add or remove your preferred actions.

Example

The example below will create a new view that extends the ProductBundlesQuoteDataGroupList view and append a row action to the Group List's vertical elipsis action list.

First, create your custom view type that will extend the ProductBundlesQuoteDataGroupList view. This will contain the JavaScript for your action.

./custom/modules/ProductBundles/clients/base/views/quote-data-group-list/quote-data-group-list.js

```
  ({
    extendsFrom: 'ProductBundlesQuoteDataGroupListView',

    initialize: function(options) {
        this.events = _.extend({}, this.events, options.def.events, {
            'click [name="gl-custom-action"]': 'actionClicked'
        });

        this._super('initialize', [options]);
    },

    /**
     * Click event
     */
    actionClicked: function() {
        app.alert.show('success_alert', {
            level: 'success',
            title: 'List Header Row Action was clicked!'
        });
    },
})
```

This code will append a click event to our field named gl-custom-action and trigger the actionClicked method. Once completed, add your new row action to the group list in the $viewdefs['ProductBundles']['base']['view']['quote-data-group-

list']['selection']['actions'] index.

./custom/modules/ProductBundles/clients/base/views/quote-data-group-list/quote-data-group-list.php

```
  <?php

$viewdefs['ProductBundles']['base']['view']['quote-data-group-
list'] = array(
    'selection' => array(
        'type' => 'multi',
        'actions' => array(
            array(
                'type' => 'rowaction',
                'name' => 'edit_row_button',
                'label' => 'LBL_EDIT_BUTTON',
                'tooltip' => 'LBL_EDIT_BUTTON',
                'acl_action' => 'edit',
            ),
            array(
                'type' => 'rowaction',
                'name' => 'delete_row_button',
                'label' => 'LBL_DELETE_BUTTON',
                'tooltip' => 'LBL_DELETE_BUTTON',
                'acl_action' => 'delete',
            ),
            array(
                'type' => 'rowaction',
                'name' => 'gl-custom-action',
                'label' => 'LBL_GL_CUSTOM_ACTION',
                'tooltip' => 'LBL_GL_CUSTOM_ACTION_TOOLTIP',
                'acl_action' => 'edit',
            ),
        ),
    ),
);
```

Finally, create labels under Quotes for the label and tooltip indexes. To accomplish this, create a language extension:

./custom/Extension/modules/Quotes/Ext/Language/en_us.gh-custom-action.php

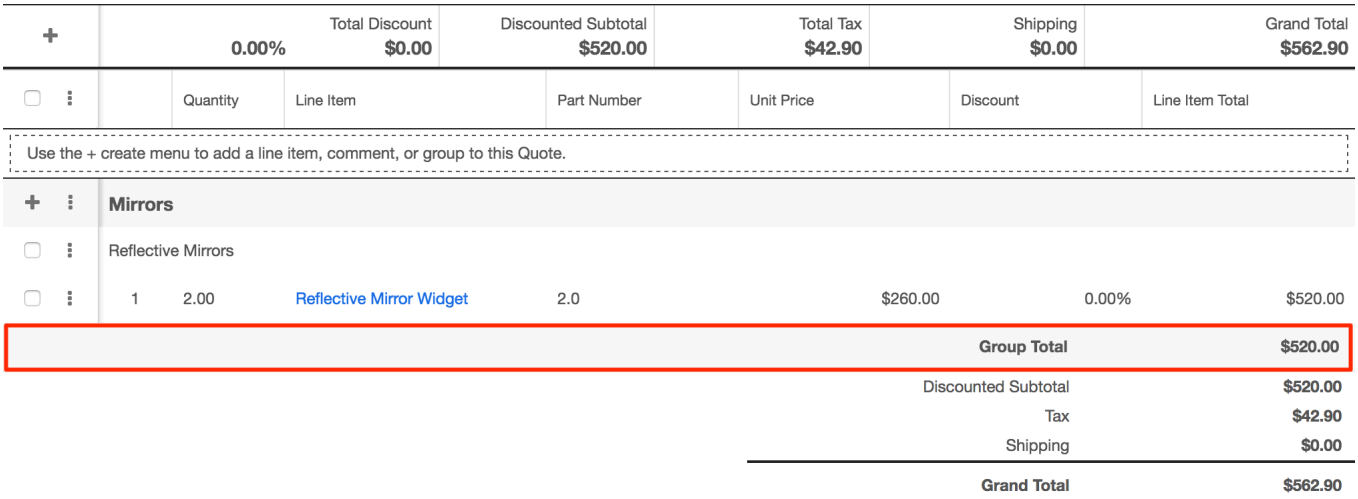
```
  <?php
```

```
$mod_strings['LBL_GL_CUSTOM_ACTION'] = 'Custom Action';
$mod_strings['LBL_GL_CUSTOM_ACTION_TOOLTIP'] = 'Custom Action Tooltip'
;
```

Once the files are in place, navigate to Admin > Repair > Quick Repair and Rebuild. Your changes will now be reflected in the system.

Group Footer

The Group Footer contains the total for each grouping of quoted line items.



Modifying Fields in the Group Footer

To modify the GroupFooter fields, you can copy ./modules/ProductBundles/clients/base/views/quote-data-group-footer/quote-data-group-footer.php to ./custom/modules/ProductBundles/clients/base/views/quote-data-group-footer/quote-data-group-footer.php or you may create a metadata extension in ./custom/Extension/modules/ProductBundles/clients/base/views/quote-data-group-footer/. Next, modify the $viewdefs['ProductBundles']['base']['view']['quote-data-group-footer'] index to add or remove your preferred fields.

Example

The example below will append the bundle stage (product_bundles.bundle_stage) field to the Group Footer. It's important to note that when adding additional fields, that changes to the corresponding .hbs file may be necessary to correct any formatting issues.

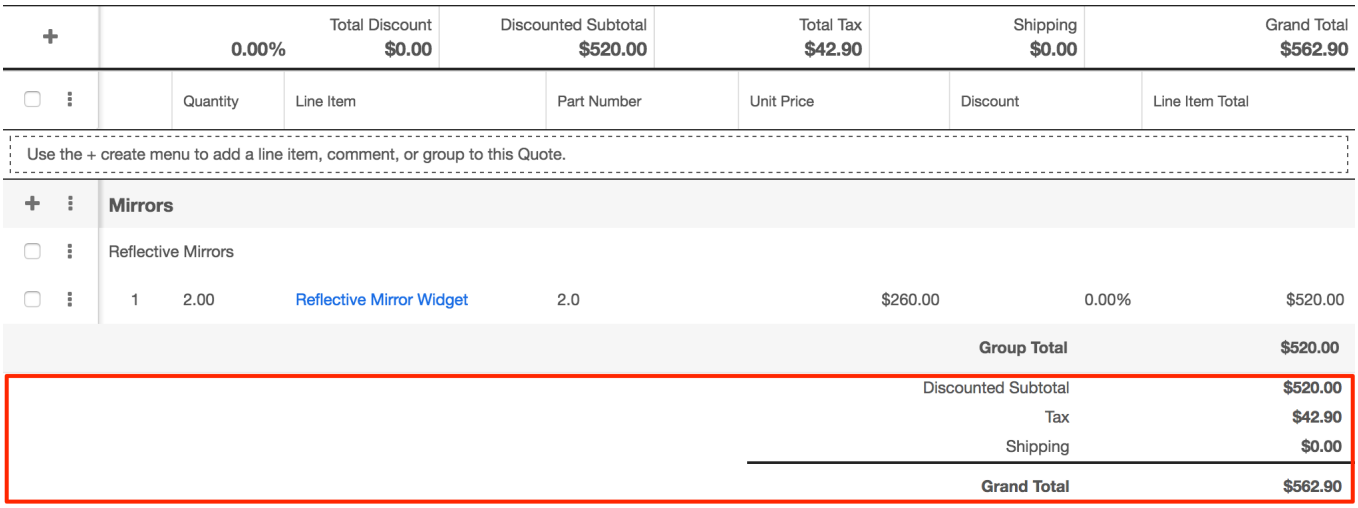
./custom/modules/ProductBundles/clients/base/views/quote-data-group-header/quote-data-group-footer.php

```
  <?php

$viewdefs['ProductBundles']['base']['view']['quote-data-group-footer'] = array(
    'panels' => array(
        array(
            'name' => 'panel_quote_data_group_footer',
            'label' => 'LBL_QUOTE_DATA_GROUP_FOOTER',
            'fields' => array(
                'bundle_stage',
                array(
                    'name' => 'new_sub',
                    'label' => 'LBL_GROUP_TOTAL',
                    'type' => 'currency',
                ),
            ),
        ),
    ),
);
```

If adding custom fields from the Product Bundles module to the view, you will also need to add that field to the related_fields array as outlined in the Record View documentation below. Once the files are in place, navigate to Admin > Repair > Quick Repair and Rebuild. Your changes will now be reflected in the system.

Grand Totals Footer

The Grand Total Footer contains the calculated totals for the quote. It mimics the information found in the Grand Total Header.



Modifying Fields in the Grand Totals Footer

To modify the Grand Totals Footer fields, you can copy ./modules/Quotes/clients/base/views/quote-data-grand-totals-footer/quote-data-grand-totals-footer.php to ./custom/modules/Quotes/clients/base/views/quote-data-grand-totals-footer/quote-data-grand-totals-footer.php or you may create a metadata extension in ./custom/Extension/modules/Quotes/clients/base/views/quote-data-grand-totals-footer/. Next, modify the $viewdefs['Quotes']['base']['view']['quote-data-grand-totals-footer']['panels'][0]['fields'] index to add or remove your preferred fields.

Example

The example below will remove the "Shipping" field (quotes.shipping) field from the layout.

./custom/modules/Quotes/clients/base/views/quote-data-grand-totals-footer/quote-data-grand-totals-footer.php

```
  <?php

$viewdefs['Quotes']['base']['view']['quote-data-grand-totals-
footer'] = array(
    'panels' => array(
        array(
            'name' => 'panel_quote_data_grand_totals_footer',
            'label' => 'LBL_QUOTE_DATA_GRAND_TOTALS_FOOTER',
            'fields' => array(
```

```
                'quote_num',
                array(
                    'name' => 'new_sub',
                    'type' => 'currency',
                ),
                array(
                    'name' => 'tax',
                    'type' => 'currency',
                    'related_fields' => array(
                        'taxrate_value',
                    ),
                ),
                array(
                    'name' => 'total',
                    'label' => 'LBL_LIST_GRAND_TOTAL',
                    'type' => 'currency',
                    'css_class' => 'grand-total',
                ),
            ),
        ),
    ),
);
```

If adding custom fields from the Quotes module to the view, you will also need to add that field to the related_fields array as outlined in the Record View documentation below. Once the files are in place, navigate to Admin > Repair > Quick Repair and Rebuild. Your changes will now be reflected in the system.

Record View

The record view for Quotes is updated and used like the standard Record View for all modules. The one major difference to note is that the JavaScript models used by the previous views above, are derived from the Model defined in the record view metadata.

Adding Custom Related Fields to Model

In the Record View metadata, the first panel panel_header containing the picture and name fields for the Quote record, contains a related_fields property in the name definition that defines the related Product Bundles and Products data that is pulled into the JavaScript model. When custom fields are added to the above mentioned views you will need to add them to the related_fields property in their respective related module so that they are properly loaded during the page load.

The following example adds the custom_product_bundle_field_c field from the Product Bundles module, and the custom_product_field_c from the Products module into the retrieved Model.

./custom/modules/Quotes/clients/base/views/record/record.php

```
  <?php
$viewdefs['Quotes']['base']['view']['record'] = array(
    ...
    'panels' => array(
        array(
            'name' => 'panel_header',
            'label' => 'LBL_PANEL_HEADER',
            'header' => true,
            'fields' => array(
                array(
                    'name' => 'picture',
                    'type' => 'avatar',
                    'size' => 'large',
                    'dismiss_label' => true,
                    'readonly' => true,
                ),
                array(
                   'name' => 'name',
                   'events' => array(
                       'keyup' => 'update:quote',
                   ),
                   'related_fields' => array(
                       array(
                           'name' => 'bundles',
                           'fields' => array(
                              'id',
                              'bundle_stage',
                              'currency_id',
                              'base_rate',
                              'currencies',
                              'name',
                              'deal_tot',
                              'deal_tot_usdollar',
                              'deal_tot_discount_percentage',
                              'new_sub',
                              'new_sub_usdollar',
                              'position',
                              'related_records',
                              'shipping',
                              'shipping_usdollar',
```

```
                            'subtotal',
                            'subtotal_usdollar',
                            'tax',
                            'tax_usdollar',
                            'taxrate_id',
                            'team_count',
                            'team_count_link',
                            'team_name',
                            'taxable_subtotal',
                            'total',
                            'total_usdollar',
                            'default_group',
                            'custom_product_bundle_field_c',
                            array(
                                'name' => 'product_bundle_items',
                                'fields' => array(
                                    'name',
                                    'quote_id',
                                    'description',
                                    'quantity',
                                    'product_template_name',
                                    'product_template_id',
                                    'deal_calc',
                                    'mft_part_num',
                                    'discount_price',
                                    'discount_amount',
                                    'tax',
                                    'tax_class',
                                    'subtotal',
                                    'position',
                                    'currency_id',
                                    'base_rate',
                                    'discount_select',
                                    'custom_product_field_c',
                                ),
                                'max_num' => -1,
                            ),
                        ),
                        'max_num' => -1,
                        'order_by' => 'position:asc',
                    ),
                ),
            ),
        ),
    ),
    ...
```

```
  ),
);
```

Quote PDFs

Ungrouped Quoted Line Items and Notes

As of Sugar 7.9, the quotes module allows users to add quoted line items and notes without first adding a group. Adding at least one group to your quote was mandatory in earlier versions. This document will cover how to update your custom PDF templates to support ungrouped QLIs and notes. Ungrouped items are technically added to a default group. The goal is to exclude this group when no items exist in it.

PDF Manager Templates

In your PDF Manager templates, you may have code similar to what is shown below to iterate over the groups in a quote:

```
  {foreach from=$product_bundles item="bundle"}
    ....
{/foreach}
```

To correct this for 7.9, we will need to add an if statement to check to see if the group is empty. If it is empty, it will be ignored in your PDF. The benefit is that it will also exclude any other empty groups in your quote and clean the generated PDF.

```
  {foreach from=$product_bundles item="bundle"}
    {if $bundle.products|@count}
        ....
    {/if}
{/foreach}
```

Smarty Templates

In Smarty templates, you may have code similar to what is shown below to iterate over the groups in a quote:

```
  {literal}{foreach from=$product_bundles item="bundle"}{/literal}
```

```
    ...
{literal}{/foreach}{/literal}
```

To correct this for 7.9, we will need to add an if statement to check to see if the group is empty. If it is empty, it will be ignored in your PDF. The benefit is that it will also exclude any other empty groups in your quote and clean the generated PDF.

```
  {literal}{foreach from=$product_bundles item="bundle"}{/literal}
    {literal}{if $bundle.products|@count}{/literal}
        ...
    {literal}{/if}{/literal}
{literal}{/foreach}{/literal}
```

Creating Custom PDF Templates

With Sugar, there are generally two routes that can be taken to create custom PDF templates. The first and most recommended route is to use the PDF Manager found in the Admin > PDF Manager. The second route is to extend the Sugarpdf class and write your own template using PHP and the TCPDF library. This method should only be used if you are unable to accomplish your business needs through the PDF Manager.

Creating the TCPDF Template

The first step is to create your custom TCPDF template in ./custom/modules/Quotes/sugarpdf/. For our example, we will extend QuotesSugarpdfStandard, found at ./modules/Quotes/sugarpdf/sugarpdf.standard.php, to a new file in ./custom/modules/Quotes/sugarpdf/ to create a custom invoice. The QuotesSugarpdfStandard class sets up our general quote requirements and extends the Sugarpdf class. Technical documentation on templating can be found in the Sugar PDF documentation. Our class will be named QuotesSugarpdfCustomInvoice as the naming must be in the format of <module>Sugarpdf<pdf view>.

./custom/modules/Quotes/sugarpdf/sugarpdf.custominvoice.php

```
  <?php

if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry Poin
t');
```

```
require_once('modules/Quotes/sugarpdf/sugarpdf.standard.php');

class QuotesSugarpdfCustomInvoice extends QuotesSugarpdfStandard
{
    function preDisplay()
    {
        global $mod_strings, $timedate;
        parent::preDisplay();

        $quote[0]['TITLE'] = $mod_strings['LBL_PDF_INVOICE_NUMBER'];
        $quote[1]['TITLE'] = $mod_strings['LBL_PDF_QUOTE_DATE'];
        $quote[2]['TITLE'] = $mod_strings['LBL_PURCHASE_ORDER_NUM'];
        $quote[3]['TITLE'] = $mod_strings['LBL_PAYMENT_TERMS'];
        $quote[0]['VALUE']['value'] = format_number_display($this->bea
n->quote_num, $this->bean->system_id);
        $quote[1]['VALUE']['value'] = $timedate->nowDate();
        $quote[2]['VALUE']['value'] = $this->bean->purchase_order_num;
        $quote[3]['VALUE']['value'] = $this->bean->payment_terms;
        // these options override the params of the $options array.
        $quote[0]['VALUE']['options'] = array();
        $quote[1]['VALUE']['options'] = array();
        $quote[2]['VALUE']['options'] = array();
        $quote[3]['VALUE']['options'] = array();

        $html = $this->writeHTMLTable($quote, true, $this->headerOptio
ns);
        $this->SetHeaderData(PDF_HEADER_LOGO, PDF_HEADER_LOGO_WIDTH, $
mod_strings['LBL_PDF_INVOICE_TITLE'], $html);
    }

    /**
     * This method build the name of the PDF file to output.
     */
    function buildFileName()
    {
        global $mod_strings;

        $fileName = preg_replace("#[^A-
Z0-9\-_\.]#i", "_", $this->bean->shipping_account_name);

        if (!empty($this->bean->quote_num)) {
            $fileName .= "_{$this->bean->quote_num}";
        }

        $fileName = $mod_strings['LBL_INVOICE'] . "_{$fileName}.pdf";
```

```
        if (isset($_SERVER['HTTP_USER_AGENT']) && preg_match("/MSIE/",
 $_SERVER['HTTP_USER_AGENT'])) {
            //$fileName = $locale->translateCharset($fileName, $locale
->getExportCharset());
            $fileName = urlencode($fileName);
        }

        $this->fileName = $fileName;
    }
}
```

Next, register the layout. This is a two step process. The first step is to create ./custom/modules/Quotes/Layouts.php which will map our view action to the physical PHP file.

./custom/modules/Quotes/Layouts.php

```
  <?php

$layouts['CustomInvoice'] = 'custom/modules/Quotes/sugarpdf/sugarpdf.c
ustominvoice.php';
```

The second step is to update the layouts_dom dropdown list by navigating to Admin > Dropdown Editor. Once there, create a new entry in the list with an Item Name of "CustomInvoice" and a Display Label of your choice. It's also recommended to remove the Quote and Invoice Templates if they are not being used to avoid confusion. This will create a ./custom/Extension/application/Ext/Language/en_us.sugar_layouts_dom.php file that should look similar to:

./custom/Extension/application/Ext/Language/en_us.sugar_layouts_dom.php

```
  <?php

$app_list_strings['layouts_dom']=array (
  'CustomInvoice' => 'Custom Invoice',
);
```

Once the layout is registered, we need to extend the pdfaction field for the Quotes

module to render our custom pdf templates in the record view. An example of this is shown below:

./custom/modules/Quotes/clients/base/fields/pdfaction/pdfaction.js

```
/**
 * @class View.Fields.Base.Quotes.PdfactionField
 * @alias SUGAR.App.view.fields.BaseQuotesPdfactionField
 * @extends View.Fields.Base.PdfactionField
 */
({
    extendsFrom: 'PdfactionField',

    /**
     * @inheritdoc
     * Create PDF Template collection in order to get available templa
te list.
     */
    initialize: function(options) {
        this._super('initialize', [options]);
    },

    /**
     * Define proper filter for PDF template list.
     * Fetch the collection to get available template list.
     * @private
     */
    _fetchTemplate: function() {
        this.fetchCalled = true;
        var collection = this.templateCollection;
        collection.filterDef = {'$and': [{
            'base_module': this.module
        }, {
            'published': 'yes'
        }]};
        collection.fetch({
            success: function(){
                var models = [];
                _.each(app.lang.getAppListStrings('layouts_dom'), func
tion(template, id){
                    var model = new Backbone.Model({
                        id: id,
                        name: template
                    });
                    models.push(model);
```

```
                });
                collection.add(models);
            }
        });
    },

    /**
     * Build download link url.
     *
     * @param {String} templateId PDF Template id.
     * @return {string} Link url.
     * @private
     */
    _buildDownloadLink: function(templateId) {
        var sugarpdf = this._isUUID(templateId)? 'pdfmanager' : templa
teId;
        var urlParams = $.param({
            'action': 'sugarpdf',
            'module': this.module,
            'sugarpdf': sugarpdf,
            'record': this.model.id,
            'pdf_template_id': templateId
        });
        return '?' + urlParams;
    },

    /**
     * Build email pdf link url.
     *
     * @param {String} templateId PDF Template id.
     * @return {string} Email pdf url.
     * @private
     */
    _buildEmailLink: function(templateId) {
        var sugarpdf = this._isUUID(templateId)? 'pdfmanager' : templa
teId;
        return '#' + app.bwc.buildRoute(this.module, null, 'sugarpdf',
 {
                'sugarpdf': sugarpdf,
                'record': this.model.id,
                'pdf_template_id': templateId,
                'to_email': '1'
            });
    },

    /**
```

```
     * tests to see if a templateId is a uuid or template name.
     *
     * @param {String} templateId PDF Template id
     * @return {boolean} true if uuid, false if not
     * @private
     */
    _isUUID: function(templateId) {
        var regex = /^[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-
f]{4}-[0-9a-f]{12}$/i;
        return regex.test(templateId);
    }
})
```

Once the files are in place, navigate to Admin > Repair > Quick Repair and Rebuild. Your changes will now be reflected in the system and navigating to a url in the format of index.php?module=Quotes&record=<record id>&action=sugarpdf&sugarpdf=CustomInvoice will generate the pdf document.

Hiding PDF Buttons

In some circumstances, users may not have a need to generate PDFs from Quotes. If this should occur, a developer can copy ./modules/Quotes/clients/base/views/record/record.php to ./custom/modules/Quotes/clients/base/views/record/record.php if it doesn't already exist. Next, find the array with a name of main_dropwdown in the $viewdefs['Quotes']['base']['view']['record']['buttons'] index. Once found, you will then need to locate the buttons index within that array. You will then need to remove the following arrays from that index:

```
  array(
    'type' => 'pdfaction',
    'name' => 'download-pdf',
    'label' => 'LBL_PDF_VIEW',
    'action' => 'download',
    'acl_action' => 'view',
),
array(
    'type' => 'pdfaction',
    'name' => 'email-pdf',
    'label' => 'LBL_PDF_EMAIL',
    'action' => 'email',
    'acl_action' => 'view',
),
```

Your file should look similar to what is shown below:

custom/modules/Quotes/clients/base/views/record/record.php

```
  <?php

$viewdefs['Quotes']['base']['view']['record'] = array(
    'buttons' => array(
        array(
            'type' => 'button',
            'name' => 'cancel_button',
            'label' => 'LBL_CANCEL_BUTTON_LABEL',
            'css_class' => 'btn-invisible btn-link',
            'showOn' => 'edit',
            'events' => array(
                'click' => 'button:cancel_button:click',
            ),
        ),
        array(
            'type' => 'rowaction',
            'event' => 'button:save_button:click',
            'name' => 'save_button',
            'label' => 'LBL_SAVE_BUTTON_LABEL',
            'css_class' => 'btn btn-primary',
            'showOn' => 'edit',
            'acl_action' => 'edit',
        ),
        array(
            'type' => 'actiondropdown',
            'name' => 'main_dropdown',
            'primary' => true,
            'showOn' => 'view',
            'buttons' => array(
                array(
                    'type' => 'rowaction',
                    'event' => 'button:edit_button:click',
                    'name' => 'edit_button',
                    'label' => 'LBL_EDIT_BUTTON_LABEL',
                    'acl_action' => 'edit',
                ),
                array(
                    'type' => 'shareaction',
                    'name' => 'share',
                    'label' => 'LBL_RECORD_SHARE_BUTTON',
                    'acl_action' => 'view',
                ),
```

```
                array(
                    'type' => 'divider',
                ),
                array(
                    'type' => 'convert-to-opportunity',
                    'event' => 'button:convert_to_opportunity:click',
                    'name' => 'convert_to_opportunity_button',
                    'label' => 'LBL_QUOTE_TO_OPPORTUNITY_LABEL',
                    'acl_module' => 'Opportunities',
                    'acl_action' => 'create',
                ),
                array(
                    'type' => 'divider',
                ),
                array(
                    'type' => 'rowaction',
                    'event' => 'button:historical_summary_button:click
',
                    'name' => 'historical_summary_button',
                    'label' => 'LBL_HISTORICAL_SUMMARY',
                    'acl_action' => 'view',
                ),
                array(
                    'type' => 'rowaction',
                    'event' => 'button:audit_button:click',
                    'name' => 'audit_button',
                    'label' => 'LNK_VIEW_CHANGE_LOG',
                    'acl_action' => 'view',
                ),
                array(
                    'type' => 'rowaction',
                    'event' => 'button:find_duplicates_button:click',
                    'name' => 'find_duplicates_button',
                    'label' => 'LBL_DUP_MERGE',
                    'acl_action' => 'edit',
                ),
                array(
                    'type' => 'divider',
                ),
                array(
                    'type' => 'rowaction',
                    'event' => 'button:delete_button:click',
                    'name' => 'delete_button',
                    'label' => 'LBL_DELETE_BUTTON_LABEL',
                    'acl_action' => 'delete',
                ),
```

```
            ),
        ),
        array(
            'name' => 'sidebar_toggle',
            'type' => 'sidebartoggle',
        ),
    ),
    ...
);
```

Once the files are in place, navigate to Admin > Repair > Quick Repair and Rebuild. Your changes will now be reflected in the system.

SugarBPM

Introduction

SugarBPM™ automation suite is the next-generation workflow management tool for Sugar. Introduced in 7.6, SugarBPM is loosely based on BPMN process notation standards and provides a simple drag-and-drop user interface along with an intelligent flow designer to allow for the creation of easy yet powerful process definitions.

Note: SugarBPM™ is not available for Sugar Professional.

SugarBPM enables administrators to streamline common business processes by managing approvals, sales processes, call triaging, and more. The easy-to-use workflow tool adds advanced BPM functionality to the core Sugar software stack.

A business process is a set of logically related tasks that are performed in order to achieve a specific organizational goal. It presents all of the tasks that must be completed in a simplified and streamlined format. SugarBPM empowers Sugar administrators, allowing for the automation of vital business processes for their organization. The SugarBPM automation suite features an extensive toolbox of modules that provide the ability to easily create digital forms and map out fully functioning workflows.

SugarBPM Modules

SugarBPM is broken down into four distinct modules, each with its own area of responsibility. Three of the four SugarBPM modules have access control settings that restrict its visibility to System Administrators, Module Administrators, and

Module Developers.

Process Definitions

A process definition defines the steps in an overall business process. Process definitions are created by either a Sugar administrator, a module Administrator or a module Developer. The process definition consists of a collection of activities and their relationships, criteria to indicate the start and end of the process, and information about the individual activities (e.g., participants) contained within the business process.

Business Rules

A business rule is a reusable set of conditions and outcomes that can be embedded in a process definition. The set of rules may enforce business policy, make a decision, or infer new data from existing data. For example, if Sally manages all business opportunities of $10,000 or more, and Chris manages all business opportunities under $10,000, a business rule can be created and used by all relevant process definitions based on the same target module to ensure that the assignment policy is respected. In the case of an eventual personnel change, only the business rule will need to be edited to affect all related processes.

Email Templates

A process email template is required in order to include a Send Message event in a process definition. The SugarCRM core product includes several places where email templates can be created for different purposes, but SugarBPM requires all sent messages to be created via the Process Email Templates module.

Processes

A process is a running instance of a process definition. A single process begins every time a process definition meeting certain criteria is executed. For example, a single process definition could be created to automate quote approvals, but because users may engage in several quote approvals per day, each approval will be represented by a separate process instance, all governed by the single process definition.

Database Tables

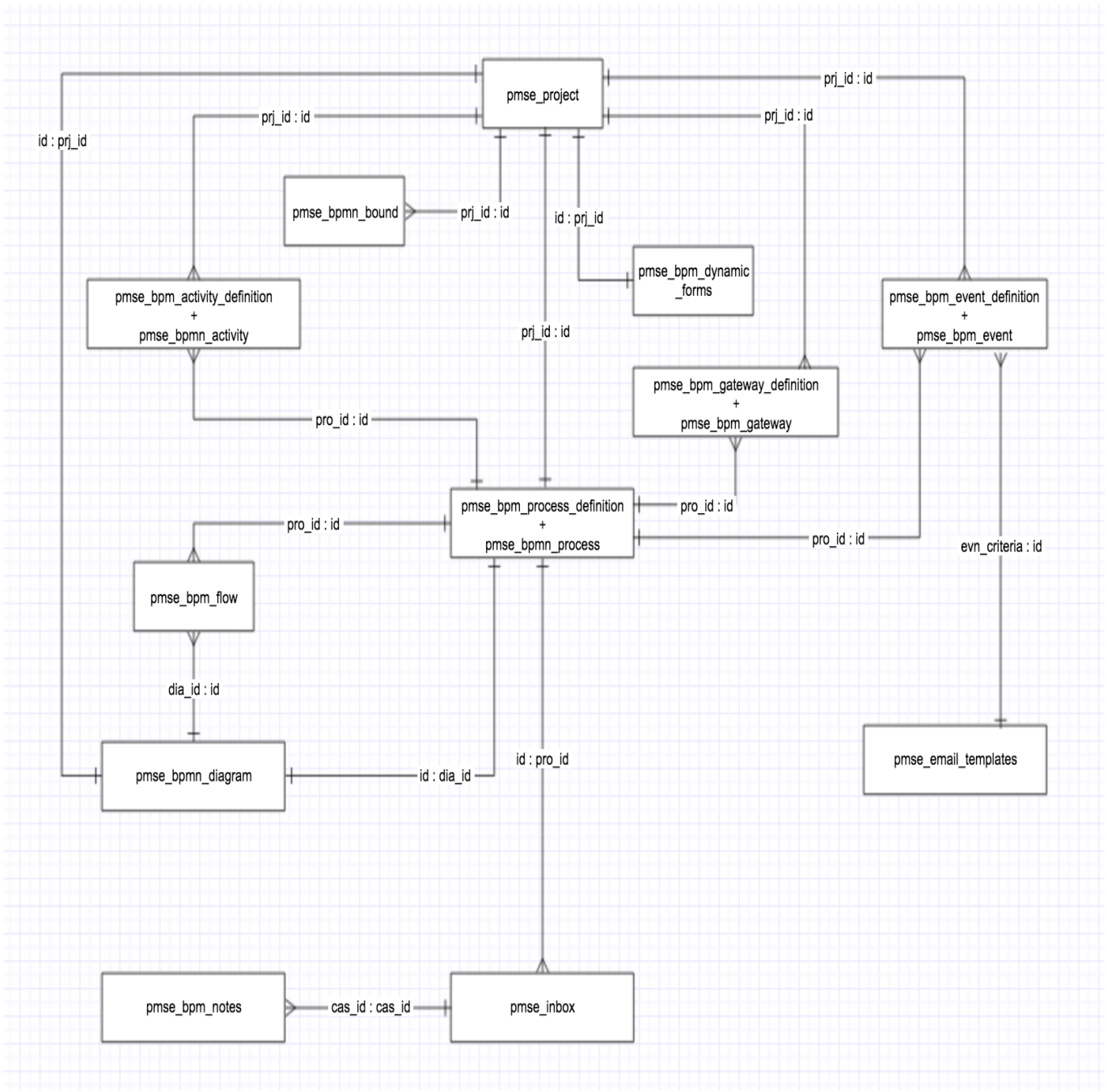
Information surrounding various portions of SugarBPM can be found in the following SugarBPM database tables:

Table name	Description
pmse_bpm_access_management	This table is not used.
pmse_bpm_activity_definition	Holds definition data for an activity. Maps directly to the pmse_bpmn_activity table and may, in the future, be merged with the pmse_bpmn_activity table to prevent forked data backends.
pmse_bpm_activity_step	This table is not used.
pmse_bpm_activity_user	This table is not used.
pmse_bpm_config	This table is not used.
pmse_bpm_dynamic_forms	Holds module specific form information - basically viewdefs for a module - for a process.
pmse_bpm_event_definition	Holds definition data for an event. Maps directly to the pmse_bpmn_event table and may, in the future, be merged with the pmse_bpmn_event table to prevent forked data backends.
pmse_bpm_flow	Holds information about triggered process flows as well as state of any process that is currently running.
pmse_bpm_form_action	Holds information about a process that has been acted upon.
pmse_bpm_gateway_definition	Holds definition data for a gateway. Maps directly to the pmse_bpmn_event table and may, in the future, be merged with the pmse_bpmn_gateway table to prevent forked data backends.
pmse_bpm_group	This table is not used.
pmse_bpm_group_user	This table is not used.
pmse_bpm_notes	Holds notes saved for a process record.
pmse_bpm_process_definition	Holds definition data for a process definition. Maps directly to the pmse_bpmn_process table and may, in the future, be merged with the pmse_bpmn_process table to prevent forked data backends.
pmse_bpm_related_dependency	Holds information about dependencies

Table name	Description
	related to an event
pmse_bpm_thread	Holds information related to process threads for running processes.
pmse_bpmn_activity	Please see pmse_bpm_activity_definition
pmse_bpmn_artifact	Holds BPM artifact information, like Comments on a process diagram. At present, SugarBPM only supports the text annotation artifact.
pmse_bpmn_bound	Data related to element boundaries on the diagram.
pmse_bpmn_data	This table is not used.
pmse_bpmn_diagram	Data related to a process definition's diagram.
pmse_bpmn_documentation	This table is not used.
pmse_bpmn_event	Please see pmse_bpm_event_definition
pmse_bpmn_extension	This table is not used.
pmse_bpmn_flow	Holds information about the connecting flows of a diagram.
pmse_bpmn_gateway	Please see pmse_bpm_gateway_definition
pmse_bpmn_lane	This table is not used.
pmse_bpmn_laneset	This table is not used.
pmse_bpmn_participant	This table is not used.
pmse_bpmn_process	Please see pmse_bpm_process_definition
pmse_business_rules	Holds business rule record data.
pmse_emails_templates	Holds email template record data.
pmse_inbox	Holds information about processes that are currently running or that have completed.
pmse_project	Holds process definition record information.

Relationships

The following Entity Relationship Diagram highlights the relationships between the various SugarBPM tables.



-Relationships" width="1654" recordid="e3710de4-b5b1-11e6-a3de-005056bc231a" style="width: nullpx; height: NaNpx;" />

Flows

When a process definition is created the following 5 tables get populated:

- pmse_bpm_dynamic_forms contains dyn_view_defs which is the JSON encoded string of module-specific form information
- pmse_bpm_process_definition contains the module associated with and the status of a process definition

- pmse_bpmn_diagram contains process definition name as well and a diagram uid
- pmse_bpmn_process contains process definition name. The same Id is shared between pmse_bpm_process_definition and pmse_bpmn_process dia_id is a foreign key related to pmse_bpmn_diagram
- pmse_project contains process definition name, project id, module, status. Other tables have a foreign key of prj_id in them.

Upon adding a start/end/send message event:

- pmse_bpmn_event contains event name
- pmse_bpm_event_definition Shares id with pmse_bpmn_event
- pmse_bpmn_bound

When Settings is changed to "New Records Only"

- pmse_bpm_related_dependency

Upon adding an action or activity:

- pmse_bpm_activity_definition contains the action name
    - The act_fields column contains the JSON encoded list of fields which will be changed
- pmse_bpm_activity contains action name. Shares id with pmse_bpm_activity_definition. act_script_type contains the type of action e.g. CHANGE_FIELD
- pmse_bpmn_bound

Upon adding a connector between start event and action:

- pmse_bpmn_flow contains origin and destination info

Upon adding a Gateway:

- pmse_bpm_gateway_definition
- pmse_bpm_gateway shares id with pmse_bpm_gateway_definition
- pmse_bpmn_flow flo_condition contains JSON encoded string of conditions required to proceed ahead with an action or activity
- pmse_bpmn_bound

When a process runs:

- pmse_inbox cas_status indicates the status of a process
- pmse_bpm_thread
- pmse_bpm_flow Stores the entire flow that the process elements went through (so if 3 elements and 2 connectors are in the process definition

then 5 records will exist)

After Process has run and appears in Process Management:

- pmse_bpm_case_data
- pmse_bpm_form_action contains frm_action, cas_pre_data , and cas_data

Add a comment:

- pmse_bpmn_artifact contains the comment
- pmse_bpm_bound

Add a business rule to a process definition:

- pmse_business_rules rst_source_definition contains the conditions of the business rule
- pmse_bpm_activity_definition act_fields contains the business rule id

Add an email template to a process definition:

- pmse_email_templates
- pmse_bpm_event_definition evn_criteria contains the email template id

Extension and Customization

In Sugar 7.8, Sugar introduced the Process Manager library to support extending SugarBPM in an upgrade-safe way. For more information on extending SugarBPM, or on using the Process Manager library, please read the Process Manager documentation.

Caveats

In Sugar 7.8, the Process Manager library brought Registry Object to maintain the state of SugarBPM processes during a PHP Process. This change introduced a limitation in SugarBPM that prevents the same process definition from running on multiple records inside the same PHP process. For certain customizations in Sugar, if you are updating multiple records in a module using SugarBean, you may want them to trigger the defined process definitions for each record.

The following code can be added to your customization to allow for that functionality:

```
  use Sugarcrm\Sugarcrm\ProcessManager\Registry;

//custom code
```

```
Registry\Registry::getInstance()->drop('triggered_starts');

$bean->save();
```

The 'triggered_starts' registry contains the Start Event IDs that have been triggered previously in the PHP process, thus allowing the SugarBean::save() method to trigger SugarBPM and go through the same Start Event again.

Extending SugarBPM (Process Manager)

Introduction

From its earliest version, Sugar has been a tool that allows for customizing and extending in a way that allows developers to shape and mold it to a specific business need. Most components of Sugar are customizable or extensible through the use of custom classes (for custom functionality) or extensions (for custom metadata). However, a recent addition to the product, the SugarBPM™ automation suite, did not fit this paradigm. In response, the Process Manager Library was introduced to give developers the ability to customize any portion of SugarBPM where object instantiation was previously handled using the 'new' operator.

Note: SugarBPM™ is not available in Sugar Professional. For an introduction to the SugarBPM modules and architecture, please refer to the SugarBPM documentation in this guide.

Process Manager

By way of ProcessManager\Factory we can now develop custom versions of most any PMSE* class and have that class used in place of the core PMSE* class.

Limitations

The current implementation of the Process Manager Library only handles server-side customizations by way of the Factory class. These customizations can be applied to any instantiable pmse_* object, however, static classes and singleton classes, such as the following, are not affected by the library:

- PMSE
- PMSEEngineUtils
- PMSELogger

While some effort has been made to allow for the creation of custom actions in the Process Definition designer, as of Sugar 7.8, that is the only client customization allowance that will be implemented.

Library Structure

The Process Manager Library is placed under the Sugarcrm\Sugarcrm\ProcessManager namespace and contains the following directories and classes:

- Factory
- Exception/
    - BaseException
    - DateTimeException
    - ExceptionInterface
    - ExecutionException
    - InvalidDataException
    - NotAuthorizedException
    - RuntimeException
- Field/
    - Evaluator/
        - AbstractEvaluator
        - Base
        - Currency
        - Datetime
        - Decimal
        - EvaluatorInterface
        - Int
        - Multienum
        - Relate
- Registry/
    - Registry
    - RegistryInterface

Examples

Getting a SugarBPM object

Almost all objects in SugarBPM can be instantiated through the Process Manager Factory. When loading a class, the Factory getPMSEObject method will look in the following directories as well as the custom versions of these directories for files that match the object name being loaded:

- modules/pmse_Business_Rules/
- modules/pmse_Business_Rules/clients/base/api/

- modules/pmse_Emails_Templates/
- modules/pmse_Emails_Templates/clients/base/api/
- modules/pmse_Inbox/clients/base/api/
- modules/pmse_Inbox/engine/
- modules/pmse_Inbox/engine/parser/
- modules/pmse_Inbox/engine/PMSEElements/
- modules/pmse_Inbox/engine/PMSEHandlers/
- modules/pmse_Inbox/engine/PMSEPreProcessor/
- modules/pmse_Inbox/engine/wrappers/
- modules/pmse_Project/clients/base/api/
- modules/pmse_Project/clients/base/api/wrappers/
- modules/pmse_Project/clients/base/api/wrappers/PMSEObservers/

What this means is that virtually any class that is found by name in these directories can be instantiated and returned via the getPMSEObject() method. This also means that customizations to any of these classes can be easily implemented by creating a Custom version of the class under the appropriate ./custom directory path.

For example, to create a custom version of PMSEProjectWrapper, you would create a CustomPMSEProjectWrapper class at ./custom/modules/pmse_Project/clients/base/api/wrappers/CustomPMSEProjectWrapper.php. This allows you to have full control of your instance of SugarBPM while giving you the flexibility to add any classes you want and consuming them through the factory.

./custom/modules/pmse_Project/clients/base/api/wrappers/CustomPMSEProjectWrapper.php

```
  <?php

require_once 'modules/pmse_Project/clients/base/api/wrappers/PMSEProjectWrapper.php';

class CustomPMSEProjectWrapper extends PMSEProjectWrapper
{
}
```

This can now be consumed by simply calling the getPMSEObject() method:

```
  <?php

// Set the process manager library namespace
use \Sugarcrm\Sugarcrm\ProcessManager\Factory;
```

```
// Get our wrapper
$wrapper = ProcessManager\Factory::getPMSEObject('PMSEProjectWrapper')
;
```

Getting a SugarBPM Element Object

Element objects in SugarBPM generally represent some part of the Process engine, and often time map to design elements. All Element objects can be found in the modules/pmse_Inbox/engine/PMSEElements/ directory.

Getting a SugarBPM Element object can be done easily by calling the getElement() method of the Process Manager Factory:

```
  <?php

use \Sugarcrm\Sugarcrm\ProcessManager\Factory;

// Get a default PMSEElement object
$element = ProcessManager\Factory::getElement();
```

To get a specific object, you can pass the object name to the getElement() method:

```
  $activity = ProcessManager\Factory::getElement('PMSEActivity');
```

Alternatively, you can remove the PMSE prefix and call the method with just the element name:

```
  $gateway = ProcessManager\Factory::getElement('Gateway');
```

To create a custom SugarBPM Element, you can simply create a new file at ./custom/modules/pmse_Inbox/engine/PMSEElements/ where the file name is prefixed with Custom.

./custom/modules/pmse_Inbox/engine/PMSEElements/CustomPMSEScriptTask.php

```
  <?php

require_once 'modules/pmse_Inbox/engine/PMSEElements/PMSEScriptTask.ph
```

```
p';

use Sugarcrm\Sugarcrm\ProcessManager;

class CustomPMSEScriptTask extends PMSEScriptTask
{
}
```

Note: All SugarBPM Element classes must implement the PMSERunnable interface. Failure to implement this interface will result in an exception when using the Factory to get an Element object.

To get a custom ScriptTask object via the Factory, just call getElement():

```
$obj = ProcessManager\Factory::getElement('ScriptTask');
```

To make your own custom widget element, you can implement the PMSERunnable interface:

./custom/modules/pmse_Inbox/engine/PMSEElements/CustomPMSEWidget.php

```
  <?php

require_once 'modules/pmse_Inbox/engine/PMSEElements/PMSERunnable.php'
;

use Sugarcrm\Sugarcrm\ProcessManager;

class CustomPMSEWidget implements PMSERunnable
{
}
```

To get a Widget object via the Factory, just call getElement():

```
  $obj = ProcessManager\Factory::getElement('Widget');
```

Getting and Using a Process Manager Field Evaluator Object

Process Manager Field Evaluator objects determine if a field on a record has changed and if a field on a record is empty according to the field type. The purpose of these classes can grow to include more functionality in the future.

Getting a field evaluator object is also done through the Factory:

```
  <?php

use \Sugarcrm\Sugarcrm\ProcessManager\Factory;

// Get a Datetime Evaluator. Type can be date, time, datetime or datet
imecombo
$eval = ProcessManager\Factory::getFieldEvaluator(['type'  => 'date'])
;
```

A more common way of getting a field evaluator object is to pass the field definitions for a field on a SugarBean into the Factory:

```
  $eval = ProcessManager\Factory::getFieldEvaluator($bean ->field_defs
[$field]);
```

```
$eval = ProcessManager\Factory::getFieldEvaluator($bean ->
field_defs[$field]);
```

Once the evaluator object is fetched, you may initialize it with properties:

```
  /*
 This is commonly used for field change evaluations
 */

// $bean is a SugarBean object
// $field is the name of the field being evaluated
// $data is an array of data that is used in the evaluation
$eval ->init($bean, $field, $data);

// Has the field changed?
$changed  = $eval->hasChanged();
```

Or you can set properties onto the evaluator:

```
  /*
 This is commonly used for empty evaluations
 */

// Set the bean onto the evaluator
$eval->setBean($bean);

// And set the name onto the evaluator
$eval->setName($field);

// Are we empty?
$isEmpty  = $eval->isEmpty();
```

Creating and consuming a custom field evaluator is as easy as creating a custom class and extending the Base class:

./custom/src/ProcessManager/Field/Evaluator/CustomWidget.php

```
  <?php

namespace Sugarcrm\Sugarcrm\ProcessManager\Field\Evaluator;

/**
 * Custom widget field evaluator
 *  @package ProcessManager
 */
class CustomWidget extends Base
{
}
```

Alternatively, you can create a custom evaluator that extends the AbstractEvaluator parent class if you implement the EvaluatorInterface interface.

./custom/src/ProcessManager/Field/Evaluator/CustomWidget.php

```
  <?php

namespace Sugarcrm\Sugarcrm\ProcessManager\Field\Evaluator;

/**
 * Custom widget field evaluator
 *  @package ProcessManager
 */
class CustomWidget extends AbstractEvaluator  implements EvaluatorInte
rface
{
}
```

And at this point, you can use the Factory to get your widget object:

```
  // Get a custom widget evaluator object
$eval  = ProcessManager\Factory::getFieldEvaluator(['type'  => 'widget
']);
```

Getting an Exception Object with Logging

Getting a SugarBPM exception with logging is as easy as calling a single Factory method. Currently, the Process Manager library supports the following exception types:

- DateTime
- Execution
- InvalidData
- NotAuthorized
- Runtime

As a fallback, the Base exception can be used. All exceptions log their message to the PMSE log when created, and all exceptions implement ExceptionInterface.

The most basic way to get a ProcessManager exception is to call the getException() method on the Factory with no arguments:

```
  <?php

use \Sugarcrm\Sugarcrm\ProcessManager\Factory;

// Get a BaseException object, with a default message
```

```
// of 'An unknown Process Manager exception had occurred'
$exception = ProcessManager\Factory::getException();
```

To get a particular type of exception, or to set your message, you can add the first two arguments:

```
  if ($error) {
     throw ProcessManager\Factory::getException('InvalidData', $error)
;
}
```

It is possible to set an exception code, as well as a previous exception, when creating an Exception object:

```
  // Assume $previous is an ExecutionException
throw ProcessManager\Factory ::getException('Runtime', 'Cannot carry o
ut the action', 255, $previous);
```

Finally, to create your own custom exception classes, you must add a new custom file in the following path with the file named appropriately:

./custom/src/ProcessManager/Exception/CustomEvaluatorException.php

```
  <?php

namespace Sugarcrm\Sugarcrm\ProcessManager\Exception;

/**
 * Custom Evaluator exception
 *  @package ProcessManager
 */
class CustomEvaluatorException  extends BaseException  implements Exce
ptionInterface
{
}
```

And to consume this custom exception, call the Factory:

```
  $exception = ProcessManager\Factory::getException('Evaluator', 'Coul
d not evaluate the expression');
```

Maintaining State Throughout a Process

The Registry class implements the RegistryInterface which exposes the following public methods:

- set($key, $value, $override = false)
- get($key, $default = null)
- has($key)
- drop($key)
- getChanges($key)
- reset()

To use the Registry class, simply use the namespace to get the singleton:

```
  <?php

use \Sugarcrm\Sugarcrm\ProcessManager\Registry;

$registry = Registry\Registry::getInstance();
```

To set a value into the registry, simply add it along with a key:

```
  $registry->set('reg_key', 'Hold on to this');
```

NOTE: To prevent named index collision, it is advisable to write indexes to contain a namespace indicator, such as a prefix.

```
  $registry->set('mykeyindex:reg_key', 'Hold on to this');
```

Once a value is set in the registry, it is immutable unless the set($key, $value, $override = false) method is called with the override argument set to true:

```
  // Sets the reg_key value
```

```
$registry->set('reg_key', 'Hold on to this');

// Since the registry key reg_key is already set, this will do nothing
$registry->set('reg_key', 'No wait!');

// To forcefully set it, add a true for the third argument
$registry->set('reg_key', 'No wait!', true);
```

When a key is changed on the registry, these changes are logged and can be inspected using getChanges():

```
  // Set and reset a value on the registry
$registry->set('key', 'Foo');
$registry->set('key', 'Bar', true);
$registry->set('key', 'Baz', true);

// Get the changes for this key
$changes  = $registry ->getChanges('key');

/*
$changes will be:
 array(
     0 => array(
         'from' => 'Foo',
         'to' => 'Bar',
     ),
     1 => array(
         'from' => 'Bar',
         'to' => 'Baz',
     ),
 */
```

To get the value of the key you just set, call the get() method:

```
  $value = $registry->get('reg_key');
```

To get a value of a key with a default, you can pass the default value as the second argument to get():

```
  // Will return either the value for attendee_count or 0
```

```
$count  = $registry->get('attendee_count', 0);
```

To see if the registry currently holds a value, call the has() method:

```
  if ($registry->has('field_list')) {
    // Do something here
}
```

To remove a value from the registry, use the drop() method. This will add an entry to the changelog, provided the key was already set.

```
  $registry->drop('key');
```

Finally, to clear the entire registry of all values and changelog entries, use the reset() method.

```
  $registry->reset();
```

Note: This is very destructive. Please use with caution because you can purge settings that you didn't own.

Entry Points

Overview

Entry points, defined in ./include/MVC/Controller/entry_point_registry.php, were used to ensure that proper security and authentication steps are applied consistently across the entire application. While they are still used in some areas of Sugar, any developers using custom entry points should adjust their customizations to use the latest REST API endpoints instead.

Accessing Entry Points

Available custom entry points can be accessed using a URL pattern as follows:

```
http://{sugar url}/index.php?entryPoint={entry point name}
```

The entry point name will be the specified index in the $entry_point_registry array. Access to this entry point outside of sugar will be dependent on the auth parameter defined in the $entry_point_registry array as well. For more information, refer to the Creating Custom Entry Points page.

Creating Custom Entry Points

Overview

As of 6.3.x, entry points can be created using the extension framework. The entry point extension directory, located at ./custom/Extension/application/Ext/EntryPointRegistry/, is compiled into ./custom/application/Ext/EntryPointRegistry/entry_point_registry.ext.php after a Quick Repair and Rebuild. Additional information can be found in the extensions EntryPointRegistry section.

Custom Entry Points

Prior to 6.3.x, an entry point could be added by creating the file ./custom/include/MVC/Controller/entry_point_registry.php. This method of creating entry points is still compatible but is not recommended from a best practices standpoint as duplicating ./include/MVC/Controller/entry_point_registry.php to ./custom/include/MVC/Controller/entry_point_registry.php will prevent any upgrader updates to ./include/MVC/Controller/entry_point_registry.php from being reflected in the system. Entry point registries contain two properties:

- **file** - The path to the entry point.
- **auth** - A Boolean value that determines whether or not the user must be authenticated in order to access the entry point.

```
  $entry_point_registry['customEntryPoint'] = array(
    'file' => 'path/to/customEntryPoint.php',
    'auth' => true
);
```

Example

The first step is to create the actual entry point. This is where all of the logic for your entry point will be located. This file can be located anywhere you choose. For my example, I will create:

./custom/customEntryPoint.php

```
  <?php

if(!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry Point
');

echo "Hello World!";
```

Next, we will need to create our extension in the application extensions. This will be located at:
./custom/Extension/application/Ext/EntryPointRegistry/customEntryPoint.php

```
  <?php

$entry_point_registry['customEntryPoint'] = array(
    'file' => 'custom/customEntryPoint.php',
    'auth' => true
);
```

Finally, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then generate the file
./custom/application/Ext/EntryPointRegistry/entry_point_registry.ext.php
containing your registry entry. We are now able to access our entry point by navigating to:

```
http://{sugar url}/index.php?entryPoint=customEntryPoint
```

Job Queue

Overview

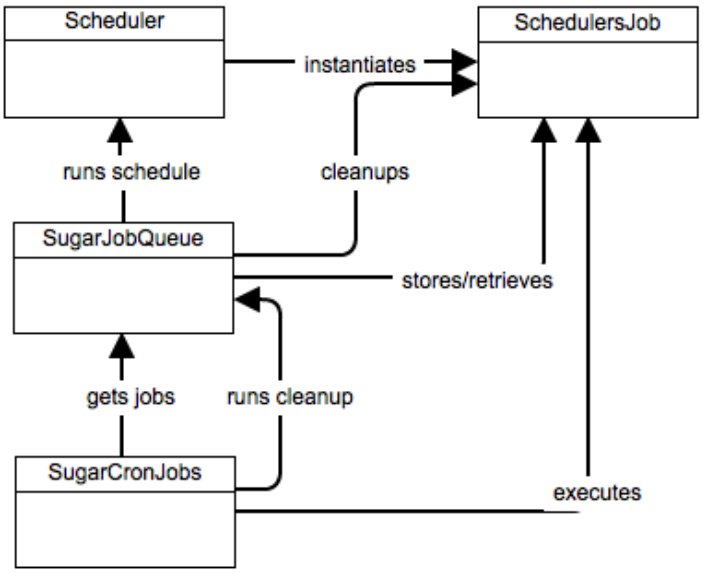
The Job Queue executes automated tasks in Sugar through a scheduler, which integrates with external UNIX systems and Windows systems to run jobs that are scheduled through those systems. Jobs are the individual runs of the specified function from a scheduler.

The Job Queue is composed of the following parts:

- **SugarJobQueue** : Implements the queue functionality. The queue contains

the various jobs.

- **SchedulersJob** : A single instance of a job. This represents a single executable task and is held in the SugarJobQueue.
- **Scheduler** : This is a periodically occurring job.
- **SugarCronJobs** : The cron process that uses SugarJobQueue to run jobs. It runs periodically and does not support parallel execution.



Stages

Schedule Stage

On the scheduling stage (checkPendingJobs in Scheduler class), the queue checks if any schedules are qualified to run at this time and do not have job instance already in the queue. If such schedules exist, a job instance will immediately be created for each.

Execution Stage

The SQL queue table is checked for any jobs in the "Pending" status. These will be set to "Running"' and then executed in accordance to its target and settings.

Cleanup Stage

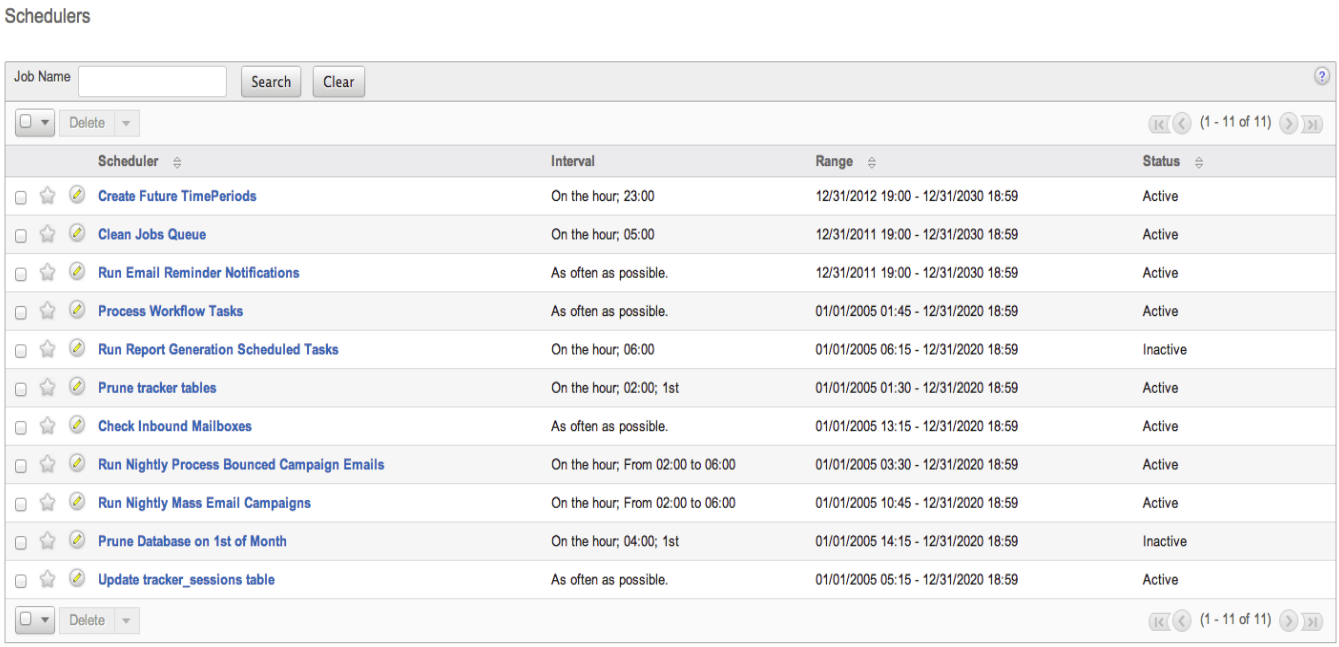
The queue is checked for jobs that are in the "Running" state longer than the defined timeout. Such jobs are considered "Failed" jobs (they may be re-queued if their definition includes re-queuing on failure).

Schedulers

Overview

Sugar provides a Scheduler service that can execute predefined functions asynchronously on a periodic basis. The Scheduler integrates with external UNIX systems and Windows systems to run jobs that are scheduled through those systems. The typical configuration is to have a UNIX cron job or a Windows scheduled job execute the Sugar Scheduler service every couple of minutes. The Scheduler service checks the list of Schedulers defined in the Scheduler Admin screen and executes any that are currently due.

A series of schedulers are defined by default with every Sugar installation. For detailed information on these stock schedulers, please refer to the Schedulers documentation.



Config Settings

- cron.max_cron_jobs - Determines the maximum number of jobs executed per cron run
- cron.max_cron_runtime - Determines the maximum amount of time a job can run before forcing a failure
- cron.min_cron_interval - Specified the minimum amount of time between cron runs

Considerations

- Schedulers execute the next time the cron runs after the interval of time has passed, which may not be at the exact time specified on the scheduler.
- If you see the message "Job runs too frequently, throttled to protect the system" in the Sugar log, the cron is running too frequently.
- If you would prefer the cron to run more frequently, set the cron.min_cron_interval setting to 0 and disable throttling completely.

Creating Custom Schedulers

Overview

In addition to the default schedulers that are packaged with Sugar, developers can create custom scheduler jobs.

Defining the Job Label

The first step to create a custom scheduler is creating a label extension file. This will add the display text for the scheduler job when creating a new scheduler in Admin > Scheduler. The file path of our file will be in the format of ./custom/Extension/modules/Schedulers/Ext/Language/<language key>.<name>.php. For our example, name the file en_us.custom_job.php.

./custom/Extension/modules/Schedulers/Ext/Language/en_us.custom_job.php

```
  <?php

$mod_strings['LBL_CUSTOM_JOB'] = 'Custom Job';
```

Defining the Job Function

Next, define the custom job's function using the extension framework. The file path of the file will be in the format of ./custom/Extension/modules/Schedulers/Ext/ScheduledTasks/<function_name>.php. For this example, name the file custom_job.php. Prior to 6.3.x, job functions were added by creating the file ./custom/modules/Schedulers/_AddJobsHere.php. This method of creating functions is still compatible but is not recommended from a best practices standpoint.

./custom/Extension/modules/Schedulers/Ext/ScheduledTasks/custom_job.php

```
  <?php
```

```
array_push($job_strings, 'custom_job');
function custom_job()
{
    //logic here
    //return true for completed
    return true;
}
```

Using the New Job

Once the files are in place, navigate to Admin > Repair > Quick Repair and Rebuild. This will rebuild the extension directories with our additions. Next, navigate to Admin > Scheduler > Create Scheduler. In the Jobs dropdown, there will be a new custom job in the list.

Scheduler Jobs

Overview

Jobs are the individual runs of the specified function from a scheduler. This article will outline the various parts of a Scheduler Job.

Properties

- **name** : Name of the job
- **scheduler_id** : ID of the scheduler that created the job. This may be empty as schedulers are not required to create jobs
- **execute_time** : Time when job is ready to be executed
- **status** : Status of the job
- **resolution** : Notes whether or not the job was successful
- **message** : Contains messages produced by the job, including errors
- **target** : Function or URL called by the job
- **data** : Data attached to the job
- **requeue** : Boolean to determine whether the job will be replaced in the queue upon failure
- **retry_count :** Determines how many times the system will retry the job before failure
- **failure_count** : The number f times the job has failed
- **job_delay** : The delay (in seconds) between each job run
- **assigned_user_id** : User ID of which the job will be executed as
- **client** : The name of the client owning the job
- **percent_complete** : For postponed jobs, this can be used to determine

how much of the job has been completed

Creating a Job

To create job, you must first create an instance of SchedulesJobs class and use submitJob in SugarJobQueue. An example is shown below:

```
  <?php

$jq = new SugarJobQueue();
$job = new SchedulersJob();
$job->name = "My Job";
$job->target = "function::myjob";
$jobid = $jq->submitJob($job);

echo "Created job $jobid!";
```

Job Targets

Job target contains two components, type and name, separated by "::". Type can be:

- **function :** Name or static method name (using :: syntax). This function will be passed the job object as the first parameter and if **data** is not empty, it will be passed as the second parameter.
- **url** : External URL to call when running the job

Running the Job

The job is run via the runJob() function in SchedulersJob. This function will return a boolean success value according to the value returned by the target function. For URL targets, the HTTP error code being less than 400 will return success.

If the function updated the job status from 'running', the return value of a function is ignored. Currently, the postponing of a URL job is not supported.

Job status

Jobs can be in following statuses:

- **queued** : The job is waiting in the queue to be executed
- **running** : The job is currently being executed
- **done** : The job has executed and completed

Job Resolution

Job resolution is set when the job is finished and can be:

- **queued** : The job is still not finished
- **success** : The job has completed successfully
- **failure** : The job has failed
- **partial** : The job is partially done but needs to be completed

Job Logic Hooks

The scheduler jobs module has two additional logic hooks that can be used to monitor jobs. The additional hooks that can be used are shown below:

- **job_failure_retry** : Executed when a job fails but will be retried
- **job_failure** : Executed when the job fails for the final time

You can find more information on these hooks in the Job Queue Hooks section.

Creating Custom Jobs

Overview

How to create and execute your own custom jobs.

How it Works

As of 6.3.0, custom job functions can be created using the extension framework. The function for the job will be defined in ./custom/Extension/modules/Schedulers/Ext/ScheduledTasks/. Files in this directory are compiled into ./custom/modules/Schedulers/Ext/ScheduledTasks/scheduledtasks.ext.php and then included for use in ./modules/Schedulers/_AddJobsHere.php.

Creating the Job

Use the section below to introduce the technology in relation to the product and describe its benefits.

This first step is to create your jobs custom key. For this example we will be using 'custom_job' as our job key.

./custom/Extension/modules/Schedulers/Ext/ScheduledTasks/custom_job.php

```
  <?php

//add the job key to the list of job strings
array_push($job_strings, 'custom_job');

function custom_job()
{
    //custom job code

    //return true for completed
    return true;
}
```

Next, we will need to define our jobs label string:

./custom/Extension/modules/Schedulers/Ext/Language/en_us.custom_job.php

```
  <?php

$mod_strings['LBL_CUSTOM_JOB'] = 'Custom Job';
```

Finally, We will need to navigate to:

```
Admin / Repair / Quick Repair and Rebuild
```

Once a Quick Repair and Rebuild has been run, the new job will be available for use when creating new schedulers in:

```
Admin / Scheduler
```

Queuing Logic Hook Actions

Overview

This example will demonstrate how to pass tasks to the job queue. This enables you to send longer running jobs such as sending emails, calling web services, or doing other resource intensive jobs to be handled asynchronously by the cron in the background.

Example

This example will queue an email to be sent out by the cron when an account record is saved. First, we will create a before_save logic hook on accounts.

./custom/modules/Accounts/logic_hooks.php

```
<?php

    $hook_version = 1;
    $hook_array = Array();

    $hook_array['before_save'][] = Array();
    $hook_array['before_save'][] = Array(
        1,
        'Queue Job Example',
        'custom/modules/Accounts/Accounts_Save.php',
        'Accounts_Save',
        'QueueJob'
    );

?>
```

In our logic hook, we will create a new SchedulersJob and submit it to the SugarJobQueue targeting our custom AccountAlertJob that we will create next.

./custom/modules/Accounts/Accounts_Save.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    require_once 'include/SugarQueue/SugarJobQueue.php';
    class Accounts_Save
    {
        function QueueJob(&$bean, $event, $arguments)
        {
            //create the new job
            $job = new SchedulersJob();
            //job name
            $job->name = "Account Alert Job - {$bean->name}";
            //data we are passing to the job
```

```
            $job->data = $bean->id;
            //function to call
            $job->target = "function::AccountAlertJob";

            global $current_user;
            //set the user the job runs as
            $job->assigned_user_id = $current_user->id;

            //push into the queue to run
            $jq = new SugarJobQueue();
            $jobid = $jq->submitJob($job);
        }
    }

?>
```

Next, we will need to define the Job. This will be done by creating a new function to execute our code. We will put this file in the ./custom/Extension/modules/Schedulers/Ext/ScheduledTasks/ directory with the name AccountAlertJob.php.

./custom/Extension/modules/Schedulers/Ext/ScheduledTasks/AccountAlertJob.php

```
<?php

function AccountAlertJob($job)
{
    if (!empty($job->data))
    {
        $bean = BeanFactory::getBean('Accounts', $job->data);

        $emailObj = new Email();
        $defaults = $emailObj->getSystemDefaultEmail();
        $mail = new SugarPHPMailer();
        $mail->setMailerForSystem();
        $mail->From = $defaults['email'];
        $mail->FromName = $defaults['name'];
        $mail->Subject = from_html($bean->name);
        $mail->Body = from_html("Email alert that '{$bean->name}' was
saved");
        $mail->prepForOutbound();
        $mail->AddAddress('example@sugar.crm');

        if($mail->Send())
```

```
        {
            //return true for completed
            return true;
        }
    }

    return false;
}
```

Finally, navigate to Admin / Repair / Quick Repair and Rebuild. The system will then generate the file ./custom/modules/Schedulers/Ext/ScheduledTasks/scheduledtasks.ext.php containing our new function. We are now able to queue and run the scheduler job from a logic hook.

Access Control Lists

Overview

Access Control Lists (ACLs) are used to control access to the modules, data, and actions available to users in Sugar. By default, Sugar comes with a default ACL Strategy that is configurable by an Admin through the Roles module via Admin > Role Management. Sugar also comes with other ACL strategies that are located in the ./data/acl/ folder, and are used in other parts of the system such as the Administration module, Emails module, Locked Fields functionality, and many others. They can also be leveraged by customizations and custom ACL strategies can be implemented to control your own customizations.

Actions

Actions are the use cases in which a User interacts with a particular module in the system. ACLs control access by using different logic to determine if a user can do an action. Below is a list of actions that are utilized by Sugar and will be requested actions against a custom ACL strategy implementation. You can add your own actions to be checked against in a custom strategy, but the following list should always be considered:

- **index, list, listview** - for module ListView access
- **detail, detailview, view** - module DetailView access
- **popupeditview**, **editview** - module EditView access
- **edit, save** - editing module data
- **import** - import into the module

- **export** - export from the module
- **delete** - delete module record
- **team_security** - should this module have team security enabled?
- **field** - access field ACLs. The field action is specified via context array, see below
- **subpanel** - checks if subpanel should be displayed. Should have "subpanel" context option.

Note: Action names are not case sensitive, although the standard practice is to use them in all lowercase

Field Actions

When using the field action, the action for the particular field that is being checked is passed via Context. The field actions that are used in Sugar are:

- **access** - any access to field data
- **read, detail** - read access
- **write, edit** - write access

Context

The context array is used throughout Sugar's ACL architecture as a way to pass extra contextual data that is used by an ACL Strategy to determine the access request. The following context parameters are used in stock contexts, and should be used as a guideline for custom ACL implementations as parameters that should be accommodated or used:

- **bean (*SugarBean*)** - the current SugarBean object
- **user (*User*)** - the user to check access for, otherwise defaults to the current user.
- **user_id (*String*)** - the current user ID (also overrides global current user). If the current user object is available, use the **user** context, since it has precedent.
- **owner_override (*Bool*)** - if set to true, apply ACLs as if the current user were the owner of the object.
- **subpanel (*aSubPanel*)** - used in subpanel action to provide aSubPanel object describing the panel.
- **field**, **action (*String*)** - used to specify field name and action name for field ACL requests.
- **massupdate (*Bool*)** - true if save operation is a mass update

SugarACL

The SugarACL Class, ./data/SugarACL.php, is the static API for checking access for

an action against a module.

checkAccess()

The checkAccess() method is used to check a users access for a given action against a module.

```
SugarACL::checkAccess('Accounts','edit');
```

Arguments

Name	Type	Required	Description
$module	String	true	The name of the module
$action	String	true	The name of the action. See Actions section.
$context	Array	false	The associative array of context data. See Context section.

Returns

Boolean

checkField()

The checkField() method is used to check a users access for a given field against a module.

```
SugarACL::checkField('Accounts','account_type','view');
```

Arguments

Name	Type	Required	Description
$module	String	true	The name of the module
$field	String	true	The name of the field
$action	String	true	The name of the

			action. See Actions section.
$context	Array	false	The associative array of context data. See Context section.

Returns

Boolean

getFieldAccess()

The getFieldAccess() method is used to get the access level for a specific

```
$access = SugarACL::getFieldAccess('Accounts','account_type');
```

Arguments

Name	Type	Required	Description
$module	String	true	The name of the module
$field	String	true	The name of the field
$context	Array	false	The associative array of context data. See Context section.

Returns

Integer

The integers are represented as constants in the SugarACL class, and correspond as follows:

- SugarACL::ACL_NO_ACCESS = 0 - access denied
- SugarACL::ACL_READ_ONLY = 1 - read only access
- SugarACL::ACL_READ_WRITE = 4 - full access

filterModuleList()

The filterModuleList() method is used to filter the list of modules available for a

given action. For example, if you wanted to get all the modules the current user had edit access to, you might call the following in code:

```
    global $app_list_strings;
$modules = $app_list_strings['module_list'];
$editableModules = SugarACL::filterModuleList($modules,'edit');
```

Arguments

Name	Type	Required	Description
$modules	Array	true	The list of modules you want to filter
$action	String	false	The action to check for, See Actions section. **Defaults to 'access' action**
$use_value	Boolean	false	Whether to the module name in the $modules array is in the Key or the Value of the array. **Defaults to false** For example, if filtering an array of modules, where the modules are defined in the values of the array: `$modules = array ( 'Accounts', 'Cases', 'Administration');$accessModules = SugarACL::filterModuleList($modules,'access',true);`

Returns

Array

The filtered list of modules.

disabledModuleList()

Similar to the previous method, the disableModuleList() method filters a list of modules to those that the user does not have access to.

```
    global $app_list_strings;
$modules = $app_list_strings['module_list'];
$disabledModules = SugarACL::disabledModuleList($modules,'access');
```

Arguments

Name	Type	Required	Description
$modules	Array	true	The list of modules you want to filter
$action	String	false	The action to check for, See Actions section. **Defaults to 'access' action**
$use_value	Boolean	false	Whether to the module name in the $modules array is in the Key or the Value of the array. **Defaults to false** For example, if filtering an array of modules, where the modules are defined in the values of the array: `$modules = array ( 'Accounts', 'Cases', 'Administration');$accessModules = SugarACL::filterModuleList($modules,'access',tru`

			e);

Returns

Array

The filtered list of modules.

moduleSupportsACL()

The moduleSupportsACL() method is used to check if a module has ACLs defined on the vardefs. A good use case for this method is to not run any ACL checks if module has no ACL definitions.

```
SugarACL::moduleSupportsACL('Accounts');
```

Arguments

Name	Type	Required	Description
$module	String	true	The name of the module

Returns

Boolean

listFilter()

The listFilter() method allows for filtering an array of fields for a module to remove those which the user does not have access to. The removal can be done in a couple of different ways, provided by the $options argument. The filtering of the array is done in place, rather than returning the filtered list.

```
    $Account = BeanFactory::getBean('Accounts','12345');
$fields = array(
  'account_type' => 'foobar',
  'status' => 'New',
  'name' => 'Customer'
);

$context = array(
   'bean' => $Account
);
```

```
$options = array(
    'blank_value' => true
);

SugarACL::listFilter('Accounts',$fields,$context,$options);
echo <pre>json_encode($fields)</pre>;
```

Should the user not have access to the 'status' field, the above might output:

```
    {
  "account_type": "foobar",
  "status": "",
  "name": "Customer"
}
```

Arguments

Name	Type	Required	Description
$module	String	true	The name of the module
$list	Array	true	The array, as a reference, which will be filtered
$context	Array	false	The associative array of context data. See Context section.
$options	Array	false	An associative array containing some of the following options: • **blank_value (*Bool*)** - instead of removing inaccessible field put '' there • **add_acl (*Bool*)** -

			instead of removing fields add 'acl' value with access level • **suffix (*String*)** - strip suffix from field names • **min_access (*Int*)** - require this level of access for the field. **Defaults to SugarACL::ACL_READ_ONLY or 0** • **use_value (*Bool*)** - look for field name in value, not in the key of the list

Returns

Void

SugarBean Usage

The SugarBean class also comes with some methods for working with the ACLs in regards to the current SugarBean context. These methods automatically provide the 'bean' context parameter and provide a wrapper that utilizes the SugarACL class

getACLCategory()

The getACLCategory() method is used by all of the ACL helper methods that exist on the SugarBean and is used to determine the module name that should be checked against for the current Bean. By default, it will return the acl_category property set on the Bean, otherwise if that is not set, it will use the module_dir property that is configured on the Bean. This method can be overridden by a custom Bean implementation, but the following code shows which properties on a Bean implementation should be set for usage with ACLs.

```
    <?php

class testBean extends SugarBean
{
    public $module_dir = 'testBean';
    //all the other SugarBean class logic...
}

class test_SubmoduleBean extends SugarBean
{
    public $module_dir = 'testBean/submodule';
    public $acl_category = 'test_SubmoduleBean';
    //all the other SugarBean class logic...
}

$test = new testBean();
//echoes testBean
echo $test->getACLCategory();

$submodule = new test_SubmoduleBean();
//echoes test_SubmoduleBean
echo $submodule->getACLCategory();
```

Arguments

None

Returns

String

ACLAccess()

The ACLAccess() method is a wrapper for SugarACL::checkField() method, which provides the Bean context as the current SugarBean.

```
    $Account = BeanFactory::getBean('Accounts','12345');
if ($Account->ACLAccess('edit')){
   //do something if User has edit access
}
```

Arguments

Name	Type	Required	Description
$action	String	false	The name of the action to check. See Actions section.
$context	Array	false	The associative array of context data. See Context section.

Returns

Boolean

ACLFieldAccess()

The ACLFieldAccess() method is a wrapper for SugarACL::checkAccess() method, which provides the Bean context as the current SugarBean.

```
    $Account = BeanFactory::getBean('Accounts','12345');
if ($Account->ACLFieldAccess('account_type','edit')){
   //do something if User has account_type edit access
}
```

Arguments

Name	Type	Required	Description
$field	String	true	The name of the field
$action	String	false	The name of the action to check, see Field Actions section. **Defaults to 'access'**

$context	Array	false	The associative array of context data. See Context section.

Returns

Boolean

ACLFieldGet()

The ACLFieldGet() method is a wrapper for SugarACL::getFieldAccess() method, which provides the Bean context as the current SugarBean.

```
    $Account = BeanFactory::getBean('Accounts','12345');
$accountTypeAccess = $Account->ACLFieldGet('account_type');
```

Arguments

Name	Type	Required	Description
$field	String	true	The name of the field
$context	Array	false	The associative array of context data. See Context section.

Returns

Boolean

ACLFilterFields()

The ACLFilterFields() method uses the SugarACL::checkField() method to blank out those fields which a user doesn't have access to on the current SugarBean.

```
    $Account = BeanFactory::getBean('Accounts','12345');
$Account->status = 'Old';
$Account->ACLFilterFields('edit');

echo $Account->status;
```

Should the user not have 'edit' access to the 'status' field, the above wouldn't output any data since the status field would be blank.

Arguments

Name	Type	Required	Description
$action	String	true	The name of the action to check. See Actions section.
$context	Array	false	The associative array of context data. See Context section.

Returns

Void

ACLFilterFieldList()

The ACLFilterFieldList() method is a wrapper for SugarACL::listFilter() method, which provides the Bean context as the current SugarBean.

```
    $Account = BeanFactory::getBean('Accounts','12345');
$fields = array(
  'account_type' => 'foobar',
  'status' => 'New',
  'name' => 'Customer'
);

$options = array(
    'blank_value' => true
);

$Account->ACLFilterFieldList($fields,array(),$options);
echo "<pre>".json_encode($fields)."</pre>";
```

Should the user not have access to the 'status' field, the above might output:

```
    {
  "account_type": "foobar",
  "status": "",
```

```
  "name": "Customer"
}
```

Arguments

Name	Type	Required	Description
$list	Array	true	The array, as a reference, which will be filtered
$context	Array	false	The associative array of context data. See Context section.
$options	Array	false	An associative array containing some of the following options: • **blank_value (*Bool*)** - instead of removing inaccessible field put '' there • **add_acl (*Bool*)** - instead of removing fields add 'acl' value with access level • **suffix (*String*)** - strip suffix from field names • **min_access (*Int*)** - require this level of

			access for the field. **Defaults to SugarACL::ACL_READ_ONLY or 0** • **use_value (*Bool*)** - look for field name in value, not in the key of the list

Returns

Void

Legacy Methods

When working in the Sugar codebase there might be areas that still utilize the following legacy ACLController class. The following gives a brief overview of a few methods that might still be used but should be avoided in future custom development.

ACLController::checkAcess()

The ACLController::checkAccess() method is just a wrapper for SugarACL::checkAccess() method and has been left in place for backward compatibility.

ACLController::moduleSupportsACL()

The ACLController::moduleSupportsACL() method is just a wrapper for SugarACL::moduleSupportsACL() method and has been left in place for backward compatibility.

ACLController::displayNoAccess()

This method does an echo to display a "no access" message for pages.

Teams

Overview

Teams provide the ability to limit access at the record level, allowing sharing flexibility across functional groups. Developers can manipulate teams programmatically provided they understand Sugar's visibility framework.

For more information on teams, please refer to the Team Management documentation.

Database Tables

Table	Description
teams	Each record in this table represents a team in the system.
team_sets	Each record in this table represents a unique team combination. For example, each user's private team will have a corresponding team set entry in this table. A team set may also be comprised of one or more teams.
team_sets_teams	The team_sets_teams table maintains the relationships to determine which teams belong to a team set. Each table that previously used the team_id column to maintain team security now uses the team_set_id column's value to associate the record to team(s).
team_sets_modules	This table is used to manage team sets and keep track of which modules have records that are or were associated to a particular team set.

Never modify these tables without going through the PHP object methods. Manipuating the team sets with direct SQL may cause undesired side effects in the system due to how the validations and security methods work.

Module Team Fields

In addition to the team tables, each module table also contains a team_id and team_set_id column. The team_set_id contains the id of a record in the team_sets table that contains the unique combination of teams associated with the record.

The team_id will contain the id of the primary team designated for a record.

Team Sets (team_set_id)

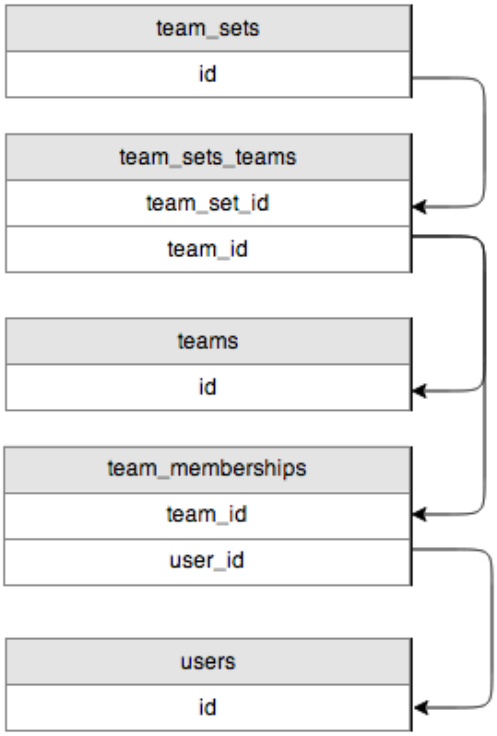
As mentioned above, Sugar implemented this feature not as a many-to-many relationship but as a one-to-many relationship. On each table that had a team_id field we added a 'team_set_id' field. We have also added a team_sets table, which maintains the team_set_id, and a team_sets_teams table, which relates a team set to the teams. When performing a team based query we use the 'team_set_id' field on the module table to join to team_sets_teams.team_set_id and then join all of the teams associated with that set. Given the list of teams from team_memberships we can then decide if the user has access to the record.

Primary Team (team_id)

The team_id is still being used, not only to support backwards compatibility with workflow and reports, but also to provide some additional features. When displaying a list, we use the team set to determine whether the user has access to the record. When displaying the data, we show the team from team id in the list. When the user performs a mouseover on that team, Sugar performs an Ajax call to display all of the teams associated with the record. This team_id field is designated as the Primary Team because it is the first team shown in the list, and for sales territory management purposes, can designate the team that actually owns the record and can report on it.

Team Security

The team_sets_teams table allows the system to check for permissions on multiple teams. The following diagram illustrates table relationships in SugarBean's add_team_security_where_clause method.



Using the team_sets_teams table the system will determine which teams are associated with the team_set_id and then look in the team_memberships table for users that belong to the team(s).

TeamSetLink

Typically, any relationship in a class is handled by the data/Link2.php class. As a part of dynamic teams, we introduced the ability to provide your own custom Link class to handle some of the functionality related to managing relationships. The team_security parent vardefs in the Sugar objects contain the following in the 'teams' field definition:

```
  'link_class' => 'TeamSetLink',
'link_file' => 'modules/Teams/TeamSetLink.php',
```

The link_class entry defines the class name we are using, and the link_file tells us where that class file is located. This class extends the legacy Link.php and overrides some of the methods used to handle relationships such as 'add' and 'delete'.

Manipulating Teams Programmatically

Overview

How to manipulate team relationships.

Fetching Teams

To fetch teams related to a bean, you will need to retrieve an instance of a TeamSet object and use the getTeams() method to retrieve the teams using the team_set_id. An example is shown below:

```
      //Create a TeamSet bean - no BeanFactory
require_once 'modules/Teams/TeamSet.php';
$teamSetBean = new TeamSet();

//Retrieve the bean
$bean = BeanFactory::getBean($module, $record_id);

//Retrieve the teams from the team_set_id
$teams = $teamSetBean->getTeams($bean->team_set_id);
```

Adding Teams

To add a team to a bean, you will need to load the team's relationship and use the add() method. This method accepts an array of team ids to add. An example is shown below:

```
      //Retrieve the bean
$bean = BeanFactory::getBean($module, $record_id);

//Load the team relationship
$bean->load_relationship('teams');

//Add the teams
$bean->teams->add(
    array(
        $team_id_1,
        $team_id_2
    )
);
```

Considerations

- If adding teams in a logic hook, the recommended approach is to use an after_save hook rather than a before_save hook as the $_REQUEST may reset any changes you make.

Removing Teams

To remove a team from a bean, you will need to load the team's relationship and use the remove() method. This method accepts an array of team ids to remove. An example is shown below:

```
      //Retrieve the bean
$bean = BeanFactory::getBean($module, $record_id);

//Load the team relationship
$bean->load_relationship('teams');

//Remove the teams
$bean->teams->remove(
    array(
        $team_id_1,
        $team_id_2
    )
);
```

Considerations

- If removing teams in a logic hook, the recommended approach is to use an after_save hook rather than a before_save hook as the $_REQUEST may reset any changes you make.

Replacing Team Sets

To replace all of the teams related to a bean, you will need to load the team's relationship and use the replace() method. This method accepts an array of team ids. An example is shown below:

```
      //Retrieve the bean
$bean = BeanFactory::getBean($module, $record_id);

//Load the team relationship
$bean->load_relationship('teams');
```

```
//Set the primary team
$bean->team_id = $team_id_1

//Replace the teams
$bean->teams->replace(
    array(
        $team_id_1,
        $team_id_2
    )
);

//Save to update primary team
$bean->save()
```

Considerations

- If replacing teams in a logic hook, the recommended approach is to use an after_save hook rather than a before_save hook as the $_REQUEST or workflow may reset any changes you make.
- This method does not replace (or set) the primary team for the record. When replacing teams, you need to also make sure that the primary team, determined by the team_id field, is set appropriately and included in the replacement ids. If this is being done in a logic hook you should set the primary team in a before_save hook and replace the team set in the after_save hook.
- When using an after_save hook, be sure to call $bean->teams->setSaved(false) to explicitly reset the save state. This ensures that the updates (create, update or delete) to the team sets are applied.

Example:

```
      //before save function

public function before_save_hook($bean, $event, $arguments)
{
    $bean->team_id = $team_id_1;
}

//after save function
public function after_save_hook($bean, $event, $arguments)
{
 $bean->teams->setSaved(false); // Manually reset TeamSet state for sa
```

```
ve

    $bean->teams->replace(
        array(
            $team_id_1,
            $team_id_2
        )
    );
}
```

Creating and Retrieving Team Set IDs

To create or retrieve the team_set_id for a group of teams, you will need to retrieve an instance of a TeamSet object and use the addTeams() method. If a team set does not exist, this method will create it and return an id. An example is below:

```
      //Create a TeamSet bean - no BeanFactory
require_once 'modules/Teams/TeamSet.php';
$teamSetBean = new TeamSet();

//Retrieve/create the team_set_id
$team_set_id = $teamSetBean->addTeams(
    array(
        $team_id_1,
        $team_id_2
    )
);
```

Adding Additional Access

To enable additional access for a record, you can either set the team id or team set to the beans acl_team_set_id field. An example of this is shown below:

```
      require_once 'modules/Teams/TeamSet.php';
// Create a new teamset or fetch the id of an existing teamset
$teamSetBean = new TeamSet();
$teamSetId = $teamSetBean->addTeams([
    'East', // Demo Team ID
    'West', // Demo Team ID
]);

$bean = BeanFactory::getBean('Accounts', '15bcf01c-1e1e-11e8-9e13-f45c
```

```
89a8598f');

// Set additional access
$bean->acl_team_set_id = $teamSetId; // if set to NULL, this will remo
ve any existing value
$bean->save();
```

Visibility Framework

Overview

The visibility framework provides the capability to alter the queries Sugar uses to retrieve records from the database. This framework can allow for additional restrictions or specific logic to meet business requirements of hiding or showing only specific records. Visibility classes only apply to certain aspects of Sugar record retrieval, e.g. List Views, Dashlets, and Filter Lookups.

Custom Row Visibility

Custom visibility class files are stored under ./custom/data/visibility/. The files in this directory can be enabled or disabled by modifying a module's visibility property located in ./custom/Extension/modules/<module>/Ext/Vardefs/. Every enabled visibility class is merged into the module's definition, allowing multiple layers of logic to be added to a module.

Visibility Class

To add custom row visibility, you must create a visibility class that will extend the core SugarVisibility class ./data/SugarVisibility.php . The visibility class has the ability to override the following methods:

Name	Description
addVisibilityQuery	Add visibility clauses to a SugarQuery object.
addVisibilityFrom	[Deprecated] Add visibility clauses to the FROM part of the query. This method should still be implemented, as not all objects have been switched over to use addVisibilityQuery() method.
addVisibilityFromQuery	[Deprecated] Add visibility clauses to the FROM part of SugarQuery. This method should still be implemented, as

	not all objects have been switched over to use addVisibilityQuery() method.
addVisibilityWhere	[Deprecated] Add visibility clauses to the WHERE part of the query. This method should still be implemented, as not all objects have been switched over to use addVisibilityQuery() method.
addVisibilityWhereQuery	[Deprecated] Add visibility clauses to the WHERE part of SugarQuery. This method should still be implemented, as not all objects have been switched over to use addVisibilityQuery() method.

The visibility class should also implement Sugarcrm\Sugarcrm\Elasticsearch\Provider\Visibility\StrategyInterface so that the visibility rules are also applied to the global search. StrategyInterface has the following functions that should be implemented:

Name	**Description**
elasticBuildAnalysis	Build Elasticsearch analysis settings. This function can be empty if you do not need any special analyzers.
elasticBuildMapping	Build Elasticsearch mapping. This function should contain a mapping of fields that should be analyzed.
elasticProcessDocumentPreIndex	Process document before it's being indexed. This function should perform any actions to the document that needs to be completed before it is indexed.
elasticGetBeanIndexFields	Bean index fields to be indexed. This function should return an array of the fields that need to be indexed as part of your custom visibility.
elasticAddFilters	Add visibility filters. This function should apply the Elastic filters.

Example

The following example creates a custom visibility filter that determines whether Opportunity records should be displayed based on their Sales Stage. Opportunity records with Sales Stage set to Closed Won or Closed Lost will not be displayed in the Opportunities module or global search for users with the Demo Visibility role.

/custom/data/visibility/FilterOpportunities.php:

```
  <?php

use Sugarcrm\Sugarcrm\Elasticsearch\Provider\Visibility\StrategyInterf
ace as ElasticStrategyInterface;
use Sugarcrm\Sugarcrm\Elasticsearch\Provider\Visibility\Visibility;
use Sugarcrm\Sugarcrm\Elasticsearch\Analysis\AnalysisBuilder;
use Sugarcrm\Sugarcrm\Elasticsearch\Mapping\Mapping;
use Sugarcrm\Sugarcrm\Elasticsearch\Adapter\Document;

/**
 *
 * Custom visibility class for Opportunities module:
 *
 * This demo allows to restrict access to opportunity records based on
 the
 * user's role and configured filtered sales stages.
 *
 * The following $sugar_config parameters are available:
 *
 * $sugar_config['custom']['visibility']['opportunities']['target_role
']
 * This parameter takes a string containing the role name for which
 * the filtering should apply.
 *
 * $sugar_config['custom']['visibility']['opportunities']['filter_sale
s_stages']
 * This parameters takes an array of filtered sales stages. If current
 user is
 * member of the above configured role, then the opportunities with th
e sale
 * stages as configured in this array will be inaccessible.
 *
 *
 * Example configuration given that 'Demo Visibility' role exists (con
fig_override.php):
 *
 * $sugar_config['custom']['visibility']['opportunities']['target_role
'] = 'Demo Visibility';
 * $sugar_config['custom']['visibility']['opportunities']['filter_sale
s_stages'] = array('Closed Won', 'Closed Lost');
 *
 */
class FilterOpportunities extends SugarVisibility implements ElasticSt
rategyInterface
```

```
{
    /**
     * The target role name
     * @var string
     */
    protected $targetRole = '';
    /**
     * Filtered sales stages
     * @var array
     */
    protected $filterSalesStages = array();

    /**
     * {@inheritdoc}
     */
    public function __construct(SugarBean $bean, $params = null)
    {
        parent::__construct($bean, $params);
        $config = SugarConfig::getInstance();
        $this->targetRole = $config->get(
            'custom.visibility.opportunities.target_role',
            $this->targetRole
        );
        $this->filterSalesStages = $config->get(
            'custom.visibility.opportunities.filter_sales_stages',
            $this->filterSalesStages
        );
    }

    /**
     * {@inheritdoc}
     */
    public function addVisibilityWhere(&$query)
    {
        if (!$this->isSecurityApplicable()) {
            return $query;
        }
        $whereClause = sprintf(
            "%s.sales_stage NOT IN (%s)",
            $this->getTableAlias(),
            implode(',', array_map(array($this->bean->db, 'quoted'), $
this->filterSalesStages))
        );
        if (!empty($query)) {
            $query .= " AND $whereClause ";
        } else {
```

```
            $query = $whereClause;
        }
        return $query;
    }

    /**
     * {@inheritdoc}
     */
    public function addVisibilityWhereQuery(SugarQuery $sugarQuery, $o
ptions = array())
    {
        $where = null;
        $this->addVisibilityWhere($where, $options);
        if (!empty($where)) {
            $sugarQuery->where()->addRaw($where);
        }
        return $sugarQuery;
    }

    /**
     * Check if we can apply our security model
     * @param User $user
     * @return false|User
     */
    protected function isSecurityApplicable(User $user = null)
    {
        $user = $user ?: $this->getCurrentUser();
        if (!$user) {
            return false;
        }
        if (empty($this->targetRole) || empty($this->filterSalesStages
)) {
            return false;
        }
        if (!is_string($this->targetRole) || !is_array($this->filterSa
lesStages)) {
            return false;
        }
        if (!$this->isUserMemberOfRole($this->targetRole, $user)) {
            return false;
        }
        if ($user->isAdminForModule("Opportunities")) {
            return false;
        }
        return $user;
    }
```

```
    /**
     * Get current user
     * @return false|User
     */
    protected function getCurrentUser()
    {
        return empty($GLOBALS['current_user']) ? false : $GLOBALS['cur
rent_user'];
    }

    /**
     * Check if given user has a given role assigned
     * @param string $targetRole Name of the role
     * @param User $user
     * @return boolean
     */
    protected function isUserMemberOfRole($targetRole, User $user)
    {
        $roles = ACLRole::getUserRoleNames($user->id);
        return in_array($targetRole, $roles) ? true : false;
    }

    /**
     * Get table alias
     * @return string
     */
    protected function getTableAlias()
    {
        $tableAlias = $this->getOption('table_alias');
        if (empty($tableAlias)) {
            $tableAlias = $this->bean->table_name;
        }
        return $tableAlias;
    }

    /**
     * {@inheritdoc}
     */
    public function elasticBuildAnalysis(AnalysisBuilder $analysisBuil
der, Visibility $provider)
    {
        // no special analyzers needed
    }

    /**
```

```
     * {@inheritdoc}
     */
    public function elasticBuildMapping(Mapping $mapping, Visibility $
provider)
    {
        $mapping->addNotAnalyzedField('visibility_sales_stage');
    }

    /**
     * {@inheritdoc}
     */
    public function elasticProcessDocumentPreIndex(Document $document,
 \SugarBean $bean, Visibility $provider)
    {
        // populate the sales_stage into our explicit filter field
        $sales_stage = isset($bean->sales_stage) ? $bean->sales_stage
: '';
        $document->setDataField('visibility_sales_stage', $sales_stage
);
    }

    /**
     * {@inheritdoc}
     */
    public function elasticGetBeanIndexFields($module, Visibility $pro
vider)
    {
        // make sure to pull sales_stage regardless of search
        return array('sales_stage');
    }

    /**
     * {@inheritdoc}
     */
    public function elasticAddFilters(User $user, Elastica\Query\BoolQ
uery $filter,
                                      Sugarcrm\Sugarcrm\Elasticsearch\
Provider\Visibility\Visibility $provider)
    {
        if (!$this->isSecurityApplicable($user)) {
            return;
        }
        // apply elastic filter to exclude the given sales stages
        $filter->addMustNot($provider->createFilter(
            'OpportunitySalesStages',
            array(
```

```
                'filter_sales_stages' => $this->filterSalesStages,
            )
        ));
    }
}
```

./custom/Extension/modules/Opportunities/Ext/Vardefs/opp_visibility.php

```
  <?php

if (!defined('sugarEntry') || !sugarEntry) {
    die('Not A Valid Entry Point');
}

$dictionary['Opportunity']['visibility']['FilterOpportunities'] = true
;
```

./custom/src/Elasticsearch/Provider/Visibility/Filter/OpportunitySalesStagesFilter.php

```
  <?php

namespace Sugarcrm\Sugarcrm\custom\Elasticsearch\Provider\Visibility\F
ilter;

use Sugarcrm\Sugarcrm\Elasticsearch\Provider\Visibility\Filter\FilterI
nterface;
use Sugarcrm\Sugarcrm\Elasticsearch\Provider\Visibility\Visibility;

/**
 *
 * Custom opportunity filter by sales_stage
 *
 * This logic can exist directly in the FilterOpportunities visibility
 class.
 * However by abstracting the (custom) filters makes it possible to re-
use
 * them in other places as well.
 */
class OpportunitySalesStagesFilter implements FilterInterface
{
    /**
```

```
     * @var Visibility
     */
    protected $provider;

    /**
     * {@inheritdoc}
     */
    public function setProvider(Visibility $provider)
    {
        $this->provider = $provider;
    }

    /**
     * {@inheritdoc}
     */
    public function buildFilter(array $options = array())
    {
        return new \Elastica\Query\Terms(
            'visibility_sales_stage',
            $options['filter_sales_stages']
        );
    }
}
```

After creating the above files, log in to your Sugar instance as an administrator and navigate to **Administration > Repair > Quick Repair and Rebuild**.

Next, perform a full reindex by navigating to **Administration > Search** and selecting the "delete existing data" option.

Execute a cron to process all of the queued records into Elasticsearch by doing the following:

1. Open a command line client and navigate to your Sugar directory.
2. Execute chmod +x bin/sugarcrm to ensure bin/sugarcrm is executable.
3. Execute php cron.php to consume the queue.
4. Execute bin/sugarcrm elastic:queue to see if the queue has finished.

Repeat steps 3 and 4 until the queue has 0 records.

This example requires the Sales Stage field to be part of the Opportunities module. Navigate to **Administration > Opportunities** and ensure the **Opportunities** radio button is selected.

Create a new role named "Demo Visibility" and assign a user to this role. Note: if you are using the sample data, do NOT use Jim as he has admin permission on the Opportunities module and will be able to view all records. We recommend using Max.

Configure your instance to filter opportunities for a given sales stages for this role by adding the following to ./config_override.php:

```
  <?php

$sugar_config['custom']['visibility']['opportunities']['target_role']
= 'Demo Visibility';
$sugar_config['custom']['visibility']['opportunities']['filter_sales_s
tages'] = array('Closed Won', 'Closed Lost');
```

Log in as the user to whom you assigned the Demo Visibility role. Observe that opportunity records in the sales stages "Closed Won" and "Closed Lost" are no longer accessible.

You can download a module loadable package of this example here.

Tags

Overview

The tagging feature allows a user to apply as many "tags" as they choose on any record they want to categorize. This allows the user to effectively group records together from any source within the application and relate them together. Please note that the tag field and Tags module do not support customization at this time.

The Tags Module

The Tags module is a very simple module based on the Basic SugarObject template. It is not studio editable, is part of the default module list, and is available to the application upon installation. By default, the Tags module is set up such that only an Administrator can perform any administrative tasks with the Tags module. In other words, a regular user will have a very restrictive set of permissions inside the Tags module, namely: create, export, view, and share. All other actions within the Tags modules must be carried out by an Administrative user.

The "tag" Field Type

The back end utilizes two new field types: the relatecollection field and the tag field. The tag field is a relatecollection type field with a few added enhancements specific to the tagging implementation:

1. Enforces uniqueness of a tag when created
2. Establishes the necessary relationship between the Tags module and the module record being tagged
3. Collects and formats a tag collection in a way the client understands
4. Format the tag field for consumption by and from the import process
5. Handles proper query setting for a search that is filtered by tags

The Taggable Implementation

The taggable implementation is applied to all SugarObject templates so that it is available on all custom modules created and is also applied to all sidecar enabled modules. Any module that implements a SugarObject template will be taggable by default. Any module that doesn't implement a SugarObject template can easily apply it using the uses property of the module vardefs:

```
  $dictionary[$module] = array(
    ...
    'uses' => array(
        'taggable',
    ),
    ...
);
```

There may be instances where a module should not implement tagging yet implements the SugarObject template. To remove tagging from a module you can use the module vardefs ignore_templates property:

```
  $dictionary[$module] = array(
    ...
    'ignore_templates' => array(
        'taggable',
    ),
    ...
);
```

The Tagging Relationship Structure

The tagging relationship schema is similar in nature to the email_addresses

relationship schema in that it is used to represent a collection of 0-N Tags module records related to a module:

Column	Description
id	The GUID for the relationship between the tag record and the module record
tag_id	The id of the tag record used in this relationship
bean_id	The id of the module record used in this relationship
bean_module	The module for the record being tagged
date_modified	The date the relationship was created/modified
deleted	A tinyint(1) boolean value that indicates whether this relationship is deleted

Tagging in Action

For adding tags from the UI, please refer to the Tags documentation.

Tagging Records from the API

Tagging a record via the API is done by sending a PUT request to the /<module>/<record> API endpoint with a tag property set to an array of key/value pairs that include a tag id (optional) and a tag name:

```
  {
    "name": "Record Name",
    "tag": [
        {"id": "Test Tag", "name": "Test Tag"},
        {"name": "Test Tag 2"},
        {"id": "1234-56-7890", "name": "Test Tag 3"}
    ]
}
```

After the record is created/modified and the tag values are applied, the response will contain the returned collection of tags for that record complete with their ids:

```
{
    "name": "Record Name",
    "tag": [
```

```
        {"id": "9876-54-3210", "name": "Test Tag"},
        {"id": "4321-56-0987", "name": "Test Tag 2"},
        {"id": "1234-56-7890", "name": "Test Tag 3"}
    ]
}
```

You can visit How to Manipulate Tags (CRUD) for a full example demonstrating CRUD actions for tags.

Mass Updating Tags on Records Via the API

Mass updating records with tags are as simple as sending a MassUpdate PUT request to /<module>/MassUpdate with a payload similar to:

```
  {
    "massupdate_params": {
        "uid": ["12345-67890", "09876-54321"],
        "tag": [
            { "id": "23456-78901", "name": "MyTag1" },
            { "id": "34567-89012", "name": "MyTag2" }
        ]
    }
}
```

This request will override all existing tags on the named records. To append tags to a record, send the "tag_type" request argument set to a value of 1:

```
  {
    "massupdate_params": {
        "uid": ["12345-67890", "09876-54321"],
        "tag": [
            { "id": "23456-78901", "name": "MyTag1" },
            { "id": "34567-89012", "name": "MyTag2" }
        ],
        "tag_type": "1"
    }
}
```

More information on this API endpoint can be found in the /<module>/MassUpdate

PUT documentation.

Fetching Tags on a Record

By default, the tag field is added to all Sidecar module record views. That means when a request is made for a single record through the API, that single record will return the "tag" field, which will be a collection of key:value pair objects.

For example, when a request is made to the /Accounts/:record GET endpoint, the tags associated with the Accountrecord will be returned in the tag field as an array of objects:

```
  {
    "id": "<record>",
    "name": "Record Name",
    "tag": [
        { "id": "9876-54-3210", "name": "Test Tag" },
        { "id": "4321-56-0987", "name": "Test Tag 2" },
        { "id": "1234-56-7890", "name": "Test Tag 3" }
    ]
}
```

Filtering a Record List by Tags

Filtering a list of records by tags can be done simply by sending a filter request to the ModuleApi endpoint for a module. For example, to filter the Accounts list where the tag field has a tag by the name of "Tradeshow", you can send a request to the /Accounts GET endpoint with the following request arguments:

```
  {
    "view": "list",
    "filter": [
        {
            "tag": {
                "$in": [
                    {
                        "name": "Tradeshow"
                    }
                ]
            }
        }
    ]
}
```

Currently, the tag field filter definitions support the following filter operators:

- Is any of ($in)
- Is not any of ($not_in)
- Is empty ($empty)
- Is not empty ($not_empty)

Fetching a list of Tags from the Tags module

Fetch a list of tag records from the Tags module is done the same way as fetch a list of records from any other module, namely by sending a GET request to the /Tags ModuleApi endpoint. More information can be found in the /<module> GET documentation.

Manipulating Tags Programmatically

Getting Tags Related to a Record

Here is an example that demonstrates how to get all the tags and its ids related to a contact record:

```
  // Creating a Bean for Contacts
$bean = BeanFactory::getBean("Contacts");
// Creating a Bean for Tags
$tagBean = BeanFactory::newBean('Tags');
// Get all the tags related to Contacts Bean by givin Contact ID. You
can provide more than one Record ID.
$tags = $tagBean->getRelatedModuleRecords($bean, ["<CONTACT_RECORD_ID>
"]);
```

Creating a New Tag and Adding to a Record

Here is an example that demonstrates how to add create a tag and add to a contact record.

In order to add a new tag first, we will create the tag bean. Then using load_relationship function we will add the newly created tag id to the contacts

```
  // Creating new Tag Bean
$tagBean = BeanFactory::newBean("Tags");

// Setting its name
$tagBean->name = "New Tag";
$tagBean->save();
```

```
// Retrieving the Contacts Bean
$bean = BeanFactory::getBean("Contacts", "<RECORD_ID>");

// Getting tag field and its properties
$tagField = $bean->getTagField();
$tagFieldProperties = $bean->field_defs[$tagField];

// Identifying relation link
$link = $tagFieldProperties['link'];

// Loading relationship
if ($bean->load_relationship($link)) {
    // Adding newly created Tag Bean
    $bean->$link->add($tagBean->id);
}
```

Removing Tags from a Record

Here is an example that demonstrates how to remove a tag from a contact record:

```
  // Getting the Contacts Bean
$bean = BeanFactory::getBean("Contacts", "<RECORD_ID>");

// Getting tag field and its properties
$tagField = $bean->getTagField();
$tagFieldProperties = $bean->field_defs[$tagField];

// Identifying relation link
$link = $tagFieldProperties['link'];

// Loading relationship
if($bean->load_relationship($link)){
    // Removing the Tag ID
    $bean->$link->delete($bean->id, "<TAG_RECORD_ID>");
}
```

Synchronizing Tags by Name With API Helpers

If you have multiple tags that you need to add and delete at the same time, then you can use SugarFieldTag->apiSave method. Here is an example that demonstrates how to sync tags by using SugarFieldTag Class for a Contact record.

It is important to know that since this is a sync; you will need to keep the existing tags if you want them to still exist in the record.

```
  // Getting the Contacts Bean
$bean = BeanFactory::getBean("Contacts", "<RECORD_ID>");

// Getting Tag Field ID
$tagField = $bean->getTagField();

// Getting Tag Field Properties
$tagFieldProperties = $bean->field_defs[$tagField];

// Preparing the latest Tags to be sync with the record
$tags = [
    // Note: Already attached tags will be automatically removed from
the record
    // If you want to keep some of the existing tags then you will nee
d to keep them in the array
    "tag" => [
        // Since this tag is already added, it will be preserved
        'already added tag' => 'Already Added Tag',

        // The new tags to add - All other tags that previously existe
d will be deleted
        'new tag' => 'New Tag',
        'new tag2' => 'New Tag2',
    ],
];
// Building SugarFieldTag instance
$SugarFieldTag = new SugarFieldTag();

// Passing the arguments to save the Tags
$SugarFieldTag->apiSave($bean, $tags, $tagField, $tagFieldProperties);
```

TinyMCE

Overview

This article demonstrates how to work with the TinyMCE rich-editor field and customize its settings.

TinyMCE Field Type (htmleditable_tinymce)

Sugar provides a rich-text editor field using TinyMCE. This editor is used available

when customizing templates and composing emails. As Sugar does not provide a way to create and use these field in Studio, we will demonstrate how to convert a text area field into a TinyMCE editor in the following sections.

Converting a Text Area to a TinyMCE Editor

As an example, we will use the Accounts' description field. The description fields type is by default text area. First of all, you will need to change the type of the field to htmleditable_tinymce using the displayParams vardef key.

```
  $dictionary['Account']['fields']['description']['displayParams']['ty
pe'] = 'htmleditable_tinymce';
```

Additionally, having a rich-text editor will require you to have a longer database field to store the HTML content. You will need to change the dbType to longtext and its length to 4294967295.

```
  $dictionary['Account']['fields']['description']['dbType'] = 'longtex
t';
$dictionary['Account']['fields']['description']['len'] = '4294967295';
```

After making these changes, the vardef definition will look as follows:

./custom/Extension/modules/Accounts/Ext/Vardefs/tinymce_description.php

```
  <?php
$dictionary['Account']['fields']['description']['displayParams']['type
'] = 'htmleditable_tinymce';
$dictionary['Account']['fields']['description']['dbType'] = 'longtext'
;
$dictionary['Account']['fields']['description']['len'] = '4294967295';
```

Last, you will need to navigate Admin > Repairs and Perform Quick Repair and Rebuild.

Configuring the TinyMCE Editor

Configuring the TinyMCE editors can be done using the tinyConfig vardef key. The tinyConfig key maps to the TinyMCE configuration options. For example, if you

would add a plugin into TinyMCE editor, you would modify the tinyConfig.plugins vardef key. An example of a vardef definition that will change the accounts description field to a custom TinyMCE Editor and modify its default TinyMCE settings is shown below.

./custom/Extension/modules/Accounts/Ext/Vardefs/tinymce_description.php

```
  <?php
$dictionary['Account']['fields']['description']['displayParams']['type
'] = 'htmleditable_tinymce';
$dictionary['Account']['fields']['description']['dbType'] = 'longtext'
;
$dictionary['Account']['fields']['description']['len'] = '4294967295';
$dictionary['Account']['fields']['description']['tinyConfig'] = array(
    'plugins' => 'paste,autoresize,visualblocks,textcolor,table,emotic
ons,autolink',
    'table_default_attributes' => array(
        'border' => '1',
    ),
    'target_list' => array(
        array(
            'title' => 'New page',
            'value' => '_blank',
        ),
    ),
    'default_link_target' => '_blank',
    'extended_valid_elements' => "a[href|target|data-mce-href]",
    'codesample_languages' => array(
        array('text' => 'HTML/XML', 'value' => 'markup'),
        array('text' => 'JavaScript', 'value' => 'javascript'),
        array('text' => 'CSS', 'value' => 'css'),
        array('text' => 'PHP', 'value' => 'php'),
    ),
    'toolbar1' => 'formatselect | bold italic underline strikethrough
| bullist numlist | forecolor backcolor ',
    'toolbar2' => 'table tabledelete emoticons codesample | tableprops
 tablerowprops tablecellprops | tableinsertrowbefore tableinsertrowaft
er tabledeleterow | tableinsertcolbefore tableinsertcolafter tabledele
tecol | visualblocks removeformat',
);
```

Since you are making changes in the vardef, you will need to navigate Admin > Repairs and run Quick Repair and Rebuild. Once you perform Quick Repair and Rebuild, the description of the Account records will have TinyMCE Editor that

allows you to write rich-text content.

Plugins

Sugar is using TinyMCE v4. Most of the settings that you can define in tinyConfig are exist in TinyMCE documentation. However, it is important to know that plugins are limited with plugins that are provided by Sugar. These plugins can be found in ./include/javascript/tinymce4/plugins.

Configuring the TinyMCE Editors used in BWC Modules

If you are looking for a solution with older TinyMCE that Sugar uses. You can try to create an override file and modify default settings. There are two different overrides that can be applied to buttons and default settings.

Overriding Buttons

The first override file is for the toolbar buttons. To do this, you must create ./custom/include/tinyButtonConfig.php. Within this file, you will be able to override the button layout for the TinyMCE editor.

There are 3 different layouts you can change:

- **default** : Used when creating an email template
- **email_compose** : Used when composing an email from the full form under the Emails module
- **email_compose_light** : Used when doing a quick compose from a listview or subpanel

Example File

./custom/include/tinyButtonConfig.php

```
  <?php

    //create email template
    $buttonConfigs['default'] = array(
        'buttonConfig' => "code,help,separator,bold,italic,underline,s
trikethrough,separator,justifyleft,justifycenter,justifyright,justifyf
ull,separator,forecolor,backcolor,separator,styleselect,formatselect,f
ontselect,fontsizeselect,",
        'buttonConfig2' => "cut,copy,paste,pastetext,pasteword,selecta
ll,separator,search,replace,separator,bullist,numlist,separator,outden
t,indent,separator,ltr,rtl,separator,undo,redo,separator, link,unlink,
anchor,image,separator,sub,sup,separator,charmap,visualaid",
```

```
        'buttonConfig3' => "tablecontrols,separator,advhr,hr,removefor
mat,separator,insertdate,inserttime,separator,preview"
    );

    //Standard email compose
    $buttonConfigs['email_compose'] = array(
        'buttonConfig' => "code,help,separator,bold,italic,underline,s
trikethrough,separator,bullist,numlist,separator,justifyleft,justifyce
nter,justifyright,justifyfull,separator,forecolor,backcolor,separator,
styleselect,formatselect,fontselect,fontsizeselect,",
        'buttonConfig2' => "", //empty
        'buttonConfig3' => "" //empty
    );

    //Quick email compose
    $buttonConfigs['email_compose_light'] = array(
        'buttonConfig' => "code,help,separator,bold,italic,underline,s
trikethrough,separator,bullist,numlist,separator,justifyleft,justifyce
nter,justifyright,justifyfull,separator,forecolor,backcolor,separator,
styleselect,formatselect,fontselect,fontsizeselect,",
        'buttonConfig2' => "", //empty
        'buttonConfig3' => "" //empty
    );
```

Overriding Default Settings

The second override file is for basic TinyMCE functionality. To do this, you must create ./custom/include/tinyMCEDefaultConfig.php. TinyMCE has quite a few settings that can be altered, so the best reference for configuration settings can be found in the TinyMCE Configuration Documentation.

Example File

./custom/include/tinyMCEDefaultConfig.php

```
  <?php

    $defaultConfig = array(
        'convert_urls' => false,
        'valid_children' => '+body[style]',
        'height' => 300,
        'width'    => '100%',
        'theme'    => 'advanced',
        'theme_advanced_toolbar_align' => "left",
```

```
        'theme_advanced_toolbar_location' => "top",
        'theme_advanced_buttons1' => "",
        'theme_advanced_buttons2' => "",
        'theme_advanced_buttons3' => "",
        'strict_loading_mode' => true,
        'mode'    => 'exact',
        'language' => 'en',
        'plugins' => 'advhr,insertdatetime,table,preview,paste,searchr
eplace,directionality',
        'elements' => '',
        'extended_valid_elements' => 'style[dir|lang|media|title|type]
,hr[class|width|size|noshade],@[class|style]',
        'content_css' => 'include/javascript/tiny_mce/themes/advanced/
skins/default/content.css',
    );
```

Creating Plugins

Another alternative to customizing the TinyMCE is to create a plugin. Your plugins will be stored in ./include/javascript/tiny_mce/plugins/. You can find more information on creating plugins on the TinyMCE website.

the use and customization of TinyMCE.

Modifying the TinyMCE Editor

Overview

This article is a brief overview on how to modify the default settings for the TinyMCE editor by creating an override file. There are two different overrides that can be applied to buttons and default settings.

Overriding Buttons

The first override file is for the toolbar buttons. To do this, you must create ./custom/include/tinyButtonConfig.php. Within this file, you will be able to override the button layout for the TinyMCE editor.

There are 3 different layouts you can change:

- **default** : Used when creating an email template
- **email_compose** : Used when composing an email from the full form under

the Emails module
- **email_compose_light** : Used when doing a quick compose from a listview or subpanel

Example File

./custom/include/tinyButtonConfig.php

```
<?php

    //create email template
    $buttonConfigs['default'] = array(
        'buttonConfig' => "code,help,separator,bold,italic,underline,s
trikethrough,separator,justifyleft,justifycenter,justifyright,justifyf
ull,separator,forecolor,backcolor,separator,styleselect,formatselect,f
ontselect,fontsizeselect,",
        'buttonConfig2' => "cut,copy,paste,pastetext,pasteword,selecta
ll,separator,search,replace,separator,bullist,numlist,separator,outden
t,indent,separator,ltr,rtl,separator,undo,redo,separator, link,unlink,
anchor,image,separator,sub,sup,separator,charmap,visualaid",
        'buttonConfig3' => "tablecontrols,separator,advhr,hr,removefor
mat,separator,insertdate,inserttime,separator,preview"
    );

    //Standard email compose
    $buttonConfigs['email_compose'] = array(
        'buttonConfig' => "code,help,separator,bold,italic,underline,s
trikethrough,separator,bullist,numlist,separator,justifyleft,justifyce
nter,justifyright,justifyfull,separator,forecolor,backcolor,separator,
styleselect,formatselect,fontselect,fontsizeselect,",
        'buttonConfig2' => "", //empty
        'buttonConfig3' => "" //empty
    );

    //Quick email compose
    $buttonConfigs['email_compose_light'] = array(
        'buttonConfig' => "code,help,separator,bold,italic,underline,s
trikethrough,separator,bullist,numlist,separator,justifyleft,justifyce
nter,justifyright,justifyfull,separator,forecolor,backcolor,separator,
styleselect,formatselect,fontselect,fontsizeselect,",
        'buttonConfig2' => "", //empty
        'buttonConfig3' => "" //empty
    );
```

Overriding Default Settings

The second override file is for basic TinyMCE functionality. To do this, you must create ./custom/include/tinyMCEDefaultConfig.php. TinyMCE has quite a few settings that can be altered, so the best reference for configuration settings can be found in the TinyMCE Configuration Documentation.

Example File

./custom/include/tinyMCEDefaultConfig.php

```
<?php

    $defaultConfig = array(
        'convert_urls' => false,
        'valid_children' => '+body[style]',
        'height' => 300,
        'width'    => '100%',
        'theme'    => 'advanced',
        'theme_advanced_toolbar_align' => "left",
        'theme_advanced_toolbar_location' => "top",
        'theme_advanced_buttons1' => "",
        'theme_advanced_buttons2' => "",
        'theme_advanced_buttons3' => "",
        'strict_loading_mode' => true,
        'mode'    => 'exact',
        'language' => 'en',
        'plugins' => 'advhr,insertdatetime,table,preview,paste,searchr
eplace,directionality',
        'elements' => '',
        'extended_valid_elements' => 'style[dir|lang|media|title|type]
,hr[class|width|size|noshade],@[class|style]',
        'content_css' => 'include/javascript/tiny_mce/themes/advanced/
skins/default/content.css',
    );
```

Creating Plugins

Another alternative to customizing the TinyMCE is to create a plugin. Your plugins will be stored in ./include/javascript/tiny_mce/plugins/. You can find more information on creating plugins on the TinyMCE website.

SugarPDF

Overview

An overview of SugarPDF and how it relates to TCPDF. As of version 6.7.x, Sugar includes a Smarty template engine called PDF Manager that is accessible by navigating to Admin > PDF Manager. The PDF Manager allows administrators to create and manage templates for deployed modules without having to write custom code. The following sections are only specific to developers looking to create their own custom TCPDF templates using PHP.

PDF Classes

The various classes used in generating PDFs within Sugar.

TCPDF

Sugar uses the TCPDF 4.6.013 library, located in ./vendor/tcpdf/, as the core engine to generate PDFs. This library is extended by Sugarpdf which is used by the core application to generate PDFs.

Sugarpdf

The Sugarpdf class, located in ./include/Sugarpdf/Sugarpdf.php, extends the TCPDF class. Within this class, we have overridden certain functions to integrate sugar features. Some key methods that were overridden are listed below:

Method	Description
Header	Overridden to enable custom logos.
SetFont	Overridden to allow for the loading of custom fonts. The custom fonts direction is defined by the K_PATH_CUSTOM_FONTS var. By default, this value is set to ./custom/vendor/tcpdf/fonts/
Cell	Overridden to apply the prepare_string() function. This allows for the ability to clean the string before sending to PDF.

Some key additional methods that were added are listed below:

Method	Description

predisplay	Preprocessing before the display method is called. Is intended to setup general PDF document properties like margin, footer, header, etc.
display	Performs the actual PDF content generation. This is where the logic to display output to the PDF should be placed.
process	Calls predisplay and display.
writeCellTable	Method to print a table using the cell print method of TCPDF
writeHTMLTable	Method to print a table using the writeHTML print method of TCPDF

Custom PDF classes that extend Sugarpdf can be located in the following directories:

- ./include/Sugarpdf/sugarpdf/sugarpdf.<pdf view>.php
- ./modules/<module>/sugarpdf/sugarpdf.<pdf view>.php
- ./custom/modules/<module>/sugarpdf/sugarpdf.<pdf view>.php

PDF Manager classes that extend Sugarpdf are located in the following directories:

- ./include/Sugarpdf/sugarpdf/sugarpdf.smarty.php
- ./include/Sugarpdf/sugarpdf/sugarpdf.pdfmanager.php
- ./custom/include/Sugarpdf/sugarpdf/sugarpdf.pdfmanager.php

Each extended class will define a specific PDF view that is accessible with the following URL parameters:

- module=<module>
- action=sugarpdf
- sugarpdf=<pdf view>

Within each custom PDF class, the display method will need to be redefined. If you would like, It is also possible to override other methods like Header. The process method of this class will be called from ViewSugarpdf. When a PDF is being generated, the most relevant sugarpdf.<pdf action>.pdf class is determined by the SugarpdfFactory.

ViewSugarpdf

The ViewSugarpdf class, located in ./include/MVC/View/views/view.sugarpdf.php, is used to create the SugarViews

that generate PDFs. These views can be found and/or created in one of the following directory paths:

- ./modules/<module>/views/view.sugarpdf.php
- ./custom/modules/<module>/views/view.sugarpdf.php

SugarViews can be called by navigating to a URL in the format of:

```
http://<url>/index.php?module=<module>&action=sugarpdf&sugarpdf=<pdf a
ction>
```

As of 6.7, PDFs are mainly generated using the PDF Manager templating system. To generate the PDF stored in the PDF Manager, you would call a URL in the format of:

```
http://<url>/index.php?module<module>&record=<record id>&action=sugarp
df&sugarpdf=pdfmanager&pdf_template_id=<template id>
```

SugarpdfFactory

The ViewSugarpdf class uses the SugarpdfFactory class, located in ./include/Sugarpdf/SugarpdfFactory.php, to find the most relevant sugarpdf.<pdf action>.pdf class which generates the PDF document for a given PDF view and module. If one is not found, then the core Sugarpdf class is used.

The SugarpdfFactory class loads the first class found for the specified PDF action as determined by the following order:

- ./custom/modules/<module>/sugarpdf/sugarpdf.<pdf view>.php
- ./modules/<module>/sugarpdf/sugarpdf.<pdf view>.php
- ./custom/include/Sugarpdf/sugarpdf/sugarpdf.<pdf view>.php
- ./include/Sugarpdf/sugarpdf/sugarpdf.<pdf view>.php
- ./include/Sugarpdf/sugarpdf.php

SugarpdfHelper

The SugarpdfHelper, located in ./include/Sugarpdf/SugarpdfHelper.php, is included by Sugarpdf. This is a utility file that contains many of the functions we use to generate PDFs.
Available functions are:

- **wrapTag**, **wrapTD**, **wrapTable, etc.** : These functions help to create an

HTML code.
- **prepare_string** : This function prepare a string to be ready for the PDF printing.
- **format_number_sugarpdf** : This function is a copy of format_number() from currency with a fix for sugarpdf.

PdfManagerHelper

The PdfManagerHelper, located in ./modules/PdfManager/PdfManagerHelper.php, is primarily utilized by the pdfmanager Sugarpdf view. This class file contains methods useful for accessing PDF Manager templates. Some of the primary methods are:

- **getAvailableModules** : Returns an array of available modules for use with PdfManager.
- **getFields** : Takes an module name and returns a list of fields and links available for this module in PdfManager.
- **getPublishedTemplatesForModule** : Returns an array of the available templates for a specific module.

FontManager

The FontManagerclass, located in ./include/Sugarpdf/FontManager.php, is a stand-alone class that manages all the fonts for TCPDF. More information can be found in the PDF Fonts section below.

Functionality:

- List all the available fonts from the OOB font directory and the custom font directory (it can create a well-formatted list for select tag).
- Get the details of each listed font (Type, size, encoding,...) by reading the font php file.
- Add a new font to the custom font directory from a font file and a metric file.
- Delete a font from the custom font directory.

Generating a PDF

To generate a custom PDF in Sugar, you will need to create a PDF view that extends the Sugarpdf class. To accomplish this, create the file:

./custom/modules/<module>/sugarpdf/sugarpdf.<pdf view>.php

```
<?php
```

```
if(!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry Point
');

require_once('include/Sugarpdf/Sugarpdf.php');

class <module>Sugarpdf<pdf view> extends Sugarpdf
{
    /**
     * Override
     */
    function process(){
        $this->preDisplay();
        $this->display();
        $this->buildFileName();
    }

    /**
     * Custom header
     */
    public function Header()
    {
        $this->SetFont(PDF_FONT_NAME_MAIN, 'B', 16);
        $this->MultiCell(0, 0, 'TCPDF Header',0,'C');
        $this->drawLine();
    }

    /**
     * Custom header
     */
    public function Footer()
    {
        $this->SetFont(PDF_FONT_NAME_MAIN, '', 8);
        $this->MultiCell(0,0,'TCPDF Footer', 0, 'C');
    }

    /**
     * Predisplay content
     */
    function preDisplay()
    {
        //Adds a predisplay page
        $this->AddPage();
        $this->SetFont(PDF_FONT_NAME_MAIN,'',PDF_FONT_SIZE_MAIN);
        $this->Ln();
        $this->MultiCell(0,0,'Predisplay Content',0,'C');
    }
```

```
    /**
     * Main content
     */
    function display()
    {
        //add a display page
        $this->AddPage();
        $this->SetFont(PDF_FONT_NAME_MAIN,'',PDF_FONT_SIZE_MAIN);
        $this->Ln();
        $this->MultiCell(0,0,'Display Content',0,'C');
    }

    /**
     * Build filename
     */
    function buildFileName()
    {
        $this->fileName = 'example.pdf';
    }

    /**
     * This method draw an horizontal line with a specific style.
     */
    protected function drawLine()
    {
        $this->SetLineStyle(array('width' => 0.85 / $this->getScaleFac
tor(), 'cap' => 'butt', 'join' => 'miter', 'dash' => 0, 'color' => arr
ay(220, 220, 220)));
        $this->MultiCell(0, 0, '', 'T', 0, 'C');
    }
}
```

This file will contain the markup for the PDF. The main things to note are the Header(), Footer() and display() methods as they contain most of the styling and display logic. The method buildFileName() will generate the document's name when downloaded by the user.

Once in place, navigate to Admin > Repair > Quick Repair and Rebuild. The PDF will now be accessible by navigating to the following url in your browser:

```
http://{sugar url}/index.php?module=<module>&action=sugarpdf&sugarpdf=
<pdf action>
```

PDF Settings

This section will outline how to configure the PDF settings. These settings determine the widths, heights, images, pdf metadata, and the UI configurations found in:

```
Admin > PDF Manager > Edit Report PDF Template
```

Settings

The default PDF settings for TCPDF can be found in ./include/Sugarpdf/sugarpdf_default.php. You can add additional custom settings by creating the following file:

./custom/include/Sugarpdf/sugarpdf_default.php

```
  <?php

$sugarpdf_default["PDF_NEW_SETTING"] = "Value";
```

You should note that values specified here will be the default values. Once edited, the updated values are stored in the database config table under the category "sugarpdf" and a name matching the setting name.

```
category: sugarpdf
name: pdf_new_setting
value: Value
```

Displaying and Editing Settings

A select set of settings can be edited within the Sugar UI by navigating to:

```
  Admin > PDF Manager > Edit Report PDF Template
```

The settings here are managed in the file ./modules/Configurator/metadata/SugarpdfSettingsdefs.php. A brief description of the settings parameters are listed below:

- **label** : This is the display label for the setting.
- **info_label** : Hover info text for the display label.
- **value** : The PDF Setting.
- **class** : Determines which panel the setting resides in. Possible values are 'basic' and 'logo'. 'advanced' is not currently an available value.
- **type** : Determines the settings display widget. Possible values are: 'image', 'text', 'percent', 'multiselect', 'bool', 'password', and 'file'.

Custom settings can be added to this page by creating ./custom/modules/Configurator/metadata/SugarpdfSettingsdefs.php and specifying a new setting index. An example is below:

./custom/modules/Configurator/metadata/SugarpdfSettingsdefs.php

```
  <?php

    //Retrieve setting info from db
    defineFromConfig("PDF_NEW_SETTING", $sugarpdf_default["PDF_NEW_SET
TING"]);

    //Add setting display
    $SugarpdfSettings['sugarpdf_pdf_new_setting'] = array(
        "label" => $mod_strings["LBL_PDF_NEW_SETTING_TITLE"],
        "info_label" => $mod_strings["LBL_PDF_NEW_SETTING_INFO"],
        "value" => PDF_NEW_SETTING,
        "class" => "basic",
        "type" => "text",
    );
```

You should note that the $SugarpdfSettings index should following the format sugarpdf_pdf_<setting name>. If your setting does not follow this format, it will not be saved or retrieved from the database correctly.

Once the setting is defined, you will need to define the display text for the UI setting.

./custom/modules/Configurator/language/en_us.lang.php

Once finished, navigate to Admin > Repair > Quick Repair and Rebuild. This will rebuild the language files for your display text.

PDF Fonts

The stock fonts for TCPDF are stored in the directory ./vendor/tcpdf/fonts/. If you would like to add additional fonts to the system, they should be added to the ./custom/vendor/tcpdf/fonts/ directory.

Font Cache

The font list is built by the font manager with the listFontFiles() or getSelectFontList() methods. The list is then saved in the cache as ./cache/Sugarpdf/cachedFontList.php. This caching process prevents unnecessary parsing of the fonts folder. The font cache is automatically cleared when the delete() or add() methods are used. When you create a module loader package to install fonts you will have to call the clearCachedFile() method in a post_execute and post_uninstall action in the manifest.php to clear the font cache.

DateTime

Overview

The SugarDateTime class, located in, ./include/SugarDateTime.php, extends PHP's built in DateTime class and can be used to perform common operations on date and time values.

Setting

With existing SugarDateTime objects, it is recommended to clone the object before modifying or performing operations on it.

```
$new_datetime = clone $datetime;
```

Another option is to instantiate a new SugarDateTime object.

```
// defaults to now
$due_date_time = new SugarDateTime();

$due_date_time->setDate($dueYear, $dueMonth, $dueDay);
$due_date_time->setTime($dueHour, $dueMin);
$due_date_time->setTimezone($dueTZ);
```

Note : When creating a new SugarDateTime object, the date and time will default

to the current date and time in the timezone configured in PHP.

SugarDateTime Objects can also be modified by passing in common English textual date/time strings to manipulate the value.

```
$due_date = clone $date;
$end_of_the_month->modify("last day of this month");
$end_of_next_month->modify("last day of next month");
$thirty_days_later->modify("+30 days");
```

Formatting

When formatting SugarDateTime objects into a string for display or logging purposes, there are a few options you can utilize through the formatDateTime() method.

Return the date/time formatted for the database:

```
$db_datetime_string = formatDateTime("datetime", "db");
```

Return the date/time formatted using a user's preference:

```
global $current_user;
$user_datetime_str
ing = formatDateTime("datetime", "user", $current_user);
```

Return the date/time formatted following ISO-8601 standards:

```
$iso_datetime_string = formatDateTime("datetime", "iso");
```

There are times when it is needed to return only certain parts of a date or time. The format() method accepts a string parameter to define what parts or information about the date/time should be returned.

```
// 28 Dec 2016
echo $due_date_time->format("j M Y");
```

```
// 2016-12-28 19:01:09
echo $due_date_time->asDb();

// The date/time 2016-12-28T11:01:09-08:00 is set in the timezone of A
merica/Los_Angeles
echo "The date/time " . $due_date_time->format("c") . " is set in the
timezone of " . $due_date_time->format("e");

// The time when this date/time object was created is 11:01:09 AM -080
0
echo "The time when this date/time object was created is " . $due_date
_time->format("H:i:s A O");

// There are 31 days in the month of December
echo "There are " . $due_date_time->format("t") . " days in the month
of " . $due_date_time->format("F");

// There are 366 days in the year of 2016
echo "There are " . $due_date_time->__get("days_in_year") . " days in
the year of " . $due_date_time->format("Y");

// Between Wed, 28 Dec 2016 11:01:09 -0800 and the end of the year, th
ere are 4 days.
echo "Between " . $due_date_time . " and the end of the year, there ar
e " . ($due_date_time->__get("days_in_year") - $due_date_time->format(
"z")) . " days.";
```

For more information on the available options, please refer to http://php.net/manual/en/function.date.php

TimeDate Class

The TimeDate class, located in ./include/TimeDate.php, utilizes the SugarDateTime class to provide an extended toolset of methods for date/time usage.

Setting

With existing TimeDate objects, it is recommended to clone the object before modifying or performing operations on it.

```
$new_timedate = clone $timedate;
```

Another option is to instantiate a new TimeDate object.

```
// current time in UTC
$timedate = new TimeDate();
$now_utcTZ = $timedate->getNow();

// current time in user's timezone
$timedate = new TimeDate($current_user);
$now_userTZ = $timedate->getNow(true);
```

Note : When creating a new TimeDate object, the date and time will default to the current date and time in the UTC timezone unless a user object is passed in.

Formatting

TimeDate objects can return the Date/Time in either a string format or in a SugarDateTime object. The TimeDate object has many formatting options; some of the most common ones are :

getNow	Get the current date/time as a *SugarDateTime* object
fromUser	Get *SugarDateTime* object from user's date/time string
asUser	Format *DateTime* object as user's date/time string
fromDb	Get *SugarDateTime* object from a DB date/time string
fromString	Create a *SugarDateTime* object from a string
get_db_date_time_format	Gets the current database date and time format
to_display_date_time	Format a string as user's display date/time

getNow

returns *SugarDateTime* object

Parameters

Name	Data Type	Default	Description
$userTz	*bool*	false	Whether to use the user's timezone or not. False will use UTC

Returns a SugarDateTime object set to the current date/time. If there is a user object associate to the TimeDate object, passing true to the $userTz parameter will return the object in user's timezone. If no user is associated or false is passed, the timezone returned will be in UTC.

Example

```
$timedate = new TimeDate($current_user);

// returns current date/time in UTC
$now_utcTZ = $timedate->getNow();

// returns current date/time in the user's timezone
$now_userTZ = $timedate->getNow(true);
```

fromUser

returns SugarDateTime object

Parameters

Name	Data Type	Default	Description
$date	*string*	n/a	Date/Time string to convert to an object
$user	*User*	null	User to utilize formatting and timezone info from

Returns a SugarDateTime object converted from the passed in string. If there is a user object passed in as a parameter will return the object in user's timezone. If no user is passed in, the timezone returned will be in UTC.

Example

```
$date = "2016-12-28 13:09";
$timedate = new TimeDate();
$datetime = $timedate->fromUser($date, $current_user);
```

asUser

returns *string*

Parameters

Name	Data Type	Default	Description
$date	*string*	n/a	Date/Time string to convert to an object
$user	*User*	null	User to utilize formatting and timezone info from

Returns a string of the date and time formatted based on the user's profile settings. If there is a user object passed in as a parameter it will return the object in user's timezone. If no user is passed in, the timezone returned will be in UTC.

Example

```
$datetime = new datetime("2016-12-28 13:09");
$timedate = new TimeDate();
$formattedDate = $timedate->asUser($datetime, $current_user);
```

fromDb

returns *SugarDateTime*

Parameters

Name	Data Type	Default	Description
$date	*string*	n/a	Date/Time string in database format to convert to an object

Returns a SugarDateTime object converted from the passed in database formatted string.

Note : If the string does not match the database format exactly, this function will return *boolean* false.

Example

```
// Y-m-d H:i:s
$db_datetime = "2016-12-28 13:09:01";
$timedate = new TimeDate();
$datetime = $timedate->fromDb($db_datetime);

// returns bool(false)
$wrong_datetime = "2016-12-28 13:09";
$timedate = new TimeDate();
$datetime = $timedate->fromDb($timedate);
```

fromString

returns *SugarDateTime*

Parameters

Name	Data Type	Default	Description
$date	*string*	n/a	Date/Time string to convert to an object
$user	*User*	null	User to utilize timezone info from

Returns a SugarDateTime object converted from the passed in database formatted string. If there is a user object passed in as a parameter it will return the object in user's timezone. If no user is passed in, the timezone returned will be in UTC.

Example

```
$datetime_str = "December 28th 2016 13:09";
$timedate = new TimeDate();
$datetime = $timedate->fromString($datetime_str);
```

get_db_date_time_format

returns *string*

Parameters

N/a

Returns a string of the current database date and time format settings.

Note : For just the date format use get_db_date_format() and for just the time format use get_db_time_format().

Example

```
$timedate = new TimeDate();

// Y-m-d H:i:s
$db_format = $timedate->get_db_date_time_format();
```

to_display_date_time

returns *string*

Parameters

Name	Data Type	Default	Description
$date	*string*	n/a	Date/Time string in database format
$meridiem	*boolean*	true	Deprecated -- Remains for backwards compatibility only
$convert_tz	*boolean*	true	Convert to user's timezone
$user	*User*	null	User to utilize formatting and timezone info from

Returns a string of the date and time formatted based on the user's profile settings. If no user is passed in, the current user's default will be used. If $convert_tz is true the string returned will be in the passed in user's timezone. If no user is passed in, the timezone returned will be in UTC.

Note : If the string does not match the database format exactly, this function will return boolean false.

Example

```
$datetime_str = "2016-12-28 13:09:01";

// 12-28-2016 07:09
$timedate = new TimeDate();
```

```
$datetime = $timedate->to_display_date_time($datetime_str);

// 2016-12-28 13:09
$timedate = new TimeDate();
$datetime = $timedate->to_display_date_time($datetime_str, true, false
, $diff_user);
```

Parsing

In addition to formatting, TimeDate objects can also return Date/Time values based on special parameters. The TimeDate object has many date parsing options; some of the most common ones are :

parseDateRange	Gets the start and end date/time from a range keyword

parseDateRange

returns *array*

Parameters

Name	Data Type	Default	Description
$range	*string*	n/a	String from a predetermined list of available ranges
$user	*User*	null	User to utilize formatting and timezone info from
$adjustForTimezone	*boolean*	true	Convert to user's timezone

Returns an array of SugarDateTime objects containing the start and Date/Time values calculated based on the entered parameter. If no user is passed in, the current user's default will be used. If $adjustForTimezone is true the string returned will be in the passed in user's timezone. If NULL is passed in as the user, the timezone returned will be in UTC. The available values for $range are :

Range String	Description
yesterday	Yesterday's start and end date/time
today	Today's start and end date/time

tomorrow	Tomorrows's start and end date/time
last_7_days	Start and end date/time of the last seven days
next_7_days	Start and end date/time of the next seven days
last_30_days	Start and end date/time of the last thirty days
next_30_days	Start and end date/time of the next thirty days
next_month	Start and end date/time of next month
last_month	Start and end date/time of last month
this_month	Start and end date/time of the current month
last_year	Start and end date/time of last year
this_year	Start and end date/time of the current year
next_year	Start and end date/time of next year

Example

```
$timedate = new TimeDate($current_user);

// returns today's start and end date/time in UTC
$today_utcTZ = $timedate->parseDateRange("today", NULL, false);

// returns today's start and end date/time in the user's timezone
$today_userTZ = $timedate->parseDateRange("today", $current_user, true
);
```

Shortcuts

Overview

Shortcuts is a framework to add shortcut keys to the application. When shortcut keys are registered, they are registered to a particular shortcut session. Shortcut sessions group shortcut keys, and they can be activated, deactivated, saved, and restored. Global shortcut keys can be registered, but they are not tied to a particular shortcut session. They are rather applied everywhere in the application and can only be applied once.

The Shortcut framework is implemented on top of Mousetrap library.

Shortcut Sessions

In order to register a shortcut in Sugar, a shortcut session must first be created by adding ShortcutSession plugin to the top-level layout JavaScript controller.

```
  plugins: ['ShortcutSession']
```

Then, the top-level layout needs to list which shortcuts are allowed in the shortcut session. Shortcuts can be listed in the top-level layout JavaScript controller :

./custom/clients/layout/my-layout/my-layout.js

```
  shortcuts: [
    'Sidebar:Toggle',
    'Record:Save',
    'Record:Cancel',
    'Record:Action:More'
]
```

Shortcuts can also be listed in the metadata :

./custom/clients/layout/my-layout/my-layout.php

```
  'shortcuts' => array(
    'Sidebar:Toggle',
    'Record:Save',
    'Record:Cancel',
    'Record:Action:More'
),
```

Register

Once a shortcut session is created, shortcut keys can be registered by adding the following in your JavaScript code :

```
  app.shortcuts.register('<unique shortcut ID>', '<shortcut keys>', <c
allback function>, <current component>, <call on focus?>);
```

Since shortcut keys should only be registered once in your component, they should be registered inside initialize() or someplace where re-rendering the component would not register more than once.

Shortcut IDs

Shortcut IDs should be unique across the application. They should be namespaced by convention, for example Record:Next, Sidebar:Toggle, List:Edit. If the namespace starts with Global:, it assumes that the shortcut is global.

Global shortcuts

Global shortcuts are shortcut keys that are applied once and are available everywhere in the application.

```
  app.shortcuts.register(app.shortcuts.GLOBAL + 'Footer:Help', '?', fu
nction() {}, this, false);
```

Shortcut Keys

There are only certain keys that are supported under the Shortcut framework. The following keyboard keys can be used :

- All alphanumeric characters and symbols
- shift, ctrl, alt, option, meta, command
- backspace, tab, enter, return, capslock, esc, escape, space, pageup, pagedown, end, home, ins, del
- left, up, right, down

Additional Features

In addition to standard keys, the Shortcut framework also supports some additional features :

- **Key combinations** : 'ctrl+s'
- **Multiple keys** : ['ctrl+a', 'command+a']
- **Key sequences** : 'f a'

Input Focus

The fifth parameter in the register method specifies whether or not the shortcut key should be fired when the user is focused in an input field. It is false by default.

```
  app.shortcuts.register('Record:Save', ['ctrl+s','command+s'], functi
```

```
on() {}, this, true);
```

Limitations

Shortcut keys do not work on side panels, like dashboards and previews.

Shortcuts Help

Anytime a new shortcut is created, a help text should be provided in ./clients/base/layouts/shortcuts/shortcuts.php with a shortcut ID and a language string.

```
  'List:Favorite' => 'LBL_SHORTCUT_FAVORITE_RECORD',
```

Shortcuts help is displayed when Shortcuts button is clicked.

Advanced Features

Enable/disable dynamically

Shortcuts feature can be enabled and disabled dynamically via code by calling app.shortcuts.enable() and app.shortcuts.disable().

Dynamically saving, creating new, and restoring sessions

A new shortcut session is created when a user visits a top-level layout with ShortcutSession plugin. A new session can be created dynamically:

```
  app.shortcuts.createSession(<array of shorcut IDs>, <component>);
```

But before creating a new session, the current session should be saved first.

```
  app.shortcuts.saveSession();
app.shortcuts.createSession(<array of shorcut IDs>, <component>);
```

Once a new session is created, shortcut keys can be registered to it. When the need for the session is done, the previous shortcut session can be restored.

```
  app.shortcuts.restoreSession();
```

Example

The following example will be to add some custom shortcuts onto a custom layout. For more information on how to create a custom layout, please refer to the Creating Layouts documentation.

First, on our custom JavaScript controller for our layout, we must enable the ShortcutSession plugin as well as list the shortcuts we will be enabling :

./custom/clients/base/layouts/my-layout/my-layout.js

```
  ({
    plugins: ['ShortcutSession'],
    shortcuts: [
        'Global:MyLayout:MyCallback',
    ],
})
```

Next, on the custom view being rendered on our layout, we will register the new shortcuts as well as define a callback method :

./custom/clients/base/views/my-view/my-view.js

```
...
initialize: function(options){
    this._super('initialize', [options]);

    // call myCallback method when command + a is pressed
    app.shortcuts.register(app.shortcuts.GLOBAL + 'MyLayout:MyCallback
', 'command+a', this.myCallback, this, false);

    app.shortcuts.saveSession();
    app.shortcuts.createSession([
        'MyLayout:InlineCall'
    ], this);

    // call inline code when control + m or command + m is pressed
    app.shortcuts.register('MyLayout:InlineCall', ['ctrl+m','command+m
'], function() {
  console.log("Ctrl or Command m has been pressed");
```

```
    }, this, false);
},
myCallback: function(){
    console.log("MyCallback called from Command a");
},
...
```

The last step will be to create a new label for our shortcut help text and register the help so it displays when "Shortcuts" is click in the footer of the page :

./custom/Extension/application/Ext/Language/en_us.LBL_MYLAYOUT_SHORTCUT_HELP.php

```
  <?php

$app_strings['LBL_MYLAYOUT_MYCALLBACK_HELP'] = "Activates my Callback Method";
$app_strings['LBL_MYLAYOUT_MYINLINECALL_HELP'] = "Activates Inline Code";
```

./custom/Extension/application/Ext/clients/base/layouts/shortcuts/shortcuts.php

```
  <?php

$viewdefs['base']['layout']['shortcuts']['help']['Global:MyLayout:MyCallback'] = 'LBL_MYLAYOUT_MYCALLBACK_HELP';
$viewdefs['base']['layout']['shortcuts']['help']['MyLayout:InlineCall'] = 'LBL_MYLAYOUT_MYINLINECALL_HELP';
```

Navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and add the new shortcut to the custom layout.

Themes

Overview

How to customize the Sugar theme.

Theme Variables

Sugar's theme variables, defined in ./styleguide/themes/clients/base/default/variables.php, determine the color of the primary action buttons, links, and header navigation. An example of the definition is shown below:

./styleguide/themes/clients/base/default/variables.php

```
  $lessdefs = array(
    'colors' => array(
        /**
         * Secondary Color:
         * color of the navbar
         * ------------------------
         * was @secondary
         */
        'NavigationBar' => '#fff',

        /**
         * Primary Button Color:
         * color of the primary button
         * -------------------------
         * was @primaryBtn
         */
        'PrimaryButton' => '#0679c8',
    ),
);
```

Variable Descriptions

It is important to understand what the purpuse and utility of each variable is before you apply in your code, below is an in-depth description:

@NavigationBar

- Used to customize the mega menu background colour
- Defaults to Sugar's @white (#ffffff)
- Note: This variable is only supported in light mode

@PrimaryButton

- Used to customize the background colour for primary buttons (elements using `btn btn-primary`)

- Defaults to Sugar's @blue (#0679c8)

@LinkColor

- Used to customize the text color of link (anchor) elements
- Defaults to Sugar's @blue (#0679c8)
- Note: This variable is only supported in light mode

Custom Themes

Modifications to the theme can be made by creating ./custom/themes/clients/base/default/variables.php. Within this file, you can define the custom hex codes for the colors you would like to use. You should note that this is limited to the @NavigationBar, and @PrimaryButton less variables. An example is shown below.

./custom/themes/clients/base/default/variables.php

Note: Developers cannot override existing bootstrap less variables using this file.

Adding CSS

Sugar allows you to customize CSS using the less language in ./custom/themes/custom.less. As Sugar makes use of the Bootstrap library, you have access to the features of Bootstrap and can make use of its variables to create your own CSS. An example is shown below.

./custom/themes/custom.less

```
  //You can import any less file you want and define your own file str
ucture
//@import 'anyotherfile.less

@myCustomBgColor: lighten(@NavigationBar,10%); //variable defined on a
 custom variable.

.myCustomClass {
    color: @linkColor; //bootstrap variable
    background-color: @myCustomBgColor;
}
```

Overriding CSS Definitions

You can use the ./custom/themes/custom.less file to override CSS classes. The

following example overrides the record label font size.

./custom/themes/custom.less

```
  /*
* Changes record field label sizes to 13px;
*/
.record-cell .record-label{
    font-size:13px;
}
```

Overriding the MegaMenu

As the MegaMenu has limited color customization within ./custom/themes/clients/base/default/variables.php, you may want to customize the look and feel further. The following example will automatically set the menu's link color to a contrasting color based on the @NavigationBar variable and determine the hover and active colors for the tabs.

./custom/themes/custom.less

```
  // Workaround for luminance calculation
// Use luma() function once Sugar upgraded Lessphp to 1.7+ (see: http:
//lesscss.org/functions/#color-channel-luminance)
@luma: 1 - ( (0.299 * red(@NavigationBar)) + (0.587 * green(@Navigatio
nBar)) + (0.114 * blue(@NavigationBar)))/255;

/**
 * LESS GUARDS
 */
// General Nav Colors
.mixin-color() {
    // Darker Colors
    & when (@luma > 0.5) {
        color: darken(contrast(@NavigationBar), 10%) !important;
    }
    // Bright Colors
    & when (@luma <= 0.5) {
        color: lighten(contrast(@NavigationBar), 10%) !important;
        // color: darken(@NavigationBar, 30%) !important;
    }
}

// Nav Fa Icon Colors
```

```
.mixin-fa-color(){
    // Darker Colors
    & when (@luma > 0.5) {
        color: darken(contrast(@NavigationBar), 10%) !important;
    }
    // Bright Colors
    & when (@luma <= 0.5) {
        // color: lighten(@NavigationBar, 30%);
        color: lighten(contrast(@NavigationBar), 10%) !important;
        // color: darken(@NavigationBar, 30%) !important;
    }
}

// Hover Button Groups Background colors
.mixin-background-color-hover(){
    // Dark Colors
    & when (@luma > 0.5) {
        background-color: lighten(@NavigationBar, 15%) !important;
    }
    // Bright Colors
    & when (@luma <= 0.5) {
        background-color: darken(@NavigationBar, 15%) !important;
    }
}

// Hover Button Groups colors
.mixin-color-hover(){
    // Dark Colors
    & when (@luma > 0.5) {
        color: darken(contrast(@NavigationBar), 10%) !important;
    }
    // Bright Colors
    & when (@luma <= 0.5) {
        // color: lighten(@NavigationBar, 20%) !important;
        color: lighten(contrast(@NavigationBar), 10%) !important;
    }
}

// Active Button Group Background Colors
.mixin-background-color-active(){
    // Dark Colors
    & when (@luma > 0.5) {
        background-color: lighten(@NavigationBar, 10%) !important;
    }
    // Bright Colors
    & when (@luma <= 0.5) {
```

```
        background-color: darken(@NavigationBar, 10%) !important;
    }
}
// Active Button Group Hover Colors
.mixin-color-active-hover(){
    // Dark Colors
    & when (@luma > 0.5) {
        color: darken(contrast(@NavigationBar), 5%) !important;
    }
    // Bright Colors
    & when (@luma <= 0.5) {
        color: lighten(contrast(@NavigationBar), 5%) !important;
    }
}

// Open Button Group Background Colors
.mixin-background-color-open(){
    // Dark Colors
    & when (@luma > 0.5) {
        background-color: lighten(@NavigationBar, 20%) !important;
    }
    // Bright Colors
    & when (@luma <= 0.5) {
        background-color: darken(@NavigationBar, 20%) !important;
    }
}

// Open Button Group Hover Colors
.mixin-color-open-hover(){
    // Dark Colors
    & when (@luma > 0.5) {
        color: darken(contrast(@NavigationBar), 15%) !important;
    }
    // Bright Colors
    & when (@luma <= 0.5) {
        color: lighten(contrast(@NavigationBar), 15%) !important;
    }
}

// Background/Foreground Dropdown Menu Hover
.mixin-background-foreground-dropdown-menu-hover(){
    // Dark Colors
    & when (@luma > 0.5) {
        background-color: lighten(@NavigationBar, 15%) !important;
        color: darken(contrast(@NavigationBar), 15%) !important;
        .fa {
```

```
            color: darken(contrast(@NavigationBar), 15%) !important;
        }
    }
    // Bright Colors
    & when (@luma <= 0.5) {
        background-color: @NavigationBar !important;
        color: lighten(contrast(@NavigationBar), 5%) !important;
        .fa {
            color: lighten(contrast(@NavigationBar), 5%) !important;
        }
    }
}

/**
 * LESS Definitions
 */

// Nav Button Group
.module-list .megamenu > .dropdown .module-name{
    .mixin-color;
}

// Home Button Caret
.navbar .megamenu > .dropdown.active .btn-group:not(.open).home .fa-
caret-down,

// More Button Caret
.module-list .megamenu > .dropdown.more .fa,

// Module Toggle caret
.navbar .megamenu > .dropdown .btn-group > .dropdown-toggle .fa {
    .mixin-fa-color;
}

// Nav Button Group Hover
.megamenu .dropdown .btn-group{
    &:hover, &:focus{
        .mixin-background-color-hover;
        .btn,
        > .dropdown-toggle .fa{
            .mixin-color-hover;
        }
    }
}

// Active Button Group
```

```
.navbar .megamenu > .dropdown.active .btn-group{
    .mixin-background-color-active;
    > .dropdown-toggle .fa,
    > a.btn{
        .mixin-fa-color;
    }
}

// Active Button Group Hover
.navbar .megamenu > .dropdown.active .btn-group:hover{
    .mixin-color-active-hover;
    > .dropdown-toggle .fa,
    > a.btn{
        .mixin-color-active-hover;
    }
}

// Open Nav Button Group
.navbar .megamenu > .dropdown .btn-group.open{
    .mixin-background-color-open;
    > .dropdown-toggle .fa,
    > a.btn{
        .mixin-fa-color;
    }
}

// Open Nav Button Group Hover
.navbar .megamenu > .dropdown .btn-group.open:hover{
    .mixin-color-open-hover;
    > .dropdown-toggle .fa,
    > a.btn{
        .mixin-color-open-hover;
    }
}

// Nav Button Group Dropdown Menu
.navbar .megamenu > .dropdown .dropdown-menu li a{
    &:hover, &:focus{
        .mixin-background-foreground-dropdown-menu-hover;
    }
}
```

Web Accessibility

Overview

Learn about the Sugar Accessibility Plugin for Sidecar.

Introduction

Making your application accessible -- per the standards defined by the W3C's WAI specifications -- is a hard thing to do and even harder to maintain. The goal of the **Sugar Accessibility Plugin for Sidecar** is to automatically apply rules to your rendered HTML, so that you don't have to be concerned with all of the intricacies of accessibility.

With respect to programmatically applying accessibility rules, you can generally assume that the rules fall into one of three categories:

1. Rules that are dependent on the context of the element's use and cannot be applied programmatically because the context is never clear.
2. Rules that can be applied programmatically, but only when the context is clear.
3. Rules that can always be applied programmatically.

We plan to continue to develop this plugin to address more and more accessibility concerns in an abstract way, with the intention of completely covering the latter two cases. In the meantime, the plugin handles two very specific cases: and (2) One from the third category.

How It Works

The plugin listens to the render event for all View.Component objects that are created. Anytime a component is rendered, the plugin runs its own plugins (hereinafter referred to as "helpers") on the component. Each of these helpers is responsible for determining if any modifications are necessary in order for the component's HTML to meet accessibility standards and then carrying out those changes in the DOM. This behavior is done automatically as a part of the sidecar framework, so you do not need to do anything to start using it.

Sometimes, a component that you write will modify the HTML after it has been rendered. The plugin has no means of becoming aware of these changes to the HTML and you will have to tell it to look for rules to apply. One example is found inView.Fields.Base.ActionmenuField.

When the user selects all records in a list view, an alert is flashed indicating that all visible records are now selected. Within this alert, a link can be clicked to select all records, even those that are not currently visible. A new onclick event listener is registered for this link. And because this link is added to the DOM after the

component is rendered, the author of the component must make sure that the new HTML meets accessibility requirements. Here is how that is done:

```
var $el = $('a#select-all');
$el.on('click', function() {...});
app.accessiblity.run($el, 'click');
```

This will only run the click helper. If you want to run all helpers, then call app.accessiblity.run($el) (without the second parameter). But be aware that some helpers only support View.Component objects, while others support eitherView.Component objects or jQuery DOM elements. So running all helpers on a jQuery DOM element may fail, or at least fail to apply some accessibility rules as expected.

When the logger is configured for debug mode, messages are logged indicating which helpers are being run and which helpers could not be run when intended.

Plugins

A plugin (or helper) is a module that applies an accessibility rule to a View.Component (or jQuery DOM element). At runtime, these helpers can be found in app.accessibility.helpers and implement a run method. The run method takes aView.Component (or jQuery DOM element) and then checks its HTML to determine if anything needs to be done in order to make the HTML compliant with accessibility standards related to the rule or task with which the helper is concerned. If any changes are necessary, the helper then modifies the HTML to comply.

Click

The click helper is responsible for making an element compliant with accessibility standards when click events are bound to said element. Since this helper only deals with onclick events, it only inspects elements within the component that include onclick event listeners.

If the tag name cannot be determined for an element being inspected, then there is no way of knowing whether or not the element is accessible. Thus, the element is assumed to be compliant.

Inherently focusable elements are those elements that require no intervention. These elements include:

- button

- input
- select
- textarea

Conditionally focusable elements are those elements that require intervention under certain circumstances. In the case of<a> and <area> tags, these elements are compliant as long as they contain an href attribute. These elements include:

- a
- area

All other elements are not inherently focusable and require a tabindex attribute of -1 if a tabindex attribute does not already exist. This helper adds tabindex="-1" to any elements within the component that are not compliant.

When the logger is configured for debug mode, messages are logged...

1. In the event that no onclick events were found within the component. Thus, no action is taken.
2. In the event that an element being inspected has no tag name.
3. To report the type of element being made compliant.
4. To report the type of element that is already found to be compliant.

Label

The label helper adds an aria-label to the form element found within the component. This helper only inspects elements that can be found via the component's fieldTag selector and is consdered a "best effort" approach.

> This helper will only work on View.Field components since it is extremely unlikely for the fieldTag selector forView.Layout or View.View components to match form elements.

A form element is considered to be compliant if the aria-label attribute is already present or if its tag is not one that requires a label. These elements include:

- button
- input
- select
- textarea

This helper adds aria-label="{label}" to the element that needs to be made compliant. View.Field.label is the label that is assigned to the attribute. The

component must be a View.Component. Plain jQuery DOM elements are not sufficient since they do not include alabel property.

API

SUGAR.accessibility.init()

Initializes the accessibility module to execute all accessibility helpers on components as they are rendered. This is called by the application during bootstrapping.

SUGAR.accessibility.run()

Loads the accessibility helpers that are to be run and executes them on the component.

Arguments

Name	Type	Required	Description
component	View.Component/jQuery	true	The element to test for accessibility compliance.
helper	String/Array	false	One or more names of specified helpers to run. All registered helpers will be run if undefined.

Returns

Void

SUGAR.accessibility.whichHelpers()

Get the helpers registered on a specific element.

Name	Type	Required	Description
helper	String/Array	true	One or more names of specified helpers to run.

Returns

Array - The accessibility helpers that were requested. Filters out any named

helpers that are not registered. All registered helpers are returned if no helper names are provided as a parameter.

SUGAR.accessibility.getElementTag()

Generates a human-readable string for identifying an element. For example, a[name="link"][class="btn btn-link"][href="http://www.sugarcrm.com/"]. Primarily used for logging purposes, this method is useful for debugging.

Arguments

Name	Type	Required	Description
$el	jQuery	true	The element for which the tag should be generated.

Returns

String - A string representing an element's tag, with all attributes. The element's selector, if one exists, is returned when a representation cannot be reasonably generated.

Validation Constraints

Overview

This article will cover how to add validation constraints to your custom code.

Constraints

Validation constraints allow developers to validate user input

Symfony Validation Constraints

The Symfony's Validation library, located in ./vendor/symfony/validator/, contains many open source constraints. You can find the full list of constraints documented in Symphony's Validation Constraints. They are listed below for your reference.

Basic Constraints	Collection Constraints

- NotBlank
- Blank
- NotNull
- IsNull
- IsTrue
- IsFalse
- Type

String Constraints

- Email
- Length
- Url
- Regex
- Ip
- Uuid

Number Constraints

- Range

Comparison Constraints

- EqualTo
- NotEqualTo
- IdenticalTo
- NotIdenticalTo
- LessThan
- LessThanOrEqual
- GreaterThan
- GreaterThanOrEqual

Date Constraints

- Choice
- Collection
- Count
- UniqueEntity
- Language
- Locale
- Country

File Constraints

- File
- Image

Financial and other Number Constraints

- Bic
- CardScheme
- Currency
- Luhn
- Iban
- Isbn
- Issn

Other Constraints

- Callback
- Expression
- All
- UserPassword
- Valid

• Date • DateTime • Time	

Sugar Constraints

Sugar contains its own set of validation constraints. These constraints extend from Symfony's validation constraints and are located in ./src/Security/Validator/Constraints of your Sugar installation.

• Bean/ModuleName • Bean/ModuleNameValidator • ComponentName • ComponentNameValidator • Delimited • DelimitedValidator • DropdownList • DropdownListValidator • File • FileValidator • Guid • GuidValidator • InputParameters • InputParametersValidator • JSON	• JSONValidator • Language • LanguageValidator • LegacyCleanString • LegacyCleanStringValidator • Mvc/ModuleName • Mvc/ModuleNameValidator • PhpSerialized • PhpSerializedValidator • Sql/OrderBy • Sql/OrderByValidator • Sql/OrderDirection • Sql/OrderDirectionValidator • SugarLogic/FunctionName • SugarLogic/FunctionNameValidator

Custom Constraints

If you find that the existing constraints do not meet your needs, you can also create your own. These constraints exist under ./custom/src/Security/Validator/Constraints/. The following section will outline the details. The example below will demonstrate how to create a phone number validation constraint.

The constraint file will contain your validations error message.

./custom/src/Security/Validator/Constraints/Phone.php

```
<?php
```

```
namespace Sugarcrm\Sugarcrm\custom\Security\Validator\Constraints;

use Symfony\Component\Validator\Constraint;

/**
 *
 * @see PhoneValidator
 *
 */
class Phone extends Constraint
{
    public $message = 'Phone number violation: %msg%';
}
```

The validator file will contain the logic to determine whether the value meets the requirements.

./custom/src/Security/Validator/Constraints/PhoneValidator.php

```
<?php

namespace Sugarcrm\Sugarcrm\custom\Security\Validator\Constraints;

use Symfony\Component\Validator\Constraint;
use Symfony\Component\Validator\ConstraintValidator;
use Symfony\Component\Validator\Exception\UnexpectedTypeException;

/**
 *
 * Phone validator
 *
 */
class PhoneValidator extends ConstraintValidator
{
    /*
     * Phone Pattern Examples;
     * +1 12 3456789
     * +1.12.3456789
     */
    const PHONE_PATTERN = '/^([+]?\d+(?:[ \.]\d+)*)$/';

    /**
     * {@inheritdoc}
```

```
     */
    public function validate($value, Constraint $constraint)
    {
        if (!$constraint instanceof Phone) {
            throw new UnexpectedTypeException($constraint, __NAMESPACE
__ . '\Phone');
        }

        if (null === $value || '' === $value) {
            return;
        }

        if (!is_scalar($value) && !(is_object($value) && method_exists
($value, '__toString'))) {
            throw new UnexpectedTypeException($value, 'string');
        }

        $value = (string) $value;

        // check for allowed characters
        if (!preg_match(self::PHONE_PATTERN, $value)) {
            $this->context->buildViolation($constraint->message)
                ->setParameter('%msg%', 'invalid format')
                ->setInvalidValue($value)
                ->addViolation();
            return;
        }
    }
}
```

Once the files are in place, navigate to Admin > Repairs and run a Quick Repair and Rebuild. Once completed, your constraint will be ready for use.

Using Constraints in API End Points

In this section, we will create a custom endpoint and trigger the custom phone constraint we created above. The code below will create a REST API endpoint path of /phoneCheck:

./custom/clients/base/api/MyEndpointsApi.php

```
<?php

if (!defined('sugarEntry') || !sugarEntry) {
```

```
    die('Not A Valid Entry Point');
}

use Sugarcrm\Sugarcrm\Security\Validator\ConstraintBuilder;
use Sugarcrm\Sugarcrm\Security\Validator\Validator;

class MyEndpointsApi extends SugarApi
{
    public function registerApiRest()
    {
        return array(
            //POST
            'MyEndpointsApi' => array(
                //request type
                'reqType' => 'POST',
                //endpoint path
                'path' => array('phoneCheck'),
                //endpoint variables
                'pathVars' => array(''),
                //method to call
                'method' => 'phoneCheck',

                //minimum api version
                'minVersion' => 10,
                //short help string to be displayed in the help docume
ntation
                'shortHelp' => 'An example of a POST endpoint to valid
ate phone numbers',
                //long help to be displayed in the help documentation
                'longHelp' => 'custom/clients/base/api/help/MyEndPoint
_phoneCheck_help.html',
            ),
        );
    }

    /**
     * Method to be used for my MyEndpointsApi/phoneCheck endpoint
     */
    public function phoneCheck($api, $args)
    {
        $validator = Validator::getService();
        /**
         * Validating Phone Number
         */
        $phoneContraintBuilder = new ConstraintBuilder();
        $phoneConstraints = $phoneContraintBuilder->build(
```

```
            array(
                'Assert\Phone',
            )
        );
        $errors = $validator->validate($args['phone'], $phoneConstrain
ts);
        if (count($errors) > 0) {
            /*
             * Uses a __toString method on the $errors variable which
is a
             * ConstraintViolationList object. This gives us a nice st
ring
             * for debugging.
             */
            $errorsString = (string) $errors;

            // include/api/SugarApiException.php
            throw new SugarApiExceptionInvalidParameter($errorsString)
;
        }

        //custom logic
        return $args;
    }
}
```

After creating the endpoint, navigate to Admin > Repairs and run a Quick Repair and Rebuild. The API will then be ready to use.

Valid Payload - POST to /phoneCheck

The example below demonstrates a valid phone number post the /phoneCheck endpoint.

```
{
    phone: "+1.23.456.789"
}
```

Result

```
{
```

```
    phone: "+1.23.456.789"
}
```

Invalid Payload - POST to /phoneCheck

The example below demonstrates an invalid phone number post the /phoneCheck endpoint.

```
{
    phone: "Invalid+1.23.456.789"
}
```

Result

```
{
    "error":"invalid_parameter",
    "error_message":"Invalid+1.23.456.789:\n    Phone number violation
: invalid format\n"
}
```

CLI

Overview

Sugar on-premise deployments include a command line interface tool built using the Symfony Console Framework. Sugar's CLI is intended to be an administrator or developer level power tool to execute PHP code in the context of Sugar's code base. These commands are version specific and can be executed on a preinstalled Sugar instance or on an installed Sugar instance. Sugar's CLI is not intended to be used as a tool to interact remotely with an instance nor is it designed to interact with multiple instances.

Commands

Sugar Commands are an implementation of Console Commands. They extend the base console classes to execute Sugar specific actions. Each instance of Sugar is shipped with a list of predefined commands. The current list of commands is shown below.

Command	Description

help	Displays help for a command
list	Lists commands
aclcache:dump	Dumps ACL cache contents into cache/acldumps folder
elastic:explain	Execute global search explain queries for debugging purposes. As this will generate a lot of output in JSON format, it is recommended to redirect the output to a file for later processing
elastic:indices	Show Elasticsearch index statistics
elastic:queue	Show Elasticsearch queue statistics
elastic:queue_cleanup	Cleanup records from Elasticsearch queue
elastic:refresh_enable	Enable refresh on all indices
elastic:refresh_status	Show Elasticsearch index refresh status
elastic:refresh_trigger	Perform a manual refresh on all indices
elastic:replicas_enable	Enable replicas on all indices
elastic:replicas_status	Show Elasticsearch index refresh status
elastic:routing	Show Elasticsearch index routing
idm-mode:manage	enable, disable IDM mode, or move config data to DB
password:config	Show password hash configuration
password:reset	Reset user password for local user authentication
password:weak	Show users having weak or non-compliant password hashes
search:fields	Show search engine enabled fields
search:module	Enable/disable given module for search
search:reindex	Schedule SearchEngine reindex
search:silent_reindex	Create mappings and index the data
search:silent_reindex_mp	Create mappings and index the data using multi-processing
search:status	Show search engine availability and enabled modules
team-security:rebuild	Rebuild denormalized team security data
team-security:status	Display the status of the denormalized

	data
teamset:backup	Backs up the team_sets related tables
teamset:prune	Prune all team set tables of unused team sets. Original tables will be backed up automatically. DO NOT USE while users are logged into the system
teamset:restore_from_backup	Restores all team set tables from backups. DO NOT USE while users are logged into the system
teamset:scan	Scan all module tables for unused team sets and report the number found
teamset:sql	Print the sql query used to search for unused teamsets

Note : For advanced users, a bash autocompletion script for CLI is located at ./src/Console/Resources/sugarcrm-bash-completion for inclusion in ~/.bashrc. More details on using the script can be found in the file's header comments.

Usage

The command executable, located at ./bin/sugarcrm, is executed from the root of the Sugar instance. The command signature is shown below:

```
  php bin/sugarcrm <command> [options] [arguments]
```

Help

The command console has built-in help documentation. If you pass in a command that isn't found, you will be shown the default help documentation.

```
  SugarCRM Console version <version>

Usage:
  command [options] [arguments]

Options:
  -h, --help            Display this help message
  -q, --quiet           Do not output any message
  -V, --version         Display this application version
      --ansi            Force ANSI output
      --no-ansi         Disable ANSI output
  -n, --no-interaction  Do not ask any interactive question
```

```
      --profile         Display timing and memory usage information
  -v|vv|vvv, --verbose  Increase the verbosity of messages: 1 for norm
al output, 2 for more verbose output and 3 for debug

Available commands:
  help              Displays help for a command
  list              Lists commands
 aclcache
  aclcache:dump               Dumps ACL cache contents into cache/acl
dumps folder.
 elastic
  elastic:explain             Execute global search explain queries f
or debugging purposes. As this will generate a lot of output in JSON f
ormat, it is recommended to redirect the output to a file for later pr
ocessing.
  elastic:indices             Show Elasticsearch index statistics
  elastic:queue               Show Elasticsearch queue statistics
  elastic:queue_cleanup       Cleanup records from Elasticsearch queu
e.
  elastic:refresh_enable      Enable refresh on all indices
  elastic:refresh_status      Show Elasticsearch index refresh status
  elastic:refresh_trigger     Perform a manual refresh on all indices
  elastic:replicas_enable     Enable replicas on all indices
  elastic:replicas_status     Show Elasticsearch index refresh status
  elastic:routing             Show Elasticsearch index routing
 idm-mode
  idm-mode:manage             enable, disable IDM mode, or move confi
g data to DB
 password
  password:config             Show password hash configuration
  password:reset              Reset user password for local user auth
entication.
  password:weak               Show users having weak or non-
compliant password hashes
 search
  search:fields               Show search engine enabled fields
  search:module               Enable/disable given module for search
  search:reindex              Create mappings and schedule a reindex
  search:silent_reindex       Create mappings and index the data
  search:silent_reindex_mp    Create mappings and index the data usin
g multi-processing
  search:status               Show search engine availability and ena
bled modules
 team-security
  team-security:rebuild       Rebuild denormalized team security data
  team-
```

```
security:status         Display the status of the denormalized data
 teamset
  teamset:backup               Backs up the team_sets related tables.
  teamset:prune                Prune all team set tables of unused tea
m sets. Original tables will be backed up automatically. DO NOT USE wh
ile users are logged into the system!
  teamset:restore_from_backup  Restores all team set tables from backu
ps. DO NOT USE while users are logged into the system!
  teamset:scan                 Scan all module tables for unused team
sets and report the number found.
  teamset:sql                  Print the sql query used to search for
unused teamsets
```

Additional help documentation is also available for individual commands. Some examples of accessing the help are shown below.

Passing the word "help" before a command name:

```
  php bin/sugarcrm help <command>
```

Passing the "-h" option:

```
  php bin/sugarcrm <command> -h
```

An example of the list help documentation is shown below.

```
  Usage:
  list [options] [--] []

Arguments:
  namespace            The namespace name

Options:
      --xml            To output list as XML
      --raw            To output raw command list
      --format=FORMAT  The output format (txt, xml, json, or md) [defa
ult: "txt"]

Help:
 The list command lists all commands:
```

```
   php bin/sugarcrm list

 You can also display the commands for a specific namespace:

   php bin/sugarcrm list test

 You can also output the information in other formats by using the --f
ormat option:

   php bin/sugarcrm list --format=xml

 It's also possible to get raw list of commands (useful for embedding
command runner):

   php bin/sugarcrm list --raw
```

Custom Commands

Sugar allows developers the ability to create custom commands. These commands are registered using the Extension Framework. Each custom command will extend the stock Command class and implement a specific mode interface. The file system location of the command code can exist anywhere in the instance's file system including a separate composer loaded repository.

Best Practices

The following are some common best practices developers should keep in mind when adding new commands :

- Do not extend from existing commands
- Do not duplicate an already existing command
- Pick the correct mode for the command
- Limit the logic in the command
- Do not put reusable components in the command itself

Each command requires specific sections of code in order to properly create the command. These requirements are listed in the sections below.

Command Namespace

It is recommended to name any custom commands using a proper "command namespace". These namespaces are different than PHP namespaces and reduce

the chances of collisions occurring between vendors. An example of this would be to prefix any commands with a namespace format of vendor:category:command E.g. MyDevShop:repairs:fixMyModule

Note : The CLI framework does not allow overriding of existing commands.

PHP Namespace

The header of the PHP command file will need to define the following namespace:

```
namespace Sugarcrm\Sugarcrm\custom\Console\Command;
```

In addition to the namespace, the following use commands are required for different operations :

Name	Description	Example
Command	Every command will extend the Command class	use Symfony\Component\Console\Command\Command;
Mode	Whether the command is an Instance or Standalone Mode command	use Sugarcrm\Sugarcrm\Console\CommandRegistry\Mode\InstanceModeInterface; use Sugarcrm\Sugarcrm\Console\CommandRegistry\Mode\StandaloneModeInterface;
Input	Accepts Input from the command	use Symfony\Component\Console\Input\InputInterface;
Output	Sends Output from the command	use Symfony\Component\Console\Output\OutputInterface;

Mode

When creating Sugar commands, developers will need to specify the mode or set of modes for the command. These modes help prepare the appropriate resources for execution of the command.

Mode	Description

Instance Mode	Commands that require an installed Sugar instance.
Standalone Mode	Commands that do not require an installed Sugar instance.

Note : Multiple modes can be selected.

Class Definition

The custom command class definition should extend the stock command class as well as implement a mode's interface.

```
  // Instance Mode Class Definition
class <classname> extends Command implements InstanceModeInterface
{
    //protected methods
}

// Standalone Mode Class Definition
class <classname> extends Command implements StandaloneModeInterface
{
    //protected methods
}

// Implementing both Modes
class <classname> extends Command implements InstanceModeInterface, St
andaloneModeInterface
{
    //protected methods
}
```

Methods

Each command class implements two protected methods :

Method	Description
configure()	Runs on the creation of the command. Useful to set the name, description, and help on the command.
execute(InputInterface $input, OutputInterface $output)	The code to run for executing the command. Accepts an input and output parameter.

An example of implementing the protected methods is shown below:

```
  protected function configure()
{
    $this->setName('sugardev:helloworld')
         ->setDescription('Hello World')
         ->setHelp('This command accepts no paramters and returns "Hel
lo World".')
    ;
}

protected function execute(InputInterface $input, OutputInterface $out
put)
{
    $output->writeln("Hello world -> " . $this->getApplication()->getM
ode());
}
```

Registering Command

After creating the custom command file, register it with the Extension Framework . This file will be located in ./custom/Extension/application/Ext/Console/. An example of registering a command is below:

```
  <?php

Sugarcrm\Sugarcrm\Console\CommandRegistry\CommandRegistry::getInstance
()
   ->addCommand(new Sugarcrm\Sugarcrm\custom\Console\Command\<Command
Class Name>());
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The custom command will now be available.

Example

The following sections demonstrate how to create a "Hello World" example. This command does not alter the Sugar system and will only output display text. First, create the command class under the appropriate namespace.

./custom/src/Console/Command/HelloWorldCommand.php

```
  <?php

namespace Sugarcrm\Sugarcrm\custom\Console\Command;

use Sugarcrm\Sugarcrm\Console\CommandRegistry\Mode\InstanceModeInterfa
ce;
use Symfony\Component\Console\Command\Command;
use Symfony\Component\Console\Input\InputInterface;
use Symfony\Component\Console\Output\OutputInterface;

/**
 *
 * Hello World Example
 *
 */
class HelloWorldCommand extends Command implements InstanceModeInterfa
ce
{

    protected function configure()
    {
        $this
            ->setName('sugardev:helloworld')
            ->setDescription('Hello World')
            ->setHelp('This command accepts no paramters and returns "
Hello World".')
        ;
    }

    protected function execute(InputInterface $input, OutputInterface
$output)
    {
        $output->writeln("Hello world -> " . $this->getApplication()->
getMode());
    }
}
```

Next, register the new command:

./custom/Extension/application/Ext/Console/RegisterHelloWorldCommand.php

```
  <?php

// Register HelloWorldCommand
```

```
Sugarcrm\Sugarcrm\Console\CommandRegistry\CommandRegistry::getInstance
()
    ->addCommand(new Sugarcrm\Sugarcrm\custom\Console\Command\HelloWor
ldCommand());
```

Navigate to Admin > Repair > Quick Repair and Rebuild. The new command will be executable from the root of the Sugar instance. It can be executed by running the following command:

```
  php bin/sugarcrm sugardev:helloworld
```

Result:

```
  Hello world -> InstanceMode
```

Help text can be displayed by executing the following command:

```
  php bin/sugarcrm help sugardev:helloworld
```

Result:

```
  Usage:
  sugardev:helloworld

Options:
  -h, --help            Display this help message
  -q, --quiet           Do not output any message
  -V, --version         Display this application version
      --ansi            Force ANSI output
      --no-ansi         Disable ANSI output
  -n, --no-interaction  Do not ask any interactive question
      --profile         Display timing and memory usage information
  -v|vv|vvv, --verbose  Increase the verbosity of messages: 1 for norm
al output, 2 for more verbose output and 3 for debug

Help:
 This command accepts no paramters and returns "Hello World".
```

Download the module loadable example package here.

Performance Tuning

The following sections detail ways to enhance the performance of your Sugar instance.

Sugar Performance

Overview

As your company uses Sugar over time, the size of your database will naturally grow and, without the proper maintenance, performance will inevitably begin to degrade. The purpose of this article is to review some of the most common recommendations for increasing the performance in Sugar to help increase the system's efficiency for your users.

Note: This guide is intended for on-site installations. Customers hosted on Sugar's cloud service should file a support case for any performance issues.

General Settings in Sugar

The following recommendations can be modified in the Sugar admin interface

- **Do Not Set Listview and Subpanel Items Per Page to Excessive Settings.** Under Admin > System Settings, there are two settings 'Listview items per page' and 'Subpanel items per page'. The defaults for these settings are 20 and 10 respectively. When increasing these values, it should be expected that general system wide performance will be impacted. We generally recommend keeping listview settings to 100 or less and subpanel settings to be set to 10 or less to keep system performance optimal.
- **Make sure 'Developer Mode' is disabled under Admin > System Settings.** This setting should never be enabled in a production environment as it causes cached files to be rebuilt on every page load.
- **Set the 'Log Level' to 'Fatal' and 'Maximum log size' to '10M' under Admin > System Settings.** The log level should only be set to more verbose levels when troubleshooting the application as it will cause a performance degradation as user activity increases.
- **Ensure the scheduled job, 'Prune Database on the 1st of Month', is set to 'Active'.** This will go through your database and delete any records that have been deleted by your users. Sugar only soft deletes records when a user deletes a record and over time, this will cause performance

degradation if these records are not removed from the database.
- **Make sure 'Tracker Performance' and 'Tracker Queries' are disabled under Admin > Tracker.** These settings are intended to help diagnose performance issues and should never be left enabled in a production environment.
- **Ensure large scheduler jobs are running at slower intervals under Admin > Scheduler.** Jobs such as 'Check Inbound Mailboxes' can decrease overall performance if they are running every minute and polling a lot of data. It is important to set these jobs to every 5 or 10 minutes to help offset the performance impacts for your users.

Sugar Performance Settings

General Settings

Disable client IP verification : Eliminates the system checking to see if the user is accessing Sugar from the IP address of their last page load.

```
$sugar_config['verify_client_ip'] = false;
```

BWC Modules

For modules running in Backward Compatibility mode, the following settings can be used to speed up performance:

Drop the absolute totals from listviews : Eliminates performing expensive count queries on the database when populating listviews and subpanels.

```
$sugar_config['disable_count_query'] = true;
```

Disable automatic searches on listviews : Forces a user to perform a search when they access a listview rather than loading the results from their last search.

```
$sugar_config['save_query'] = 'populate_only';
```

Hide all subpanels : Increases performance by collapsing all subpanels when accessing a detailview every time and not querying for data until a user explicitly expands a subpanel

```
$sugar_config['hide_subpanels'] = true;
```

Hide subpanels per session : Increases performance by collapsing all subpanels when accessing a detailview when the user logs in but any subpanels expanded during the user's session will remain expanded until the user logs out

```
$sugar_config['hide_subpanels_on_login'] = true;
```

General Environment Checks

Depending on your environment and version of Sugar, there may be additional changes your system administrator can make to improve performance.

- **If your instance is running Sugar 6.3.x or lower, we highly recommend upgrading to 6.4.x or 6.5.x.** Beginning in 6.4.x, we made a number of query improvement to address overall application performance. With 6.5.x, we drastically improved the UI to improve the speed of page loads.
- **If your instance of Sugar is version 6.3.x or higher with a MySQL database, we highly recommend upgrading the MySQL 5.5.** MySQL 5.5 offers performance improvements in a number of areas over 5.1 such as subselects in queries.
- **If you are using MySQL as your database, we strongly recommend using InnoDB.** InnoDB is tested to be better performing than MyISAM and should be the default configuration for using MySQL with Sugar.
- **If you are running PHP 5.2.x, we strongly recommend upgrading to a supported version of PHP 5.3.** Our current list of supported PHP versions can be found on our Supported Platforms page.
- **If you are using a single server setup (web server and database on the same server), we have the following recommendations.**
    - The server should have a minimum of 8 GB of RAM and roughly follow the 60/40 rule (60% for database / 40% for web server). On a 8 GB server, this would mean 4.8 GB for database and 3.2 GB for web server.
    - Make sure the following parameters are set for MySQL (assumption is that the engine is InnoDB)
        - innodb_buffer_pool_size = 4294967296 (4 GB in size)
        - innodb_log_buffer_size = anywhere from 10485760 (10 MB - Minimal writes) to 104857600 (100 MB - Lots of writes)
- **If you are using IE 8 or lower, we recommend upgrading to IE 9 or using Google Chrome.** Earlier versions of IE 8 exhibit poor performance with our application and we recommend updating your browser to IE 9 or changing to Chrome.

PHP Caching

Whether your instance of Sugar is deployed on a Linux or Windows server, you should utilize opcode caching to ensure optimal performance. For Linux servers, APC is the recommended opcode cache for PHP with the following guidelines and settings:

- Use the latest stable version.
- apc.shm size should be close to your program size. For Sugar, that's at least 150 MB (default for apc.shm is 32 MB). When in doubt, more is always better.
- apc.stat_ctime should be enabled. This will ensure file changes are noticed. You should note that this may increase the stat() activity to your NFS.
- apc.file_update_protection should be enabled. This helps the system when trying to add multiple files to the cache at the same time.
- If your installation of Sugar is located on a network filesystem such as NFS or CIFS, make sure apc.stat is enabled.
- apc.ttl should be set to 0. This parameter disables garbage collection and can cause fragmentation. Earlier APC releases had locking issues that made caches with many entries take forever to be garbage collected.
- apc.shm_segments should be set to the default of 1. If you think you really need multiple shm_segments, you must also read the documentation on apc.mmap_file_mask as well and understand and set that value accordingly. If you don't understand apc.mmap_file_mask, you should leave apc.shm_segments at the default value.
- APC ships with an additional apc.php file that when hit with a browser, will show settings, cache information, and fragmentation. If you suspect APC problems, this is a great tool to start checking things out.

PHP Profiling

Overview

As of the 6.6.2 release, Sugar introduced the ability to profile via XHProf, which is an easy-to-use, hierarchical profiler for PHP. This allows developers to better manage and understand customer performance issues introduced by their customizations. This tool enables quick and accurate identification of the sources of performance sinks within the code by generating profiling logs. Profiling gives you the ability to see the call stack for the entire page load with timing details around function and method calls as well as statistics on call frequency.

Assuming XHProf is installed and enabled in your PHP configuration (which you can learn how to do in the PHP Manual), you can enable profiling in Sugar by adding the following parameters to the ./config_override.php file:

```
$sugar_config['xhprof_config']['enable'] = true;
$sugar_config['xhprof_config']['log_to'] = '{instance server path}/cac
he/xhprof';
// x where x is a number and 1/x requests are profiled. So to sample a
ll requests set it to 1
$sugar_config['xhprof_config']['sample_rate'] = 1;
// array of function names to ignore from the profile (pass into xhpro
f_enable)
$sugar_config['xhprof_config']['ignored_functions'] = array();
// flags for xhprof
$sugar_config['xhprof_config']['flags'] = XHPROF_FLAGS_CPU + XHPROF_FL
AGS_MEMORY;
```

Please note that with the above 'log_to' parameter, you would need to create the ./cache/xhprof/ directory in your instance directory with proper permissions and ownership for the Apache user. You can also opt to leave the xhprof_config.log_to parameter empty and set the logging path via the xhprof.output_dir parameter in the php.ini file.

Once the above parameters are set, XHProf profiling will log all output to the indicated directory and allow you to research any performance related issues encountered in the process of developing and maintaining the application.

Integrating Sugar With New Relic APM for Performance Management

Overview

Sugar® 7 includes support for New Relic APM™, a third-party Application Performance Management (APM) tool that can facilitate deep insight into your Sugar instance in order to troubleshoot sluggish response times. This article explains how to set up and use New Relic in conjunction with Sugar for powerful performance management capabilities.

Note: This article pertains to on-site installations of Sugar only. SugarCloud customers who are experiencing performance-related issues should contact the Sugar Support team for assistance.

Prerequisites

- To install and configure New Relic for use with your Sugar instance, you

must have Sugar hosted on-site and have access to the root directory.
- You must be a New Relic account holder. To sign up for New Relic, please visit newrelic.com to find the subscription level best suited for your needs.

Steps to Complete

New Relic can provide useful information outside of the Sugar integration, but the feedback it provides will be limited to the instance's PHP file structure, which could make troubleshooting your instance a challenge. Follow these instructions to set up and use New Relic for PHP with your Sugar instance.

Installing the New Relic for PHP Agent

First, install the New Relic for PHP agent. For the most current installation steps, please refer to the Getting Started Guide on the documentation site for New Relic for PHP.

Configuring Sugar to Work With New Relic for PHP

Enable the Sugar integration with New Relic by editing the ./config_override.php file. Add the following lines to the end of the file contents (explanation follows):

```
  $sugar_config['metrics_enabled'] = 1;
$sugar_config['metric_providers']['SugarMetric_Provider_Newrelic'] = '
include/SugarMetric/Provider/Newrelic.php';
$sugar_config['metric_settings']['SugarMetric_Provider_Newrelic']['app
licationname'] = "SugarCRM New Relic";
```

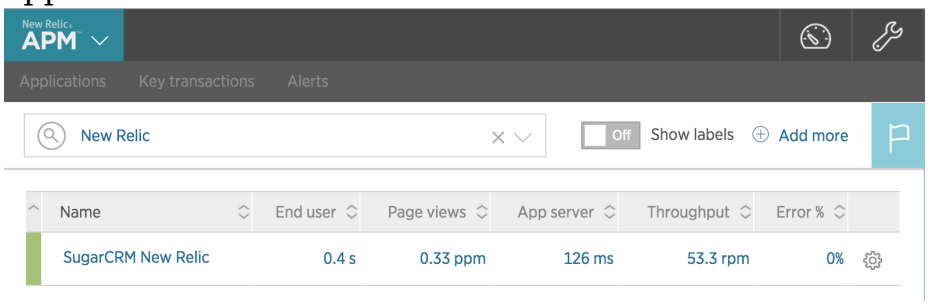
- The first line of the configuration enables metrics collection for your Sugar instance.
- The second line specifies the path where the New Relic provider files can be found.
  Note: When overwriting the New Relic provider, this path must be changed to the location where the files are located in the ./custom/ directory.
- The last line allows you to configure a custom application name so that you may make a distinction between production, staging and development environments. This name will be displayed in your New Relic application list. Simply replace the text inside the double quotes with your desired application name. In the example above, we name the application, "SugarCRM New Relic".

Using New Relic for PHP

New Relic integrated with Sugar enables you to view the exact functions that

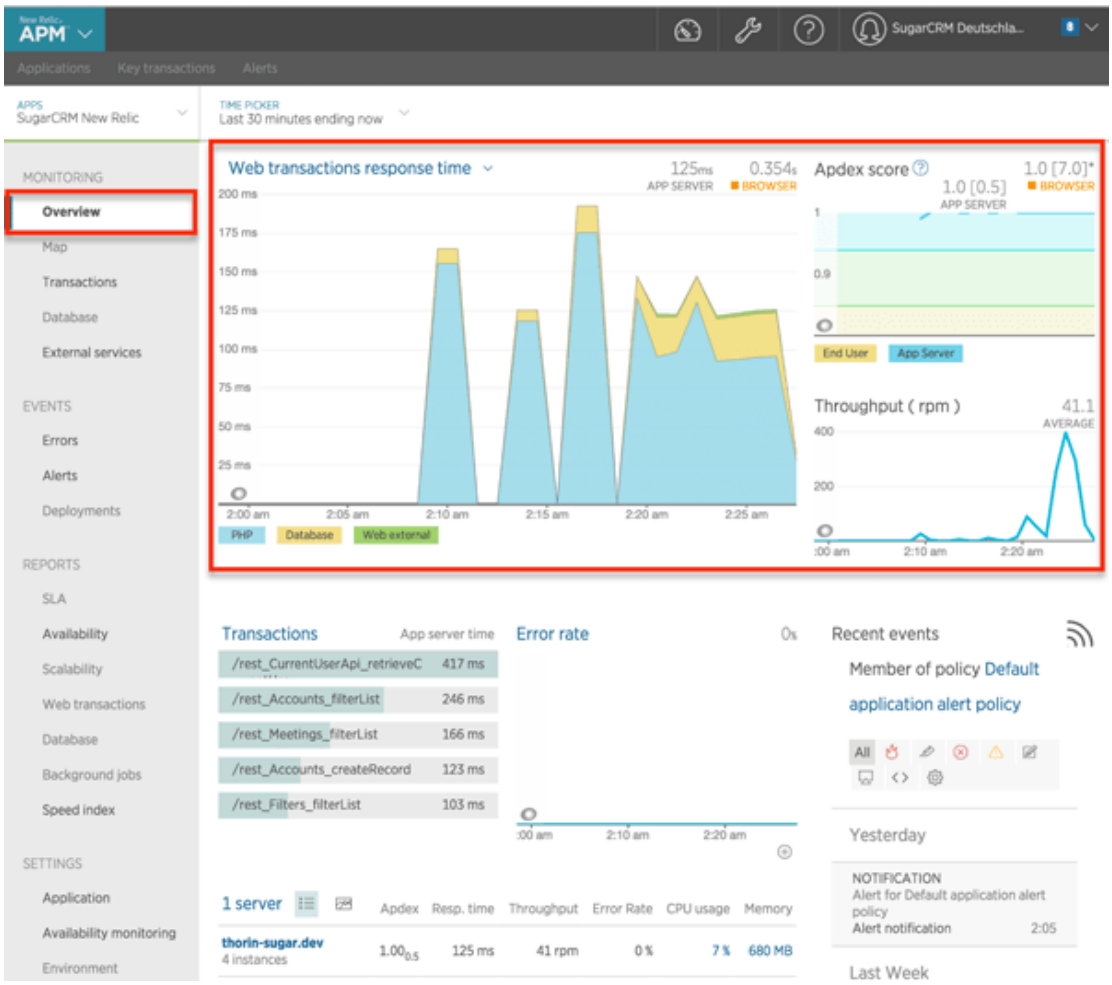
cause unusual performance behavior in your instance, such as unexpected triggering of logic hooks, database queries that should be optimized, or customizations that are responding slower than expected.

Shortly after completing the configuration steps above, a new application will appear in the New Relic APM interface's application list. Click on the appropriate application's name to view the overview dashboard.

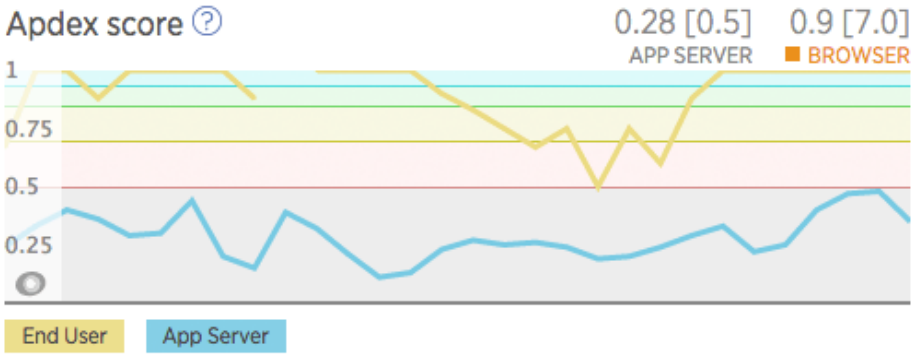


Overview Dashboard

The dashboard gives you a quick overview of server response times over a selected period. By default, it will show data collected within the last 30 minutes. To the right of this chart is the Apdex (short for application performance index) chart. Apdex provides an easy way to measure whether performance meets user expectations.



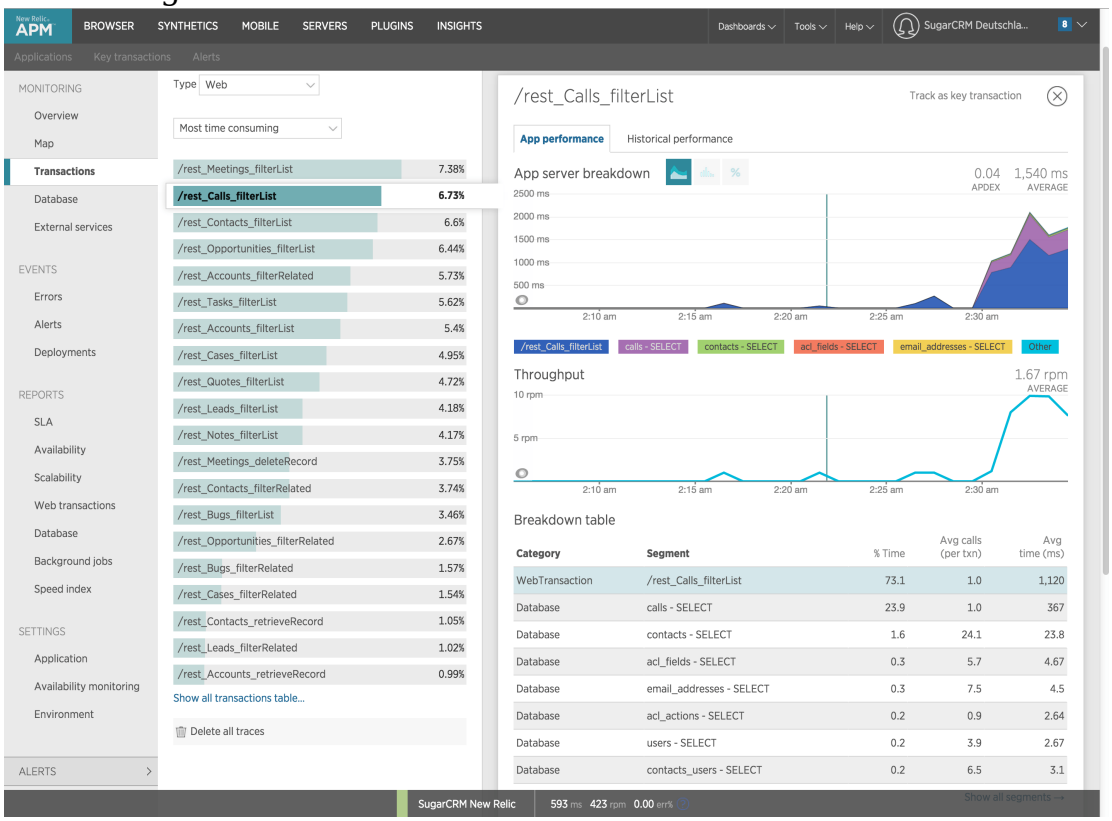
In this instance, the Apdex threshold, or T-value, is configured to 0.5 seconds. This means that an app server response time of 0.5 seconds or less is satisfactory for the users, a response time between 0.5 seconds and 2.0 seconds is tolerable, and any value higher than 2.0 seconds becomes frustrating.



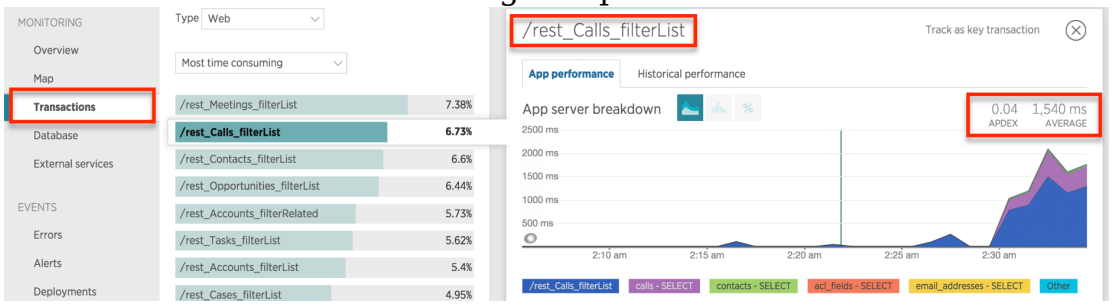
The default Apdex T-value for New Relic is 0.5 seconds but it can be configured to match your current environment and user expectations. For information on changing the Apdex T-value, please refer to the Change Your Apdex Settings article in the New Relic documentation.

Transactions

The Transactions listing is one of the most powerful tools available in New Relic. It will reveal the specific calls or actions that are taking the most time and resources from the server, and speculate as to why. Select "Transactions" in the menu on the left to see a full overview of all the calls done in sugar, sorted by the most time consuming.



In this case, rest_Calls_filterList is selected, which monitors the length of time that it takes to call the Calls module's list view. The performance data for this transaction is displayed on the right. As you can see at the top of the chart, calling the Calls listview has an average response time of 1.5 seconds.

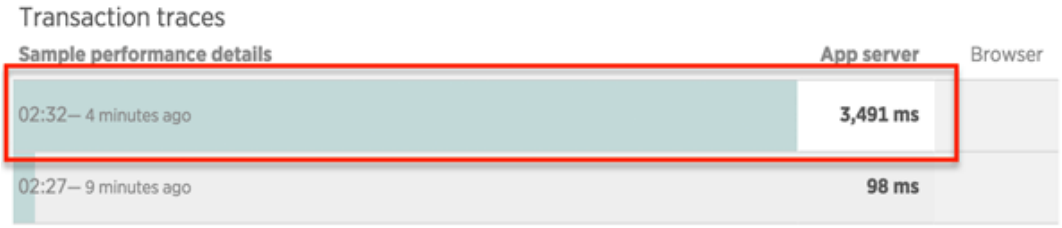


Refer to the breakdown table in the lower part of the screen to see which part of the call is taking the most time, with a representation of the performed actions on the database per segment.

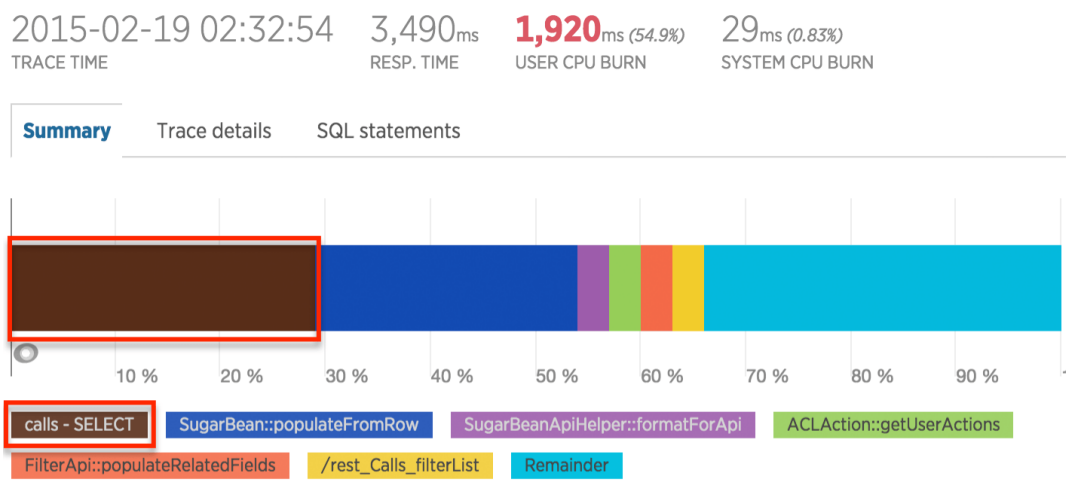
Category	Segment	% Time	(per txn)	time (ms)
WebTransaction	/rest_Calls_filterList	73.1	1.0	1,120
Database	calls - SELECT	23.9	1.0	367
Database	contacts - SELECT	1.6	24.1	23.8
Database	acl_fields - SELECT	0.3	5.7	4.67
Database	email_addresses - SELECT	0.3	7.5	4.5
Database	acl_actions - SELECT	0.2	0.9	2.64
Database	users - SELECT	0.2	3.9	2.67
Database	contacts_users - SELECT	0.2	6.5	3.1

Show all segments →

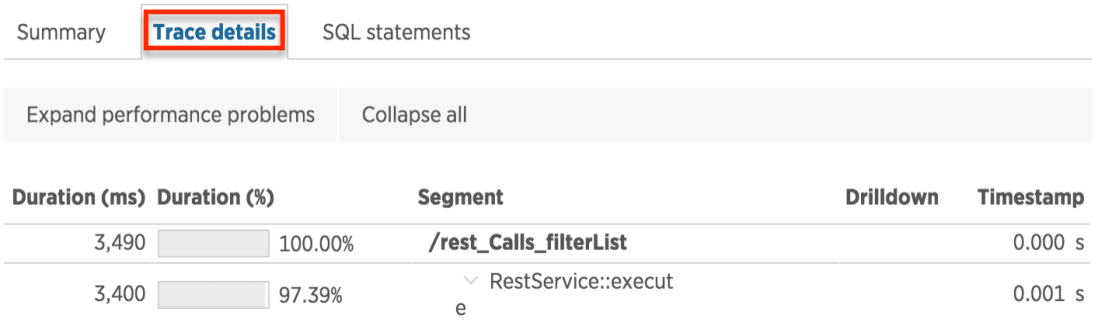
Below the breakdown table is a list of transaction traces. New Relic will automatically generate a transaction trace when a response time is in the frustrating zone of the Apdex or slower. Click on the transaction trace to learn what is having the negative impact on the performance for this activity.



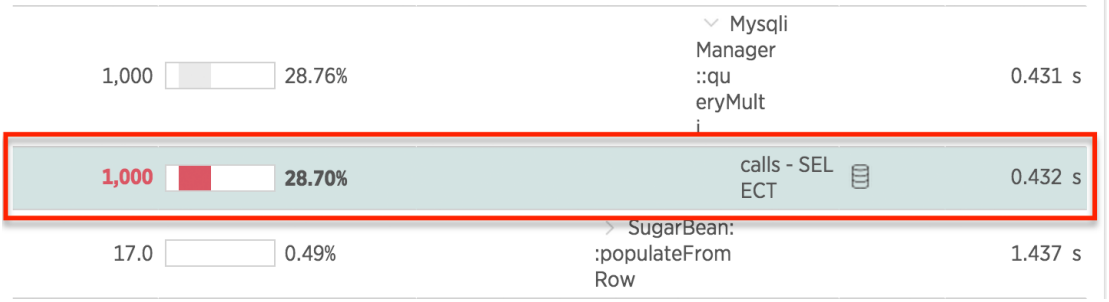
The transaction trace shows how long various components took to load. In this case, the SELECT query on the Calls table took a significant amount of time.



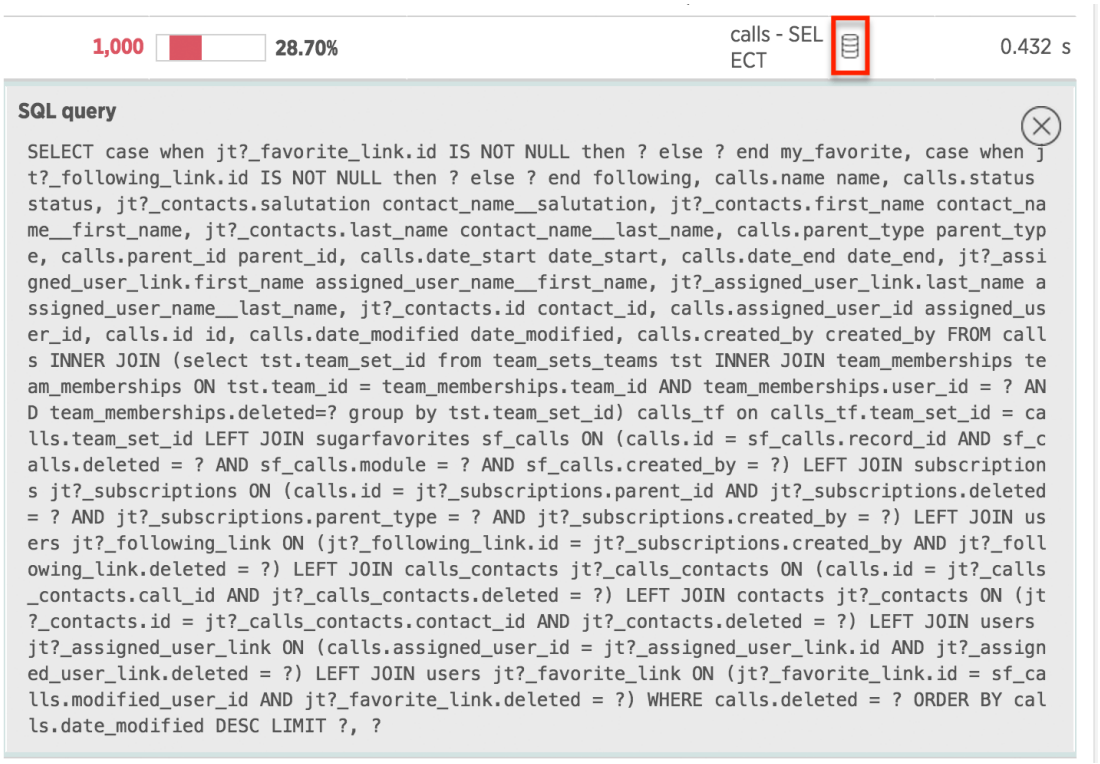
Click on the Trace Details tab above the chart to investigate further. The details page displays specific functions that are being called and how long they took to execute. By drilling down in the tree to the child functions, it is possible to find the root cause of the impaired performance.



Scroll down to find the SELECT query on calls took 1 second to load.



Click on the database icon next to the action to reveal the SQL query called by Sugar.



To view all SQL queries at once, click on the SQL Statements tab.

Summary

For critical business applications like CRM, an APM tool can you help keep your system running fast so end users stay productive and your organization sees a maximum return on investment. An integrated APM will monitor, analyze, and visualize the response times of your application to identify bottlenecks so that your team can proactively address them.

To learn more about using New Relic for PHP, please visit the New Relic documentation website.

Backward Compatibility

Overview

As of Sugar® 7, modules are built using the Sidecar framework. Any modules that still use the MVC architecture and have not yet migrated to the Sidecar framework are set in what Sugar refers to as "backward compatibility" (BWC) mode.

For the list of Studio-enabled modules in backward compatibility for your version of Sugar, please refer to the User Interface (Legacy Modules) documentation for

your version of Sugar.

URL Differences

There are some important differences between the legacy MVC and Sidecar URLs. When a module is set in backward compatibility, the URL will contain "/#bwc/" in the path and use query string parameters to identify the module and action. An example of the Sidecar BWC URL is shown below.

```
http://{site url}/#bwc/index.php?module=<module>&action=<action>
```

You can see that this differs from the standard route of:

```
http://{site url}/#<module>/<record id>
```

Studio Differences

There are some important differences between the legacy MVC modules and Sidecar modules. When a module is in backward compatibility mode, an asterisk is placed next to the modules name in Studio. Modules in backward compatibility will use the legacy MVC metadata layouts, located in ./modules/<module>/metadata/, as follows:

- Layouts
    - Edit View
    - Detail View
    - List View
- Search
    - Basic Search
    - Advanced Search

These layouts differ from Sidecar in many ways. The underlying metadata is very different and you should notice that the MVC layouts have a separation between the Edit View and Detail View whereas the Sidercar layouts make use of the Record View. The Sidecar metadata for Sugar is located in ./modules/<module>/clients/base/views/.

- Layouts
    - Record View
    - List View
- Search

- Search

Enabling Backward Compatibility

Overview

How to enable backward compatibility for a module.

Enabling Backward Compatibility

Backward Compatibility Mode is not a permanent solution for modules with legacy customizations. If you should need to temporarily get a module working due to legacy customizations, you can follow the steps below to enable the legacy MVC framework. Please note that switching stock Sugar modules from the Sidecar framework to backward compatibility mode is not supported and may result in unexpected behaviors in the application.

Enabling BWC

To enable backward compatibility, you must first create a file in ./custom/Extension/application/Ext/Include/ for the module. If the module is custom, there will already be an existing file in this folder pertaining to the module that you can edit.

./custom/Extension/application/Ext/Include/<file>.php

```
  <?php

$bwcModules[] = '<module key>';
```

Once the file is in place, you will need to navigate to Admin > Repair > Quick Repair and Rebuild. The Quick Repair can wait until you have completed the following sections for this customization.

Updating the MegaMenu Module Link

Once you have enabled backward compatibility for a module, you will then need to manually update the module's link in the MegaMenu. This will control the navigation when a user clicks the actual module name on the MegaMenu.

./custom/modules/<module>/clients/base/layouts/records/records.php

```
  <?php

$viewdefs['<module key>']['base']['layout']['records'] = array(
    'name' => 'bwc',
    'type' => 'bwc',
    'components' =>
    array(
        array(
            'view' => 'bwc',
        ),
    ),
);
```

Once the file is in place, you will need to navigate to Admin > Repair > Quick Repair and Rebuild. The Quick Repair can wait until you have completed the following section for this customization.

Updating the MegaMenu Sub-Navigation Links

Once you have updated the MegaMenu module link, you will then need to manually update the module's MegaMenu action links.

The module's deployed sub-navigation links for a module will be similar to the file shown below:

./modules/<module>/clients/base/menus/header/header.php

```
  <?php

$moduleName = '<module key>';

$viewdefs[$moduleName]['base']['menu']['header'] = array(
    array(
        'route' => "#$moduleName/create",
        'label' => 'LNK_NEW_RECORD',
        'acl_action' => 'create',
        'acl_module' => $moduleName,
        'icon' => 'icon-plus',
    ),
    array(
        'route' => "#$moduleName",
        'label' => 'LNK_LIST',
        'acl_action' => 'list',
        'acl_module' => $moduleName,
```

```
        'icon' => 'icon-reorder',
    ),
    array(
        'route' => "#bwc/index.php?module=Import&action=Step1&import_m
odule=$moduleName&return_module=$moduleName&return_action=index",
        'label' => 'LBL_IMPORT',
        'acl_action' => 'import',
        'acl_module' => $moduleName,
        'icon' => '',
    ),
);
```

This file should be duplicated to the custom directory and edited to adjust the URLs to the BWC format. The example below demonstrates the changes needed to the duplicated file.

./custom/modules/<module>/clients/base/menus/header/header.php

```
  <?php

$moduleName = '<module key>';

$viewdefs[$moduleName]['base']['menu']['header'] = array(
    array(
        'route' => "#bwc/index.php?module=$moduleName&action=EditView"
,
        'label' => 'LNK_NEW_RECORD',
        'acl_action' => 'create',
        'acl_module' => $moduleName,
        'icon' => 'icon-plus',
    ),
    array(
        'route' => "#bwc/index.php?module=$moduleName&action=ListView"
,
        'label' => 'LNK_LIST',
        'acl_action' => 'list',
        'acl_module' => $moduleName,
        'icon' => 'icon-reorder',
    ),
    array(
        'route' => "#bwc/index.php?module=Import&action=Step1&import_m
odule=$moduleName&return_module=$moduleName&return_action=index",
        'label' => 'LBL_IMPORT',
        'acl_action' => 'import',
```

```
        'acl_module' => $moduleName,
        'icon' => '',
    ),
);
```

Once the file is in place, you will need to navigate to Admin > Repair > Quick Repair and Rebuild.

Verifying BWC Is Enabled

To verify that backward compatibility is enabled, you can inspect App.metadata.get().modules after running a Quick Repair and Rebuild in your browser's console window:

Using Developer Tools in Google Chrome

1. Open Developer Tools
    - This can be done in the following ways:

        1. Command + Option + i
        2. View > Developer > Developer Tools
        3. Right-click on the web page and selecting "Inspect Element"
2. Select the "Console" tab
3. Run the following command, replacing <module key> and verify that it returns true:

```
App.metadata.get().modules.<module key>.isBwcEnabled
```

Using Firebug in Google Chrome or Mozilla Firefox

1. Open Firebug
2. Select the "Console" tab
3. Run the following command, replacing <module key>, and verify that it

returns true:

```
App.metadata.get().modules.<module key>.isBwcEnabled
```

Converting Legacy Modules To Sidecar

How to upgrade a legacy module to be Sidecar enabled.

Overview

After upgrading your instance to Sugar 7.x, some custom modules may be left in the legacy backward compatible format. This is normally due to the module containing a custom view or file that Sugar does not recognize as being a supported customization for Sidecar. To get your module working with Sidecar, you will need to remove the unsupported customization and run the UpgradeModule.php script.

Note: The following commands should be run on a sandbox instance before being applied to any production environment. It is also important to note that this script should only be run against custom modules. Stock modules in backward compatibility should remain in backward compatibility.

Steps to Complete

The following steps require access to both the Sugar filesystem as well as an administrative user.

Note: As of Sugar 10.0, the UpgradeModule.php script has been removed from the codebase. We are providing a zip of the files that were removed so that it is still possible to run the upgrade.

1. Begin by downloading this zip file that contains the previously removed code
2. Unzip the file and move the contents into your Sugar application at ./modules/UpgradeWizard/
3. To upgrade a custom module, identify the module's unique key. This key can be easily found by identifying the module's folder name in ./modules/<module key name>/. The module's key name will be in the format of abc_module.
4. Next, change to the ./modules/UpgradeWizard/ path relative to your Sugar root directory in your terminal or command line application:

```
cd <sugar root>/modules/UpgradeWizard/
```

5. Run the UpgradeModule.php script, passing in the sugar root directory and the unique key of the legacy module:

   ```
   php UpgradeModule.php <sugar root> <module key>
   ```

   An example is shown below:

   ```
   php UpgradeModule.php /var/www/html/sugarcrm/ abc_module
   ```

6. The script with then output a series of messages identifying issues that need to be corrected. Once the issues have been addressed, you can then run the UpgradeModule.php script again to confirm no errors have been found.
7. Once completed, log into your instance and navigate to Admin > Repair > Quick Repair and Rebuild. Test the custom module to ensure it is working as expected. There is no guarantee that the module will be fully functional in Sidecar and may therefore require additional development effort to ensure compatibility.

Integration

Overview

How to integrate with Sugar APIs

Best Practices

Overview

Best practices when integrating and migrating Sugar.

Latency When Posting Data To Sugar

When integrating with Sugar, it is best to avoid long-running web requests. While performance-draining requests are not always obvious, once an issue has been identified, it is best to move the processing to the backend.

By default, we expect a request to an endpoint such as POST /Accounts to be

straightforward, however, adding Workflows, Logic Hooks, and Sugar Logic may stress the otherwise simple process and cause issues with webheads and other resources being used in the process. If you are experiencing these symptoms, the following sections may help you.

Disabling Related Calculation Fields

When a calculated field in Sugar uses the related function in the Sugar Logic, this will cause the calculated field to be executed when the related module is updated. This can cause a cascading effect through the system to update related calculated fields. When this happens you may receive a 502 Gateway Error. You can disable the related calculation field updates temporarily or permanently by adding the following line to the config_override.php file:

```
$sugar_config['disable_related_calc_fields'] = true;
```

Note : This is a global setting that will affect all modules. If you have a calculated field in Accounts that sums up all Opportunities for the account, setting this value to true will no longer update the opportunity account sum in Accounts until the account record itself is modified. However, if this setting is left disabled, the sum would update any time a related opportunity or the account is modified.

More information on this setting can be found in the core configuration settings.

Disabling Logic Hooks

When data is being migrated into Sugar, logic hooks may be adding unnecessary time to your API requests. It is highly recommended for you to disable any unnecessary logic hooks from your system during an initial import. Logic hook definitions may be located in the following files and/or directories:

- ./custom/modules/logic_hooks.php
- ./custom/modules/<module>/logic_hooks.php
- ./custom/Extension/application/Ext/LogicHooks/
- ./custom/Extension/modules/<module>/Ext/LogicHooks/

Disabling Workflows

When data is being migrated into Sugar, workflows may be adding unnecessary time to your API requests. It is highly recommended for you to disable any unnecessary workflows from your system during an initial import in Admin > Workflows.

Queuing Data in the Job Queue

One solution for long running requests is to send the data to the Job Queue to be processed by the cron. To accomplish this, make a file customization to add to the target processing function. For this article's use case, create ./custom/Extension/modules/Schedulers/Ext/ScheduledTasks/CustomCreateAccountJob.php and send any stored data to it for processing. Enter the following code in the php file:

```
  <?php

function CustomCreateAccountJob($job)
{
 if (!empty($job->data)) {
  $account = BeanFactory::newBean('Accounts');
  $fields = json_decode($job->data, true);

  foreach($fields as $field => $value) {
      $account->$field = $value;
  }

  $account->save();

  if (!empty($account->id)) {
       return true;
  }
  }

  return false;
}
```

Next, modify the request to pass the new account data to the SchedulerJobs module in the data field and specify the new function in the target field, as follows:

```
  curl -v -H "oauth-token: 4afd4aea-df99-7cec-4d94-560a97cda9f8" -H "C
ontent-Type: application/json" -X POST -d '{"assigned_user_id": "1", "
name":"Queue Create Account", "status":"queued", "data":"{\"name\": \"
Example Account\"}", "target":"function::CustomCreateAccountJob"}' htt
p://<site_url>rest/v10/SchedulersJobs
```

This solution will queue data for processing and free up system resources to send more requests. For more information, please refer to the Scheduler Jobs

documentation.

Web Services

Overview

Web Services allow for communication between different applications and platforms. Sugar currently supports REST and SOAP APIs. The following sections will outline how to interact with the APIs and what versions of the API we recommend for use.

Versioning

API versioning is the process of creating a new set of API endpoints for new functionality while leaving pre-existing endpoints available for third-party applications and integrations to continue using. This helps to extend the application in an upgrade-safe manner.

Quick Reference

When working with the Web Service API, you should be using the latest REST API specific to your release. A quick reference of this can be found below:

Release[1]	**REST Version**	**REST URL**	**SOAP Version**[2]	**SOAP URL**
12.0.x	v11.16	/rest/v11_16/	v4.1	/service/v4_1/soap.php
11.3.x	v11.15	/rest/v11_15/	v4.1	/service/v4_1/soap.php
11.2.x	v11.14	/rest/v11_14/	v4.1	/service/v4_1/soap.php
11.1.x	v11.13	/rest/v11_13/	v4.1	/service/v4_1/soap.php
11.0.x	v11.12	/rest/v11_12/	v4.1	/service/v4_1/soap.php
10.3.x	v11.11	/rest/v11_11/	v4.1	/service/v4_1/soap.php
10.2.x	v11.10	/rest/v11_10/	v4.1	/service/v4_1/soap.php
10.1.x	v11.9	/rest/v11_9/	v4.1	/service/v4_1/s

				oap.php
10.0.x	v11.8	/rest/v11_8/	v4.1	/service/v4_1/soap.php
9.3.x	v11.7	/rest/v11_7/	v4.1	/service/v4_1/soap.php
9.2.x	v11.6	/rest/v11_6/	v4.1	/service/v4_1/soap.php
9.1.x	v11.5	/rest/v11_5/	v4.1	/service/v4_1/soap.php
9.0.x	v11.4	/rest/v11_4/	v4.1	/service/v4_1/soap.php
8.3.x	v11.4	/rest/v11_4/	v4.1	/service/v4_1/soap.php
8.2.x	v11.3	/rest/v11_3/	v4.1	/service/v4_1/soap.php
8.1.x	v11.2	/rest/v11_2/	v4.1	/service/v4_1/soap.php
8.0.x	v11.1	/rest/v11_1/	v4.1	/service/v4_1/soap.php
7.11.x	v11	/rest/v11/	v4.1	/service/v4_1/soap.php
7.10.x	v11	/rest/v11/	v4.1	/service/v4_1/soap.php
7.9.x	v10	/rest/v10/	v4.1	/service/v4_1/soap.php
7.8.x	v10	/rest/v10/	v4.1	/service/v4_1/soap.php
7.7.x	v10	/rest/v10/	v4.1	/service/v4_1/soap.php
6.5.x	v4.1	/service/v4_1/rest.php	v4.1	/service/v4_1/soap.php

[1] Some of the releases in this table may no longer be officially-supported Sugar versions, but we have included them for archival purposes. For more information, please refer to the Supported Versions resource page.
[2] As of Sugar 7.0, SOAP support is no longer offered with the new APIs. The legacy APIs will remain accessible in the product, however, any existing integrations should be updated to use the latest REST endpoints.

REST API

Overview

v10 - v11.16 API documentation.

What Is REST?

REST stands for 'Representational State Transfer'. As of 7.x, REST is a core component of Sugar that defines how all information is exchanged within the application. The v10+ API is separate from the v1 - v4_1 REST APIs in that it has been rebuilt with the latest REST standards. Most functionality in the system, whether fetching or posting data, is interacting with the API in some way.

Getting Started

How to Access the REST Service

The base endpoint for the REST service can be found at https://<site_url>/rest/v**{version}**/.

Note: version refers to the version of the API you are accessing.

For your reference, the help documentation for all versioned endpoints can be found by navigating to https://<site_url>/rest/v**{version}**/help. Once you have identified your instance's base endpoint, we can begin by authenticating.

Authentication

Sugar uses two-legged OAuth2 for authentication. You simply need to do a POST to /rest/<version>/oauth2/token with the following parameters:

grant_type	String	Type of request. Available grant types are "password" and "refresh_token".
client_id	String	The client_id of "sugar" will automatically create an OAuth Key in the system and can be used for "password" authentication. The client_id of "support_portal" will

		create an OAuth Key if the portal system is enabled and will allow for portal authentication. Other client_id's can be created by the administrator in the OAuthKeys section in the Administration section and can be used in the future for additional grant types, if the client secret is filled in, it will be checked to validate the use of the client id.
client_secret	String	The client's secret key.
username	String	The username of the user authenticating to the system.
password	String	The plaintext password the user authenticating to the system.
platform	String	Defaults to "base" allows you to have custom meta-data per platform. If using a value other than "base", you should make sure it is registered using the Platform extension or configure an API platform in Administration panel.

First, we are going to log in using a grant_type of "password" and a platform of "custom". Normally, when logging into Sugar, users log in with a platform type of "base". We are using "custom" to avoid any potential login conflicts.

```
  curl -X POST -H Cache-Control:no-cache -H "Content-
Type: application/json" -d '{
    "grant_type":"password",
    "client_id":"sugar",
    "client_secret":"",
    "username":"<username>",
    "password":"<password>",
    "platform":"custom"
}' https:/<site_url>/rest/<version>/oauth2/token
```

Once you get the response you will need to hold onto the access_token and the refresh_token. Anytime the access_token is invalidated, you will want to make another request to the token endpoint with a grant_type of "refresh_token". Store just the refresh_token in long-term storage – not the username and password. The response from the server will be as follows:

```
  {
    "access_token": "5ee48ec7-023e-ecff-5184-530bd0358868",
    "expires_in": 3600,
    "token_type": "bearer",
    "scope": null,
    "refresh_token": "5f197357-0167-f7a6-7912-530bd03275b6",
    "refresh_expires_in": 1209600,
    "download_token": "5f531625-e301-e3ea-1b11-530bd098be41"
}
```

Avoiding Login Conflicts

Login conflicts often occur when building integrations or running data migrations with the platform of "base" or any other client type that is in use. This is due to the fact that Sugar uses the same REST API to power all the various clients such as Sugar, Portal, Mobile, and even the Outlook Plugin. Due to this, you need to let the API know you aren't conflicting with another client that may be in use. The way to accomplish this is the /rest/<version>/oauth2/token call by changing the platform parameter to something other than "base", "mobile", or "portal". It is best to name it something that describes and identifies your current integration.

Input / Output Data Types

The default input / output datatype for REST is JSON.

Date Handling

Date and date time inputs should be formatted following the ISO 8601 format. If the time zone is not included in a request, Sugar will assume the time zone of the user making the request.

Filter on a specific date:

```
  {
    "date_start": "2015-08-12"
```

```
}
```

Filter on a date keyword using $dateRange:

```
  {
    "date_start": {
        "$dateRange": "today"
    }
}
```

Filter on date range using manual time zones:

```
  {
    "date_start": {
        "$dateBetween": [
            "2015-09-10T00:00:00+10:00",
            "2015-09-10T23:59:59+10:00"
        ]
    }
}
```

Endpoints

The following sections contain the in-app help documentation for the REST endpoints.

/ GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate

indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

Name	Type	Description	Required
filter	String	The filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query parameters in one of two ways for GET requests: 1. By specifying individual filter arguments as distinct parameters. Example: filter[0][id]=1. 2. By specifying the whole filter as a single JSON-encoded string. Note that this syntax is currently not supported on certain modules. Example: filter=[{"id":"	False

Name	Type	Description	Required
		1"}].	
filter_id	String	Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND.	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
fields	String	Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters (applicable to link and collection fields). The fields id and date_modified will always be returned. Example: name,account_type,description,{"name":"opportunities","fields":["id","name","sales_status"],"order_by":"date_closed:desc"} For more details on additional field parameters, see Relate API and Collection API.	False

Name	Type	Description	Required
view	String	Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list". Example: record	False
order_by	String	How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon. Example: name:DESC,account_type:DESC,date_modified:ASC	False
q	String	A search expression, will search on this module. Cannot be used at the same time as a filter expression or id.	False
deleted	Boolean	Boolean to show deleted records in the result set.	False
nulls_last	Boolean	Boolean to return records with null	False

Name	Type	Description	Required
		values in order_by fields last in the result set.	

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
   "filter":[
      {
         "name":"Nelson Inc"
      }
   ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
   "filter":[
      {
         "name":{
            "$starts":"Nelson"
         }
      }
   ]
}
```

Below is a list of operation types:

Operation	Description
$equals	Performs an exact match on that field.
$not_equals	Performs an exact match on that field.
$not_equals	Matches on non-matching values.
$starts	Matches on anything that starts with the value.
$ends	Matches anything that ends with the value.
$contains	Matches anything that contains the value
$in	Finds anything where field matches one of the values as specified as an array.
$not_in	Finds anything where field does not matches any of the values as specified as an array.
$is_null	Checks if the field is null. This operation does not need a value specified.
$not_null	Checks if the field is not null. This operation does not need a value specified.
$lt	Matches when the field is less than the value.
$lte	Matches when the field is less than or equal to the value.
$gt	Matches when the

Operation		Description	
			field is greater than the value.
	$gte		Matches when the field is greater than or equal to the value.

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "$and" and "$or".

Example

```
{
   "filter":[
      {
         "$or":[
            {
               "name":"Nelson Inc"
            },
            {
               "name":"Nelson LLC"
            }
         ]
      }
   ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "$favorite" and "$owner".

Example

```
{
```

```
   "filter":[
      {
         "$favorite":"_this"
      }
   ]
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Response

```
{
   "next_offset":-1,
   "records":[
      {
         "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
         "name":"Dale Spivey",
         "date_modified":"2013-02-28T05:03:00+00:00",
         "description":"",
         "opportunities": [
            {
               _module: "Opportunities",
               "id": "b0701501-1fab-8ae7-3942-540da93f5017",
               "name": "360 Vacations - 228 Units",
               "date_modified": "2014-09-08T16:05:00+03:00",
               "sales_status": "New"
            },
         ],
         "_acl": {
            "fields": {
            }
```

```
            }
         },
         {
            "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
            "name":"Florence Haddock",
            "date_modified":"2013-02-26T19:12:00+00:00",
            "description":"",
            "opportunities": [
               {
                  _module: "Opportunities"
                  date_modified: "2014-09-08T16:05:00+03:00"
                  id: "9ce7c088-8ee4-7cd3-18f1-540da944d4c0"
                  name: "360 Vacations - 312 Units"
                  sales_status: "New"
               },
            ],
            "_acl": {
               "fields": {
               }
            }
         }
      ]
}
```


Change Log

Version	Change
v10	Added /<module>/filter GET endpoint.

/ POST

Overview

Create a new record of a specified type.

Request Arguments

Name	Type	Description	Required
<record field>	<record field type>	The name value list of fields to populate.	True

Request

```
{
    "name":"Example Account",
    "account_type":"Customer",
    "description":"My Example Account",
    "meetings": {
    {
        "add": ["21e3385e-404f-b470-407e-54044e3d8023"],
        "create": [
        {
            "name": "Test Meeting"
        }
      ]
    }
  }
}
```

Link fields

It is possible to add record relations to other records and create new related records.

Action	Type	Description
add	List	List of related records ID. See RelateRecordApi for details.
create	List	List of name to value arrays of related records.

Response Arguments

Name	Type	Description
<record field>	<record field type>	Returns the fields for the newly created record.

Response

```
{
   "id":"e91b1fa7-1bd8-3c71-be96-512e643f9ca4",
   "name":"Example Account",
```

```
"date_entered":"2013-02-27T19:56:00+00:00",
"date_modified":"2013-02-27T19:56:00+00:00",
"modified_user_id":"1",
"modified_by_name":"Administrator",
"created_by":"1",
"created_by_name":"Administrator",
"description":"My Example Account",
"img":"",
"deleted":false,
"assigned_user_id":"",
"assigned_user_name":"",
"team_name":[
   {
      "id":1,
      "name":"Global",
      "name_2":"",
      "primary":true
   }
],
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"account_type":"Customer",
"industry":"",
"annual_revenue":"",
"phone_fax":"",
"billing_address_street":"",
"billing_address_street_2":"",
"billing_address_street_3":"",
"billing_address_street_4":"",
"billing_address_city":"",
"billing_address_state":"",
"billing_address_postalcode":"",
"billing_address_country":"",
"rating":"",
"phone_office":"",
"phone_alternate":"",
"website":"",
"ownership":"",
"employees":"",
"ticker_symbol":"",
"shipping_address_street":"",
"shipping_address_street_2":"",
"shipping_address_street_3":"",
"shipping_address_street_4":"",
```

```
   "shipping_address_city":"",
   "shipping_address_state":"",
   "shipping_address_postalcode":"",
   "shipping_address_country":"",
   "email1":"",
   "parent_id":"",
   "sic_code":"",
   "parent_name":"",
   "email_opt_out":"",
   "invalid_email":"",
   "email":[

   ],
   "campaign_id":"",
   "campaign_name":"",
   "my_favorite":false,
   "_acl":{
      "fields":{

      }
   }
}
```

Change Log

Version	Change
v10 (7.6.0)	Added support for link fields.
v10	Added /<module> POST endpoint.

//:lhs_sync_key_field_value/link_by_sync_keys/:link_name/:rhs_sync_key_field_value DELETE

Overview

Removes a relationship based on sync_key. If both the LHS and RHS records can be found with the respective fields, and those records are related, then remove the relationship of the RHS record to the LHS record.

Request Arguments

Name	Type	Description	Required
module	String	The module the record belongs to	True
lhs_sync_key_field_value	String	A unique ID for the LHS record identifying it in an external system	True
link_name	String	A name for the link	True
rhs_sync_key_field_value	String	A unique ID for the RHS record identifying it in an external system	True

Request

```
  /<module>/:lhs_sync_key_field_value/link_by_sync_keys/:link_name/:rhs_sync_key_field_value
```

Response Arguments

Name	Type	Description
record	String	The ID of the record.
related_record	String	The ID of the related record.

Response

Status 200

```
  {
    "record": "a0328573-a252-a27c-3530-4e4297d4c9e1",
    "related_record": "a0328573-bc54-a554-3530-4e4297d4c9e1"
}
```

Status 422

```
  {
    "error": "invalid_parameter",
    "error_message": "Could not find record with :xhs_sync_key_field_value in module: <module>"
```

```
}
```

Change Log

Version	Change
v11_10	Added /<module>/:lhs_sync_key_field_value/link_by_sync_keys/:link_name/:rhs_sync_key_field_value DELETE endpoint.

//:lhs_sync_key_field_value/link_by_sync_keys/:link_name/:rhs_sync_key_field_value POST

Overview

Creates a relationship based on sync_key. If both the LHS and RHS records can be found with the respective fields, then relate the RHS record to the LHS record.

Request Arguments

Name	Type	Description	Required
module	String	The module the record belongs to	True
lhs_sync_key_field_value	String	A unique ID for the LHS record identifying it in an external system	True
link_name	String	A name for the link	True
rhs_sync_key_field_value	String	A unique ID for the RHS record identifying it in an external system	True

Request

```
  /<module>/:lhs_sync_key_field_value/link_by_sync_keys/:link_name/:rhs_sync_key_field_value
```

Response Arguments

Name	Type	Description
record	String	The ID of the record.
related_record	String	The ID of the related record.

Response

Status 200

```
  {
    "record": "a0328573-a252-a27c-3530-4e4297d4c9e1",
    "related_record": "a0328573-bc54-a554-3530-4e4297d4c9e1"
}
```

Status 422

```
  {
    "error": "invalid_parameter",
    "error_message": "Could not find record with :xhs_sync_key_field_v
alue in module: <module>"
}
```

Change Log

Version	Change
v11_10	Added /<module>/:lhs_sync_key_field_value/link_by_sync_keys/:link_name/:rhs_sync_key_field_value POST endpoint.

//:record DELETE

Overview

Delete a record of a specified type.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

Name	Type	Description
id	String	Returns the ID of the deleted record.

Response

```
{
    "id": "11cf0d0a-40af-8cb1-9da0-5057a5f511f9"
}
```

Change Log

Version	Change
v10	Added /<module>/:record DELETE endpoint.

//:record GET

Overview

Retrieves a record.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

Name	Type	Description
<record field>	<record field type>	Returns the fields for the selected record.

Response

```
{
   "id":"bdd59d85-687b-1739-b00a-512d09f6db9e",
   "name":"Insight Marketing Inc",
```

```
"date_entered":"2013-02-26T19:12:00+00:00",
"date_modified":"2013-02-26T19:12:00+00:00",
"modified_user_id":"1",
"modified_by_name":"Administrator",
"created_by":"1",
"created_by_name":"Administrator",
"description":"",
"img":"",
"last_activity_date":"2013-02-26T19:12:00+00:00",
"deleted":false,
"assigned_user_id":"seed_max_id",
"assigned_user_name":"Max Jensen",
"team_name":[
   {
      "id":"East",
      "name":"East",
      "name_2":"",
      "primary":false
   },
   {
      "id":1,
      "name":"Global",
      "name_2":"",
      "primary":false
   },
   {
      "id":"West",
      "name":"West",
      "name_2":"",
      "primary":true
   }
],
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"account_type":"Customer",
"industry":"Electronics",
"annual_revenue":"",
"phone_fax":"",
"billing_address_street":"345 Sugar Blvd.",
"billing_address_street_2":"",
"billing_address_street_3":"",
"billing_address_street_4":"",
"billing_address_city":"San Mateo",
"billing_address_state":"CA",
```

```
   "billing_address_postalcode":"56019",
   "billing_address_country":"USA",
   "rating":"",
   "phone_office":"(927) 136-9572",
   "phone_alternate":"",
   "website":"www.sectionvegan.de",
   "ownership":"",
   "employees":"",
   "ticker_symbol":"",
   "shipping_address_street":"345 Sugar Blvd.",
   "shipping_address_street_2":"",
   "shipping_address_street_3":"",
   "shipping_address_street_4":"",
   "shipping_address_city":"San Mateo",
   "shipping_address_state":"CA",
   "shipping_address_postalcode":"56019",
   "shipping_address_country":"USA",
   "email1":"kid.support.vegan@example.info",
   "parent_id":"",
   "sic_code":"",
   "parent_name":"",
   "email_opt_out":false,
   "invalid_email":false,
   "email":[
      {
         "email_address":"kid.support.vegan@example.info",
         "opt_out":"0",
         "invalid_email":"0",
         "primary_address":"1"
      },
      {
         "email_address":"phone.kid@example.cn",
         "opt_out":"0",
         "invalid_email":"0",
         "primary_address":"0"
      }
   ],
   "campaign_id":"",
   "campaign_name":"",
   "my_favorite":false,
   "_acl":{
      "fields":{

      }
   }
}
```

Change Log

Version	Change
v10	Added /<module>/:record GET endpoint.

//:record PUT

Overview

Update a record of the specified type.

Request Arguments

Name	Type	Description	Required
<record field>	<record field type>	The name value list of fields to populate.	True

Request

```
{
    "name":"New Account Name",
    "phone_office":"(555) 888-5555",
    "website":"www.newsite.com",
    "meetings": {
    {
        "add": ["21e3385e-404f-b470-407e-54044e3d8024"],
        "delete": ["21e3385e-404f-b470-407e-54044e3d8023"],
        "create": [
        {
            "name": "Test Meeting"
        }
      ]
    }
}
```

Link fields

It is possible to add or delete record relations to other records and create new related records.

Action	Type	Description
add	List	List of related records ID. See RelateRecordApi for details.
delete	List	List of related records ID.
create	List	List of name to value arrays of related records.

Response Arguments

Name	Type	Description
<record field>	<record field type>	Returns the fields for the updated record.

Response


```
{
   "id":"f222265a-b755-da89-0bc7-512d09b800b6",
   "name":"New Account Name",
   "date_entered":"2013-02-26T19:12:00+00:00",
   "date_modified":"2013-02-27T22:49:00+00:00",
   "modified_user_id":"1",
   "modified_by_name":"Administrator",
   "created_by":"1",
   "created_by_name":"Administrator",
   "description":"",
   "img":"",
   "last_activity_date":"2013-02-26T19:12:00+00:00",
   "deleted":false,
   "assigned_user_id":"seed_sarah_id",
   "assigned_user_name":"Sarah Smith",
   "team_name":[
      {
         "id":1,
         "name":"Global",
         "name_2":"",
         "primary":false
      },
```

```
      {
         "id":"West",
         "name":"West",
         "name_2":"",
         "primary":true
      }
   ],
   "linkedin":"",
   "facebook":"",
   "twitter":"",
   "googleplus":"",
   "account_type":"Customer",
   "industry":"Hospitality",
   "annual_revenue":"",
   "phone_fax":"",
   "billing_address_street":"9 IBM Path",
   "billing_address_street_2":"",
   "billing_address_street_3":"",
   "billing_address_street_4":"",
   "billing_address_city":"San Francisco",
   "billing_address_state":"CA",
   "billing_address_postalcode":"43635",
   "billing_address_country":"USA",
   "rating":"",
   "phone_office":"(555) 888-5555",
   "phone_alternate":"",
   "website":"www.newsite.com",
   "ownership":"",
   "employees":"",
   "ticker_symbol":"",
   "shipping_address_street":"9 IBM Path",
   "shipping_address_street_2":"",
   "shipping_address_street_3":"",
   "shipping_address_street_4":"",
   "shipping_address_city":"San Francisco",
   "shipping_address_state":"CA",
   "shipping_address_postalcode":"43635",
   "shipping_address_country":"USA",
   "email1":"kid86@example.net",
   "parent_id":"",
   "sic_code":"",
   "parent_name":"",
   "email_opt_out":false,
   "invalid_email":false,
   "email":[
      {
```

```
            "email_address":"kid86@example.net",
            "opt_out":"0",
            "invalid_email":"0",
            "primary_address":"1"
         },
         {
            "email_address":"the.dev@example.name",
            "opt_out":"0",
            "invalid_email":"0",
            "primary_address":"0"
         }
      ],
      "campaign_id":"",
      "campaign_name":"",
      "my_favorite":false,
      "_acl":{
         "fields":{

         }
      }
}
```


Change Log

Version	Change
v10	Added /<module>/:record PUT endpoint.

//:record/audit GET

Overview

Returns data changes for a specific record.

Request Arguments

Name	Type	Description	Required
record	String	The record id.	True
module	String	The name of the module.	True

Query Parameters

Name	Type	Description	Required
max_num	Integer	A maximum number of records to return. The default is all records.	False
offset	Integer	The number of records to skip over before records are returned. The default is 0. The parameter will be ignored if the max_num parameter is missing.	False

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Response


```
{
 "next_offset":-1,
 "records":[
    {
       "id":"b569e480-237a-5921-4382-512ff555fee7",
       "parent_id":"ef1b40aa-5815-4f8d-e909-512d09617ac8",
       "date_created":"03\/01\/2013 12:26am",
       "created_by":"admin",
       "field_name":"Team ID:",
       "data_type":"team_list",
```

```
            "before_value_string":"East",
            "after_value_string":"Will"
         },
         {
            "id":"bc0bee72-8398-a0a6-d731-512ff5bfebc7",
            "parent_id":"ef1b40aa-5815-4f8d-e909-512d09617ac8",
            "date_created":"03\/01\/2013 12:26am",
            "created_by":"admin",
            "field_name":"Teams",
            "data_type":"id",
            "before_value_string":"East, Global, West",
            "after_value_string":"Jim, Will"
         }
      ]
}
```


Change Log

Version	Change
v11_11	Added optional max_num and offset parameters to /Audit GET endpoint.
v10	Added /Audit GET endpoint.

//:record/children GET

Overview

Retrieves all children of selected record.

Request Arguments

Name	Type	Description	Required
module	String	The name of sugar module that contains a nested set data and implements the NestedSetInterface.	True
:record	String	The ID of record	True

Response Arguments

This endpoint does not return any response arguments.

Response


```
[{
    "id": "045c1de6-327b-11e4-818b-5404a67f3363",
    "name": "Financial",
    "date_entered": null,
    "date_modified": null,
    "modified_user_id": null,
    "created_by": null,
    "description": null,
    "deleted": "0",
    "source_id": null,
    "source_type": null,
    "source_meta": null,
    "root": "935d3e07-327a-11e4-818b-5404a67f3363",
    "lft": "11",
    "rgt": "14",
    "level": "2"
}, {
    "id": "0f65a6c7-327b-11e4-818b-5404a67f3363",
    "name": "Agreements",
    "date_entered": null,
    "date_modified": null,
    "modified_user_id": null,
    "created_by": null,
    "description": null,
    "deleted": "0",
    "source_id": null,
    "source_type": null,
    "source_meta": null,
    "root": "935d3e07-327a-11e4-818b-5404a67f3363",
    "lft": "15",
    "rgt": "16",
    "level": "2"
}, {
    "id": "14d30bf3-327b-11e4-818b-5404a67f3363",
    "name": "Clients",
    "date_entered": null,
    "date_modified": null,
    "modified_user_id": null,
    "created_by": null,
    "description": null,
```

```
    "deleted": "0",
    "source_id": null,
    "source_type": null,
    "source_meta": null,
    "root": "935d3e07-327a-11e4-818b-5404a67f3363",
    "lft": "17",
    "rgt": "18",
    "level": "2"
}]
```


Change Log

Version	Change
v10	Added /<module>/:record/children GET endpoint.

//:record/collection/:collection_name GET

Overview

Lists related records from multiple links at a time.

Summary

This endpoint will return a set of related records fetched from multiple links defined by :collection_name. The records will be filtered, sorted and truncated to max_num as a single collection. Collections may use a field mapping. For instance, the due_date of Tasks may be referred to as date_end. In this case the alias (date_end) should be used in filter and order_by expressions instead of real field name. Note that response data still contains the original fields for the module.

Request Arguments

Name	Type	Description	Required
fields	String	See Filter API. If collection definition uses aliases, then aliases should be used instead of real field names.	False
view	String	See Filter API.	False

Name	Type	Description	Required
max_num	Integer	See Filter API.	False
filter	String	See Filter API. If collection definition uses aliases, then aliases should be used instead of real field names.	False
order_by	String	See Filter API. If collection definition uses aliases, then aliases should be used instead of real field names.	False
offset	Map	Individual offset for each link which the collection consists of. -1 (or any negative value) denotes that the link should be skipped. If link offset is not specified, it defaults to 0.	False

Response Arguments

Name	Type	Description
next_offset	Map	The next offset to retrieve records. -1 will be returned for the given link when there are no more records.
records	Array	The record result set.
errors	Map	Errors encountered while making requests for the individual links. These errors are returned for the client to consume, but do not affect the response status of the request.

Response


```
{
    "next_offset": {
        "calls": 1,
        "meetings": -1,
        "tasks": -1,
    },
    "records": [
        {
            "_module": "Calls",
            "_link": "calls",
            "id": "8703fbf3-0ffa-c288-8d2c-512f943ecdc3",
            "name": "Discuss review process",
            "date_end": "2014-02-26T19:12:00+00:00"
        },
        {
            "_module": "Tasks",
            "_link": "tasks",
            "id": "e1c495cb-af17-1b37-dd66-512f934fe155",
            "name": "Introduce all players",
            "due_date": "2014-02-26T19:12:00+00:00"
        },
        {
            "_module": "Tasks",
            "_link": "tasks",
            "id": "456b7848-9959-5a64-cd34-512d0938addd",
            "name": "Follow-up on proposal",
            "due_date": "2014-02-26T19:12:00+00:00"
        }
    ],
    "errors": {
        "meetings": {
            "code": 403,
            "error": "not_authorized",
            "error_message": "You are not authorized to perform this a
ction. Contact your administrator if you need access."
        }
    }
}
```


Change Log

Version	Change
v10	Added /<module>/:record/collection/:collection_name GET endpoint.

//:record/collection/:collection_name/count GET

Overview

Counts related records from multiple links at a time.

Summary

This endpoint will return count of records related by multiple links defined by :collection_name. The records may be filtered.

Request Arguments

Name	Type	Description	Required
filter	String	See Filter API. If collection definition uses aliases, then aliases should be used instead of real field names.	False

Response Arguments

Name	Type	Description
record_count	Map	Count of related records.

Response

```
{
   "record_count": {
      "calls": 1,
      "meetings": 2,
      "tasks": 3
   }
}
```

Change Log

Version	Change
v10	Added /<module>/:record/collection/:collection_name/count GET endpoint.

//:record/favorite DELETE

Overview

Removes a record of a specified type as a favorite for the current user.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

Name	Type	Description
<record field>	<record field type>	Returns the fields for the selected record.

Response


```
{
   "id":"bdd59d85-687b-1739-b00a-512d09f6db9e",
   "name":"Insight Marketing Inc",
   "date_entered":"2013-02-26T19:12:00+00:00",
   "date_modified":"2013-02-26T19:12:00+00:00",
   "modified_user_id":"1",
   "modified_by_name":"Administrator",
   "created_by":"1",
   "created_by_name":"Administrator",
   "description":"",
   "img":"",
   "last_activity_date":"2013-02-26T19:12:00+00:00",
   "deleted":false,
   "assigned_user_id":"seed_max_id",
   "assigned_user_name":"Max Jensen",
   "team_name":[
      {
         "id":"East",
         "name":"East",
         "name_2":"",
```

```
            "primary":false
        },
        {
            "id":1,
            "name":"Global",
            "name_2":"",
            "primary":false
        },
        {
            "id":"West",
            "name":"West",
            "name_2":"",
            "primary":true
        }
    ],
    "linkedin":"",
    "facebook":"",
    "twitter":"",
    "googleplus":"",
    "account_type":"Customer",
    "industry":"Electronics",
    "annual_revenue":"",
    "phone_fax":"",
    "billing_address_street":"345 Sugar Blvd.",
    "billing_address_street_2":"",
    "billing_address_street_3":"",
    "billing_address_street_4":"",
    "billing_address_city":"San Mateo",
    "billing_address_state":"CA",
    "billing_address_postalcode":"56019",
    "billing_address_country":"USA",
    "rating":"",
    "phone_office":"(927) 136-9572",
    "phone_alternate":"",
    "website":"www.sectionvegan.de",
    "ownership":"",
    "employees":"",
    "ticker_symbol":"",
    "shipping_address_street":"345 Sugar Blvd.",
    "shipping_address_street_2":"",
    "shipping_address_street_3":"",
    "shipping_address_street_4":"",
    "shipping_address_city":"San Mateo",
    "shipping_address_state":"CA",
    "shipping_address_postalcode":"56019",
    "shipping_address_country":"USA",
```

```
   "email1":"kid.support.vegan@example.info",
   "parent_id":"",
   "sic_code":"",
   "parent_name":"",
   "email_opt_out":false,
   "invalid_email":false,
   "email":[
      {
         "email_address":"kid.support.vegan@example.info",
         "opt_out":"0",
         "invalid_email":"0",
         "primary_address":"1"
      },
      {
         "email_address":"phone.kid@example.cn",
         "opt_out":"0",
         "invalid_email":"0",
         "primary_address":"0"
      }
   ],
   "campaign_id":"",
   "campaign_name":"",
   "my_favorite":false,
   "_acl":{
      "fields":{

      }
   }
}
```


Change Log

Version	Change
v10	Added /<module>/:record/favorite DELETE endpoint.

//:record/favorite PUT

Overview

Updates a record of a specified type as a favorite for the current user.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

Name	Type	Description
<record field>	<record field type>	Returns the fields for the selected record.

Response


```
{
   "id":"bdd59d85-687b-1739-b00a-512d09f6db9e",
   "name":"Insight Marketing Inc",
   "date_entered":"2013-02-26T19:12:00+00:00",
   "date_modified":"2013-02-26T19:12:00+00:00",
   "modified_user_id":"1",
   "modified_by_name":"Administrator",
   "created_by":"1",
   "created_by_name":"Administrator",
   "description":"",
   "img":"",
   "last_activity_date":"2013-02-26T19:12:00+00:00",
   "deleted":false,
   "assigned_user_id":"seed_max_id",
   "assigned_user_name":"Max Jensen",
   "team_name":[
      {
         "id":"East",
         "name":"East",
         "name_2":"",
         "primary":false
      },
      {
         "id":1,
         "name":"Global",
         "name_2":"",
         "primary":false
      },
      {
         "id":"West",
         "name":"West",
         "name_2":"",
         "primary":true
```

```
   }
],
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"account_type":"Customer",
"industry":"Electronics",
"annual_revenue":"",
"phone_fax":"",
"billing_address_street":"345 Sugar Blvd.",
"billing_address_street_2":"",
"billing_address_street_3":"",
"billing_address_street_4":"",
"billing_address_city":"San Mateo",
"billing_address_state":"CA",
"billing_address_postalcode":"56019",
"billing_address_country":"USA",
"rating":"",
"phone_office":"(927) 136-9572",
"phone_alternate":"",
"website":"www.sectionvegan.de",
"ownership":"",
"employees":"",
"ticker_symbol":"",
"shipping_address_street":"345 Sugar Blvd.",
"shipping_address_street_2":"",
"shipping_address_street_3":"",
"shipping_address_street_4":"",
"shipping_address_city":"San Mateo",
"shipping_address_state":"CA",
"shipping_address_postalcode":"56019",
"shipping_address_country":"USA",
"email1":"kid.support.vegan@example.info",
"parent_id":"",
"sic_code":"",
"parent_name":"",
"email_opt_out":false,
"invalid_email":false,
"email":[
   {
      "email_address":"kid.support.vegan@example.info",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"1"
   },
```

```
      {
         "email_address":"phone.kid@example.cn",
         "opt_out":"0",
         "invalid_email":"0",
         "primary_address":"0"
      }
   ],
   "campaign_id":"",
   "campaign_name":"",
   "my_favorite":true,
   "_acl":{
      "fields":{

      }
   }
}
```

Change Log

Version	Change
v10	Added /<module>/:record/favorite PUT endpoint.

//:record/file GET

Overview

Lists all populated fields of type "file" or of type "image" for a record.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

Name	Type	Description
field	Array	The fields containing file properties.
field.content-type	String	The files content type.
field.content-length	Integer	The files content length.

Name	Type	Description
field.name	String	The files ID.
field.width	Integer	The width of the image.
field.height	Integer	The height of the image.
field.uri	String	The URI of the file.

Response


```
{
   "picture":{
      "content-type":"image\/png",
      "content-length":72512,
      "name":"1a7b8f5c-b11c-0094-c8d8-512e9daaa983",
      "width":933,
      "height":519,
      "uri":"http:\/\/sugarcrm\/rest\/v10\/Contacts\/fa300a0e-0ad1-b32
2-9601-512d0983c19a\/file\/picture"
   },
   "record": {
      "my_favorite": false,
      "following": true,
      "id": "fa300a0e-0ad1-b322-9601-512d0983c19a",
      "name": "Test",
      "date_modified": "2014-05-16T03:49:12+02:00",
      "modified_user_id": "1",
      "modified_by_name": "admin",
      "created_by": "1",
      "created_by_name": "Administrator",
      "doc_owner": "1",
      "description": "",
      "deleted": false,
      "assigned_user_id": "1",
      "assigned_user_name": "Administrator",
      "team_count": "",
      "team_name": [
         {
            "id": 1,
            "name": "Global",
            "name_2": "",
            "primary": true
         }
      ],
      "email": [],
      "email1": "",
```

```
"email2": "",
"invalid_email": "",
"email_opt_out": "",
"email_addresses_non_primary": "",
"salutation": "",
"first_name": "",
"last_name": "Test",
"full_name": "Test",
"title": "",
"facebook": "",
"twitter": "",
"googleplus": "",
"department": "",
"do_not_call": false,
"phone_home": "",
"phone_mobile": "",
"phone_work": "",
"phone_other": "",
"phone_fax": "",
"primary_address_street": "",
"primary_address_city": "",
"primary_address_state": "",
"primary_address_postalcode": "",
"primary_address_country": "",
"alt_address_street": "",
"alt_address_city": "",
"alt_address_state": "",
"alt_address_postalcode": "",
"alt_address_country": "",
"assistant": "",
"assistant_phone": "",
"picture": "",
"email_and_name1": "",
"lead_source": "",
"account_name": "",
"account_id": "",
"dnb_principal_id": "",
"opportunity_role_fields": "",
"opportunity_role_id": "",
"opportunity_role": "",
"reports_to_id": "",
"report_to_name": "",
"birthdate": "",
"portal_name": "",
"portal_active": false,
"portal_password": null,
```

```
      "portal_password1": null,
      "portal_app": "",
      "preferred_language": "en_us",
      "campaign_id": "",
      "campaign_name": "",
      "c_accept_status_fields": "",
      "m_accept_status_fields": "",
      "accept_status_id": "",
      "accept_status_name": "",
      "accept_status_calls": "",
      "accept_status_meetings": "",
      "sync_contact": false,
      "mkto_sync": false,
      "mkto_id": null,
      "mkto_lead_score": null,
      "_acl": {
         "fields": {}
      },
      "_module": "Contacts"
  }
}
```


Change Log

Version	Change
v10	Added /<module>/:record/file GET endpoint.

//:record/file/:field DELETE

Overview

Removes an attachment from a field for a record and subsequently removes the file from the file system.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

Name	Type	Description
field	Array	The fields containing file properties.
field.content-type	String	The files content type.
field.content-length	Integer	The files content length.
field.name	String	The files ID.
field.width	Integer	The width of the image.
field.height	Integer	The height of the image.
field.uri	String	The URI of the file.

Response

```
{
    "picture":{}
}
```

Change Log

Version	Change
v10	Added /<module>/:record/file/:field DELETE endpoint.

//:record/file/:field GET

Overview

Retrieves an attached file for a specific field on a record.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

The response from this request is an HTTP response with a forced download. This can be used in rendering images through the API or in offering immediate file downloads.

Response

```
Cache-Control: no-cache, must-revalidate Connection: keep-alive Conten
t-Encoding: gzip Content-Type: application/octet-stream Date: Mon, 30
Jan 2012 17:00:46 GMT Expires: Mon, 30 Jan 2012 17:00:45 GMT Last-Modi
fied: Thu, 10 Nov 2011 19:01:31 GMT Transfer-
Encoding: chunked Vary: Accept-Encoding...
```

Change Log

Version	Change
v10	Added /<module>/:record/file/:field GET endpoint.

//:record/file/:field POST

Overview

Attaches a file to a field on a record.

Request Arguments

Name	Type	Description	Required
format	String	The data format. Currently accepts 'sugar-html-json'.	True
delete_if_fails	Boolean	Indicates whether the API is to mark related record deleted if the file upload fails.	False
oauth_token	String	The oauth_token value.	False - Required if only if delete_if_fails is true.
<attachment field>	String	The field and file to populate. Example: {"":"@\/path\/to\/ExampleDocument.txt"}	True

Request

```
{
   "format":"sugar-html-json",
   "delete_if_fails":true,
   "oauth_token":"43b6b327-cc70-c301-3299-512ffb99ad97",
   "<attachment field>":"@\/path\/to\/ExampleDocument.txt"
}
```

Response Arguments

Name	Type	Description
content-type	String	The files content type.
content-length	Integer	The files content length.
name	String	The files name.
uri	String	The URI of the file.

Response

```
{
   "content-type":"text\/plain",
   "content-length":16,
   "name":"ExampleDocument.txt",
   "uri":"http:\/\/sugarcrm\/rest\/v10\/Notes\/ca66c92f-5a8b-28b4-4fc8
-512d099b790b\/file\/<attachment field>?format=sugar-html-json&delete_
if_fails=1&oauth_token=6ec91cf3-1f97-25b9-e0b1-512f8971b2d4"
}
```

Change Log

Version	Change
v10	Added /<module>/:record/file/:field POST endpoint.

//:record/file/:field PUT

Overview

This endpoint takes a file or image and saves it to a record that already contains

an attachment in the specified field. The PUT method is very similar to the POST method but differs slightly in how the request is constructed. PUT requests should send the file data as the body of the request. Optionally, a filename query parameter can be sent with the request to assign a name. Additionally, the PUT method can accept base64 encoded file data which will be decoded by the endpoint if the content_transfer_encoding parameter is set to 'base64'.

NOTE: In lieu of the content_transfer_encoding parameter, a request header of X-Content-Transfer-Encoding can also be sent with a value of 'base64'. In the event both the content transfer encoding header and request parameter are sent, the header will be used.

Request Arguments

Name	Type	Description	Required
filename	String	Filename to save the document as	False
content_transfer_encoding	String	When set to 'base64', indicates the file contents are base64 encoded and will result in the file contents being base64 decoded before being saved	False

Request

PUT /rest/v10/Notes/abcd-1234/file/field/ HTTP/1.1 Host: localhost Connection: keep-alive Content-Length: 23456 Content-Type: application/document-doc ...This is where the bin data would be

Response Arguments

Name	Type	Description
field	Array	The fields containing file properties.
field.content-type	String	The files content type.
field.content-length	Integer	The files content length.
field.name	String	The files name.
field.width	Integer	The width of the image.
field.height	Integer	The height of the image.

Name	Type	Description
field.uri	String	The URI of the file.

Response

```
{
   "picture":{
      "content-type":"image\/png",
      "content-length":72512,
      "name":"1a7b8f5c-b11c-0094-c8d8-512e9daaa983",
      "width":933,
      "height":519,
      "uri":"http:\/\/sugarcrm\/rest\/v10\/Contacts\/fa300a0e-0ad1-b32
2-9601-512d0983c19a\/file\/picture"
   }
    "attachment":{
        "content-type":"application/document-doc",
        "content-length":"873921",
        "name":"myFile.doc",
        "uri": "http:\/\/sugarcrm\/rest\/v10\/Contacts\/f2f9aa4d-99a8-
e86e-f4d5-512d0986effa\/file\/attachment"
    }
}
```

Change Log

Version	Change
v10	Added /<module>/:record/file/:field PUT endpoint.

//:record/link POST

Overview

Creates relationships to a pre-existing record.

Request Arguments

Name	Type	Description	Required
<relationship	<string>	Link between	True

Name	Type	Description	Required
link>		targeted and related records.	
<record ID>	<string>	The name value list of related records. Each item of the list may be either string equal to related item ID, or map containing record ID ("id" key is required in this case) and addition relationship properties.	True

Request

```
{
    link_name: "accounts"
    ids: [
        "da6a3741-2a81-ba7f-f249-512d0932e94e",
        {
            "id": "e689173e-c953-1e14-c215-512d0927e7a2",
            "role": "owner"
        }
    ]
}
```

Response Arguments

Name	Type	Description
record	Array	The record linked to related records.
related_records	Array	Records that were associated.

Response

```
    "record": {
        "id": "da6a3741-2a81-ba7f-f249-512d0932e94e",
```

```
"name": "Slender Broadband Inc - 1000 units",
"date_entered": "2013-02-26T19:12:00+00:00",
"date_modified": "2013-02-26T19:12:00+00:00",
"modified_user_id": "1",
"modified_by_name": "Administrator",
"created_by": "1",
"created_by_name": "Administrator",
"description": "",
"img": "",
"last_activity_date": "2013-02-28T18:21:00+00:00",
"deleted": false,
"assigned_user_id": "seed_max_id",
"assigned_user_name": "Max Jensen",
"team_name": [
    {
        "id": "East",
        "name": "East",
        "name_2": "",
        "primary": false
    },
    {
        "id": "West",
        "name": "West",
        "name_2": "",
        "primary": true
    }
],
"opportunity_type": "",
"account_name": "Slender Broadband Inc",
"account_id": "181461c6-dc81-1115-1fe0-512d092e8f15",
"campaign_id": "",
"campaign_name": "",
"lead_source": "Campaign",
"amount": "25000",
"base_rate": "1",
"amount_usdollar": "25000",
"currency_id": "-99",
"currency_name": "",
"currency_symbol": "",
"date_closed": "2013-02-27",
"date_closed_timestamp": "1361992480",
"next_step": "",
"sales_stage": "Needs Analysis",
"sales_status": "New",
"probability": "90",
"best_case": "25000",
```

```
        "worst_case": "25000",
        "commit_stage": "include",
        "my_favorite": false,
        "_acl": {
            "fields": {

            }
        }
    },
    "related_records": [
        {
            "id": "e689173e-c953-1e14-c215-512d0927e7a2",
            "name": "Gus Dales",
            "date_entered": "2013-02-26T19:12:00+00:00",
            "date_modified": "2013-02-26T19:12:00+00:00",
            "modified_user_id": "1",
            "modified_by_name": "Administrator",
            "created_by": "1",
            "created_by_name": "Administrator",
            "description": "",
            "img": "",
            "deleted": false,
            "assigned_user_id": "seed_sally_id",
            "assigned_user_name": "Sally Bronsen",
            "team_name": [
                {
                    "id": "West",
                    "name": "West",
                    "name_2": "",
                    "primary": true
                }
            ],
            "salutation": "",
            "first_name": "Gus",
            "last_name": "Dales",
            "full_name": "Gus Dales",
            "title": "Director Operations",
            "linkedin": "",
            "facebook": "",
            "twitter": "",
            "googleplus": "",
            "department": "",
            "do_not_call": false,
            "phone_home": "(661) 120-2292",
            "email": [
                {
```

```
                    "email_address": "section.sugar.section@example.it
",
                    "opt_out": "1",
                    "invalid_email": "0",
                    "primary_address": "0"
                },
                {
                    "email_address": "support.qa.kid@example.co.uk",
                    "opt_out": "0",
                    "invalid_email": "0",
                    "primary_address": "1"
                }
            ],
            "phone_mobile": "(294) 447-9707",
            "phone_work": "(036) 840-3216",
            "phone_other": "",
            "phone_fax": "",
            "email1": "support.qa.kid@example.co.uk",
            "email2": "section.sugar.section@example.it",
            "invalid_email": false,
            "email_opt_out": false,
            "primary_address_street": "48920 San Carlos Ave",
            "primary_address_street_2": "",
            "primary_address_street_3": "",
            "primary_address_city": "Persistance",
            "primary_address_state": "CA",
            "primary_address_postalcode": "54556",
            "primary_address_country": "USA",
            "alt_address_street": "",
            "alt_address_street_2": "",
            "alt_address_street_3": "",
            "alt_address_city": "",
            "alt_address_state": "",
            "alt_address_postalcode": "",
            "alt_address_country": "",
            "assistant": "",
            "assistant_phone": "",
            "picture": "",
            "email_and_name1": "",
            "lead_source": "Support Portal User Registration",
            "account_name": "Arts & Crafts Inc",
            "account_id": "d43243c6-9b8e-2973-aee2-512d09bc34b4",
            "opportunity_role_fields": "",
            "opportunity_role_id": "",
            "opportunity_role": "Technical Advisor",
            "reports_to_id": "",
```

```
        "report_to_name": "",
        "portal_name": "GusDales145",
        "portal_active": true,
        "portal_password": "$1$JxYr6tmM$b.O6.KF42jP46RadSwz0N0",
        "portal_password1": "",
        "portal_app": "",
        "preferred_language": "en_us",
        "campaign_id": "",
        "campaign_name": "",
        "c_accept_status_fields": "",
        "m_accept_status_fields": "",
        "accept_status_id": "",
        "accept_status_name": "",
        "sync_contact": "",
        "my_favorite": false,
        "_acl": {
            "fields": {
            }
        }
    },
    {
        "id": "da6a3741-2a81-ba7f-f249-512d0932e94e",
        "name": "Slender Broadband Inc - 1000 units",
        "date_entered": "2013-02-26T19:12:00+00:00",
        "date_modified": "2013-02-26T19:12:00+00:00",
        "modified_user_id": "1",
        "modified_by_name": "Administrator",
        "created_by": "1",
        "created_by_name": "Administrator",
        "description": "",
        "img": "",
        "last_activity_date": "2013-02-28T18:36:00+00:00",
        "deleted": false,
        "assigned_user_id": "seed_max_id",
        "assigned_user_name": "Max Jensen",
        "team_name": [
            {
                "id": "East",
                "name": "East",
                "name_2": "",
                "primary": false
            },
            {
                "id": "West",
                "name": "West",
                "name_2": "",
```

```
                "primary": true
            }
        ],
        "opportunity_type": "",
        "account_name": "Slender Broadband Inc",
        "account_id": "181461c6-dc81-1115-1fe0-512d092e8f15",
        "campaign_id": "",
        "campaign_name": "",
        "lead_source": "Campaign",
        "amount": "25000",
        "base_rate": "1",
        "amount_usdollar": "25000",
        "currency_id": "-99",
        "currency_name": "",
        "currency_symbol": "",
        "date_closed": "2013-02-27",
        "date_closed_timestamp": "1361992480",
        "next_step": "",
        "sales_stage": "Needs Analysis",
        "sales_status": "New",
        "probability": "90",
        "best_case": "25000",
        "worst_case": "25000",
        "commit_stage": "include",
        "my_favorite": false,
        "_acl": {
            "fields": {

            }
        }
    }
]
}
```


Change Log

Version	Change
v10 (7.6.0)	Added support for additional relationship values.
v10 (7.1.5)	Added /<module>/:record/link POST endpoint.

//:record/link/:link_name GET

Overview

Lists related filtered records.

Summary

This endpoint will return a set of related records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Care will need to be taken to make sure this filter has appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

Name	Type	Description	Required
filter	String	The filter expression. Filter expressions are explained below.	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
fields	String	Comma delimited list of fields to return. The field date_modified will always be returned. This argument can be combined with the view argument. Example: name,account_type,description	False

Name	Type	Description	Required
view	String	Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list". Example: record	False
order_by	String	How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon. Example: name:DESC,account_type:DESC,date_modified:ASC	False

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
   "filter":[
      {
```

```
         "name":"Nelson Inc"
      }
   ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
   "filter":[
      {
         "name":{
            "$starts":"Nelson"
         }
      }
   ]
}
```

Below is a list of operation types:

Operation	Description
$equals	Performs an exact match on that field.
$not_equals	Performs an exact match on that field.
$not_equals	Matches on non-matching values.
$starts	Matches on anything that starts with the value.
$ends	Matches anything that ends with the value.
$contains	Matches anything that contains the value
$in	Finds anything

Operation		Description	
			where field matches one of the values as specified as an array.
	$not_in		Finds anything where field does not matches any of the values as specified as an array.
	$is_null		Checks if the field is null. This operation does not need a value specified.
	$not_null		Checks if the field is not null. This operation does not need a value specified.
	$lt		Matches when the field is less than the value.
	$lte		Matches when the field is less than or equal to the value.
	$gt		Matches when the field is greater than the value.
	$gte		Matches when the field is greater than or equal to the value.

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "$and" and "$or".

Example

```
{
   "filter":[
```

```
      {
         "$or":[
            {
               "name":"Nelson Inc"
            },
            {
               "name":"Nelson LLC"
            }
         ]
      }
   ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "$favorite" and "$owner".

Example

```
{
   "filter":[
      {
         "$favorite":"_this"
      }
   ]
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.

Name	Type	Description
records	Array	An array of results containing matched records.

Repsonse


```
{
   "next_offset":-1,
   "records":[
      {
         "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
         "name":"Dale Spivey",
         "date_entered":"2013-02-26T19:12:00+00:00",
         "date_modified":"2013-02-28T05:03:00+00:00",
         "modified_user_id":"1",
         "modified_by_name":"Administrator",
         "created_by":"1",
         "created_by_name":"Administrator",
         "description":"",
         "img":"",
         "deleted":false,
         "assigned_user_id":"seed_sally_id",
         "assigned_user_name":"Sally Bronsen",
         "team_name":[
            {
               "id":"East",
               "name":"East",
               "name_2":"",
               "primary":false
            },
            {
               "id":1,
               "name":"Global",
               "name_2":"",
               "primary":false
            },
            {
               "id":"West",
               "name":"West",
               "name_2":"",
               "primary":true
            }
         ],
         "salutation":"",
```

```
"first_name":"Dale",
"last_name":"Spivey",
"full_name":"Dale Spivey",
"title":"VP Operations",
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(523) 825-4311",
"email":[
   {
      "email_address":"sugar.dev.sugar@example.co.jp",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"1"
   },
   {
      "email_address":"the.support@example.biz",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"0"
   }
],
"phone_mobile":"(373) 861-0757",
"phone_work":"(212) 542-9596",
"phone_other":"",
"phone_fax":"",
"email1":"sugar.dev.sugar@example.co.jp",
"email2":"the.support@example.biz",
"invalid_email":false,
"email_opt_out":false,
"primary_address_street":"345 Sugar Blvd.",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Denver",
"primary_address_state":"CA",
"primary_address_postalcode":"87261",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
```

```
      "alt_address_country":"",
      "assistant":"",
      "assistant_phone":"",
      "picture":"",
      "email_and_name1":"",
      "lead_source":"Campaign",
      "account_name":"Smallville Resources Inc",
      "account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",
      "opportunity_role_fields":"",
      "opportunity_role_id":"",
      "opportunity_role":"",
      "reports_to_id":"",
      "report_to_name":"",
      "portal_name":"DaleSpivey97",
      "portal_active":true,
      "portal_password":"$1$yKMAONHM$Y5S.8CY.WZCZCwfGD1a1Q\/",
      "portal_password1":"",
      "portal_app":"",
      "preferred_language":"en_us",
      "campaign_id":"",
      "campaign_name":"",
      "c_accept_status_fields":"",
      "m_accept_status_fields":"",
      "accept_status_id":"",
      "accept_status_name":"",
      "sync_contact":"",
      "my_favorite":false,
      "_acl":{
         "fields":{

         }
      }
   },
   {
      "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
      "name":"Florence Haddock",
      "date_entered":"2013-02-26T19:12:00+00:00",
      "date_modified":"2013-02-26T19:12:00+00:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "description":"",
      "img":"",
      "deleted":false,
      "assigned_user_id":"seed_sally_id",
```

```
"assigned_user_name":"Sally Bronsen",
"team_name":[
   {
      "id":"East",
      "name":"East",
      "name_2":"",
      "primary":false
   },
   {
      "id":1,
      "name":"Global",
      "name_2":"",
      "primary":false
   },
   {
      "id":"West",
      "name":"West",
      "name_2":"",
      "primary":true
   }
],
"salutation":"",
"first_name":"Florence",
"last_name":"Haddock",
"full_name":"Florence Haddock",
"title":"Director Sales",
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(729) 845-3137",
"email":[
   {
      "email_address":"dev.vegan@example.de",
      "opt_out":"1",
      "invalid_email":"0",
      "primary_address":"0"
   },
   {
      "email_address":"section71@example.it",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"1"
   }
```

```
],
"phone_mobile":"(246) 233-1382",
"phone_work":"(565) 696-6981",
"phone_other":"",
"phone_fax":"",
"email1":"section71@example.it",
"email2":"dev.vegan@example.de",
"invalid_email":false,
"email_opt_out":false,
"primary_address_street":"111 Silicon Valley Road",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Denver",
"primary_address_state":"CA",
"primary_address_postalcode":"79900",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
"assistant":"",
"assistant_phone":"",
"picture":"",
"email_and_name1":"",
"lead_source":"Support Portal User Registration",
"account_name":"Smallville Resources Inc",
"account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",
"opportunity_role_fields":"",
"opportunity_role_id":"",
"opportunity_role":"",
"reports_to_id":"",
"report_to_name":"",
"portal_name":"FlorenceHaddock169",
"portal_active":true,
"portal_password":"$1$nWFhTbK6$JF9BCGSqL\/NCrbhueX5ia0",
"portal_password1":"",
"portal_app":"",
"preferred_language":"en_us",
"campaign_id":"",
"campaign_name":"",
"c_accept_status_fields":"",
"m_accept_status_fields":"",
"accept_status_id":"",
```

```
         "accept_status_name":"",
         "sync_contact":"",
         "my_favorite":false,
         "_acl":{
            "fields":{

            }
         }
      }
   ]
}
```

Change Log

Version	Change
v10	Added /<module>/:record/link/:link_name/filter GET endpoint.

//:record/link/:link_name POST

Overview

Creates a related record.

Request Arguments

Name	Type	Description	Required
<record field>	<record field type>	The name value list of fields to populate.	True

Request

```
{
    "first_name":"Bill",
    "last_name":"Edwards"
}
```

Response Arguments

Name	Type	Description
record	Array	The record associated to the newly created record.
related_record	Array	The record that was created and associated.

Response


```
{
   "record":{
      "id":"da6a3741-2a81-ba7f-f249-512d0932e94e",
      "name":"Slender Broadband Inc - 1000 units",
      "date_entered":"2013-02-26T19:12:00+00:00",
      "date_modified":"2013-02-26T19:12:00+00:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "description":"",
      "img":"",
      "deleted":false,
      "assigned_user_id":"seed_max_id",
      "assigned_user_name":"Max Jensen",
      "team_name":[
         {
            "id":"East",
            "name":"East",
            "name_2":"",
            "primary":false
         },
         {
            "id":"West",
            "name":"West",
            "name_2":"",
            "primary":true
         }
      ],
      "opportunity_type":"",
      "account_name":"Slender Broadband Inc",
      "account_id":"181461c6-dc81-1115-1fe0-512d092e8f15",
      "campaign_id":"",
      "campaign_name":"",
      "lead_source":"Campaign",
```

```
    "amount":"25000",
    "base_rate":"1",
    "amount_usdollar":"25000",
    "currency_id":"-99",
    "currency_name":"",
    "currency_symbol":"",
    "date_closed":"2013-02-27",
    "date_closed_timestamp":"1361992480",
    "next_step":"",
    "sales_stage":"Needs Analysis",
    "sales_status":"New",
    "probability":"90",
    "best_case":"25000",
    "worst_case":"25000",
    "commit_stage":"include",
    "my_favorite":false,
    "_acl":{
       "fields":{

       }
    }
 },
 "related_record":{
    "id":"e1c495cb-af17-1b37-dd66-512f934fe155",
    "name":"Bill Edwards",
    "date_entered":"2013-02-28T17:25:00+00:00",
    "date_modified":"2013-02-28T17:25:00+00:00",
    "modified_user_id":"1",
    "modified_by_name":"Administrator",
    "created_by":"1",
    "created_by_name":"Administrator",
    "description":"",
    "img":"",
    "deleted":false,
    "assigned_user_id":"",
    "assigned_user_name":"",
    "team_name":[
       {
          "id":1,
          "name":"Global",
          "name_2":"",
          "primary":true
       }
    ],
    "salutation":"",
    "first_name":"Bill",
```

```
"last_name":"Edwards",
"full_name":"Bill Edwards",
"title":"",
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"",
"email":[

],
"phone_mobile":"",
"phone_work":"",
"phone_other":"",
"phone_fax":"",
"email1":"",
"email2":"",
"invalid_email":"",
"email_opt_out":"",
"primary_address_street":"",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"",
"primary_address_state":"",
"primary_address_postalcode":"",
"primary_address_country":"",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
"assistant":"",
"assistant_phone":"",
"picture":"",
"email_and_name1":"",
"lead_source":"",
"account_id":"",
"opportunity_role_fields":"",
"opportunity_role_id":"",
"opportunity_role":"",
"reports_to_id":"",
"report_to_name":"",
```

```
      "portal_name":"",
      "portal_active":false,
      "portal_password":"",
      "portal_password1":"",
      "portal_app":"",
      "preferred_language":"en_us",
      "campaign_id":"",
      "campaign_name":"",
      "c_accept_status_fields":"",
      "m_accept_status_fields":"",
      "accept_status_id":"",
      "accept_status_name":"",
      "sync_contact":"",
      "my_favorite":false,
      "_acl":{
         "fields":{

         }
      }
   }
}
```

Change Log

Version	Change
v10	Added /<module>/:record/link/:link_name POST endpoint.

//:record/link/:link_name/:remote_id DELETE

Overview

Deletes an existing relationship between two records.

Request Arguments

This endpoint does not accept any arguments.

Response Arguments

Name	Type	Description
record	Array	The record to disassociate from the related record.
related_record	Array	The record that was disassociated.

Response


```
{
   "record":{
      "id":"da6a3741-2a81-ba7f-f249-512d0932e94e",
      "name":"Slender Broadband Inc - 1000 units",
      "date_entered":"2013-02-26T19:12:00+00:00",
      "date_modified":"2013-02-26T19:12:00+00:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "description":"",
      "img":"",
      "last_activity_date":"2013-02-28T18:36:00+00:00",
      "deleted":false,
      "assigned_user_id":"seed_max_id",
      "assigned_user_name":"Max Jensen",
      "team_name":[
         {
            "id":"East",
            "name":"East",
            "name_2":"",
            "primary":false
         },
         {
            "id":"West",
            "name":"West",
            "name_2":"",
            "primary":true
         }
      ],
      "opportunity_type":"",
      "account_name":"Slender Broadband Inc",
      "account_id":"181461c6-dc81-1115-1fe0-512d092e8f15",
      "campaign_id":"",
      "campaign_name":"",
      "lead_source":"Campaign",
```

```
    "amount":"25000",
    "base_rate":"1",
    "amount_usdollar":"25000",
    "currency_id":"-99",
    "currency_name":"",
    "currency_symbol":"",
    "date_closed":"2013-02-27",
    "date_closed_timestamp":"1361992480",
    "next_step":"",
    "sales_stage":"Needs Analysis",
    "sales_status":"New",
    "probability":"90",
    "best_case":"25000",
    "worst_case":"25000",
    "commit_stage":"include",
    "my_favorite":false,
    "_acl":{
       "fields":{

       }
    }
 },
 "related_record":{
    "id":"e689173e-c953-1e14-c215-512d0927e7a2",
    "name":"Gus Dales",
    "date_entered":"2013-02-26T19:12:00+00:00",
    "date_modified":"2013-02-26T19:12:00+00:00",
    "modified_user_id":"1",
    "modified_by_name":"Administrator",
    "created_by":"1",
    "created_by_name":"Administrator",
    "description":"",
    "img":"",
    "deleted":false,
    "assigned_user_id":"seed_sally_id",
    "assigned_user_name":"Sally Bronsen",
    "team_name":[
       {
          "id":"West",
          "name":"West",
          "name_2":"",
          "primary":true
       }
    ],
    "salutation":"",
    "first_name":"Gus",
```

```
"last_name":"Dales",
"full_name":"Gus Dales",
"title":"Director Operations",
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(661) 120-2292",
"email":[
   {
      "email_address":"section.sugar.section@example.it",
      "opt_out":"1",
      "invalid_email":"0",
      "primary_address":"0"
   },
   {
      "email_address":"support.qa.kid@example.co.uk",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"1"
   }
],
"phone_mobile":"(294) 447-9707",
"phone_work":"(036) 840-3216",
"phone_other":"",
"phone_fax":"",
"email1":"support.qa.kid@example.co.uk",
"email2":"section.sugar.section@example.it",
"invalid_email":false,
"email_opt_out":false,
"primary_address_street":"48920 San Carlos Ave",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Persistance",
"primary_address_state":"CA",
"primary_address_postalcode":"54556",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
```

```
      "assistant":"",
      "assistant_phone":"",
      "picture":"",
      "email_and_name1":"",
      "lead_source":"Support Portal User Registration",
      "account_name":"Arts & Crafts Inc",
      "account_id":"d43243c6-9b8e-2973-aee2-512d09bc34b4",
      "opportunity_role_fields":"",
      "opportunity_role_id":"",
      "opportunity_role":"",
      "reports_to_id":"",
      "report_to_name":"",
      "portal_name":"GusDales145",
      "portal_active":true,
      "portal_password":"$1$JxYr6tmM$b.O6.KF42jP46RadSwz0N0",
      "portal_password1":"",
      "portal_app":"",
      "preferred_language":"en_us",
      "campaign_id":"",
      "campaign_name":"",
      "c_accept_status_fields":"",
      "m_accept_status_fields":"",
      "accept_status_id":"",
      "accept_status_name":"",
      "sync_contact":"",
      "my_favorite":false,
      "_acl":{
         "fields":{

         }
      }
   }
}
```


Change Log

Version	Change
v10	Added /<module>/:record/link/:link_name/:remote_id DELETE endpoint.

//:record/link/:link_name/:remote_id GET

Overview

Retrieves a related record with relationship role information.

Request Arguments

This endpoint does not accept any arguments.

Response Arguments

Name	Type	Description
<record field>	<record field type>	Returns the fields for the newly created record.

Response


```
{
   "id":"e689173e-c953-1e14-c215-512d0927e7a2",
   "name":"Gus Dales",
   "date_entered":"2013-02-26T19:12:00+00:00",
   "date_modified":"2013-02-26T19:12:00+00:00",
   "modified_user_id":"1",
   "modified_by_name":"Administrator",
   "created_by":"1",
   "created_by_name":"Administrator",
   "description":"",
   "img":"",
   "deleted":false,
   "assigned_user_id":"seed_sally_id",
   "assigned_user_name":"Sally Bronsen",
   "team_name":[
      {
         "id":"West",
         "name":"West",
         "name_2":"",
         "primary":true
      }
   ],
   "salutation":"",
   "first_name":"Gus",
   "last_name":"Dales",
   "full_name":"Gus Dales",
   "title":"Director Operations",
   "linkedin":"",
   "facebook":"",
```

```
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(661) 120-2292",
"email":[
   {
      "email_address":"section.sugar.section@example.it",
      "opt_out":"1",
      "invalid_email":"0",
      "primary_address":"0"
   },
   {
      "email_address":"support.qa.kid@example.co.uk",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"1"
   }
],
"phone_mobile":"(294) 447-9707",
"phone_work":"(036) 840-3216",
"phone_other":"",
"phone_fax":"",
"email1":"support.qa.kid@example.co.uk",
"email2":"section.sugar.section@example.it",
"invalid_email":false,
"email_opt_out":false,
"primary_address_street":"48920 San Carlos Ave",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Persistance",
"primary_address_state":"CA",
"primary_address_postalcode":"54556",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
"assistant":"",
"assistant_phone":"",
"picture":"",
"email_and_name1":"",
"lead_source":"Support Portal User Registration",
```

```
   "account_name":"Arts & Crafts Inc",
   "account_id":"d43243c6-9b8e-2973-aee2-512d09bc34b4",
   "opportunity_role_fields":"",
   "opportunity_role_id":"",
   "opportunity_role":"Technical Advisor",
   "reports_to_id":"",
   "report_to_name":"",
   "portal_name":"GusDales145",
   "portal_active":true,
   "portal_password":"$1$JxYr6tmM$b.O6.KF42jP46RadSwz0N0",
   "portal_password1":"",
   "portal_app":"",
   "preferred_language":"en_us",
   "campaign_id":"",
   "campaign_name":"",
   "c_accept_status_fields":"",
   "m_accept_status_fields":"",
   "accept_status_id":"",
   "accept_status_name":"",
   "sync_contact":"",
   "my_favorite":false,
   "_acl":{
      "fields":{

      }
   }
}
```


Change Log

Version	Change
v10	Added /<module>/:record/link/:link_name/:remote_id GET endpoint.

//:record/link/:link_name/:remote_id POST

Overview

Creates a relationship to a pre-existing record.

Query String Parameters

This endpoint does not accept any arguments.

Response Arguments

Name	Type	Description
record	Array	The record linked to the related record.
related_record	Array	The record that was associated.

Response


```
{
   "record":{
      "id":"da6a3741-2a81-ba7f-f249-512d0932e94e",
      "name":"Slender Broadband Inc - 1000 units",
      "date_entered":"2013-02-26T19:12:00+00:00",
      "date_modified":"2013-02-26T19:12:00+00:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "description":"",
      "img":"",
      "last_activity_date":"2013-02-28T18:21:00+00:00",
      "deleted":false,
      "assigned_user_id":"seed_max_id",
      "assigned_user_name":"Max Jensen",
      "team_name":[
         {
            "id":"East",
            "name":"East",
            "name_2":"",
            "primary":false
         },
         {
            "id":"West",
            "name":"West",
            "name_2":"",
            "primary":true
         }
      ],
      "opportunity_type":"",
      "account_name":"Slender Broadband Inc",
      "account_id":"181461c6-dc81-1115-1fe0-512d092e8f15",
```

```
    "campaign_id":"",
    "campaign_name":"",
    "lead_source":"Campaign",
    "amount":"25000",
    "base_rate":"1",
    "amount_usdollar":"25000",
    "currency_id":"-99",
    "currency_name":"",
    "currency_symbol":"",
    "date_closed":"2013-02-27",
    "date_closed_timestamp":"1361992480",
    "next_step":"",
    "sales_stage":"Needs Analysis",
    "sales_status":"New",
    "probability":"90",
    "best_case":"25000",
    "worst_case":"25000",
    "commit_stage":"include",
    "my_favorite":false,
    "_acl":{
       "fields":{

       }
    }
},
"related_record":{
    "id":"e689173e-c953-1e14-c215-512d0927e7a2",
    "name":"Gus Dales",
    "date_entered":"2013-02-26T19:12:00+00:00",
    "date_modified":"2013-02-26T19:12:00+00:00",
    "modified_user_id":"1",
    "modified_by_name":"Administrator",
    "created_by":"1",
    "created_by_name":"Administrator",
    "description":"",
    "img":"",
    "deleted":false,
    "assigned_user_id":"seed_sally_id",
    "assigned_user_name":"Sally Bronsen",
    "team_name":[
       {
          "id":"West",
          "name":"West",
          "name_2":"",
          "primary":true
       }
```

```
],
"salutation":"",
"first_name":"Gus",
"last_name":"Dales",
"full_name":"Gus Dales",
"title":"Director Operations",
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(661) 120-2292",
"email":[
   {
      "email_address":"section.sugar.section@example.it",
      "opt_out":"1",
      "invalid_email":"0",
      "primary_address":"0"
   },
   {
      "email_address":"support.qa.kid@example.co.uk",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"1"
   }
],
"phone_mobile":"(294) 447-9707",
"phone_work":"(036) 840-3216",
"phone_other":"",
"phone_fax":"",
"email1":"support.qa.kid@example.co.uk",
"email2":"section.sugar.section@example.it",
"invalid_email":false,
"email_opt_out":false,
"primary_address_street":"48920 San Carlos Ave",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Persistance",
"primary_address_state":"CA",
"primary_address_postalcode":"54556",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
```

```
      "alt_address_state":"",
      "alt_address_postalcode":"",
      "alt_address_country":"",
      "assistant":"",
      "assistant_phone":"",
      "picture":"",
      "email_and_name1":"",
      "lead_source":"Support Portal User Registration",
      "account_name":"Arts & Crafts Inc",
      "account_id":"d43243c6-9b8e-2973-aee2-512d09bc34b4",
      "opportunity_role_fields":"",
      "opportunity_role_id":"",
      "opportunity_role":"Technical Advisor",
      "reports_to_id":"",
      "report_to_name":"",
      "portal_name":"GusDales145",
      "portal_active":true,
      "portal_password":"$1$JxYr6tmM$b.O6.KF42jP46RadSwz0N0",
      "portal_password1":"",
      "portal_app":"",
      "preferred_language":"en_us",
      "campaign_id":"",
      "campaign_name":"",
      "c_accept_status_fields":"",
      "m_accept_status_fields":"",
      "accept_status_id":"",
      "accept_status_name":"",
      "sync_contact":"",
      "my_favorite":false,
      "_acl":{
         "fields":{

         }
      }
   }
}
```


Change Log

Version	Change
v10	Added /<module>/:record/link/:link_name/:remote_id POST endpoint.

//:record/link/:link_name/:remote_id PUT

Overview

Updates relationship specific information on an existing relationship.

Request Arguments

Name	Type	Description	Required
<relationship field>	<relationship field type>	The name value list of relationship fields to populate. An example is the contact_role field on Opportunities and Contacts.	True

Request

```
{
    "contact_role":"Primary Decision Maker"
}
```

Response Arguments

Name	Type	Description
record	Array	The record linked to the related record.
related_record	Array	The record that was associated.

Response

```
{
   "record":{
      "id":"da6a3741-2a81-ba7f-f249-512d0932e94e",
      "name":"Slender Broadband Inc - 1000 units",
      "date_entered":"2013-02-26T19:12:00+00:00",
      "date_modified":"2013-02-26T19:12:00+00:00",
      "modified_user_id":"1",
```

```
"modified_by_name":"Administrator",
"created_by":"1",
"created_by_name":"Administrator",
"description":"",
"img":"",
"deleted":false,
"assigned_user_id":"seed_max_id",
"assigned_user_name":"Max Jensen",
"team_name":[
   {
      "id":"East",
      "name":"East",
      "name_2":"",
      "primary":false
   },
   {
      "id":"West",
      "name":"West",
      "name_2":"",
      "primary":true
   }
],
"opportunity_type":"",
"account_name":"Slender Broadband Inc",
"account_id":"181461c6-dc81-1115-1fe0-512d092e8f15",
"campaign_id":"",
"campaign_name":"",
"lead_source":"Campaign",
"amount":"25000",
"base_rate":"1",
"amount_usdollar":"25000",
"currency_id":"-99",
"currency_name":"",
"currency_symbol":"",
"date_closed":"2013-02-27",
"date_closed_timestamp":"1361992480",
"next_step":"",
"sales_stage":"Needs Analysis",
"sales_status":"New",
"probability":"90",
"best_case":"25000",
"worst_case":"25000",
"commit_stage":"include",
"my_favorite":false,
"_acl":{
   "fields":{
```

```
      }
   }
},
"related_record":{
   "id":"e1c495cb-af17-1b37-dd66-512f934fe155",
   "name":"Bill Edwards",
   "date_entered":"2013-02-28T17:25:00+00:00",
   "date_modified":"2013-02-28T17:25:00+00:00",
   "modified_user_id":"1",
   "modified_by_name":"Administrator",
   "created_by":"1",
   "created_by_name":"Administrator",
   "description":"",
   "img":"",
   "deleted":false,
   "assigned_user_id":"",
   "assigned_user_name":"",
   "team_name":[
      {
         "id":1,
         "name":"Global",
         "name_2":"",
         "primary":true
      }
   ],
   "salutation":"",
   "first_name":"Bill",
   "last_name":"Edwards",
   "full_name":"Bill Edwards",
   "title":"",
   "linkedin":"",
   "facebook":"",
   "twitter":"",
   "googleplus":"",
   "department":"",
   "do_not_call":false,
   "phone_home":"",
   "email":[

   ],
   "phone_mobile":"",
   "phone_work":"",
   "phone_other":"",
   "phone_fax":"",
   "email1":"",
```

```
"email2":"",
"invalid_email":"",
"email_opt_out":"",
"primary_address_street":"",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"",
"primary_address_state":"",
"primary_address_postalcode":"",
"primary_address_country":"",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
"assistant":"",
"assistant_phone":"",
"picture":"",
"email_and_name1":"",
"lead_source":"",
"account_id":"",
"opportunity_role_fields":"",
"opportunity_role_id":"",
"opportunity_role":"Primary Decision Maker",
"reports_to_id":"",
"report_to_name":"",
"portal_name":"",
"portal_active":false,
"portal_password":"",
"portal_password1":"",
"portal_app":"",
"preferred_language":"en_us",
"campaign_id":"",
"campaign_name":"",
"c_accept_status_fields":"",
"m_accept_status_fields":"",
"accept_status_id":"",
"accept_status_name":"",
"sync_contact":"",
"my_favorite":false,
"_acl":{
   "fields":{

   }
```

```
        }
    }
}
```

Change Log

Version	Change
v10	Added /<module>/:record/link/:link_name/:remote_id PUT endpoint.

//:record/link/:link_name/add_record_list/:remote_id POST

Overview

Creates relationships to pre-existing record from a record list.

URL Arguments

Name	Type	Description	Required
<record ID>	<string>	Target record ID.	True.
<report ID>	<string>	Report ID for Saved Report.	True.
<relationship link>	<string>	Link between targeted and related records.	True.

Request

Response Arguments

Name	Type	Description
record	Array	The record linked to related records.
related_records	Array	Record IDs that were associated.

Response


```
    "record": {
        "id": "da6a3741-2a81-ba7f-f249-512d0932e94e",
        "name": "Slender Broadband Inc - 1000 units",
        "date_entered": "2013-02-26T19:12:00+00:00",
        "date_modified": "2013-02-26T19:12:00+00:00",
        "modified_user_id": "1",
        "modified_by_name": "Administrator",
        "created_by": "1",
        "created_by_name": "Administrator",
        "description": "",
        "img": "",
        "last_activity_date": "2013-02-28T18:21:00+00:00",
        "deleted": false,
        "assigned_user_id": "seed_max_id",
        "assigned_user_name": "Max Jensen",
        "team_name": [
            {
                "id": "East",
                "name": "East",
                "name_2": "",
                "primary": false
            },
            {
                "id": "West",
                "name": "West",
                "name_2": "",
                "primary": true
            }
        ],
        "opportunity_type": "",
        "account_name": "Slender Broadband Inc",
        "account_id": "181461c6-dc81-1115-1fe0-512d092e8f15",
        "campaign_id": "",
        "campaign_name": "",
        "lead_source": "Campaign",
        "amount": "25000",
        "base_rate": "1",
        "amount_usdollar": "25000",
        "currency_id": "-99",
        "currency_name": "",
        "currency_symbol": "",
        "date_closed": "2013-02-27",
        "date_closed_timestamp": "1361992480",
```

```
        "next_step": "",
        "sales_stage": "Needs Analysis",
        "sales_status": "New",
        "probability": "90",
        "best_case": "25000",
        "worst_case": "25000",
        "commit_stage": "include",
        "my_favorite": false,
        "_acl": {
            "fields": {

            }
        }
    },
    "related_records": [
        success: [
            "e689173e-c953-1e14-c215-512d0927e7a2",
            "da6a3741-2a81-ba7f-f249-512d0932e94e",
            "181461c6-dc81-1115-1fe0-512d092e8f15"
        ],
        error: []
    ]
}
```


Change Log

Version	Change
v10 (7.2.0)	Added /<module>/:record/link/:link_name/add_record_list/:remote_id POST endpoint.

//:record/link/:link_name/count GET

Overview

Lists related filtered records.

Summary

This endpoint will return a set of related records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Care will need to be taken to make sure this filter has appropriate indexes on the

server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

Name	Type	Description	Required
filter	String	The filter expression. Filter expressions are explained below.	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
fields	String	Comma delimited list of fields to return. The field date_modified will always be returned. This argument can be combined with the view argument. Example: name,account_type,description	False
view	String	Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be	False

Name	Type	Description	Required
		used in combination with the fields argument. Common views are "record" and "list". Example: record	
order_by	String	How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon. Example: name:DESC,account_type:DESC,date_modified:ASC	False

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
   "filter":[
      {
         "name":"Nelson Inc"
      }
   ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name"

starts with the value "Nelson".

Example

```
{
   "filter":[
      {
         "name":{
            "$starts":"Nelson"
         }
      }
   ]
}
```

Below is a list of operation types:

Operation	Description
$equals	Performs an exact match on that field.
$not_equals	Performs an exact match on that field.
$not_equals	Matches on non-matching values.
$starts	Matches on anything that starts with the value.
$ends	Matches anything that ends with the value.
$contains	Matches anything that contains the value
$in	Finds anything where field matches one of the values as specified as an array.
$not_in	Finds anything where field does not matches any of the values as specified as an array.
$is_null	Checks if the field is null. This operation does not need a value

Operation		Description	
			specified.
	$not_null		Checks if the field is not null. This operation does not need a value specified.
	$lt		Matches when the field is less than the value.
	$lte		Matches when the field is less than or equal to the value.
	$gt		Matches when the field is greater than the value.
	$gte		Matches when the field is greater than or equal to the value.

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "$and" and "$or".

Example

```
{
   "filter":[
      {
         "$or":[
            {
               "name":"Nelson Inc"
            },
            {
               "name":"Nelson LLC"
            }
         ]
      }
   ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "$favorite" and "$owner".

Example

```
{
   "filter":[
      {
         "$favorite":"_this"
      }
   ]
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Repsonse

```
{
   "next_offset":-1,
   "records":[
      {
         "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
```

```
"name":"Dale Spivey",
"date_entered":"2013-02-26T19:12:00+00:00",
"date_modified":"2013-02-28T05:03:00+00:00",
"modified_user_id":"1",
"modified_by_name":"Administrator",
"created_by":"1",
"created_by_name":"Administrator",
"description":"",
"img":"",
"deleted":false,
"assigned_user_id":"seed_sally_id",
"assigned_user_name":"Sally Bronsen",
"team_name":[
   {
      "id":"East",
      "name":"East",
      "name_2":"",
      "primary":false
   },
   {
      "id":1,
      "name":"Global",
      "name_2":"",
      "primary":false
   },
   {
      "id":"West",
      "name":"West",
      "name_2":"",
      "primary":true
   }
],
"salutation":"",
"first_name":"Dale",
"last_name":"Spivey",
"full_name":"Dale Spivey",
"title":"VP Operations",
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(523) 825-4311",
"email":[
   {
```

```
            "email_address":"sugar.dev.sugar@example.co.jp",
            "opt_out":"0",
            "invalid_email":"0",
            "primary_address":"1"
        },
        {
            "email_address":"the.support@example.biz",
            "opt_out":"0",
            "invalid_email":"0",
            "primary_address":"0"
        }
    ],
    "phone_mobile":"(373) 861-0757",
    "phone_work":"(212) 542-9596",
    "phone_other":"",
    "phone_fax":"",
    "email1":"sugar.dev.sugar@example.co.jp",
    "email2":"the.support@example.biz",
    "invalid_email":false,
    "email_opt_out":false,
    "primary_address_street":"345 Sugar Blvd.",
    "primary_address_street_2":"",
    "primary_address_street_3":"",
    "primary_address_city":"Denver",
    "primary_address_state":"CA",
    "primary_address_postalcode":"87261",
    "primary_address_country":"USA",
    "alt_address_street":"",
    "alt_address_street_2":"",
    "alt_address_street_3":"",
    "alt_address_city":"",
    "alt_address_state":"",
    "alt_address_postalcode":"",
    "alt_address_country":"",
    "assistant":"",
    "assistant_phone":"",
    "picture":"",
    "email_and_name1":"",
    "lead_source":"Campaign",
    "account_name":"Smallville Resources Inc",
    "account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",
    "opportunity_role_fields":"",
    "opportunity_role_id":"",
    "opportunity_role":"",
    "reports_to_id":"",
    "report_to_name":"",
```

```
    "portal_name":"DaleSpivey97",
    "portal_active":true,
    "portal_password":"$1$yKMAONHM$Y5S.8CY.WZCZCwfGD1a1Q\/",
    "portal_password1":"",
    "portal_app":"",
    "preferred_language":"en_us",
    "campaign_id":"",
    "campaign_name":"",
    "c_accept_status_fields":"",
    "m_accept_status_fields":"",
    "accept_status_id":"",
    "accept_status_name":"",
    "sync_contact":"",
    "my_favorite":false,
    "_acl":{
       "fields":{

       }
    }
},
{
    "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
    "name":"Florence Haddock",
    "date_entered":"2013-02-26T19:12:00+00:00",
    "date_modified":"2013-02-26T19:12:00+00:00",
    "modified_user_id":"1",
    "modified_by_name":"Administrator",
    "created_by":"1",
    "created_by_name":"Administrator",
    "description":"",
    "img":"",
    "deleted":false,
    "assigned_user_id":"seed_sally_id",
    "assigned_user_name":"Sally Bronsen",
    "team_name":[
       {
          "id":"East",
          "name":"East",
          "name_2":"",
          "primary":false
       },
       {
          "id":1,
          "name":"Global",
          "name_2":"",
          "primary":false
```

```
      },
      {
         "id":"West",
         "name":"West",
         "name_2":"",
         "primary":true
      }
   ],
   "salutation":"",
   "first_name":"Florence",
   "last_name":"Haddock",
   "full_name":"Florence Haddock",
   "title":"Director Sales",
   "linkedin":"",
   "facebook":"",
   "twitter":"",
   "googleplus":"",
   "department":"",
   "do_not_call":false,
   "phone_home":"(729) 845-3137",
   "email":[
      {
         "email_address":"dev.vegan@example.de",
         "opt_out":"1",
         "invalid_email":"0",
         "primary_address":"0"
      },
      {
         "email_address":"section71@example.it",
         "opt_out":"0",
         "invalid_email":"0",
         "primary_address":"1"
      }
   ],
   "phone_mobile":"(246) 233-1382",
   "phone_work":"(565) 696-6981",
   "phone_other":"",
   "phone_fax":"",
   "email1":"section71@example.it",
   "email2":"dev.vegan@example.de",
   "invalid_email":false,
   "email_opt_out":false,
   "primary_address_street":"111 Silicon Valley Road",
   "primary_address_street_2":"",
   "primary_address_street_3":"",
   "primary_address_city":"Denver",
```

```
            "primary_address_state":"CA",
            "primary_address_postalcode":"79900",
            "primary_address_country":"USA",
            "alt_address_street":"",
            "alt_address_street_2":"",
            "alt_address_street_3":"",
            "alt_address_city":"",
            "alt_address_state":"",
            "alt_address_postalcode":"",
            "alt_address_country":"",
            "assistant":"",
            "assistant_phone":"",
            "picture":"",
            "email_and_name1":"",
            "lead_source":"Support Portal User Registration",
            "account_name":"Smallville Resources Inc",
            "account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",
            "opportunity_role_fields":"",
            "opportunity_role_id":"",
            "opportunity_role":"",
            "reports_to_id":"",
            "report_to_name":"",
            "portal_name":"FlorenceHaddock169",
            "portal_active":true,
            "portal_password":"$1$nWFhTbK6$JF9BCGSqL\/NCrbhueX5ia0",
            "portal_password1":"",
            "portal_app":"",
            "preferred_language":"en_us",
            "campaign_id":"",
            "campaign_name":"",
            "c_accept_status_fields":"",
            "m_accept_status_fields":"",
            "accept_status_id":"",
            "accept_status_name":"",
            "sync_contact":"",
            "my_favorite":false,
            "_acl":{
               "fields":{

               }
            }
         }
      ]
}
```

Change Log

Version	Change
v10	Added /<module>/:record/link/:link_name/filter GET endpoint.

//:record/link/:link_name/filter GET

Overview

Lists related filtered records.

Summary

This endpoint will return a set of related records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Care will need to be taken to make sure this filter has appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

Name	Type	Description	Required
filter	String	The filter expression. Filter expressions are explained below.	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
fields	String	Comma delimited list of fields to return. The field date_modified will	False

Name	Type	Description	Required
		always be returned. This argument can be combined with the view argument. Example: name,account_type,description	
view	String	Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list". Example: record	False
order_by	String	How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon. Example: name:DESC,account_type:DESC,date_modified:ASC	False

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
   "filter":[
      {
         "name":"Nelson Inc"
      }
   ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
   "filter":[
      {
         "name":{
            "$starts":"Nelson"
         }
      }
   ]
}
```

Below is a list of operation types:

Operation	Description
$equals	Performs an exact match on that field.
$not_equals	Performs an exact match on that field.
$not_equals	Matches on non-matching values.
$starts	Matches on anything that starts with the value.

Operation		Description
	$ends	Matches anything that ends with the value.
	$contains	Matches anything that contains the value
	$in	Finds anything where field matches one of the values as specified as an array.
	$not_in	Finds anything where field does not matches any of the values as specified as an array.
	$is_null	Checks if the field is null. This operation does not need a value specified.
	$not_null	Checks if the field is not null. This operation does not need a value specified.
	$lt	Matches when the field is less than the value.
	$lte	Matches when the field is less than or equal to the value.
	$gt	Matches when the field is greater than the value.
	$gte	Matches when the field is greater than or equal to the value.

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all

expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "$and" and "$or".

Example

```
{
   "filter":[
      {
         "$or":[
            {
               "name":"Nelson Inc"
            },
            {
               "name":"Nelson LLC"
            }
         ]
      }
   ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "$favorite" and "$owner".

Example

```
{
   "filter":[
      {
         "$favorite":"_this"
      }
   ]
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Repsonse


```
{
   "next_offset":-1,
   "records":[
      {
         "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
         "name":"Dale Spivey",
         "date_entered":"2013-02-26T19:12:00+00:00",
         "date_modified":"2013-02-28T05:03:00+00:00",
         "modified_user_id":"1",
         "modified_by_name":"Administrator",
         "created_by":"1",
         "created_by_name":"Administrator",
         "description":"",
         "img":"",
         "deleted":false,
         "assigned_user_id":"seed_sally_id",
         "assigned_user_name":"Sally Bronsen",
         "team_name":[
            {
               "id":"East",
               "name":"East",
               "name_2":"",
               "primary":false
            },
            {
               "id":1,
               "name":"Global",
               "name_2":"",
               "primary":false
            },
            {
               "id":"West",
               "name":"West",
```

```
      "name_2":"",
      "primary":true
   }
],
"salutation":"",
"first_name":"Dale",
"last_name":"Spivey",
"full_name":"Dale Spivey",
"title":"VP Operations",
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(523) 825-4311",
"email":[
   {
      "email_address":"sugar.dev.sugar@example.co.jp",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"1"
   },
   {
      "email_address":"the.support@example.biz",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"0"
   }
],
"phone_mobile":"(373) 861-0757",
"phone_work":"(212) 542-9596",
"phone_other":"",
"phone_fax":"",
"email1":"sugar.dev.sugar@example.co.jp",
"email2":"the.support@example.biz",
"invalid_email":false,
"email_opt_out":false,
"primary_address_street":"345 Sugar Blvd.",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Denver",
"primary_address_state":"CA",
"primary_address_postalcode":"87261",
"primary_address_country":"USA",
"alt_address_street":"",
```

```
      "alt_address_street_2":"",
      "alt_address_street_3":"",
      "alt_address_city":"",
      "alt_address_state":"",
      "alt_address_postalcode":"",
      "alt_address_country":"",
      "assistant":"",
      "assistant_phone":"",
      "picture":"",
      "email_and_name1":"",
      "lead_source":"Campaign",
      "account_name":"Smallville Resources Inc",
      "account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",
      "opportunity_role_fields":"",
      "opportunity_role_id":"",
      "opportunity_role":"",
      "reports_to_id":"",
      "report_to_name":"",
      "portal_name":"DaleSpivey97",
      "portal_active":true,
      "portal_password":"$1$yKMAONHM$Y5S.8CY.WZCZCwfGD1a1Q\/",
      "portal_password1":"",
      "portal_app":"",
      "preferred_language":"en_us",
      "campaign_id":"",
      "campaign_name":"",
      "c_accept_status_fields":"",
      "m_accept_status_fields":"",
      "accept_status_id":"",
      "accept_status_name":"",
      "sync_contact":"",
      "my_favorite":false,
      "_acl":{
         "fields":{

         }
      }
   },
   {
      "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
      "name":"Florence Haddock",
      "date_entered":"2013-02-26T19:12:00+00:00",
      "date_modified":"2013-02-26T19:12:00+00:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
```

```
"created_by_name":"Administrator",
"description":"",
"img":"",
"deleted":false,
"assigned_user_id":"seed_sally_id",
"assigned_user_name":"Sally Bronsen",
"team_name":[
   {
      "id":"East",
      "name":"East",
      "name_2":"",
      "primary":false
   },
   {
      "id":1,
      "name":"Global",
      "name_2":"",
      "primary":false
   },
   {
      "id":"West",
      "name":"West",
      "name_2":"",
      "primary":true
   }
],
"salutation":"",
"first_name":"Florence",
"last_name":"Haddock",
"full_name":"Florence Haddock",
"title":"Director Sales",
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(729) 845-3137",
"email":[
   {
      "email_address":"dev.vegan@example.de",
      "opt_out":"1",
      "invalid_email":"0",
      "primary_address":"0"
   },
   {
```

```
      "email_address":"section71@example.it",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"1"
   }
],
"phone_mobile":"(246) 233-1382",
"phone_work":"(565) 696-6981",
"phone_other":"",
"phone_fax":"",
"email1":"section71@example.it",
"email2":"dev.vegan@example.de",
"invalid_email":false,
"email_opt_out":false,
"primary_address_street":"111 Silicon Valley Road",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Denver",
"primary_address_state":"CA",
"primary_address_postalcode":"79900",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
"assistant":"",
"assistant_phone":"",
"picture":"",
"email_and_name1":"",
"lead_source":"Support Portal User Registration",
"account_name":"Smallville Resources Inc",
"account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",
"opportunity_role_fields":"",
"opportunity_role_id":"",
"opportunity_role":"",
"reports_to_id":"",
"report_to_name":"",
"portal_name":"FlorenceHaddock169",
"portal_active":true,
"portal_password":"$1$nWFhTbK6$JF9BCGSqL\/NCrbhueX5ia0",
"portal_password1":"",
"portal_app":"",
"preferred_language":"en_us",
```

```
         "campaign_id":"",
         "campaign_name":"",
         "c_accept_status_fields":"",
         "m_accept_status_fields":"",
         "accept_status_id":"",
         "accept_status_name":"",
         "sync_contact":"",
         "my_favorite":false,
         "_acl":{
            "fields":{

            }
         }
      }
   ]
}
```

Change Log

Version	Change
v10	Added /<module>/:record/link/:link_name/filter GET endpoint.

//:record/link/:link_name/filter/count GET

Overview

Lists related filtered records.

Summary

This endpoint will return a set of related records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Care will need to be taken to make sure this filter has appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

Name	Type	Description	Required
filter	String	The filter expression. Filter expressions are explained below.	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
fields	String	Comma delimited list of fields to return. The field date_modified will always be returned. This argument can be combined with the view argument. Example: name,account_type,description	False
view	String	Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list". Example: record	False

Name	Type	Description	Required
order_by	String	How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon. Example: name:DESC,account_type:DESC,date_modified:ASC	False

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
   "filter":[
      {
         "name":"Nelson Inc"
      }
   ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
   "filter":[
      {
```

```
         "name":{
            "$starts":"Nelson"
         }
      }
   ]
}
```

Below is a list of operation types:

Operation	Description
$equals	Performs an exact match on that field.
$not_equals	Performs an exact match on that field.
$not_equals	Matches on non-matching values.
$starts	Matches on anything that starts with the value.
$ends	Matches anything that ends with the value.
$contains	Matches anything that contains the value
$in	Finds anything where field matches one of the values as specified as an array.
$not_in	Finds anything where field does not matches any of the values as specified as an array.
$is_null	Checks if the field is null. This operation does not need a value specified.
$not_null	Checks if the field is not null. This operation does not need a value specified.

Operation		Description
	$lt	Matches when the field is less than the value.
	$lte	Matches when the field is less than or equal to the value.
	$gt	Matches when the field is greater than the value.
	$gte	Matches when the field is greater than or equal to the value.

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "$and" and "$or".

Example

```
{
   "filter":[
      {
         "$or":[
            {
               "name":"Nelson Inc"
            },
            {
               "name":"Nelson LLC"
            }
         ]
      }
   ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "$favorite" and "$owner".

Example

```
{
   "filter":[
      {
         "$favorite":"_this"
      }
   ]
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Repsonse

```
{
   "next_offset":-1,
   "records":[
      {
         "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
         "name":"Dale Spivey",
         "date_entered":"2013-02-26T19:12:00+00:00",
         "date_modified":"2013-02-28T05:03:00+00:00",
         "modified_user_id":"1",
         "modified_by_name":"Administrator",
         "created_by":"1",
         "created_by_name":"Administrator",
```

```
"description":"",
"img":"",
"deleted":false,
"assigned_user_id":"seed_sally_id",
"assigned_user_name":"Sally Bronsen",
"team_name":[
   {
      "id":"East",
      "name":"East",
      "name_2":"",
      "primary":false
   },
   {
      "id":1,
      "name":"Global",
      "name_2":"",
      "primary":false
   },
   {
      "id":"West",
      "name":"West",
      "name_2":"",
      "primary":true
   }
],
"salutation":"",
"first_name":"Dale",
"last_name":"Spivey",
"full_name":"Dale Spivey",
"title":"VP Operations",
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(523) 825-4311",
"email":[
   {
      "email_address":"sugar.dev.sugar@example.co.jp",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"1"
   },
   {
      "email_address":"the.support@example.biz",
```

```
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"0"
   }
],
"phone_mobile":"(373) 861-0757",
"phone_work":"(212) 542-9596",
"phone_other":"",
"phone_fax":"",
"email1":"sugar.dev.sugar@example.co.jp",
"email2":"the.support@example.biz",
"invalid_email":false,
"email_opt_out":false,
"primary_address_street":"345 Sugar Blvd.",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Denver",
"primary_address_state":"CA",
"primary_address_postalcode":"87261",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
"assistant":"",
"assistant_phone":"",
"picture":"",
"email_and_name1":"",
"lead_source":"Campaign",
"account_name":"Smallville Resources Inc",
"account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",
"opportunity_role_fields":"",
"opportunity_role_id":"",
"opportunity_role":"",
"reports_to_id":"",
"report_to_name":"",
"portal_name":"DaleSpivey97",
"portal_active":true,
"portal_password":"$1$yKMAONHM$Y5S.8CY.WZCZCwfGD1a1Q\/",
"portal_password1":"",
"portal_app":"",
"preferred_language":"en_us",
"campaign_id":"",
```

```
      "campaign_name":"",
      "c_accept_status_fields":"",
      "m_accept_status_fields":"",
      "accept_status_id":"",
      "accept_status_name":"",
      "sync_contact":"",
      "my_favorite":false,
      "_acl":{
         "fields":{

         }
      }
   },
   {
      "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
      "name":"Florence Haddock",
      "date_entered":"2013-02-26T19:12:00+00:00",
      "date_modified":"2013-02-26T19:12:00+00:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "description":"",
      "img":"",
      "deleted":false,
      "assigned_user_id":"seed_sally_id",
      "assigned_user_name":"Sally Bronsen",
      "team_name":[
         {
            "id":"East",
            "name":"East",
            "name_2":"",
            "primary":false
         },
         {
            "id":1,
            "name":"Global",
            "name_2":"",
            "primary":false
         },
         {
            "id":"West",
            "name":"West",
            "name_2":"",
            "primary":true
         }
```

```
],
"salutation":"",
"first_name":"Florence",
"last_name":"Haddock",
"full_name":"Florence Haddock",
"title":"Director Sales",
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(729) 845-3137",
"email":[
   {
      "email_address":"dev.vegan@example.de",
      "opt_out":"1",
      "invalid_email":"0",
      "primary_address":"0"
   },
   {
      "email_address":"section71@example.it",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"1"
   }
],
"phone_mobile":"(246) 233-1382",
"phone_work":"(565) 696-6981",
"phone_other":"",
"phone_fax":"",
"email1":"section71@example.it",
"email2":"dev.vegan@example.de",
"invalid_email":false,
"email_opt_out":false,
"primary_address_street":"111 Silicon Valley Road",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Denver",
"primary_address_state":"CA",
"primary_address_postalcode":"79900",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
```

```
         "alt_address_state":"",
         "alt_address_postalcode":"",
         "alt_address_country":"",
         "assistant":"",
         "assistant_phone":"",
         "picture":"",
         "email_and_name1":"",
         "lead_source":"Support Portal User Registration",
         "account_name":"Smallville Resources Inc",
         "account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",
         "opportunity_role_fields":"",
         "opportunity_role_id":"",
         "opportunity_role":"",
         "reports_to_id":"",
         "report_to_name":"",
         "portal_name":"FlorenceHaddock169",
         "portal_active":true,
         "portal_password":"$1$nWFhTbK6$JF9BCGSqL\/NCrbhueX5ia0",
         "portal_password1":"",
         "portal_app":"",
         "preferred_language":"en_us",
         "campaign_id":"",
         "campaign_name":"",
         "c_accept_status_fields":"",
         "m_accept_status_fields":"",
         "accept_status_id":"",
         "accept_status_name":"",
         "sync_contact":"",
         "my_favorite":false,
         "_acl":{
            "fields":{

            }
         }
      }
   ]
}
```


Change Log

Version	Change
v10	Added

	/<module>/:record/link/:link_name/filter GET endpoint.

//:record/link/:link_name/filter/leancount GET

Overview

Lists related filtered records.

Summary

This endpoint will return a set of related records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Care will need to be taken to make sure this filter has appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

Name	Type	Description	Required
filter	String	The filter expression. Filter expressions are explained below.	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
fields	String	Comma delimited list of fields to return. The field date_modified will always be returned. This argument can be combined with the view argument. Example: name,acc	False

Name	Type	Description	Required
		ount_type,descripti on	
view	String	Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list". Example: record	False
order_by	String	How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon. Example: name:DE SC,account_type:D ESC,date_modified: ASC	False

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
   "filter":[
      {
         "name":"Nelson Inc"
      }
   ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
   "filter":[
      {
         "name":{
            "$starts":"Nelson"
         }
      }
   ]
}
```

Below is a list of operation types:

Operation	Description
$equals	Performs an exact match on that field.
$not_equals	Performs an exact match on that field.
$not_equals	Matches on non-matching values.
$starts	Matches on anything that starts with the value.
$ends	Matches anything that ends with the value.
$contains	Matches anything

Operation		Description
		that contains the value
	$in	Finds anything where field matches one of the values as specified as an array.
	$not_in	Finds anything where field does not matches any of the values as specified as an array.
	$is_null	Checks if the field is null. This operation does not need a value specified.
	$not_null	Checks if the field is not null. This operation does not need a value specified.
	$lt	Matches when the field is less than the value.
	$lte	Matches when the field is less than or equal to the value.
	$gt	Matches when the field is greater than the value.
	$gte	Matches when the field is greater than or equal to the value.

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "$and" and "$or".

Example

```
{
   "filter":[
      {
         "$or":[
            {
               "name":"Nelson Inc"
            },
            {
               "name":"Nelson LLC"
            }
         ]
      }
   ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "$favorite" and "$owner".

Example

```
{
   "filter":[
      {
         "$favorite":"_this"
      }
   ]
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional

Name	Type	Description
		results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Repsonse


```
{
   "next_offset":-1,
   "records":[
      {
         "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
         "name":"Dale Spivey",
         "date_entered":"2013-02-26T19:12:00+00:00",
         "date_modified":"2013-02-28T05:03:00+00:00",
         "modified_user_id":"1",
         "modified_by_name":"Administrator",
         "created_by":"1",
         "created_by_name":"Administrator",
         "description":"",
         "img":"",
         "deleted":false,
         "assigned_user_id":"seed_sally_id",
         "assigned_user_name":"Sally Bronsen",
         "team_name":[
            {
               "id":"East",
               "name":"East",
               "name_2":"",
               "primary":false
            },
            {
               "id":1,
               "name":"Global",
               "name_2":"",
               "primary":false
            },
            {
               "id":"West",
               "name":"West",
               "name_2":"",
               "primary":true
```

```
      }
   ],
   "salutation":"",
   "first_name":"Dale",
   "last_name":"Spivey",
   "full_name":"Dale Spivey",
   "title":"VP Operations",
   "linkedin":"",
   "facebook":"",
   "twitter":"",
   "googleplus":"",
   "department":"",
   "do_not_call":false,
   "phone_home":"(523) 825-4311",
   "email":[
      {
         "email_address":"sugar.dev.sugar@example.co.jp",
         "opt_out":"0",
         "invalid_email":"0",
         "primary_address":"1"
      },
      {
         "email_address":"the.support@example.biz",
         "opt_out":"0",
         "invalid_email":"0",
         "primary_address":"0"
      }
   ],
   "phone_mobile":"(373) 861-0757",
   "phone_work":"(212) 542-9596",
   "phone_other":"",
   "phone_fax":"",
   "email1":"sugar.dev.sugar@example.co.jp",
   "email2":"the.support@example.biz",
   "invalid_email":false,
   "email_opt_out":false,
   "primary_address_street":"345 Sugar Blvd.",
   "primary_address_street_2":"",
   "primary_address_street_3":"",
   "primary_address_city":"Denver",
   "primary_address_state":"CA",
   "primary_address_postalcode":"87261",
   "primary_address_country":"USA",
   "alt_address_street":"",
   "alt_address_street_2":"",
   "alt_address_street_3":"",
```

```
      "alt_address_city":"",
      "alt_address_state":"",
      "alt_address_postalcode":"",
      "alt_address_country":"",
      "assistant":"",
      "assistant_phone":"",
      "picture":"",
      "email_and_name1":"",
      "lead_source":"Campaign",
      "account_name":"Smallville Resources Inc",
      "account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",
      "opportunity_role_fields":"",
      "opportunity_role_id":"",
      "opportunity_role":"",
      "reports_to_id":"",
      "report_to_name":"",
      "portal_name":"DaleSpivey97",
      "portal_active":true,
      "portal_password":"$1$yKMAONHM$Y5S.8CY.WZCZCwfGD1a1Q\/",
      "portal_password1":"",
      "portal_app":"",
      "preferred_language":"en_us",
      "campaign_id":"",
      "campaign_name":"",
      "c_accept_status_fields":"",
      "m_accept_status_fields":"",
      "accept_status_id":"",
      "accept_status_name":"",
      "sync_contact":"",
      "my_favorite":false,
      "_acl":{
         "fields":{

         }
      }
   },
   {
      "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
      "name":"Florence Haddock",
      "date_entered":"2013-02-26T19:12:00+00:00",
      "date_modified":"2013-02-26T19:12:00+00:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "description":"",
```

```
"img":"",
"deleted":false,
"assigned_user_id":"seed_sally_id",
"assigned_user_name":"Sally Bronsen",
"team_name":[
   {
      "id":"East",
      "name":"East",
      "name_2":"",
      "primary":false
   },
   {
      "id":1,
      "name":"Global",
      "name_2":"",
      "primary":false
   },
   {
      "id":"West",
      "name":"West",
      "name_2":"",
      "primary":true
   }
],
"salutation":"",
"first_name":"Florence",
"last_name":"Haddock",
"full_name":"Florence Haddock",
"title":"Director Sales",
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(729) 845-3137",
"email":[
   {
      "email_address":"dev.vegan@example.de",
      "opt_out":"1",
      "invalid_email":"0",
      "primary_address":"0"
   },
   {
      "email_address":"section71@example.it",
      "opt_out":"0",
```

```
      "invalid_email":"0",
      "primary_address":"1"
   }
],
"phone_mobile":"(246) 233-1382",
"phone_work":"(565) 696-6981",
"phone_other":"",
"phone_fax":"",
"email1":"section71@example.it",
"email2":"dev.vegan@example.de",
"invalid_email":false,
"email_opt_out":false,
"primary_address_street":"111 Silicon Valley Road",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Denver",
"primary_address_state":"CA",
"primary_address_postalcode":"79900",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
"assistant":"",
"assistant_phone":"",
"picture":"",
"email_and_name1":"",
"lead_source":"Support Portal User Registration",
"account_name":"Smallville Resources Inc",
"account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",
"opportunity_role_fields":"",
"opportunity_role_id":"",
"opportunity_role":"",
"reports_to_id":"",
"report_to_name":"",
"portal_name":"FlorenceHaddock169",
"portal_active":true,
"portal_password":"$1$nWFhTbK6$JF9BCGSqL\/NCrbhueX5ia0",
"portal_password1":"",
"portal_app":"",
"preferred_language":"en_us",
"campaign_id":"",
"campaign_name":"",
```

```
         "c_accept_status_fields":"",
         "m_accept_status_fields":"",
         "accept_status_id":"",
         "accept_status_name":"",
         "sync_contact":"",
         "my_favorite":false,
         "_acl":{
            "fields":{

            }
         }
      }
   ]
}
```


Change Log

Version	Change
v10	Added /<module>/:record/link/:link_name/filter GET endpoint.

//:record/link/:link_name/leancount GET

Overview

Lists related filtered records.

Summary

This endpoint will return a set of related records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Care will need to be taken to make sure this filter has appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

Name	Type	Description	Required
filter	String	The filter expression. Filter expressions are explained below.	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
fields	String	Comma delimited list of fields to return. The field date_modified will always be returned. This argument can be combined with the view argument. Example: name,account_type,description	False
view	String	Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list". Example: record	False

Name	Type	Description	Required
order_by	String	How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon. Example: name:DESC,account_type:DESC,date_modified:ASC	False

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
   "filter":[
      {
         "name":"Nelson Inc"
      }
   ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
   "filter":[
      {
```

```
         "name":{
            "$starts":"Nelson"
         }
      }
   ]
}
```

Below is a list of operation types:

Operation	Description
$equals	Performs an exact match on that field.
$not_equals	Performs an exact match on that field.
$not_equals	Matches on non-matching values.
$starts	Matches on anything that starts with the value.
$ends	Matches anything that ends with the value.
$contains	Matches anything that contains the value
$in	Finds anything where field matches one of the values as specified as an array.
$not_in	Finds anything where field does not matches any of the values as specified as an array.
$is_null	Checks if the field is null. This operation does not need a value specified.
$not_null	Checks if the field is not null. This operation does not need a value specified.

Operation		Description
	$lt	Matches when the field is less than the value.
	$lte	Matches when the field is less than or equal to the value.
	$gt	Matches when the field is greater than the value.
	$gte	Matches when the field is greater than or equal to the value.

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "$and" and "$or".

Example

```
{
   "filter":[
      {
         "$or":[
            {
               "name":"Nelson Inc"
            },
            {
               "name":"Nelson LLC"
            }
         ]
      }
   ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "$favorite" and "$owner".

Example

```
{
   "filter":[
      {
         "$favorite":"_this"
      }
   ]
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Repsonse

```
{
   "next_offset":-1,
   "records":[
      {
         "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
         "name":"Dale Spivey",
         "date_entered":"2013-02-26T19:12:00+00:00",
         "date_modified":"2013-02-28T05:03:00+00:00",
         "modified_user_id":"1",
         "modified_by_name":"Administrator",
         "created_by":"1",
         "created_by_name":"Administrator",
```

```
"description":"",
"img":"",
"deleted":false,
"assigned_user_id":"seed_sally_id",
"assigned_user_name":"Sally Bronsen",
"team_name":[
   {
      "id":"East",
      "name":"East",
      "name_2":"",
      "primary":false
   },
   {
      "id":1,
      "name":"Global",
      "name_2":"",
      "primary":false
   },
   {
      "id":"West",
      "name":"West",
      "name_2":"",
      "primary":true
   }
],
"salutation":"",
"first_name":"Dale",
"last_name":"Spivey",
"full_name":"Dale Spivey",
"title":"VP Operations",
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(523) 825-4311",
"email":[
   {
      "email_address":"sugar.dev.sugar@example.co.jp",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"1"
   },
   {
      "email_address":"the.support@example.biz",
```

```
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"0"
   }
],
"phone_mobile":"(373) 861-0757",
"phone_work":"(212) 542-9596",
"phone_other":"",
"phone_fax":"",
"email1":"sugar.dev.sugar@example.co.jp",
"email2":"the.support@example.biz",
"invalid_email":false,
"email_opt_out":false,
"primary_address_street":"345 Sugar Blvd.",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Denver",
"primary_address_state":"CA",
"primary_address_postalcode":"87261",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
"assistant":"",
"assistant_phone":"",
"picture":"",
"email_and_name1":"",
"lead_source":"Campaign",
"account_name":"Smallville Resources Inc",
"account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",
"opportunity_role_fields":"",
"opportunity_role_id":"",
"opportunity_role":"",
"reports_to_id":"",
"report_to_name":"",
"portal_name":"DaleSpivey97",
"portal_active":true,
"portal_password":"$1$yKMAONHM$Y5S.8CY.WZCZCwfGD1a1Q\/",
"portal_password1":"",
"portal_app":"",
"preferred_language":"en_us",
"campaign_id":"",
```

```
      "campaign_name":"",
      "c_accept_status_fields":"",
      "m_accept_status_fields":"",
      "accept_status_id":"",
      "accept_status_name":"",
      "sync_contact":"",
      "my_favorite":false,
      "_acl":{
         "fields":{

         }
      }
   },
   {
      "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
      "name":"Florence Haddock",
      "date_entered":"2013-02-26T19:12:00+00:00",
      "date_modified":"2013-02-26T19:12:00+00:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "description":"",
      "img":"",
      "deleted":false,
      "assigned_user_id":"seed_sally_id",
      "assigned_user_name":"Sally Bronsen",
      "team_name":[
         {
            "id":"East",
            "name":"East",
            "name_2":"",
            "primary":false
         },
         {
            "id":1,
            "name":"Global",
            "name_2":"",
            "primary":false
         },
         {
            "id":"West",
            "name":"West",
            "name_2":"",
            "primary":true
         }
```

```
],
"salutation":"",
"first_name":"Florence",
"last_name":"Haddock",
"full_name":"Florence Haddock",
"title":"Director Sales",
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(729) 845-3137",
"email":[
   {
      "email_address":"dev.vegan@example.de",
      "opt_out":"1",
      "invalid_email":"0",
      "primary_address":"0"
   },
   {
      "email_address":"section71@example.it",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"1"
   }
],
"phone_mobile":"(246) 233-1382",
"phone_work":"(565) 696-6981",
"phone_other":"",
"phone_fax":"",
"email1":"section71@example.it",
"email2":"dev.vegan@example.de",
"invalid_email":false,
"email_opt_out":false,
"primary_address_street":"111 Silicon Valley Road",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Denver",
"primary_address_state":"CA",
"primary_address_postalcode":"79900",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
```

```
         "alt_address_state":"",
         "alt_address_postalcode":"",
         "alt_address_country":"",
         "assistant":"",
         "assistant_phone":"",
         "picture":"",
         "email_and_name1":"",
         "lead_source":"Support Portal User Registration",
         "account_name":"Smallville Resources Inc",
         "account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",
         "opportunity_role_fields":"",
         "opportunity_role_id":"",
         "opportunity_role":"",
         "reports_to_id":"",
         "report_to_name":"",
         "portal_name":"FlorenceHaddock169",
         "portal_active":true,
         "portal_password":"$1$nWFhTbK6$JF9BCGSqL\/NCrbhueX5ia0",
         "portal_password1":"",
         "portal_app":"",
         "preferred_language":"en_us",
         "campaign_id":"",
         "campaign_name":"",
         "c_accept_status_fields":"",
         "m_accept_status_fields":"",
         "accept_status_id":"",
         "accept_status_name":"",
         "sync_contact":"",
         "my_favorite":false,
         "_acl":{
            "fields":{

            }
         }
      }
   ]
}
```


Change Log

Version	Change
v10	Added

	/<module>/:record/link/:link_name/filter GET endpoint.

//:record/link/activities GET

Activities on a module's list view

Summary:

This endpoint lists activities across a particular module in the system. It **does not** use a subscription model unlike the other endpoints. It can be queried as a regular bean collection, but without search, ordering or filtering.

Query Parameters:

Param	Description	Optional
max_num	A maximum number of records to return	Optional
offset	How many records to skip over before records are returned	Optional

Input Example:

This endpoint does not accept any input

Output Example:

```
{
    "next_offset": 20,
This will be set to -1 when there are no more records after this "page
".
    "records": [{
        "id": "22fb8b16-de1d-f1dc-b15b-51240efde177",
        "date_entered": "2013-02-19T23:47:11+00:00",
```

```
        "date_modified": "2013-02-19T23:47:11+00:00",
        "created_by": "1",
        "deleted": "0",
        "parent_id": "f5bb0331-2c0f-5c7c-b4db-5123caac0056",
        "parent_type": "Contacts",
        "activity_type": "post",
This will be the type of activity performed.
        "data": {
            "value": "This is a test post on a contact I'm subscribed
to."
        },
        "comment_count": 0,
This will be set to the total number of comments on the post.
        "last_comment": {
This will be the last comment on the post, which can be used to create
 a comment model on the frontend.
            "deleted": 0,
            "data": []
        },
        "fields": [],
        "first_name": null,
        "last_name": "Administrator",
        "created_by_name":
 " Administrator" This will be the locale-
formatted full name of the user.
    }, ... ]
}
```

//:record/link/activities/filter GET

Activities on a module's list view

Summary:

This endpoint lists activities across a particular module in the system. It **does not** use a subscription model unlike the other endpoints. It can be queried as a regular bean collection, but without search, ordering or filtering.

Query Parameters:

Param	Description	Optional
max_num	A maximum number of records to return	Optional
offset	How many records to skip over before records are returned	Optional

Input Example:

This endpoint does not accept any input

Output Example:

```
{
    "next_offset": 20,
This will be set to -1 when there are no more records after this "page
".
    "records": [{
        "id": "22fb8b16-de1d-f1dc-b15b-51240efde177",
        "date_entered": "2013-02-19T23:47:11+00:00",
        "date_modified": "2013-02-19T23:47:11+00:00",
        "created_by": "1",
        "deleted": "0",
        "parent_id": "f5bb0331-2c0f-5c7c-b4db-5123caac0056",
        "parent_type": "Contacts",
        "activity_type": "post",
This will be the type of activity performed.
        "data": {
            "value": "This is a test post on a contact I'm subscribed
to."
        },
        "comment_count": 0,
This will be set to the total number of comments on the post.
        "last_comment": {
This will be the last comment on the post, which can be used to create
 a comment model on the frontend.
            "deleted": 0,
```

```
            "data": []
        },
        "fields": [],
        "first_name": null,
        "last_name": "Administrator",
        "created_by_name":
 " Administrator" This will be the locale-
formatted full name of the user.
    }, ... ]
}
```

//:record/link/history GET

Overview

Lists history filtered records.

Summary

This endpoint will return a set of history modules (meetings, calls, notes, tasks, emails) records filtered by an expression.

Request Arguments

Name	Type	Description	Required
filter	String	The filter expression. Filter expressions are explained below.	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
order_by	String	How to sort the returned records, in a comma	False

Name	Type	Description	Required
		delimited list with the direction appended to the column name after a colon. Example: name:DESC,account_type:DESC,date_modified:ASC	
deleted	Boolean	Boolean to show deleted records in the result set.	False
module_list	String	A list of modules from the valid modules [Tasks, Calls, Emails, Notes, Meetings] that need to be included	False
alias_fields	Array	A list of field aliases from the valid modules [Tasks, Calls, Emails, Notes, Meetings] that need to be fetched. These field aliases can also be used in 'order_by'. Example: { 'record_date': { 'Calls': 'date_start', 'Emails': 'date_sent' } }	False

//:record/link/related_activities GET

Overview

Returns related activity records for a specified record.

Request Arguments

Name	Type	Description	Required
module_list	String	A list of modules from the valid modules [Tasks, Calls, Emails, Notes, Meetings] that need to be included. Example: Meetings,Calls,Emails,Notes	False
order_by	String	How to sort the returned records, in a comma-delimited list with the direction appended to the column name after a colon. Example: name:DESC,account_type:DESC,date_modified:ASC	False
max_num	String	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
deleted	Boolean	Boolean to show deleted records in the result set. Default is false.	False
alias_fields	Array	A list of field aliases from the valid modules [Tasks, Calls, Emails, Notes,	False

		Meetings] that need to be fetched. These field aliases can also be used in 'order_by'. Example: { 'record_date': { 'Calls': 'date_start', 'Emails': 'date_sent' } }	

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Response


```
  {
next_offset":-1,
   "records":[
      {
         "id":"2f9abe52-f03f-11e9-858c-3c15c2d57622",
         "name":"test call",
         "date_entered":"2019-10-16T11:02:57-07:00",
         "date_modified":"2019-10-16T11:02:57-07:00",
         "description":"",
         "date_start":"2019-10-25T11:30:00-07:00",
         "status":"Planned",
         "contact_name":"",
         "contacts":{
            "name":"",
            "id":"",
            "_acl":{
               "fields":[
               ],
               "_hash":"654d337e0e912edaa00dbb0fb3dc3c17"
```

```
            }
         },
         "contact_id":"",
         "locked_fields":[
         ],
         "assigned_user_id":"1",
         "assigned_user_name":"Administrator",
         "assigned_user_link":{
            "full_name":"Administrator",
            "id":"1",
            "_acl":{
               "fields":{
                  "pwd_last_changed":{
                     "write":"no",
                     "create":"no"
                  },
                  "last_login":{
                     "write":"no",
                     "create":"no"
                  }
               },
               "delete":"no",
               "_hash":"08b99a97c2e8d792f7a44d8882b5af6d"
            }
         },
         "_acl":{
            "fields":{
            }
         },
         "_module":"Calls",
         "moduleNameSingular":"Call",
         "moduleName":"Calls"
      },
   ]
}
```


Change Log

Version	Change
v11.7	Added /<module>/:record/link/related_activities GET endpoint.

//:record/moveafter/:target PUT

Overview

Move existing node after target node.

Request Arguments

Name	Type	Description	Required
module	String	The name of sugar module that contains a nested set data and implements the NestedSetInterface.	True
target	String	The ID of record that will be used as target to move node after	True

Response Arguments

This endpoint does not return any response arguments.

Response

This endpoint return a record GUID that was moved
"1b7b868d-7357-2e29-7513-54169bdc444d"

Change Log

Version	Change
v10	Added /<module>/moveafter/:target PUT endpoint.

//:record/movebefore/:target PUT

Overview

Move existing node before target node.

Request Arguments

Name	Type	Description	Required
module	String	The name of sugar module that contains a nested set data and implements the NestedSetInterface .	True
target	String	The ID of record that will be used as target to move node before	True

Response Arguments

This endpoint does not return any response arguments.

Response

This endpoint return a record GUID that was moved "1b7b868d-7357-2e29-7513-54169bdc444d"

Change Log

Version	Change
v10	Added /<module>/movebefore/:target PUT endpoint.

//:record/movefirst/:target PUT

Overview

Move existing node as first child of target node.

Request Arguments

Name	Type	Description	Required
module	String	The name of sugar module that contains a nested set data and implements the	True

Name	Type	Description	Required
		NestedSetInterface .	
target	String	The ID of record that will be used as target to move node as first child	True

Response Arguments

This endpoint does not return any response arguments.

Response

This endpoint return a record GUID that was moved
"1b7b868d-7357-2e29-7513-54169bdc444d"

Change Log

Version	Change
v10	Added /<module>/movefirst/:target PUT endpoint.

//:record/movelast/:target PUT

Overview

Move existing node as last child of target node.

Request Arguments

Name	Type	Description	Required
module	String	The name of sugar module that contains a nested set data and implements the NestedSetInterface .	True
target	String	The ID of record that will be used as target to move	True

Name	Type	Description	Required
		node as last child	

Response Arguments

This endpoint does not return any response arguments.

Response

This endpoint return a record GUID that was moved
"1b7b868d-7357-2e29-7513-54169bdc444d"

Change Log

Version	Change
v10	Added /<module>/movelast/:target PUT endpoint.

//:record/next GET

Overview

Retrieves next sibling of selected record.

Request Arguments

Name	Type	Description	Required
module	String	The name of sugar module that contains a nested set data and implements the NestedSetInterface .	True
:record	String	The ID of record	True

Response Arguments

This endpoint does not return any response arguments.

Response

{

```
    id: "e1ae8646-ac90-104a-59d6-5412cf5009b2"
    name: "SugarCategoryExample"
    date_entered: "2014-09-12 10:47:46"
    date_modified: "2014-09-12 10:47:46"
    modified_user_id: "9c9ad14e-1789-3340-f88d-5412cf551d3b"
    created_by: "9c9ad14e-1789-3340-f88d-5412cf551d3b"
    description: null
    deleted: "0"
    source_id: null
    source_type: null
    source_meta: null
    root: "e1ae8646-ac90-104a-59d6-5412cf5009b2"
    lft: "6"
    rgt: "11"
    level: "1"
}
```


Change Log

Version	Change
v10	Added /<module>/:record/next GET endpoint.

//:record/parent GET

Overview

Retrieves parent node for selected record.

Request Arguments

Name	Type	Description	Required
module	String	The name of sugar module that contains a nested set data and implements the NestedSetInterface.	True
:record	String	The ID of record	True

Response Arguments

This endpoint does not return any response arguments.

Response

```
{
    id: "e1ae8646-ac90-104a-59d6-5412cf5009b2"
    name: "SugarCategoryExample"
    date_entered: "2014-09-12 10:47:46"
    date_modified: "2014-09-12 10:47:46"
    modified_user_id: "9c9ad14e-1789-3340-f88d-5412cf551d3b"
    created_by: "9c9ad14e-1789-3340-f88d-5412cf551d3b"
    description: null
    deleted: "0"
    source_id: null
    source_type: null
    source_meta: null
    root: "e1ae8646-ac90-104a-59d6-5412cf5009b2"
    lft: "6"
    rgt: "11"
    level: "1"
}
```

Change Log

Version	Change
v10	Added /<module>/:record/parent GET endpoint.

//:record/path GET

Overview

Retrieves all parents of selected record.

Request Arguments

Name	Type	Description	Required
module	String	The name of sugar	True

Name	Type	Description	Required
		module that contains a nested set data and implements the NestedSetInterface .	
:record	String	The ID of record	True

Response Arguments

This endpoint does not return any response arguments.

Response


```
[{
    "id": "045c03b9-327b-11e4-818b-5404a67f3363",
    "name": "Audit",
    "date_entered": null,
    "date_modified": null,
    "modified_user_id": null,
    "created_by": null,
    "description": null,
    "deleted": "0",
    "source_id": null,
    "source_type": null,
    "source_meta": null,
    "root": "935d3e07-327a-11e4-818b-5404a67f3363",
    "lft": "10",
    "rgt": "19",
    "level": "1"
}, {
    "id": "045c1de6-327b-11e4-818b-5404a67f3363",
    "name": "Financial",
    "date_entered": null,
    "date_modified": null,
    "modified_user_id": null,
    "created_by": null,
    "description": null,
    "deleted": "0",
    "source_id": null,
    "source_type": null,
    "source_meta": null,
    "root": "935d3e07-327a-11e4-818b-5404a67f3363",
    "lft": "11",
```

```
    "rgt": "14",
    "level": "2"
}]
```

Change Log

Version	Change
v10	Added /<module>/:record/path GET endpoint.

//:record/pii GET

Overview

Returns personally identifiable information (pii) fields with current data and source of data capture for a specific record.

Version Requirements

Minimum REST API Version required - 11.1

Request Arguments

Name	Type	Description	Required
module	String	The name of the module.	True
record	String	The record id.	True

Response Arguments

Name	Type	Description
records	String	Returns the set of PII field records for the given module record.

Response

```
{
  "fields": [
    {
```

```
      "field_name": "phone_mobile",
      "value": "(645) 604-0128",
      "event_type": "update",
      "date_modified": "2018-02-21T19:56:42+00:00",
      "source": {
        "subject": {
          "_type": "user",
          "id": "999b3b50-16dc-11e8-9c03-a45e60e64123",
          "_module": "Users",
          "client": {
            "_type": "rest-api"
          },
          "first_name": "Tim",
          "last_name": "Zhang",
          "name": "Tim Zhang"
        },
        "attributes": {
          "platform": "base"
        }
      }
    },
    {
      "field_name": "phone_work",
      "value": "(212) 692-3480",
      "event_type": "update",
      "date_modified": "2018-02-21T19:56:42+00:00",
      "source": {
        "subject": {
          "_type": "user",
          "id": "654b3b50-16dc-11e8-9c03-a45e60e64465",
          "_module": "Users",
          "client": {
            "_type": "rest-api"
          },
          "first_name": "Jim",
          "last_name": "Conners",
          "name": "Jim Conners"
        },
        "attributes": {
          "platform": "base"
        }
      }
    }
  ]
}
```

Change Log

Version	Change
11.1	Added /pii GET endpoint.

//:record/prev GET

Overview

Retrieves previous sibling of selected record.

Request Arguments

Name	Type	Description	Required
module	String	The name of sugar module that contains a nested set data and implements the NestedSetInterface.	True
:record	String	The ID of record	True

Response Arguments

This endpoint does not return any response arguments.

Response

```
{
    id: "e1ae8646-ac90-104a-59d6-5412cf5009b2"
    name: "SugarCategoryExample"
    date_entered: "2014-09-12 10:47:46"
    date_modified: "2014-09-12 10:47:46"
    modified_user_id: "9c9ad14e-1789-3340-f88d-5412cf551d3b"
    created_by: "9c9ad14e-1789-3340-f88d-5412cf551d3b"
    description: null
    deleted: "0"
    source_id: null
    source_type: null
    source_meta: null
    root: "e1ae8646-ac90-104a-59d6-5412cf5009b2"
```

```
    lft: "6"
    rgt: "11"
    level: "1"
}
```

Change Log

Version	Change
v10	Added /<module>/:record/prev GET endpoint.

//:record/subscribe POST

Summary:

This endpoint creates a subscription record in the Subscriptions table, from a specified record and module. It allows the user to be subscribed to activity stream messages for the record being subscribed to, or "followed".

Query Parameters:

This endpoint does not accept any parameters.

Input Example:

This endpoint does not accept any input.

Output Example:

If successful, the endpoint responds with the GUID of the subscription record.
"96ad733c-df51-dd9d-1888-514112ef0595"

//:record/unfavorite PUT

Overview

Removes a record of a specified type as a favorite for the current user.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

Name	Type	Description
<record field>	<record field type>	Returns the fields for the selected record.

Response


```
{
   "id":"bdd59d85-687b-1739-b00a-512d09f6db9e",
   "name":"Insight Marketing Inc",
   "date_entered":"2013-02-26T19:12:00+00:00",
   "date_modified":"2013-02-26T19:12:00+00:00",
   "modified_user_id":"1",
   "modified_by_name":"Administrator",
   "created_by":"1",
   "created_by_name":"Administrator",
   "description":"",
   "img":"",
   "last_activity_date":"2013-02-26T19:12:00+00:00",
   "deleted":false,
   "assigned_user_id":"seed_max_id",
   "assigned_user_name":"Max Jensen",
   "team_name":[
      {
         "id":"East",
         "name":"East",
         "name_2":"",
         "primary":false
      },
      {
         "id":1,
         "name":"Global",
         "name_2":"",
         "primary":false
      },
      {
         "id":"West",
         "name":"West",
```

```
      "name_2":"",
      "primary":true
   }
],
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"account_type":"Customer",
"industry":"Electronics",
"annual_revenue":"",
"phone_fax":"",
"billing_address_street":"345 Sugar Blvd.",
"billing_address_street_2":"",
"billing_address_street_3":"",
"billing_address_street_4":"",
"billing_address_city":"San Mateo",
"billing_address_state":"CA",
"billing_address_postalcode":"56019",
"billing_address_country":"USA",
"rating":"",
"phone_office":"(927) 136-9572",
"phone_alternate":"",
"website":"www.sectionvegan.de",
"ownership":"",
"employees":"",
"ticker_symbol":"",
"shipping_address_street":"345 Sugar Blvd.",
"shipping_address_street_2":"",
"shipping_address_street_3":"",
"shipping_address_street_4":"",
"shipping_address_city":"San Mateo",
"shipping_address_state":"CA",
"shipping_address_postalcode":"56019",
"shipping_address_country":"USA",
"email1":"kid.support.vegan@example.info",
"parent_id":"",
"sic_code":"",
"parent_name":"",
"email_opt_out":false,
"invalid_email":false,
"email":[
   {
      "email_address":"kid.support.vegan@example.info",
      "opt_out":"0",
      "invalid_email":"0",
```

```
            "primary_address":"1"
         },
         {
            "email_address":"phone.kid@example.cn",
            "opt_out":"0",
            "invalid_email":"0",
            "primary_address":"0"
         }
      ],
      "campaign_id":"",
      "campaign_name":"",
      "my_favorite":false,
      "_acl":{
         "fields":{

         }
      }
   }
```


Change Log

Version	Change
v10	Added /<module>/:record/favorite DELETE endpoint.

//:record/unsubscribe DELETE

Summary:

This endpoint deletes a subscription record in the Subscriptions table, from a specified record and module. It allows the user to be unsubscribe from activity stream messages for the record being subscribed to, or "followed".

Query Parameters:

This endpoint does not accept any parameters.

Input Example:

This endpoint does not accept any input.

Output Example:

If successful, the endpoint responds with a bool of true.

//:record/vcard GET

Overview

Downloads a vCard.

Request Arguments

This endpoint does not accept any arguments.

Response Arguments

Name	Type	Description
<vcard information>	String;	Record in vcard format

Response

```
BEGIN:VCARD
N;CHARSET=utf-8:Leone;Rosemarie;;
FN;CHARSET=utf-8: Rosemarie Leone
BDAY:
TEL;WORK;FAX:
TEL;HOME;CHARSET=utf-8:(692) 586-8287
TEL;CELL;CHARSET=utf-8:(117) 577-0969
TEL;WORK;CHARSET=utf-8:(844) 325-7679
EMAIL;INTERNET;CHARSET=utf-8:support.im@example.it
ADR;WORK;CHARSET=utf-8:;;777 West Filmore Ln;San Mateo;CA;74984;USA
ORG;CHARSET=utf-8:Q.R.&E. Corp;
TITLE;CHARSET=utf-8:Director Sales
END:VCARD
}
```

Change Log

Version	Change
v10	Added /<module>/:record/vcard GET endpoint.

//:root/tree GET

Overview

Retrieves full tree for selected root record.

Request Arguments

Name	Type	Description	Required
module	String	The name of sugar module that contains a nested set data and implements the NestedSetInterface.	True
:record	String	The ID of record	True

Response Arguments

This endpoint does not return any response arguments.

Response


```
{
"next_offset": -1,
"records": [{
    "id": "ad4ddf76-327a-11e4-818b-5404a67f3363",
    "name": "Documents",
    "date_entered": null,
    "date_modified": null,
    "modified_user_id": null,
    "created_by": null,
    "description": null,
    "deleted": "0",
    "source_id": null,
    "source_type": null,
    "source_meta": null,
    "root": "935d3e07-327a-11e4-818b-5404a67f3363",
    "lft": "2",
```

```
"rgt": "9",
"level": "1",
"children": {
    "next_offset": -1,
    "records": [{
        "id": "ad4e03f1-327a-11e4-818b-5404a67f3363",
        "name": "Engeneering",
        "date_entered": null,
        "date_modified": null,
        "modified_user_id": null,
        "created_by": null,
        "description": null,
        "deleted": "0",
        "source_id": null,
        "source_type": null,
        "source_meta": null,
        "root": "935d3e07-327a-11e4-818b-5404a67f3363",
        "lft": "3",
        "rgt": "4",
        "level": "2",
        "children": {
            "next_offset": -1,
            "records": []
        }
    }, {
        "id": "f482dd1b-327a-11e4-818b-5404a67f3363",
        "name": "Testing",
        "date_entered": null,
        "date_modified": null,
        "modified_user_id": null,
        "created_by": null,
        "description": null,
        "deleted": "0",
        "source_id": null,
        "source_type": null,
        "source_meta": null,
        "root": "935d3e07-327a-11e4-818b-5404a67f3363",
        "lft": "5",
        "rgt": "6",
        "level": "2",
        "children": {
            "next_offset": -1,
            "records": []
        }
    }, {
        "id": "f482fb7a-327a-11e4-818b-5404a67f3363",
```

```
            "name": "Management",
            "date_entered": null,
            "date_modified": null,
            "modified_user_id": null,
            "created_by": null,
            "description": null,
            "deleted": "0",
            "source_id": null,
            "source_type": null,
            "source_meta": null,
            "root": "935d3e07-327a-11e4-818b-5404a67f3363",
            "lft": "7",
            "rgt": "8",
            "level": "2",
            "children": {
                "next_offset": -1,
                "records": []
            }
        }]
    }
}, {
    "id": "045c03b9-327b-11e4-818b-5404a67f3363",
    "name": "Audit",
    "date_entered": null,
    "date_modified": null,
    "modified_user_id": null,
    "created_by": null,
    "description": null,
    "deleted": "0",
    "source_id": null,
    "source_type": null,
    "source_meta": null,
    "root": "935d3e07-327a-11e4-818b-5404a67f3363",
    "lft": "10",
    "rgt": "19",
    "level": "1",
    "children": {
        "next_offset": -1,
        "records": [{
            "id": "045c1de6-327b-11e4-818b-5404a67f3363",
            "name": "Financial",
            "date_entered": null,
            "date_modified": null,
            "modified_user_id": null,
            "created_by": null,
            "description": null,
```

```
        "deleted": "0",
        "source_id": null,
        "source_type": null,
        "source_meta": null,
        "root": "935d3e07-327a-11e4-818b-5404a67f3363",
        "lft": "11",
        "rgt": "14",
        "level": "2",
        "children": {
            "next_offset": -1,
            "records": [{
                "id": "0f658847-327b-11e4-818b-5404a67f3363",
                "name": "Invoices",
                "date_entered": null,
                "date_modified": null,
                "modified_user_id": null,
                "created_by": null,
                "description": null,
                "deleted": "0",
                "source_id": null,
                "source_type": null,
                "source_meta": null,
                "root": "935d3e07-327a-11e4-818b-5404a67f3363",
                "lft": "12",
                "rgt": "13",
                "level": "3",
                "children": {
                    "next_offset": -1,
                    "records": []
                }
            }]
        }
    }, {
        "id": "0f65a6c7-327b-11e4-818b-5404a67f3363",
        "name": "Agreements",
        "date_entered": null,
        "date_modified": null,
        "modified_user_id": null,
        "created_by": null,
        "description": null,
        "deleted": "0",
        "source_id": null,
        "source_type": null,
        "source_meta": null,
        "root": "935d3e07-327a-11e4-818b-5404a67f3363",
        "lft": "15",
```

```
            "rgt": "16",
            "level": "2",
            "children": {
                "next_offset": -1,
                "records": []
            }
        }, {
            "id": "14d30bf3-327b-11e4-818b-5404a67f3363",
            "name": "Clients",
            "date_entered": null,
            "date_modified": null,
            "modified_user_id": null,
            "created_by": null,
            "description": null,
            "deleted": "0",
            "source_id": null,
            "source_type": null,
            "source_meta": null,
            "root": "935d3e07-327a-11e4-818b-5404a67f3363",
            "lft": "17",
            "rgt": "18",
            "level": "2",
            "children": {
                "next_offset": -1,
                "records": []
            }
        }]
    }
}]
}
```


Change Log

Version	Change
v10	Added /<module>/:record/tree GET endpoint.

//Activities GET

Activities on a module's list view

Summary:

This endpoint lists activities across a particular module in the system. It uses a subscription model, and can be queried like a normal bean collection, but without search, ordering or filtering.

Query Parameters:

Param	Description	Optional
max_num	A maximum number of records to return	Optional
offset	How many records to skip over before records are returned	Optional

Input Example:

This endpoint does not accept any input

Output Example:

```
{
    "next_offset": 20,
This will be set to -1 when there are no more records after this "page
".
    "records": [{
        "id": "22fb8b16-de1d-f1dc-b15b-51240efde177",
        "date_entered": "2013-02-19T23:47:11+00:00",
        "date_modified": "2013-02-19T23:47:11+00:00",
        "created_by": "1",
        "deleted": "0",
        "parent_id": "f5bb0331-2c0f-5c7c-b4db-5123caac0056",
        "parent_type": "Contacts",
        "activity_type": "post",
This will be the type of activity performed.
        "data": {
```

```
            "value": "This is a test post on a contact I'm subscribed
to."
        },
        "comment_count": 0,
This will be set to the total number of comments on the post.
        "last_comment": {
This will be the last comment on the post, which can be used to create
 a comment model on the frontend.
            "deleted": 0,
            "data": []
        },
        "fields": [],
        "first_name": null,
        "last_name": "Administrator",
        "created_by_name":
 " Administrator" This will be the locale-
formatted full name of the user.
    }, ... ]
}
```

//Activities/filter GET

Activities on a module's list view

Summary:

This endpoint lists activities across a particular module in the system. It uses a subscription model, and can be queried like a normal bean collection, but without search, ordering or filtering.

Query Parameters:

Param	Description	Optional
max_num	A maximum number of records to return	Optional
offset	How many records to skip over before records are returned	Optional

Input Example:

This endpoint does not accept any input

Output Example:

```
{
    "next_offset": 20,
This will be set to -1 when there are no more records after this "page
".
    "records": [{
        "id": "22fb8b16-de1d-f1dc-b15b-51240efde177",
        "date_entered": "2013-02-19T23:47:11+00:00",
        "date_modified": "2013-02-19T23:47:11+00:00",
        "created_by": "1",
        "deleted": "0",
        "parent_id": "f5bb0331-2c0f-5c7c-b4db-5123caac0056",
        "parent_type": "Contacts",
        "activity_type": "post",
This will be the type of activity performed.
        "data": {
            "value": "This is a test post on a contact I'm subscribed
to."
        },
        "comment_count": 0,
This will be set to the total number of comments on the post.
        "last_comment": {
This will be the last comment on the post, which can be used to create
 a comment model on the frontend.
            "deleted": 0,
            "data": []
        },
        "fields": [],
        "first_name": null,
        "last_name": "Administrator",
        "created_by_name":
 " Administrator" This will be the locale-
formatted full name of the user.
    }, ... ]
}
```

//MassUpdate DELETE

Overview

An API to mass delete records.

Query String Parameters

Name	Type	Description	Required
massupdate_params	Array	Mass update parameters.	True
massupdate_params.uid	Array	A list of ids.	True

Request

Mass Deleting Records by ID in a Module

```
{
    "massupdate_params":{
        "uid":[
            "ebf22b86-94ea-1601-4f4f-512d09173438",
            "e3b71c55-d96b-80bb-1696-512d09672398"
        ]
    }
}
```

Response Arguments

Name	Type	Description
Status	String	The status of the mass update. Possible value is 'done'.

Output Done Example

```
{
    "status":"done"
}
```

Change Log

Version	Change
v10	Added /<module>/MassUpdate DELETE endpoint.

//MassUpdate PUT

Overview

An API to mass update records.

Query String Parameters

Name	Type	Description	Required
massupdate_params	Array	Mass update parameters.	True
massupdate_params.uid	Array	A list of ids.	True
massupdate_params.[field name]	[field type]	The field to be modified.	False
massupdate_params.team_name	Array	Team array.	False
massupdate_params.team_name_type	String	Whether to replace or add teams. Possible values are 'add' and 'replace'.	False

Request

Mass Updating Records by ID in a Module

```
{
    "massupdate_params":{
```

```
        "uid":[
            "f22d1955-97d8-387d-9866-512d09cc1520",
            "ef1b40aa-5815-4f8d-e909-512d09617ac8"
        ],
        "department":"Marketing"
    }
}
```

Mass Updating Records with Teams

```
{
   "massupdate_params":{
      "uid":[
         "f22d1955-97d8-387d-9866-512d09cc1520",
         "ef1b40aa-5815-4f8d-e909-512d09617ac8"
      ],
      "team_name":[
         {
            "id":"35eab226-c20d-48f4-4670-512d095c8c6f",
            "primary":true
         },
         {
            "id":"8f640aba-f356-7374-7eb4-512d09745551",
            "primary":false
         }
      ],
      "team_name_type":"replace"
   }
}
```

Mass Add Leads, Contacts or Prospects to a Target List

```
{
    "massupdate_params": {
        "uid": [     /* Leads, Contacts, or Prospects to Add */
            "f3d90a49-14b4-a81c-6cac-526e6c71d33e",
            "f22cde0d-9a20-89d3-ca14-526e6c3c4d08",
            "f15f10bd-1445-5e20-9b5c-526e6ceb71d0"
        ],
        "prospect_lists": [
                /* Prospect List(s) to Add them To */
```

```
            "bc5bc249-9c9c-52ad-52b9-526e71f0e18d"
        ]
    }
}
```

Response Arguments

Name	Type	Description
Status	String	The status of the mass update. Possible value is 'done'.

Output Done Example

```
{
    "status":"done"
}
```

Change Log

Version	Change
v10	Added /<module>/MassUpdate PUT endpoint.

//append/:target POST

Overview

Append new node to target node as last child.

Request Arguments

Name	Type	Description	Required
module	String	The name of sugar module that contains a nested set data and implements the	True

Name	Type	Description	Required
		NestedSetInterface .	
target	String	The ID of record that will be used as target to append new node	True

Request

```
{
    "name" : "Children Node 1"
}
```

Response Arguments

This endpoint does not return any response arguments.

Response

This endpoint returns a newly created bean
{ "my_favorite":false, "following":"", "id":"59fa8dd7-0f2c-4bfd-364f-54495f77fa3f", "name":"Default title", "date_entered":"2014-10-23T23:03:22+03:00", "date_modified":"2014-10-23T23:03:22+03:00", "modified_user_id":"1", "modified_by_name":"Administrator", "created_by":"1", "created_by_name":"Administrator", "doc_owner":"", "description":"", "deleted":false, "source_id":"", "source_type":"", "source_meta":"", "root":"be9b0c4a-8b78-1ffa-4f14-54481c2f6269", "lft":118, "rgt":119, "level":1, "_acl":{"fields":{}}, "_module":"Categories" }

Change Log

Version	Change
v10	Added /<module>/append/:target POST endpoint.

//audit/export GET

Overview

Returns a list of data changes for a specific module.

Request Arguments

Name	Type	Description	Required
module	String	The name of the module.	True

Query Parameters

Name	Type	Description	Required
max_num	Integer	A maximum number of Audit records to return. The default is 100 records.	False
offset	Integer	The number of records to skip over before records are returned. The default is 0.	False

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Response


```
{
 "next_offset":-1,
 "records":[
    {
       "id":"b569e480-237a-5921-4382-512ff555fee7",
       "parent_id":"ef1b40aa-5815-4f8d-e909-512d09617ac8",
       "date_created":"03\/01\/2013 12:26am",
```

```
         "created_by":"admin",
         "field_name":"Team ID:",
         "data_type":"team_list",
         "before_value_string":"East",
         "after_value_string":"Will"
      },
      {
         "id":"bc0bee72-8398-a0a6-d731-512ff5bfebc7",
         "parent_id":"ef1b40aa-5815-4f8d-e909-512d09617ac8",
         "date_created":"03\/01\/2013 12:26am",
         "created_by":"admin",
         "field_name":"Teams",
         "data_type":"id",
         "before_value_string":"East, Global, West",
         "after_value_string":"Jim, Will"
      }
   ]
}
```


Change Log

Version	Change
v11_11	Added /<module>/audit/export GET endpoint.

//config GET

Overview

Retrieves the config settings for a given module.

Summary

This endpoint is normally used for the forecasting module.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

Name	Type	Description
<config field>	<config field type>	The list of config fields will be returned.

Response


```
{
   "is_setup":1,
   "is_upgrade":0,
   "has_commits":1,
   "timeperiod_type":"chronological",
   "timeperiod_interval":"Annual",
   "timeperiod_leaf_interval":"Quarter",
   "timeperiod_start_date":"2013-01-01",
   "timeperiod_shown_forward":2,
   "timeperiod_shown_backward":2,
   "forecast_ranges":"show_binary",
   "buckets_dom":"commit_stage_binary_dom",
   "show_binary_ranges":{
      "include":{
         "min":70,
         "max":100
      },
      "exclude":{
         "min":0,
         "max":69
      }
   },
   "show_buckets_ranges":{
      "include":{
         "min":85,
         "max":100
      },
      "upside":{
         "min":70,
         "max":84
      },
      "exclude":{
         "min":0,
         "max":69
      }
   },
   "show_custom_buckets_ranges":{
      "include":{
         "min":85,
```

```
            "max":100
         },
         "upside":{
            "min":70,
            "max":84
         },
         "exclude":{
            "min":0,
            "max":69
         }
      },
      "sales_stage_won":[
         "Closed Won"
      ],
      "sales_stage_lost":[
         "Closed Lost"
      ],
      "show_worksheet_likely":1,
      "show_worksheet_best":1,
      "show_worksheet_worst":0,
      "show_projected_likely":1,
      "show_projected_best":1,
      "show_projected_worst":0,
      "show_forecasts_commit_warnings":1
}
```


Change Log

Version	Change
v10	Added /<module>/config GET endpoint.

//config POST

Overview

Retrieves the config settings for a given module.

Summary

This endpoint is normally used to manage the forecasting module.

Request Arguments

Name	Type	Description	Required
<config field>	<config field type>	The list of config fields to update.	False

Request

```
{
    "MyConfigOption":"MyConfigValue"
}
```

Response Arguments

Name	Type	Description
<config field>	<config field type>	The list of config fields will be returned.

Response

```
{
   "MyConfigOption":"MyConfigValue"
}
```

Change Log

Version	Change
v10	Added /<module>/config POST endpoint.

//config PUT

Overview

Retrieves the config settings for a given module.

Summary

This endpoint is normally used to manage the forecasting module.

Request Arguments

Name	Type	Description	Required
<config field>	<config field type>	The list of config fields to update.	False

Request

```
{
    "MyConfigOption":"MyConfigValue"
}
```

Response Arguments

Name	Type	Description
<config field>	<config field type>	The list of config fields will be returned.

Response

```
{
   "MyConfigOption":"MyConfigValue"
}
```

Change Log

Version	Change
v10	Added /<module>/config PUT endpoint.

//count GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

Name	Type	Description	Required
filter	String	The filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query parameters in one of two ways for GET requests: 1. By specifying individual filter arguments as distinct parameters. Example: filter[0][id]=1. 2. By specifying the whole filter as a single JSON-encoded string. Note that this syntax is	False

Name	Type	Description	Required
		currently not supported on certain modules. Example: filter=[{"id":"1"}].	
filter_id	String	Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND.	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
fields	String	Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters (applicable to link and collection fields). The fields id and date_modified will always be returned. Example: name,account_type,description,{"name":"opportunities","fields":["id","name","sales_sta	False

Name	Type	Description	Required
		tus"],"order_by":"date_closed:desc"} For more details on additional field parameters, see Relate API and Collection API.	
view	String	Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list". Example: record	False
order_by	String	How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon. Example: name:DESC,account_type:DESC,date_modified:ASC	False
q	String	A search expression, will search on this module. Cannot be used at the same	False

Name	Type	Description	Required
		time as a filter expression or id.	
deleted	Boolean	Boolean to show deleted records in the result set.	False
nulls_last	Boolean	Boolean to return records with null values in order_by fields last in the result set.	False

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
   "filter":[
      {
         "name":"Nelson Inc"
      }
   ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
   "filter":[
      {
         "name":{
            "$starts":"Nelson"
```

```
            }
        }
    ]
}
```

Below is a list of operation types:

Operation	Description
$equals	Performs an exact match on that field.
$not_equals	Performs an exact match on that field.
$not_equals	Matches on non-matching values.
$starts	Matches on anything that starts with the value.
$ends	Matches anything that ends with the value.
$contains	Matches anything that contains the value
$in	Finds anything where field matches one of the values as specified as an array.
$not_in	Finds anything where field does not matches any of the values as specified as an array.
$is_null	Checks if the field is null. This operation does not need a value specified.
$not_null	Checks if the field is not null. This operation does not need a value specified.
$lt	Matches when the field is less than the

Operation		Description	
			value.
	$lte		Matches when the field is less than or equal to the value.
	$gt		Matches when the field is greater than the value.
	$gte		Matches when the field is greater than or equal to the value.

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "$and" and "$or".

Example

```
{
   "filter":[
      {
         "$or":[
            {
               "name":"Nelson Inc"
            },
            {
               "name":"Nelson LLC"
            }
         ]
      }
   ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by

leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "$favorite" and "$owner".

Example

```
{
   "filter":[
      {
         "$favorite":"_this"
      }
   ]
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Response

```
{
   "next_offset":-1,
   "records":[
      {
         "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
         "name":"Dale Spivey",
         "date_modified":"2013-02-28T05:03:00+00:00",
         "description":"",
         "opportunities": [
            {
               _module: "Opportunities",
               "id": "b0701501-1fab-8ae7-3942-540da93f5017",
               "name": "360 Vacations - 228 Units",
               "date_modified": "2014-09-08T16:05:00+03:00",
```

```
                "sales_status": "New"
            },
         ],
         "_acl": {
            "fields": {
            }
         }
      },
      {
         "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
         "name":"Florence Haddock",
         "date_modified":"2013-02-26T19:12:00+00:00",
         "description":"",
         "opportunities": [
            {
               _module: "Opportunities"
               date_modified: "2014-09-08T16:05:00+03:00"
               id: "9ce7c088-8ee4-7cd3-18f1-540da944d4c0"
               name: "360 Vacations - 312 Units"
               sales_status: "New"
            },
         ],
         "_acl": {
            "fields": {
            }
         }
      }
   ]
}
```


Change Log

Version	Change
v10	Added /<module>/filter GET endpoint.

//customfield POST

Overview

Create a new custom field for a specified module.

Summary

This endpoint is used to create a new custom field for a specified module.

Request Arguments

Name	Type	Description	Required
data	array	The field metadata.	True

Request Examples

This example will create a decimal type custom field:

```
  {
    "localizations": {
      "en_us": {
          "LBL_AI_SCORE": "Ai Score"
      },
      ...
    },
    "data":{
      "name":"ai_score",
      "label":"LBL_AI_SCORE",
      "type":"decimal",
      "len":"18",
      "precision":8,
      "required":false,
      "reportable":false,
      "audited":false,
      "importable":false,
      "duplicate_merge":false
    }
}
```

This example will create a textField type custom field:

```
  {
    "localizations": {
      "en_us": {
          "LBL_AI_SCORE_DESC": "Ai Score Description"
      },
      ...
    },
```

```
    "data":{
      "name":"ai_score_desc",
      "label":"LBL_AI_SCORE_DESC",
      "type":"varchar",
      "help":"Text Field Help Text",
      "comments":"Text Field Comment Text",
      "default_value":"",
      "len":255,
      "required":false,
      "reportable":true,
      "audited":false,
      "importable":"true",
      "duplicate_merge":false
    }
}
```

This example will create an enum type custom field with a new dropdown list:

```
  {
    "localizations": {
      "en_us": {
          "LBL_DROPDOWN_FIELD": "New Dropdown Field",
          "LBL_DD_ITEM_ONE": "First Text",
          "LBL_DD_ITEM_TWO": "Second Text",
          "LBL_DD_ITEM_THREE": "Third Text"
      },
      ...
    },
    "data": {
      "name":"new_dropdown_field",
      "label":"LBL_DROPDOWN_FIELD",
      "type":"enum",
      "options": {
        "dropdownName": "a_new_dropdown",
        "dropdownList": [
          {"value":"First", "label":"LBL_DD_ITEM_ONE"},
          {"value":"Second", "label":"LBL_DD_ITEM_TWO"},
          {"value":"Third", "label":"LBL_DD_ITEM_THREE"}
        ]
      },
      "default_value":"First"
    }
}
```

This example will create an enum type custom field using an existing dropdown list:

```
  {
    "localizations": {
      "en_us": {
          "LBL_DROPDOWN_FIELD": "New Dropdown Field"
      },
      ...
    },
    "data": {
      "name":"new_dropdown_field",
      "label":"LBL_DROPDOWN_FIELD",
      "type":"enum",
      "options": "account_type_dom",
      "default_value":"Analyst"
    }
}
```

Response

```
  {
    "duplicate_merge_dom_value": 0,
    "required": false,
    "source": "custom_fields",
    "name": "ai_score_c",
    "vname": "LBL_AI_SCORE",
    "type": "decimal",
    "massupdate": false,
    "hidemassupdate": false,
    "no_default": false,
    "comments": "",
    "help": "",
    "importable": "",
    "duplicate_merge": "disabled",
    "audited": false,
    "reportable": false,
    "unified_search": false,
    "merge_filter": "disabled",
    "pii": false,
    "calculated": false,
    "len": 18,
    "size": "20",
    "enable_range_search": false,
```

```
    "precision": "8",
    "id": "ca9653de-0976-11eb-a558-6c400895ea84",
    "custom_module": "Opportunities"
}
```


Change Log

Version	Change
v11.11	Added /<module>/customfield POST endpoint.

//customfield/:field DELETE

Overview

Delete a custom field for a specified module.

Summary

This endpoint is used to delete a custom field for a specified module.

Request Arguments

This endpoint does not accept any request arguments.

Response


```
  {
    "name": "ai_conv_score_c"
}
```


Response Codes and Error Messages

All successful responses should return 200.

This API returns the following error responses:

- **403 (SugarAPIExceptionNotAuthorized)**: Sent if the user attempts to access a module which they are forbidden to access.
- **404 (SugarApiExceptionNot Found)**

: Sent if the requested field is not found.

- **422 (SugarApiExceptionInvalidParameter)**: Sent if the module and/or field are invalid.
- **500 (SugarApiExceptionError)**: Sent if there's an unknown error that causes the custom field to not be deleted in the module.

Change Log

Version	Change
v11.11	Added /<module>/customfield/:field DELETE endpoint.

//duplicateCheck POST

Overview

Runs a duplicate check against specified data.

Request Arguments

Name	Type	Description	Required
<record field>	<record field type>	The name value list of fields to populate.	True

Request

```
{
    "name":"Airline"
}
```

Response Arguments

Name	Type	Description
<record field>	<record field type>	Returns the potential duplicate records.

Response

```
{
   "next_offset":-1,
   "records":[
      {
         "id":"c4c26496-5dbc-67dd-bf5c-512d0923889e",
         "name":"Airline Maintenance Co",
         "date_entered":"2013-02-26T19:12:00+00:00",
         "date_modified":"2013-02-26T19:12:00+00:00",
         "modified_user_id":"1",
         "modified_by_name":"Administrator",
         "created_by":"1",
         "created_by_name":"Administrator",
         "description":"",
         "img":"",
         "last_activity_date":"2013-02-26T19:12:00+00:00",
         "deleted":false,
         "assigned_user_id":"seed_sally_id",
         "assigned_user_name":"Sally Bronsen",
         "team_name":[
            {
               "id":"East",
               "name":"East",
               "name_2":"",
               "primary":false
            },
            {
               "id":1,
               "name":"Global",
               "name_2":"",
               "primary":false
            },
            {
               "id":"West",
               "name":"West",
               "name_2":"",
               "primary":true
            }
         ],
         "linkedin":"",
         "facebook":"",
         "twitter":"",
         "googleplus":"",
         "account_type":"Customer",
         "industry":"Other",
         "annual_revenue":"",
         "phone_fax":"",
```

```
"billing_address_street":"123 Anywhere Street",
"billing_address_street_2":"",
"billing_address_street_3":"",
"billing_address_street_4":"",
"billing_address_city":"Sunnyvale",
"billing_address_state":"CA",
"billing_address_postalcode":"48566",
"billing_address_country":"USA",
"rating":"",
"phone_office":"(648) 452-3486",
"phone_alternate":"",
"website":"www.kidqa.it",
"ownership":"",
"employees":"",
"ticker_symbol":"",
"shipping_address_street":"123 Anywhere Street",
"shipping_address_street_2":"",
"shipping_address_street_3":"",
"shipping_address_street_4":"",
"shipping_address_city":"Sunnyvale",
"shipping_address_state":"CA",
"shipping_address_postalcode":"48566",
"shipping_address_country":"USA",
"email1":"vegan.im@example.tw",
"parent_id":"",
"sic_code":"",
"parent_name":"",
"email_opt_out":false,
"invalid_email":false,
"email":[
   {
      "email_address":"vegan.im@example.tw",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"1"
   },
   {
      "email_address":"phone85@example.tv",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"0"
   }
],
"campaign_id":"",
"campaign_name":"",
"my_favorite":false,
```

```
         "_acl":{
            "fields":{

            }
         },
         "duplicate_check_rank":0
      }
   ]
}
```


Change Log

Version	Change
v10	Added /<module>/duplicateCheck POST endpoint.

//enum/:field GET

Overview

Retrieves the enum values for a specific field.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

Name	Type	Description
<list values>	Array	Returns the enum values for a field.

Response


```
{
   "":"",
   "Analyst":"Analyst",
   "Competitor":"Competitor",
   "Customer":"Customer",
   "Integrator":"Integrator",
   "Investor":"Investor",
```

```
   "Partner":"Partner",
   "Press":"Press",
   "Prospect":"Prospect",
   "Reseller":"Reseller",
   "Other":"Other"
}
```

Change Log

Version	Change
v10	Added /<module>/enum/:field GET endpoint.

//export/:record_list_id GET

Overview

Returns a record set in CSV format along with HTTP headers to indicate content type.

Request Arguments

Name	Type	Description	Required
uid	Array	A list of bean ids.	False
filter	Array	The filter expression. More information on filter expressions can be found in /<module>/filter.	False
sample	Array	Indicates whether to download sample data.	False

Request

Exporting Records by Specific IDs

```
{
   "uid":"d43243c6-9b8e-2973-aee2-512d09bc34b4"
}
```

Exporting Records by a List of IDs

```
{
   "uid":[
      "d43243c6-9b8e-2973-aee2-512d09bc34b4",
      "b3e87a3f-cd8f-7b86-467a-512d09e8d240"
   ]
}
```

Exporting Records Using a Filter

```
{
   "filter":[
      {
         "name":"airline"
      }
   ]
}
```

Exporting a Sample Result Set

```
{
    "sample":true
}
```

Response Arguments

Name	Type	Description
<csv export>	String	Returns the selected records in CSV format with the content headers.

Response

```
result: HTTP/1.1 200 OK
Date: Fri, 01 Mar 2013 04:05:55 GMT
Server: Apache/2.2.22 (Unix) DAV/2 PHP/5.3.17
X-Powered-By: PHP/5.3.17
Expires: Mon, 26 Jul 1997 05:00:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-
check=0
Pragma: cache
Content-Disposition: attachment; filename=Accounts.csv
Content-transfer-encoding: binary
Last-Modified: Fri, 01 Mar 2013 04:05:55 GMT
Cache-Control: post-check=0, pre-check=0
Content-Length: 1080
Connection: close
Content-Type: application/octet-stream; charset=UTF-8

"Name","ID","Website","Email Address","Office Phone","Alternate Phone"
,"Fax","Billing Street","Billing City","Billing State","Billing Postal
 Code","Billing Country","Shipping Street","Shipping City","Shipping S
tate","Shipping Postal Code","Shipping Country","Description","Type","
Industry","Annual Revenue","Employees","SIC Code","Ticker Symbol","Par
ent Account ID","Ownership","Campaign ID","Rating","Assigned User Name
","Assigned To","Team ID","Teams","Team Set ID","Date Created","Date M
odified","Modified By","Created By","Deleted","Image","last_activity_d
ate","Linkedin Company ID","Facebook Account","Twitter Account","Googl
e Plus ID"
"Arts & Crafts Inc","d43243c6-9b8e-2973-aee2-512d09bc34b4","","","(052
) 034-1853","","","777 West Filmore Ln","Santa Monica","CA","35354","U
SA","777 West Filmore Ln","Santa Monica","CA","35354","USA","","Custom
er","Transportation","","","","","","","","","sally","seed_sally_id","
West","West","West","02/26/2013 07:12 pm","02/26/2013 07:12 pm","1","1
","0","","02/26/2013 07:12 pm","","","",""
```

Change Log

Version	Change
v10	Added /<module>/export GET endpoint.

//favorites GET

Overview

Opportunity Favorites Help

Summary

This endpoint returns all the favorite Quotes/Revenue Line Items in alphabetical order to be displayed in the "Product Catalog Quick Picks" dashlet.

Request Arguments

Name	Type	Description	Required
pageNum	integer	An integer attribute in the payload data object which gets the page number for which the items need to be displayed based on the current page or the page clicked.	False

Request

```
{
    pageNum: 0
}
```

Response Arguments

Name	Type	Description
<config field>	<config field type>	The list of config fields will be returned.

Response

```
{
    "totalPages":5,
    "pageNum":0,
    "max_num":8,
    "records":[
```

```
                {
                    "id":"c181bd32-c7f2-11e8-b4eb-7831c1c14620",
                    "name":"Allyson Gadget",
                    "date_entered":"2018-10-04 16:28:37",
                    "date_modified":"2018-10-04 16:28:37",
                    "modified_user_id":"1",
                    "created_by":"1",
                    "description":null,
                    "deleted":0,
                    "type_id":null,
                    "manufacturer_id":"c03f952a-
c7f2-11e8-b410-7831c1c14620",
                    "category_id":"c18112d8-c7f2-11e8-b753-7831c1c1462
0",
                    "mft_part_num":"Ink Conglomerate Inc 921535XYZ987"
,
                    "vendor_part_num":null,
                    "date_cost_price":null,
                    "cost_price":"534.000000",
                    "discount_price":"842.000000",
                    "list_price":"905.000000",
                    "cost_usdollar":"534.000000",
                    "discount_usdollar":"842.000000",
                    "list_usdollar":"905.000000",
                    "status":"Available",
                    "tax_class":"Taxable",
                    "date_available":"2004-10-15",
                    "website":null,
                    "weight":"20.00",
                    "qty_in_stock":26,
                    "support_name":null,
                    "support_description":null,
                    "support_contact":null,
                    "support_term":null,
                    "pricing_formula":"Fixed",
                    "pricing_factor":"1.00",
                    "assigned_user_id":null,
                    "currency_id":"-99",
                    "base_rate":"1.000000"
                },
                {
                    "id":"c18e0448-c7f2-11e8-86b0-7831c1c14620",
                    "name":"Arla Gadget",
                    "date_entered":"2018-10-04 16:28:37",
                    "date_modified":"2018-10-04 16:28:37",
                    "modified_user_id":"1",
```

```
                    "created_by":"1",
                    "description":null,
                    "deleted":0,
                    "type_id":null,
                    "manufacturer_id":"c03f952a-
c7f2-11e8-b410-7831c1c14620",
                    "category_id":"c18b8b64-c7f2-11e8-8c1a-7831c1c1462
0",
                    "mft_part_num":"24\/7 Couriers 795391XYZ987",
                    "vendor_part_num":null,
                    "date_cost_price":null,
                    "cost_price":"398.000000",
                    "discount_price":"402.020202",
                    "list_price":"941.000000",
                    "cost_usdollar":"398.000000",
                    "discount_usdollar":"402.020202",
                    "list_usdollar":"941.000000",
                    "status":"Available",
                    "tax_class":"Taxable",
                    "date_available":"2004-10-15",
                    "website":null,
                    "weight":"22.00",
                    "qty_in_stock":145,
                    "support_name":null,
                    "support_description":null,
                    "support_contact":null,
                    "support_term":null,
                    "pricing_formula":"ProfitMargin",
                    "pricing_factor":"1.00",
                    "assigned_user_id":null,
                    "currency_id":"-99",
                    "base_rate":"1.000000"
                },
                {
                    "id":"c1b60434-c7f2-11e8-8964-7831c1c14620",
                    "name":"Billie Gadget",
                    "date_entered":"2018-10-04 16:28:37",
                    "date_modified":"2018-10-04 16:28:37",
                    "modified_user_id":"1",
                    "created_by":"1",
                    "description":null,
                    "deleted":0,
                    "type_id":null,
                    "manufacturer_id":"c03f952a-
c7f2-11e8-b410-7831c1c14620",
                    "category_id":"c1b3bf58-c7f2-11e8-92e7-7831c1c1462
```

```
0",
                    "mft_part_num":"Kringle Bell IncK.A. Tower \u0026
Co 694143XYZ987",
                    "vendor_part_num":null,
                    "date_cost_price":null,
                    "cost_price":"403.000000",
                    "discount_price":"927.000000",
                    "list_price":"927.000000",
                    "cost_usdollar":"403.000000",
                    "discount_usdollar":"927.000000",
                    "list_usdollar":"927.000000",
                    "status":"Available",
                    "tax_class":"Taxable",
                    "date_available":"2004-10-15",
                    "website":null,
                    "weight":"16.00",
                    "qty_in_stock":148,
                    "support_name":null,
                    "support_description":null,
                    "support_contact":null,
                    "support_term":null,
                    "pricing_formula":"IsList",
                    "pricing_factor":"1.00",
                    "assigned_user_id":null,
                    "currency_id":"-99",
                    "base_rate":"1.000000"
                },
                {
                    "id":"c186a612-c7f2-11e8-9fee-7831c1c14620",
                    "name":"Bradford Gadget",
                    "date_entered":"2018-10-04 16:28:37",
                    "date_modified":"2018-10-04 16:28:37",
                    "modified_user_id":"1",
                    "created_by":"1",
                    "description":null,
                    "deleted":0,
                    "type_id":null,
                    "manufacturer_id":"c03f952a-
c7f2-11e8-b410-7831c1c14620",
                    "category_id":"c185dbb0-c7f2-11e8-a6b7-7831c1c1462
0",
                    "mft_part_num":"Mississippi Bank Group 133834XYZ98
7",
                    "vendor_part_num":null,
                    "date_cost_price":null,
                    "cost_price":"528.000000",
```

```
                    "discount_price":"533.333333",
                    "list_price":"745.000000",
                    "cost_usdollar":"528.000000",
                    "discount_usdollar":"533.333333",
                    "list_usdollar":"745.000000",
                    "status":"Available",
                    "tax_class":"Taxable",
                    "date_available":"2004-10-15",
                    "website":null,
                    "weight":"33.00",
                    "qty_in_stock":70,
                    "support_name":null,
                    "support_description":null,
                    "support_contact":null,
                    "support_term":null,
                    "pricing_formula":"ProfitMargin",
                    "pricing_factor":"1.00",
                    "assigned_user_id":null,
                    "currency_id":"-99",
                    "base_rate":"1.000000"
                },
                {
                    "id":"c1b4680e-c7f2-11e8-9380-7831c1c14620",
                    "name":"Brain Gadget",
                    "date_entered":"2018-10-04 16:28:37",
                    "date_modified":"2018-10-04 16:28:37",
                    "modified_user_id":"1",
                    "created_by":"1",
                    "description":null,
                    "deleted":0,
                    "type_id":null,
                    "manufacturer_id":"c03f952a-
c7f2-11e8-b410-7831c1c14620",
                    "category_id":"c1b3bf58-c7f2-11e8-92e7-7831c1c1462
0",
                    "mft_part_num":"Smallville Resources Inc 362299XYZ
987",
                    "vendor_part_num":null,
                    "date_cost_price":null,
                    "cost_price":"370.000000",
                    "discount_price":"854.370000",
                    "list_price":"863.000000",
                    "cost_usdollar":"370.000000",
                    "discount_usdollar":"854.370000",
                    "list_usdollar":"863.000000",
                    "status":"Available",
```

```
                    "tax_class":"Taxable",
                    "date_available":"2004-10-15",
                    "website":null,
                    "weight":"10.00",
                    "qty_in_stock":106,
                    "support_name":null,
                    "support_description":null,
                    "support_contact":null,
                    "support_term":null,
                    "pricing_formula":"PercentageDiscount",
                    "pricing_factor":"1.00",
                    "assigned_user_id":null,
                    "currency_id":"-99",
                    "base_rate":"1.000000"
                },
                {
                    "id":"c1baf0e8-c7f2-11e8-ab05-7831c1c14620",
                    "name":"Carley Gadget",
                    "date_entered":"2018-10-04 16:28:37",
                    "date_modified":"2018-10-04 16:28:37",
                    "modified_user_id":"1",
                    "created_by":"1",
                    "description":null,
                    "deleted":0,
                    "type_id":null,
                    "manufacturer_id":"c03f952a-
c7f2-11e8-b410-7831c1c14620",
                    "category_id":"c1b8e4ec-
c7f2-11e8-b435-7831c1c14620",
                    "mft_part_num":"Hollywood Diner Ltd 10023XYZ987",
                    "vendor_part_num":null,
                    "date_cost_price":null,
                    "cost_price":"554.000000",
                    "discount_price":"559.595960",
                    "list_price":"927.000000",
                    "cost_usdollar":"554.000000",
                    "discount_usdollar":"559.595959",
                    "list_usdollar":"927.000000",
                    "status":"Available",
                    "tax_class":"Taxable",
                    "date_available":"2004-10-15",
                    "website":null,
                    "weight":"30.00",
                    "qty_in_stock":58,
                    "support_name":null,
                    "support_description":null,
```

```
                    "support_contact":null,
                    "support_term":null,
                    "pricing_formula":"ProfitMargin",
                    "pricing_factor":"1.00",
                    "assigned_user_id":null,
                    "currency_id":"-99",
                    "base_rate":"1.000000"
                },
                {
                    "id":"c1b77544-c7f2-11e8-8347-7831c1c14620",
                    "name":"Carma Gadget",
                    "date_entered":"2018-10-04 16:28:37",
                    "date_modified":"2018-10-04 16:28:37",
                    "modified_user_id":"1",
                    "created_by":"1",
                    "description":null,
                    "deleted":0,
                    "type_id":null,
                    "manufacturer_id":"c03f952a-
c7f2-11e8-b410-7831c1c14620",
                    "category_id":"c1b3bf58-c7f2-11e8-92e7-7831c1c1462
0",
                    "mft_part_num":"Sea Region Inc 773472XYZ987",
                    "vendor_part_num":null,
                    "date_cost_price":null,
                    "cost_price":"332.000000",
                    "discount_price":"745.000000",
                    "list_price":"822.000000",
                    "cost_usdollar":"332.000000",
                    "discount_usdollar":"745.000000",
                    "list_usdollar":"822.000000",
                    "status":"Available",
                    "tax_class":"Taxable",
                    "date_available":"2004-10-15",
                    "website":null,
                    "weight":"35.00",
                    "qty_in_stock":1,
                    "support_name":null,
                    "support_description":null,
                    "support_contact":null,
                    "support_term":null,
                    "pricing_formula":"Fixed",
                    "pricing_factor":"1.00",
                    "assigned_user_id":null,
                    "currency_id":"-99",
                    "base_rate":"1.000000"
```

```
                },
                {
                    "id":"c1b98c9e-c7f2-11e8-9740-7831c1c14620",
                    "name":"Catrina Gadget",
                    "date_entered":"2018-10-04 16:28:37",
                    "date_modified":"2018-10-04 16:28:37",
                    "modified_user_id":"1",
                    "created_by":"1",
                    "description":null,
                    "deleted":0,
                    "type_id":null,
                    "manufacturer_id":"c03f952a-
c7f2-11e8-b410-7831c1c14620",
                    "category_id":"c1b8e4ec-
c7f2-11e8-b435-7831c1c14620",
                    "mft_part_num":"Jungle Systems Inc 273034XYZ987",
                    "vendor_part_num":null,
                    "date_cost_price":null,
                    "cost_price":"577.000000",
                    "discount_price":"582.770000",
                    "list_price":"946.000000",
                    "cost_usdollar":"577.000000",
                    "discount_usdollar":"582.770000",
                    "list_usdollar":"946.000000",
                    "status":"Available",
                    "tax_class":"Taxable",
                    "date_available":"2004-10-15",
                    "website":null,
                    "weight":"13.00",
                    "qty_in_stock":105,
                    "support_name":null,
                    "support_description":null,
                    "support_contact":null,
                    "support_term":null,
                    "pricing_formula":"PercentageMarkup",
                    "pricing_factor":"1.00",
                    "assigned_user_id":null,
                    "currency_id":"-99",
                    "base_rate":"1.000000"
                }
            ]
}
```

Change Log

Version	Change
v10	Added /Opportunities/favorites GET endpoint.

//favorites POST

Overview

Opportunity Favorites Help

Summary

This endpoint returns all the favorite Quotes/Revenue Line Items in alphabetical order to be displayed in the "Product Catalog Quick Picks" dashlet.

Request Arguments

Name	Type	Description	Required
pageNum	integer	An integer attribute in the payload data object which gets the page number for which the items need to be displayed based on the current page or the page clicked.	False

Request

```
{
    pageNum: 0
}
```

Response Arguments

Name	Type	Description
<config field>	<config field type>	The list of config fields will be returned.

Response


```
{
    "totalPages":5,
    "pageNum":0,
    "max_num":8,
    "records":[
                {
                    "id":"c181bd32-c7f2-11e8-b4eb-7831c1c14620",
                    "name":"Allyson Gadget",
                    "date_entered":"2018-10-04 16:28:37",
                    "date_modified":"2018-10-04 16:28:37",
                    "modified_user_id":"1",
                    "created_by":"1",
                    "description":null,
                    "deleted":0,
                    "type_id":null,
                    "manufacturer_id":"c03f952a-
c7f2-11e8-b410-7831c1c14620",
                    "category_id":"c18112d8-c7f2-11e8-b753-7831c1c1462
0",
                    "mft_part_num":"Ink Conglomerate Inc 921535XYZ987"
,
                    "vendor_part_num":null,
                    "date_cost_price":null,
                    "cost_price":"534.000000",
                    "discount_price":"842.000000",
                    "list_price":"905.000000",
                    "cost_usdollar":"534.000000",
                    "discount_usdollar":"842.000000",
                    "list_usdollar":"905.000000",
                    "status":"Available",
                    "tax_class":"Taxable",
                    "date_available":"2004-10-15",
                    "website":null,
                    "weight":"20.00",
                    "qty_in_stock":26,
                    "support_name":null,
                    "support_description":null,
                    "support_contact":null,
                    "support_term":null,
```

```
                    "pricing_formula":"Fixed",
                    "pricing_factor":"1.00",
                    "assigned_user_id":null,
                    "currency_id":"-99",
                    "base_rate":"1.000000"
                },
                {
                    "id":"c18e0448-c7f2-11e8-86b0-7831c1c14620",
                    "name":"Arla Gadget",
                    "date_entered":"2018-10-04 16:28:37",
                    "date_modified":"2018-10-04 16:28:37",
                    "modified_user_id":"1",
                    "created_by":"1",
                    "description":null,
                    "deleted":0,
                    "type_id":null,
                    "manufacturer_id":"c03f952a-
c7f2-11e8-b410-7831c1c14620",
                    "category_id":"c18b8b64-c7f2-11e8-8c1a-7831c1c1462
0",
                    "mft_part_num":"24\/7 Couriers 795391XYZ987",
                    "vendor_part_num":null,
                    "date_cost_price":null,
                    "cost_price":"398.000000",
                    "discount_price":"402.020202",
                    "list_price":"941.000000",
                    "cost_usdollar":"398.000000",
                    "discount_usdollar":"402.020202",
                    "list_usdollar":"941.000000",
                    "status":"Available",
                    "tax_class":"Taxable",
                    "date_available":"2004-10-15",
                    "website":null,
                    "weight":"22.00",
                    "qty_in_stock":145,
                    "support_name":null,
                    "support_description":null,
                    "support_contact":null,
                    "support_term":null,
                    "pricing_formula":"ProfitMargin",
                    "pricing_factor":"1.00",
                    "assigned_user_id":null,
                    "currency_id":"-99",
                    "base_rate":"1.000000"
                },
                {
```

```
                    "id":"c1b60434-c7f2-11e8-8964-7831c1c14620",
                    "name":"Billie Gadget",
                    "date_entered":"2018-10-04 16:28:37",
                    "date_modified":"2018-10-04 16:28:37",
                    "modified_user_id":"1",
                    "created_by":"1",
                    "description":null,
                    "deleted":0,
                    "type_id":null,
                    "manufacturer_id":"c03f952a-
c7f2-11e8-b410-7831c1c14620",
                    "category_id":"c1b3bf58-c7f2-11e8-92e7-7831c1c1462
0",
                    "mft_part_num":"Kringle Bell IncK.A. Tower \u0026
Co 694143XYZ987",
                    "vendor_part_num":null,
                    "date_cost_price":null,
                    "cost_price":"403.000000",
                    "discount_price":"927.000000",
                    "list_price":"927.000000",
                    "cost_usdollar":"403.000000",
                    "discount_usdollar":"927.000000",
                    "list_usdollar":"927.000000",
                    "status":"Available",
                    "tax_class":"Taxable",
                    "date_available":"2004-10-15",
                    "website":null,
                    "weight":"16.00",
                    "qty_in_stock":148,
                    "support_name":null,
                    "support_description":null,
                    "support_contact":null,
                    "support_term":null,
                    "pricing_formula":"IsList",
                    "pricing_factor":"1.00",
                    "assigned_user_id":null,
                    "currency_id":"-99",
                    "base_rate":"1.000000"
                },
                {
                    "id":"c186a612-c7f2-11e8-9fee-7831c1c14620",
                    "name":"Bradford Gadget",
                    "date_entered":"2018-10-04 16:28:37",
                    "date_modified":"2018-10-04 16:28:37",
                    "modified_user_id":"1",
                    "created_by":"1",
```

```
                    "description":null,
                    "deleted":0,
                    "type_id":null,
                    "manufacturer_id":"c03f952a-
c7f2-11e8-b410-7831c1c14620",
                    "category_id":"c185dbb0-c7f2-11e8-a6b7-7831c1c1462
0",
                    "mft_part_num":"Mississippi Bank Group 133834XYZ98
7",
                    "vendor_part_num":null,
                    "date_cost_price":null,
                    "cost_price":"528.000000",
                    "discount_price":"533.333333",
                    "list_price":"745.000000",
                    "cost_usdollar":"528.000000",
                    "discount_usdollar":"533.333333",
                    "list_usdollar":"745.000000",
                    "status":"Available",
                    "tax_class":"Taxable",
                    "date_available":"2004-10-15",
                    "website":null,
                    "weight":"33.00",
                    "qty_in_stock":70,
                    "support_name":null,
                    "support_description":null,
                    "support_contact":null,
                    "support_term":null,
                    "pricing_formula":"ProfitMargin",
                    "pricing_factor":"1.00",
                    "assigned_user_id":null,
                    "currency_id":"-99",
                    "base_rate":"1.000000"
                },
                {
                    "id":"c1b4680e-c7f2-11e8-9380-7831c1c14620",
                    "name":"Brain Gadget",
                    "date_entered":"2018-10-04 16:28:37",
                    "date_modified":"2018-10-04 16:28:37",
                    "modified_user_id":"1",
                    "created_by":"1",
                    "description":null,
                    "deleted":0,
                    "type_id":null,
                    "manufacturer_id":"c03f952a-
c7f2-11e8-b410-7831c1c14620",
                    "category_id":"c1b3bf58-c7f2-11e8-92e7-7831c1c1462
```

```
0",
                    "mft_part_num":"Smallville Resources Inc 362299XYZ
987",
                    "vendor_part_num":null,
                    "date_cost_price":null,
                    "cost_price":"370.000000",
                    "discount_price":"854.370000",
                    "list_price":"863.000000",
                    "cost_usdollar":"370.000000",
                    "discount_usdollar":"854.370000",
                    "list_usdollar":"863.000000",
                    "status":"Available",
                    "tax_class":"Taxable",
                    "date_available":"2004-10-15",
                    "website":null,
                    "weight":"10.00",
                    "qty_in_stock":106,
                    "support_name":null,
                    "support_description":null,
                    "support_contact":null,
                    "support_term":null,
                    "pricing_formula":"PercentageDiscount",
                    "pricing_factor":"1.00",
                    "assigned_user_id":null,
                    "currency_id":"-99",
                    "base_rate":"1.000000"
                },
                {
                    "id":"c1baf0e8-c7f2-11e8-ab05-7831c1c14620",
                    "name":"Carley Gadget",
                    "date_entered":"2018-10-04 16:28:37",
                    "date_modified":"2018-10-04 16:28:37",
                    "modified_user_id":"1",
                    "created_by":"1",
                    "description":null,
                    "deleted":0,
                    "type_id":null,
                    "manufacturer_id":"c03f952a-
c7f2-11e8-b410-7831c1c14620",
                    "category_id":"c1b8e4ec-
c7f2-11e8-b435-7831c1c14620",
                    "mft_part_num":"Hollywood Diner Ltd 10023XYZ987",
                    "vendor_part_num":null,
                    "date_cost_price":null,
                    "cost_price":"554.000000",
                    "discount_price":"559.595960",
```

```
                    "list_price":"927.000000",
                    "cost_usdollar":"554.000000",
                    "discount_usdollar":"559.595959",
                    "list_usdollar":"927.000000",
                    "status":"Available",
                    "tax_class":"Taxable",
                    "date_available":"2004-10-15",
                    "website":null,
                    "weight":"30.00",
                    "qty_in_stock":58,
                    "support_name":null,
                    "support_description":null,
                    "support_contact":null,
                    "support_term":null,
                    "pricing_formula":"ProfitMargin",
                    "pricing_factor":"1.00",
                    "assigned_user_id":null,
                    "currency_id":"-99",
                    "base_rate":"1.000000"
                },
                {
                    "id":"c1b77544-c7f2-11e8-8347-7831c1c14620",
                    "name":"Carma Gadget",
                    "date_entered":"2018-10-04 16:28:37",
                    "date_modified":"2018-10-04 16:28:37",
                    "modified_user_id":"1",
                    "created_by":"1",
                    "description":null,
                    "deleted":0,
                    "type_id":null,
                    "manufacturer_id":"c03f952a-
c7f2-11e8-b410-7831c1c14620",
                    "category_id":"c1b3bf58-c7f2-11e8-92e7-7831c1c1462
0",
                    "mft_part_num":"Sea Region Inc 773472XYZ987",
                    "vendor_part_num":null,
                    "date_cost_price":null,
                    "cost_price":"332.000000",
                    "discount_price":"745.000000",
                    "list_price":"822.000000",
                    "cost_usdollar":"332.000000",
                    "discount_usdollar":"745.000000",
                    "list_usdollar":"822.000000",
                    "status":"Available",
                    "tax_class":"Taxable",
                    "date_available":"2004-10-15",
```

```
                    "website":null,
                    "weight":"35.00",
                    "qty_in_stock":1,
                    "support_name":null,
                    "support_description":null,
                    "support_contact":null,
                    "support_term":null,
                    "pricing_formula":"Fixed",
                    "pricing_factor":"1.00",
                    "assigned_user_id":null,
                    "currency_id":"-99",
                    "base_rate":"1.000000"
                },
                {
                    "id":"c1b98c9e-c7f2-11e8-9740-7831c1c14620",
                    "name":"Catrina Gadget",
                    "date_entered":"2018-10-04 16:28:37",
                    "date_modified":"2018-10-04 16:28:37",
                    "modified_user_id":"1",
                    "created_by":"1",
                    "description":null,
                    "deleted":0,
                    "type_id":null,
                    "manufacturer_id":"c03f952a-
c7f2-11e8-b410-7831c1c14620",
                    "category_id":"c1b8e4ec-
c7f2-11e8-b435-7831c1c14620",
                    "mft_part_num":"Jungle Systems Inc 273034XYZ987",
                    "vendor_part_num":null,
                    "date_cost_price":null,
                    "cost_price":"577.000000",
                    "discount_price":"582.770000",
                    "list_price":"946.000000",
                    "cost_usdollar":"577.000000",
                    "discount_usdollar":"582.770000",
                    "list_usdollar":"946.000000",
                    "status":"Available",
                    "tax_class":"Taxable",
                    "date_available":"2004-10-15",
                    "website":null,
                    "weight":"13.00",
                    "qty_in_stock":105,
                    "support_name":null,
                    "support_description":null,
                    "support_contact":null,
                    "support_term":null,
```

```
                    "pricing_formula":"PercentageMarkup",
                    "pricing_factor":"1.00",
                    "assigned_user_id":null,
                    "currency_id":"-99",
                    "base_rate":"1.000000"
                }
            ]
}
```


Change Log

Version	Change
v10	Added /Opportunities/favorites POST endpoint.

//file/vcard_import POST

Overview

Imports a person record from a vcard.

Request Arguments

Name	Type	Description	Required
format	String	The data format. Currently accepts 'sugar-html-json'.	True
module	String	The name of the module to import the vcard to.	True
file	String	The vcard contents.	True

Request


```
{
   "format":"sugar-html-json",
   "module":"Contacts"
   "file":"@\/path\/to\/ExampleContact.vcf"
}
```

Response Arguments

Name	Type	Description
file	String	The id of the imported vcard.

Response

```
{
   "file":"2c9e5c09-6824-0d14-f5cb-5130321ac3cf"
}
```

Change Log

Version	Change
v10	Added /<module>/file/vcard_import POST endpoint.

//filter GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

Name	Type	Description	Required
filter	String	The filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query parameters in one of two ways for GET requests: 1. By specifying individual filter arguments as distinct parameters. Example: filter[0][id]=1. 2. By specifying the whole filter as a single JSON-encoded string. Note that this syntax is currently not supported on certain modules. Example: filter=[{"id":"1"}].	False
filter_id	String	Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND.	False

Name	Type	Description	Required
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
fields	String	Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters (applicable to link and collection fields). The fields id and date_modified will always be returned. Example: name,account_type,description,{"name":"opportunities","fields":["id","name","sales_status"],"order_by":"date_closed:desc"} For more details on additional field parameters, see Relate API and Collection API.	False
view	String	Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the	False

Name	Type	Description	Required
		server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list". Example: record	
order_by	String	How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon. Example: name:DESC,account_type:DESC,date_modified:ASC	False
q	String	A search expression, will search on this module. Cannot be used at the same time as a filter expression or id.	False
deleted	Boolean	Boolean to show deleted records in the result set.	False
nulls_last	Boolean	Boolean to return records with null values in order_by fields last in the result set.	False

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
   "filter":[
      {
         "name":"Nelson Inc"
      }
   ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
   "filter":[
      {
         "name":{
            "$starts":"Nelson"
         }
      }
   ]
}
```

Below is a list of operation types:

Operation	Description
$equals	Performs an exact match on that field.
$not_equals	Performs an exact match on that field.
$not_equals	Matches on non-matching values.
$starts	Matches on anything that starts with the

Operation		Description
		value.
	$ends	Matches anything that ends with the value.
	$contains	Matches anything that contains the value
	$in	Finds anything where field matches one of the values as specified as an array.
	$not_in	Finds anything where field does not matches any of the values as specified as an array.
	$is_null	Checks if the field is null. This operation does not need a value specified.
	$not_null	Checks if the field is not null. This operation does not need a value specified.
	$lt	Matches when the field is less than the value.
	$lte	Matches when the field is less than or equal to the value.
	$gt	Matches when the field is greater than the value.
	$gte	Matches when the field is greater than or equal to the value.

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "$and" and "$or".

Example

```
{
   "filter":[
      {
         "$or":[
            {
               "name":"Nelson Inc"
            },
            {
               "name":"Nelson LLC"
            }
         ]
      }
   ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "$favorite" and "$owner".

Example

```
{
   "filter":[
      {
         "$favorite":"_this"
      }
   ]
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Response


```
{
   "next_offset":-1,
   "records":[
      {
         "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
         "name":"Dale Spivey",
         "date_modified":"2013-02-28T05:03:00+00:00",
         "description":"",
         "opportunities": [
            {
               _module: "Opportunities",
               "id": "b0701501-1fab-8ae7-3942-540da93f5017",
               "name": "360 Vacations - 228 Units",
               "date_modified": "2014-09-08T16:05:00+03:00",
               "sales_status": "New"
            },
         ],
         "_acl": {
            "fields": {
            }
         }
      },
      {
         "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
         "name":"Florence Haddock",
         "date_modified":"2013-02-26T19:12:00+00:00",
         "description":"",
         "opportunities": [
            {
               _module: "Opportunities"
               date_modified: "2014-09-08T16:05:00+03:00"
```

```
                id: "9ce7c088-8ee4-7cd3-18f1-540da944d4c0"
                name: "360 Vacations - 312 Units"
                sales_status: "New"
             },
          ],
          "_acl": {
             "fields": {
             }
          }
       }
    ]
}
```

Change Log

Version	Change
v10	Added /<module>/filter GET endpoint.

//filter POST

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

Name	Type	Description	Required
filter	String	The filter expression. Filter	False

Name	Type	Description	Required
		expressions are explained below.	
filter_id	String	Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND.	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
fields	String	Comma delimited list of fields to return. The field date_modified will always be returned. Example: name,account_type,description	False
view	String	Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list".	False

Name	Type	Description	Required
		Example: record	
order_by	String	How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon. Example: name:DESC,account_type:DESC,date_modified:ASC	False

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
   "filter":[
      {
         "name":"Nelson Inc"
      }
   ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
   "filter":[
```

```
      {
         "name":{
            "$starts":"Nelson"
         }
      }
   ]
}
```

Below is a list of operation types:

Operation	Description
$equals	Performs an exact match on that field.
$not_equals	Performs an exact match on that field.
$not_equals	Matches on non-matching values.
$starts	Matches on anything that starts with the value.
$ends	Matches anything that ends with the value.
$contains	Matches anything that contains the value
$in	Finds anything where field matches one of the values as specified as an array.
$not_in	Finds anything where field does not matches any of the values as specified as an array.
$is_null	Checks if the field is null. This operation does not need a value specified.
$not_null	Checks if the field is not null. This operation does not need a value

Operation		Description	
			specified.
	$lt		Matches when the field is less than the value.
	$lte		Matches when the field is less than or equal to the value.
	$gt		Matches when the field is greater than the value.
	$gte		Matches when the field is greater than or equal to the value.

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "$and" and "$or".

Example

```
{
   "filter":[
      {
         "$or":[
            {
               "name":"Nelson Inc"
            },
            {
               "name":"Nelson LLC"
            }
         ]
      }
   ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "$favorite" and "$owner".

Example

```
{
   "filter":[
      {
         "$favorite":"_this"
      }
   ]
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Response

```
{
   "next_offset":-1,
   "records":[
      {
         "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
         "name":"Dale Spivey",
         "date_entered":"2013-02-26T19:12:00+00:00",
         "date_modified":"2013-02-28T05:03:00+00:00",
         "modified_user_id":"1",
         "modified_by_name":"Administrator",
         "created_by":"1",
         "created_by_name":"Administrator",
```

```
"description":"",
"img":"",
"deleted":false,
"assigned_user_id":"seed_sally_id",
"assigned_user_name":"Sally Bronsen",
"team_name":[
   {
      "id":"East",
      "name":"East",
      "name_2":"",
      "primary":false
   },
   {
      "id":1,
      "name":"Global",
      "name_2":"",
      "primary":false
   },
   {
      "id":"West",
      "name":"West",
      "name_2":"",
      "primary":true
   }
],
"salutation":"",
"first_name":"Dale",
"last_name":"Spivey",
"full_name":"Dale Spivey",
"title":"VP Operations",
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(523) 825-4311",
"email":[
   {
      "email_address":"sugar.dev.sugar@example.co.jp",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"1"
   },
   {
      "email_address":"the.support@example.biz",
```

```
            "opt_out":"0",
            "invalid_email":"0",
            "primary_address":"0"
         }
      ],
      "phone_mobile":"(373) 861-0757",
      "phone_work":"(212) 542-9596",
      "phone_other":"",
      "phone_fax":"",
      "email1":"sugar.dev.sugar@example.co.jp",
      "email2":"the.support@example.biz",
      "invalid_email":false,
      "email_opt_out":false,
      "primary_address_street":"345 Sugar Blvd.",
      "primary_address_street_2":"",
      "primary_address_street_3":"",
      "primary_address_city":"Denver",
      "primary_address_state":"CA",
      "primary_address_postalcode":"87261",
      "primary_address_country":"USA",
      "alt_address_street":"",
      "alt_address_street_2":"",
      "alt_address_street_3":"",
      "alt_address_city":"",
      "alt_address_state":"",
      "alt_address_postalcode":"",
      "alt_address_country":"",
      "assistant":"",
      "assistant_phone":"",
      "picture":"",
      "email_and_name1":"",
      "lead_source":"Campaign",
      "account_name":"Smallville Resources Inc",
      "account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",
      "opportunity_role_fields":"",
      "opportunity_role_id":"",
      "opportunity_role":"",
      "reports_to_id":"",
      "report_to_name":"",
      "portal_name":"DaleSpivey97",
      "portal_active":true,
      "portal_password":"$1$yKMAONHM$Y5S.8CY.WZCZCwfGD1a1Q\/",
      "portal_password1":"",
      "portal_app":"",
      "preferred_language":"en_us",
      "campaign_id":"",
```

```
      "campaign_name":"",
      "c_accept_status_fields":"",
      "m_accept_status_fields":"",
      "accept_status_id":"",
      "accept_status_name":"",
      "sync_contact":"",
      "my_favorite":false,
      "_acl":{
         "fields":{

         }
      }
   },
   {
      "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
      "name":"Florence Haddock",
      "date_entered":"2013-02-26T19:12:00+00:00",
      "date_modified":"2013-02-26T19:12:00+00:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "description":"",
      "img":"",
      "deleted":false,
      "assigned_user_id":"seed_sally_id",
      "assigned_user_name":"Sally Bronsen",
      "team_name":[
         {
            "id":"East",
            "name":"East",
            "name_2":"",
            "primary":false
         },
         {
            "id":1,
            "name":"Global",
            "name_2":"",
            "primary":false
         },
         {
            "id":"West",
            "name":"West",
            "name_2":"",
            "primary":true
         }
```

```
],
"salutation":"",
"first_name":"Florence",
"last_name":"Haddock",
"full_name":"Florence Haddock",
"title":"Director Sales",
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(729) 845-3137",
"email":[
   {
      "email_address":"dev.vegan@example.de",
      "opt_out":"1",
      "invalid_email":"0",
      "primary_address":"0"
   },
   {
      "email_address":"section71@example.it",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"1"
   }
],
"phone_mobile":"(246) 233-1382",
"phone_work":"(565) 696-6981",
"phone_other":"",
"phone_fax":"",
"email1":"section71@example.it",
"email2":"dev.vegan@example.de",
"invalid_email":false,
"email_opt_out":false,
"primary_address_street":"111 Silicon Valley Road",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Denver",
"primary_address_state":"CA",
"primary_address_postalcode":"79900",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
```

```
         "alt_address_state":"",
         "alt_address_postalcode":"",
         "alt_address_country":"",
         "assistant":"",
         "assistant_phone":"",
         "picture":"",
         "email_and_name1":"",
         "lead_source":"Support Portal User Registration",
         "account_name":"Smallville Resources Inc",
         "account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",
         "opportunity_role_fields":"",
         "opportunity_role_id":"",
         "opportunity_role":"",
         "reports_to_id":"",
         "report_to_name":"",
         "portal_name":"FlorenceHaddock169",
         "portal_active":true,
         "portal_password":"$1$nWFhTbK6$JF9BCGSqL\/NCrbhueX5ia0",
         "portal_password1":"",
         "portal_app":"",
         "preferred_language":"en_us",
         "campaign_id":"",
         "campaign_name":"",
         "c_accept_status_fields":"",
         "m_accept_status_fields":"",
         "accept_status_id":"",
         "accept_status_name":"",
         "sync_contact":"",
         "my_favorite":false,
         "_acl":{
            "fields":{

            }
         }
      }
   ]
}
```


Change Log

Version	Change
v10	Added /<module>/filter POST

	endpoint.

//filter/count GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

Name	Type	Description	Required
filter	String	The filter expression. Filter expressions are explained below.	False
filter_id	String	Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND.	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
fields	String	Comma delimited	False

Name	Type	Description	Required
		list of fields to return. The field date_modified will always be returned. Example: name,account_type,description	
view	String	Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list". Example: record	False
order_by	String	How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon. Example: name:DESC,account_type:DESC,date_modified:ASC	False

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
   "filter":[
      {
         "name":"Nelson Inc"
      }
   ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
   "filter":[
      {
         "name":{
            "$starts":"Nelson"
         }
      }
   ]
}
```

Below is a list of operation types:

Operation	Description
$equals	Performs an exact match on that field.
$not_equals	Performs an exact match on that field.
$not_equals	Matches on non-matching values.
$starts	Matches on anything that starts with the value.

Operation		Description
	$ends	Matches anything that ends with the value.
	$contains	Matches anything that contains the value
	$in	Finds anything where field matches one of the values as specified as an array.
	$not_in	Finds anything where field does not matches any of the values as specified as an array.
	$is_null	Checks if the field is null. This operation does not need a value specified.
	$not_null	Checks if the field is not null. This operation does not need a value specified.
	$lt	Matches when the field is less than the value.
	$lte	Matches when the field is less than or equal to the value.
	$gt	Matches when the field is greater than the value.
	$gte	Matches when the field is greater than or equal to the value.

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all

expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "$and" and "$or".

Example

```
{
   "filter":[
      {
         "$or":[
            {
               "name":"Nelson Inc"
            },
            {
               "name":"Nelson LLC"
            }
         ]
      }
   ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "$favorite" and "$owner".

Example

```
{
   "filter":[
      {
         "$favorite":"_this"
      }
   ]
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Response


```
{
   "next_offset":-1,
   "records":[
      {
         "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
         "name":"Dale Spivey",
         "date_entered":"2013-02-26T19:12:00+00:00",
         "date_modified":"2013-02-28T05:03:00+00:00",
         "modified_user_id":"1",
         "modified_by_name":"Administrator",
         "created_by":"1",
         "created_by_name":"Administrator",
         "description":"",
         "img":"",
         "deleted":false,
         "assigned_user_id":"seed_sally_id",
         "assigned_user_name":"Sally Bronsen",
         "team_name":[
            {
               "id":"East",
               "name":"East",
               "name_2":"",
               "primary":false
            },
            {
               "id":1,
               "name":"Global",
               "name_2":"",
               "primary":false
            },
            {
               "id":"West",
               "name":"West",
```

```
            "name_2":"",
            "primary":true
         }
      ],
      "salutation":"",
      "first_name":"Dale",
      "last_name":"Spivey",
      "full_name":"Dale Spivey",
      "title":"VP Operations",
      "linkedin":"",
      "facebook":"",
      "twitter":"",
      "googleplus":"",
      "department":"",
      "do_not_call":false,
      "phone_home":"(523) 825-4311",
      "email":[
         {
            "email_address":"sugar.dev.sugar@example.co.jp",
            "opt_out":"0",
            "invalid_email":"0",
            "primary_address":"1"
         },
         {
            "email_address":"the.support@example.biz",
            "opt_out":"0",
            "invalid_email":"0",
            "primary_address":"0"
         }
      ],
      "phone_mobile":"(373) 861-0757",
      "phone_work":"(212) 542-9596",
      "phone_other":"",
      "phone_fax":"",
      "email1":"sugar.dev.sugar@example.co.jp",
      "email2":"the.support@example.biz",
      "invalid_email":false,
      "email_opt_out":false,
      "primary_address_street":"345 Sugar Blvd.",
      "primary_address_street_2":"",
      "primary_address_street_3":"",
      "primary_address_city":"Denver",
      "primary_address_state":"CA",
      "primary_address_postalcode":"87261",
      "primary_address_country":"USA",
      "alt_address_street":"",
```

```
      "alt_address_street_2":"",
      "alt_address_street_3":"",
      "alt_address_city":"",
      "alt_address_state":"",
      "alt_address_postalcode":"",
      "alt_address_country":"",
      "assistant":"",
      "assistant_phone":"",
      "picture":"",
      "email_and_name1":"",
      "lead_source":"Campaign",
      "account_name":"Smallville Resources Inc",
      "account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",
      "opportunity_role_fields":"",
      "opportunity_role_id":"",
      "opportunity_role":"",
      "reports_to_id":"",
      "report_to_name":"",
      "portal_name":"DaleSpivey97",
      "portal_active":true,
      "portal_password":"$1$yKMAONHM$Y5S.8CY.WZCZCwfGD1a1Q\/",
      "portal_password1":"",
      "portal_app":"",
      "preferred_language":"en_us",
      "campaign_id":"",
      "campaign_name":"",
      "c_accept_status_fields":"",
      "m_accept_status_fields":"",
      "accept_status_id":"",
      "accept_status_name":"",
      "sync_contact":"",
      "my_favorite":false,
      "_acl":{
         "fields":{

         }
      }
   },
   {
      "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
      "name":"Florence Haddock",
      "date_entered":"2013-02-26T19:12:00+00:00",
      "date_modified":"2013-02-26T19:12:00+00:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
```

```
"created_by_name":"Administrator",
"description":"",
"img":"",
"deleted":false,
"assigned_user_id":"seed_sally_id",
"assigned_user_name":"Sally Bronsen",
"team_name":[
   {
      "id":"East",
      "name":"East",
      "name_2":"",
      "primary":false
   },
   {
      "id":1,
      "name":"Global",
      "name_2":"",
      "primary":false
   },
   {
      "id":"West",
      "name":"West",
      "name_2":"",
      "primary":true
   }
],
"salutation":"",
"first_name":"Florence",
"last_name":"Haddock",
"full_name":"Florence Haddock",
"title":"Director Sales",
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(729) 845-3137",
"email":[
   {
      "email_address":"dev.vegan@example.de",
      "opt_out":"1",
      "invalid_email":"0",
      "primary_address":"0"
   },
   {
```

```
      "email_address":"section71@example.it",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"1"
   }
],
"phone_mobile":"(246) 233-1382",
"phone_work":"(565) 696-6981",
"phone_other":"",
"phone_fax":"",
"email1":"section71@example.it",
"email2":"dev.vegan@example.de",
"invalid_email":false,
"email_opt_out":false,
"primary_address_street":"111 Silicon Valley Road",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Denver",
"primary_address_state":"CA",
"primary_address_postalcode":"79900",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
"assistant":"",
"assistant_phone":"",
"picture":"",
"email_and_name1":"",
"lead_source":"Support Portal User Registration",
"account_name":"Smallville Resources Inc",
"account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",
"opportunity_role_fields":"",
"opportunity_role_id":"",
"opportunity_role":"",
"reports_to_id":"",
"report_to_name":"",
"portal_name":"FlorenceHaddock169",
"portal_active":true,
"portal_password":"$1$nWFhTbK6$JF9BCGSqL\/NCrbhueX5ia0",
"portal_password1":"",
"portal_app":"",
"preferred_language":"en_us",
```

```
         "campaign_id":"",
         "campaign_name":"",
         "c_accept_status_fields":"",
         "m_accept_status_fields":"",
         "accept_status_id":"",
         "accept_status_name":"",
         "sync_contact":"",
         "my_favorite":false,
         "_acl":{
            "fields":{

            }
         }
      }
   ]
}
```

Change Log

Version	Change
v10	Added /<module>/filter POST endpoint.

//filter/count POST

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

Name	Type	Description	Required
filter	String	The filter expression. Filter expressions are explained below.	False
filter_id	String	Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND.	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
fields	String	Comma delimited list of fields to return. The field date_modified will always be returned. Example: name,account_type,description	False
view	String	Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields	False

Name	Type	Description	Required
		argument. Common views are "record" and "list". Example: record	
order_by	String	How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon. Example: name:DESC,account_type:DESC,date_modified:ASC	False

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
   "filter":[
      {
         "name":"Nelson Inc"
      }
   ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
   "filter":[
      {
         "name":{
            "$starts":"Nelson"
         }
      }
   ]
}
```

Below is a list of operation types:

Operation	Description
$equals	Performs an exact match on that field.
$not_equals	Performs an exact match on that field.
$not_equals	Matches on non-matching values.
$starts	Matches on anything that starts with the value.
$ends	Matches anything that ends with the value.
$contains	Matches anything that contains the value
$in	Finds anything where field matches one of the values as specified as an array.
$not_in	Finds anything where field does not matches any of the values as specified as an array.
$is_null	Checks if the field is null. This operation does not need a value specified.
$not_null	Checks if the field is

Operation		Description
		not null. This operation does not need a value specified.
	$lt	Matches when the field is less than the value.
	$lte	Matches when the field is less than or equal to the value.
	$gt	Matches when the field is greater than the value.
	$gte	Matches when the field is greater than or equal to the value.

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "$and" and "$or".

Example

```
{
   "filter":[
      {
         "$or":[
            {
               "name":"Nelson Inc"
            },
            {
               "name":"Nelson LLC"
            }
         ]
      }
   ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "$favorite" and "$owner".

Example

```
{
   "filter":[
      {
         "$favorite":"_this"
      }
   ]
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Response

```
{
   "next_offset":-1,
   "records":[
      {
         "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
         "name":"Dale Spivey",
         "date_entered":"2013-02-26T19:12:00+00:00",
```

```
"date_modified":"2013-02-28T05:03:00+00:00",
"modified_user_id":"1",
"modified_by_name":"Administrator",
"created_by":"1",
"created_by_name":"Administrator",
"description":"",
"img":"",
"deleted":false,
"assigned_user_id":"seed_sally_id",
"assigned_user_name":"Sally Bronsen",
"team_name":[
   {
      "id":"East",
      "name":"East",
      "name_2":"",
      "primary":false
   },
   {
      "id":1,
      "name":"Global",
      "name_2":"",
      "primary":false
   },
   {
      "id":"West",
      "name":"West",
      "name_2":"",
      "primary":true
   }
],
"salutation":"",
"first_name":"Dale",
"last_name":"Spivey",
"full_name":"Dale Spivey",
"title":"VP Operations",
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(523) 825-4311",
"email":[
   {
      "email_address":"sugar.dev.sugar@example.co.jp",
      "opt_out":"0",
```

```
            "invalid_email":"0",
            "primary_address":"1"
        },
        {
            "email_address":"the.support@example.biz",
            "opt_out":"0",
            "invalid_email":"0",
            "primary_address":"0"
        }
    ],
    "phone_mobile":"(373) 861-0757",
    "phone_work":"(212) 542-9596",
    "phone_other":"",
    "phone_fax":"",
    "email1":"sugar.dev.sugar@example.co.jp",
    "email2":"the.support@example.biz",
    "invalid_email":false,
    "email_opt_out":false,
    "primary_address_street":"345 Sugar Blvd.",
    "primary_address_street_2":"",
    "primary_address_street_3":"",
    "primary_address_city":"Denver",
    "primary_address_state":"CA",
    "primary_address_postalcode":"87261",
    "primary_address_country":"USA",
    "alt_address_street":"",
    "alt_address_street_2":"",
    "alt_address_street_3":"",
    "alt_address_city":"",
    "alt_address_state":"",
    "alt_address_postalcode":"",
    "alt_address_country":"",
    "assistant":"",
    "assistant_phone":"",
    "picture":"",
    "email_and_name1":"",
    "lead_source":"Campaign",
    "account_name":"Smallville Resources Inc",
    "account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",
    "opportunity_role_fields":"",
    "opportunity_role_id":"",
    "opportunity_role":"",
    "reports_to_id":"",
    "report_to_name":"",
    "portal_name":"DaleSpivey97",
    "portal_active":true,
```

```
      "portal_password":"$1$yKMAONHM$Y5S.8CY.WZCZCwfGD1a1Q\/",
      "portal_password1":"",
      "portal_app":"",
      "preferred_language":"en_us",
      "campaign_id":"",
      "campaign_name":"",
      "c_accept_status_fields":"",
      "m_accept_status_fields":"",
      "accept_status_id":"",
      "accept_status_name":"",
      "sync_contact":"",
      "my_favorite":false,
      "_acl":{
         "fields":{

         }
      }
   },
   {
      "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
      "name":"Florence Haddock",
      "date_entered":"2013-02-26T19:12:00+00:00",
      "date_modified":"2013-02-26T19:12:00+00:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "description":"",
      "img":"",
      "deleted":false,
      "assigned_user_id":"seed_sally_id",
      "assigned_user_name":"Sally Bronsen",
      "team_name":[
         {
            "id":"East",
            "name":"East",
            "name_2":"",
            "primary":false
         },
         {
            "id":1,
            "name":"Global",
            "name_2":"",
            "primary":false
         },
         {
```

```
      "id":"West",
      "name":"West",
      "name_2":"",
      "primary":true
   }
],
"salutation":"",
"first_name":"Florence",
"last_name":"Haddock",
"full_name":"Florence Haddock",
"title":"Director Sales",
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(729) 845-3137",
"email":[
   {
      "email_address":"dev.vegan@example.de",
      "opt_out":"1",
      "invalid_email":"0",
      "primary_address":"0"
   },
   {
      "email_address":"section71@example.it",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"1"
   }
],
"phone_mobile":"(246) 233-1382",
"phone_work":"(565) 696-6981",
"phone_other":"",
"phone_fax":"",
"email1":"section71@example.it",
"email2":"dev.vegan@example.de",
"invalid_email":false,
"email_opt_out":false,
"primary_address_street":"111 Silicon Valley Road",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Denver",
"primary_address_state":"CA",
"primary_address_postalcode":"79900",
```

```
         "primary_address_country":"USA",
         "alt_address_street":"",
         "alt_address_street_2":"",
         "alt_address_street_3":"",
         "alt_address_city":"",
         "alt_address_state":"",
         "alt_address_postalcode":"",
         "alt_address_country":"",
         "assistant":"",
         "assistant_phone":"",
         "picture":"",
         "email_and_name1":"",
         "lead_source":"Support Portal User Registration",
         "account_name":"Smallville Resources Inc",
         "account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",
         "opportunity_role_fields":"",
         "opportunity_role_id":"",
         "opportunity_role":"",
         "reports_to_id":"",
         "report_to_name":"",
         "portal_name":"FlorenceHaddock169",
         "portal_active":true,
         "portal_password":"$1$nWFhTbK6$JF9BCGSqL\/NCrbhueX5ia0",
         "portal_password1":"",
         "portal_app":"",
         "preferred_language":"en_us",
         "campaign_id":"",
         "campaign_name":"",
         "c_accept_status_fields":"",
         "m_accept_status_fields":"",
         "accept_status_id":"",
         "accept_status_name":"",
         "sync_contact":"",
         "my_favorite":false,
         "_acl":{
            "fields":{

            }
         }
      }
   ]
}
```

Change Log

Version	Change
v10	Added /<module>/filter POST endpoint.

//globalsearch GET

Overview

Global search

Summary

This endpoint exposes the global search capability using solely the Elasticsearch backend as an alternative to the /search endpoint. This endpoint can be used with a long list of request arguments. Instead of using GET request arguments, all described arguments can be used inside a JSON encoded request body using a the GET request. If your client has no support for GET requests with a body, the POST method can be used as an alternative.

Request Arguments

Name	Type	Description	Required
q	String	The search expression. Multiple terms can be specified at once. All enabled fields will be searched in. The results are ordered by relevance which is based on a multitude of settings based on token counts, hit ratio, (weighted) boost values and type of field. Currently no operators are supported in the search expression	False

Name	Type	Description	Required
		itself. By refining the search expression more relevant results will be returned as top results. If no search expression is given results are returned based on last modified date.	
module_list	String	Comma delimited list of modules to search. If omitted, all search enabled modules will be queried. Note that when consuming the endpoint /:module/globalsearch that this parameter is ignored. Example: Accounts,Contacts	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
highlights	Boolean	Wether or not to return highlighted results. Default is true.	False
sort	Array	Define the sort order of the results. By default the results are returned by relevance which is the preferred	False

Name	Type	Description	Required
		approach. Using this argument any search enabled field can be used to sort on. Keep in mind the not sorting by relevance may have a negative performance impact. Example: {"date_modified":"desc","name":"asc"}	

Request

```
{
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Response

```
{
}
```

Change Log

Version	Change
v10	Added /globalsearch GET/POST endpoint.
v10	Added /:module/globalsearch GET/POST endpoint.

//globalsearch POST

Overview

Global search

Summary

This endpoint exposes the global search capability using solely the Elasticsearch backend as an alternative to the /search endpoint. This endpoint can be used with a long list of request arguments. Instead of using GET request arguments, all described arguments can be used inside a JSON encoded request body using a the GET request. If your client has no support for GET requests with a body, the POST method can be used as an alternative.

Request Arguments

Name	Type	Description	Required
q	String	The search expression. Multiple terms can be specified at once. All enabled fields will be searched in. The results are ordered by relevance which is based on a multitude of settings based on token counts, hit ratio, (weighted) boost values and type of field. Currently no operators are	False

Name	Type	Description	Required
		supported in the search expression itself. By refining the search expression more relevant results will be returned as top results. If no search expression is given results are returned based on last modified date.	
module_list	String	Comma delimited list of modules to search. If omitted, all search enabled modules will be queried. Note that when consuming the endpoint /:module/globalsearch that this parameter is ignored. Example: Accounts,Contacts	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
highlights	Boolean	Wether or not to return highlighted results. Default is true.	False
sort	Array	Define the sort order of the results. By default the results are returned by	False

Name	Type	Description	Required
		relevance which is the preferred approach. Using this argument any search enabled field can be used to sort on. Keep in mind the not sorting by relevance may have a negative performance impact. Example: {"date_modified":"desc","name":"asc"}	

Request

```
{
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Response

```
{
}
```

Change Log

Version	Change
v10	Added /globalsearch GET/POST endpoint.
v10	Added /:module/globalsearch GET/POST endpoint.

//insertafter/:target POST

Overview

Insert new node after target node.

Request Arguments

Name	Type	Description	Required
module	String	The name of sugar module that contains a nested set data and implements the NestedSetInterface.	True
target	String	The ID of record that will be used as target to insert new node after	True

Request

```
{
    "name" : "Children Node 1"
}
```

Response Arguments

This endpoint does not return any response arguments.

Response

This endpoint returns a newly created bean
{ "my_favorite":false, "following":"", "id":"59fa8dd7-0f2c-4bfd-364f-54495f77fa3f", "name":"Default title", "date_entered":"2014-10-23T23:03:22+03:00", "date_modified":"2014-10-23T23:03:22+03:00", "modified_user_id":"1", "modified_by_name":"Administrator", "created_by":"1", "created_by_name":"Administrator", "doc_owner":"", "description":"", "deleted":false, "source_id":"", "source_type":"", "source_meta":"", "root":"be9b0c4a-8b78-1ffa-4f14-54481c2f6269", "lft":118, "rgt":119, "level":1, "_acl":{"fields":{}}, "_module":"Categories" }

Change Log

Version	Change
v10	Added /<module>/insertafter/:target POST endpoint.

//insertbefore/:target POST

Overview

Insert new node before target node.

Request Arguments

Name	Type	Description	Required
module	String	The name of sugar module that contains a nested set data and implements the NestedSetInterface.	True
target	String	The ID of record that will be used as target to insert new node before	True

Request

```
{
    "name" : "Children Node 1"
}
```

Response Arguments

This endpoint does not return any response arguments.

Response

This endpoint returns a newly created bean
{ "my_favorite":false, "following":"", "id":"59fa8dd7-0f2c-4bfd-364f-54495f77fa3f", "name":"Default title", "date_entered":"2014-10-23T23:03:22+03:00", "date_modified":"2014-10-23T23:03:22+03:00", "modified_user_id":"1", "modified_by_name":"Administrator", "created_by":"1", "created_by_name":"Administrator", "doc_owner":"", "description":"", "deleted":false, "source_id":"", "source_type":"", "source_meta":"", "root":"be9b0c4a-8b78-1ffa-4f14-54481c2f6269", "lft":118, "rgt":119, "level":1, "_acl":{"fields":{}}, "_module":"Categories" }

Change Log

Version	Change
v10	Added /<module>/insertbefore/:target POST endpoint.

//prepend/:target POST

Overview

Append new node to target node as first child.

Request Arguments

Name	Type	Description	Required
module	String	The name of sugar module that contains a nested set data and implements the NestedSetInterface.	True
target	String	The ID of record that will be used as target to prepend	True

Name	Type	Description	Required
		new node	

Request

```
{
    "name" : "Children Node 1"
}
```

Response Arguments

This endpoint does not return any response arguments.

Response

This enpdoint returns a newly created record GUID
{ "my_favorite":false, "following":"", "id":"59fa8dd7-0f2c-4bfd-364f-54495f77fa3f", "name":"Default title", "date_entered":"2014-10-23T23:03:22+03:00", "date_modified":"2014-10-23T23:03:22+03:00", "modified_user_id":"1", "modified_by_name":"Administrator", "created_by":"1", "created_by_name":"Administrator", "doc_owner":"", "description":"", "deleted":false, "source_id":"", "source_type":"", "source_meta":"", "root":"be9b0c4a-8b78-1ffa-4f14-54481c2f6269", "lft":118, "rgt":119, "level":1, "_acl":{"fields":{}}, "_module":"Categories" }

Change Log

Version	Change
v10	Added /<module>/prepend/:target POST endpoint.

//recent-product GET

Overview

Opportunity Recent Product Help

Summary

This endpoint retrieves top 10 recently used Quoted/Revenue Line Items based on the module to display them in the "Product Catalog Quick Picks" dashlet in the

reverse chronological order.

Request Arguments

None

Response Arguments

NONE

Response


```
{
    "next_offset":-1,

    "records":[
                {
                    "id":"c1b4680e-c7f2-11e8-9380-7831c1c14620",

                    "name":"Brain Gadget",

                    "date_entered":"2018-10-04 16:28:37",

                    "date_modified":"2018-10-04 16:28:37",

                    "modified_user_id":"1",

                    "created_by":"1",

                    "description":null,

                    "deleted":0,

                    "type_id":null,

                    "manufacturer_id":"c03f952a-
c7f2-11e8-b410-7831c1c14620",

                    "category_id":"c1b3bf58-c7f2-11e8-92e7-7831c1c1462
0",

                    "mft_part_num":"Smallville Resources Inc 362299XYZ
987",

                    "vendor_part_num":null,
```

```
            "date_cost_price":null,
            "cost_price":"370.000000",
            "discount_price":"854.370000",
            "list_price":"863.000000",
            "cost_usdollar":"370.000000",
            "discount_usdollar":"854.370000",
            "list_usdollar":"863.000000",
            "status":"Available",
            "tax_class":"Taxable",
            "date_available":"2004-10-15",
            "website":null,
            "weight":"10.00",
            "qty_in_stock":106,
            "support_name":null,
            "support_description":null,
            "support_contact":null,
            "support_term":null,
            "pricing_formula":"PercentageDiscount",
            "pricing_factor":"1.00",
            "assigned_user_id":null,
            "currency_id":"-99",
            "Base_rate":"1.000000"
        },
        {
```

```
                    "id":"c186a612-c7f2-11e8-9fee-7831c1c14620",

                    "name":"Bradford Gadget",

                    "date_entered":"2018-10-04 16:28:37",

                    "date_modified":"2018-10-04 16:28:37",

                    "modified_user_id":"1",

                    "created_by":"1",

                    "description":null,

                    "deleted":0,

                    "type_id":null,

                    "manufacturer_id":"c03f952a-
c7f2-11e8-b410-7831c1c14620",

                    "category_id":"c185dbb0-c7f2-11e8-a6b7-7831c1c1462
0",

                    "mft_part_num":"Mississippi Bank Group 133834XYZ98
7",

                    "vendor_part_num":null,

                    "date_cost_price":null,

                    "cost_price":"528.000000",

                    "discount_price":"533.333333",

                    "list_price":"745.000000",

                    "cost_usdollar":"528.000000",

                    "discount_usdollar":"533.333333",

                    "list_usdollar":"745.000000",

                    "status":"Available",

                    "tax_class":"Taxable",
```

```
        "date_available":"2004-10-15",
        "website":null,
        "weight":"33.00",
        "qty_in_stock":70,
        "support_name":null,
        "support_description":null,
        "support_contact":null,
        "support_term":null,
        "pricing_formula":"ProfitMargin",
        "pricing_factor":"1.00",
        "assigned_user_id":null,
        "currency_id":"-99",
        "Base_rate":"1.000000"
    },
    {
        "id":"c1b60434-c7f2-11e8-8964-7831c1c14620",
        "name":"Billie Gadget",
        "date_entered":"2018-10-04 16:28:37",
        "date_modified":"2018-10-04 16:28:37",
        "modified_user_id":"1",
        "created_by":"1",
        "description":null,
        "deleted":0,
        "type_id":null,
```

```
                    "manufacturer_id":"c03f952a-
c7f2-11e8-b410-7831c1c14620",

                    "category_id":"c1b3bf58-c7f2-11e8-92e7-7831c1c1462
0",

                    "mft_part_num":"Kringle Bell IncK.A. Tower \u0026
Co 694143XYZ987",

                    "vendor_part_num":null,

                    "date_cost_price":null,

                    "cost_price":"403.000000",

                    "discount_price":"927.000000",

                    "list_price":"927.000000",

                    "cost_usdollar":"403.000000",

                    "discount_usdollar":"927.000000",

                    "list_usdollar":"927.000000",

                    "status":"Available",

                    "tax_class":"Taxable",

                    "date_available":"2004-10-15",

                    "website":null,

                    "weight":"16.00",

                    "qty_in_stock":148,

                    "support_name":null,

                    "support_description":null,

                    "support_contact":null,

                    "support_term":null,
```

```
                    "pricing_formula":"IsList",
                    "pricing_factor":"1.00",
                    "assigned_user_id":null,
                    "currency_id":"-99",
                    "Base_rate":"1.000000"
                },
                {
                    "id":"c18e0448-c7f2-11e8-86b0-7831c1c14620",
                    "name":"Arla Gadget",
                    "date_entered":"2018-10-04 16:28:37",
                    "date_modified":"2018-10-04 16:28:37",
                    "modified_user_id":"1",
                    "created_by":"1",
                    "description":null,
                    "deleted":0,
                    "type_id":null,
                    "manufacturer_id":"c03f952a-
c7f2-11e8-b410-7831c1c14620",
                    "category_id":"c18b8b64-c7f2-11e8-8c1a-7831c1c1462
0",
                    "mft_part_num":"24\/7 Couriers 795391XYZ987",
                    "vendor_part_num":null,
                    "date_cost_price":null,
                    "cost_price":"398.000000",
                    "discount_price":"402.020202",
```

```
        "list_price":"941.000000",
        "cost_usdollar":"398.000000",
        "discount_usdollar":"402.020202",
        "list_usdollar":"941.000000",
        "status":"Available",
        "tax_class":"Taxable",
        "date_available":"2004-10-15",
        "website":null,
        "weight":"22.00",
        "qty_in_stock":145,
        "support_name":null,
        "support_description":null,
        "support_contact":null,
        "support_term":null,
        "pricing_formula":"ProfitMargin",
        "pricing_factor":"1.00",
        "assigned_user_id":null,
        "currency_id":"-99",
        "Base_rate":"1.000000"
    },
    {
        "id":"c181bd32-c7f2-11e8-b4eb-7831c1c14620",
        "name":"Allyson Gadget",
        "date_entered":"2018-10-04 16:28:37",
```

```
                    "date_modified":"2018-10-04 16:28:37",

                    "modified_user_id":"1",

                    "created_by":"1",

                    "description":null,

                    "deleted":0,

                    "type_id":null,

                    "manufacturer_id":"c03f952a-
c7f2-11e8-b410-7831c1c14620",

                    "category_id":"c18112d8-c7f2-11e8-b753-7831c1c1462
0",

                    "mft_part_num":"Ink Conglomerate Inc 921535XYZ987"
,

                    "vendor_part_num":null,

                    "date_cost_price":null,

                    "cost_price":"534.000000",

                    "discount_price":"842.000000",

                    "list_price":"905.000000",

                    "cost_usdollar":"534.000000",

                    "discount_usdollar":"842.000000",

                    "list_usdollar":"905.000000",

                    "status":"Available",

                    "tax_class":"Taxable",

                    "date_available":"2004-10-15",

                    "website":null,

                    "weight":"20.00",
```

```
                    "qty_in_stock":26,

                    "support_name":null,

                    "support_description":null,

                    "support_contact":null,

                    "support_term":null,

                    "pricing_formula":"Fixed",

                    "pricing_factor":"1.00",

                    "assigned_user_id":null,

                    "currency_id":"-99",

                    "base_rate":"1.000000"
                }
            ]
}
```


Change Log

Version	Change
v10	Added /Opportunities/recent_product GET endpoint.

//recent-product POST

Overview

Opportunity Recent Product Help

Summary

This endpoint retrieves top 10 recently used Quoted/Revenue Line Items based on the module to display them in the "Product Catalog Quick Picks" dashlet in the reverse chronological order.

Request Arguments

None

Response Arguments

NONE

Response


```
{
    "next_offset":-1,

    "records":[
                {
                    "id":"c1b4680e-c7f2-11e8-9380-7831c1c14620",

                    "name":"Brain Gadget",

                    "date_entered":"2018-10-04 16:28:37",

                    "date_modified":"2018-10-04 16:28:37",

                    "modified_user_id":"1",

                    "created_by":"1",

                    "description":null,

                    "deleted":0,

                    "type_id":null,

                    "manufacturer_id":"c03f952a-
c7f2-11e8-b410-7831c1c14620",

                    "category_id":"c1b3bf58-c7f2-11e8-92e7-7831c1c1462
0",

                    "mft_part_num":"Smallville Resources Inc 362299XYZ
987",

                    "vendor_part_num":null,

                    "date_cost_price":null,
```

```
            "cost_price":"370.000000",
            "discount_price":"854.370000",
            "list_price":"863.000000",
            "cost_usdollar":"370.000000",
            "discount_usdollar":"854.370000",
            "list_usdollar":"863.000000",
            "status":"Available",
            "tax_class":"Taxable",
            "date_available":"2004-10-15",
            "website":null,
            "weight":"10.00",
            "qty_in_stock":106,
            "support_name":null,
            "support_description":null,
            "support_contact":null,
            "support_term":null,
            "pricing_formula":"PercentageDiscount",
            "pricing_factor":"1.00",
            "assigned_user_id":null,
            "currency_id":"-99",
            "Base_rate":"1.000000"
        },
        {
            "id":"c186a612-c7f2-11e8-9fee-7831c1c14620",
```

```
                    "name":"Bradford Gadget",

                    "date_entered":"2018-10-04 16:28:37",

                    "date_modified":"2018-10-04 16:28:37",

                    "modified_user_id":"1",

                    "created_by":"1",

                    "description":null,

                    "deleted":0,

                    "type_id":null,

                    "manufacturer_id":"c03f952a-
c7f2-11e8-b410-7831c1c14620",

                    "category_id":"c185dbb0-c7f2-11e8-a6b7-7831c1c1462
0",

                    "mft_part_num":"Mississippi Bank Group 133834XYZ98
7",

                    "vendor_part_num":null,

                    "date_cost_price":null,

                    "cost_price":"528.000000",

                    "discount_price":"533.333333",

                    "list_price":"745.000000",

                    "cost_usdollar":"528.000000",

                    "discount_usdollar":"533.333333",

                    "list_usdollar":"745.000000",

                    "status":"Available",

                    "tax_class":"Taxable",

                    "date_available":"2004-10-15",
```

```
        "website":null,
        "weight":"33.00",
        "qty_in_stock":70,
        "support_name":null,
        "support_description":null,
        "support_contact":null,
        "support_term":null,
        "pricing_formula":"ProfitMargin",
        "pricing_factor":"1.00",
        "assigned_user_id":null,
        "currency_id":"-99",
        "Base_rate":"1.000000"
    },
    {
        "id":"c1b60434-c7f2-11e8-8964-7831c1c14620",
        "name":"Billie Gadget",
        "date_entered":"2018-10-04 16:28:37",
        "date_modified":"2018-10-04 16:28:37",
        "modified_user_id":"1",
        "created_by":"1",
        "description":null,
        "deleted":0,
        "type_id":null,
        "manufacturer_id":"c03f952a-
```

```
c7f2-11e8-b410-7831c1c14620",
                    "category_id":"c1b3bf58-c7f2-11e8-92e7-7831c1c1462
0",
                    "mft_part_num":"Kringle Bell IncK.A. Tower \u0026
Co 694143XYZ987",
                    "vendor_part_num":null,
                    "date_cost_price":null,
                    "cost_price":"403.000000",
                    "discount_price":"927.000000",
                    "list_price":"927.000000",
                    "cost_usdollar":"403.000000",
                    "discount_usdollar":"927.000000",
                    "list_usdollar":"927.000000",
                    "status":"Available",
                    "tax_class":"Taxable",
                    "date_available":"2004-10-15",
                    "website":null,
                    "weight":"16.00",
                    "qty_in_stock":148,
                    "support_name":null,
                    "support_description":null,
                    "support_contact":null,
                    "support_term":null,
                    "pricing_formula":"IsList",
```

```
                    "pricing_factor":"1.00",

                    "assigned_user_id":null,

                    "currency_id":"-99",

                    "Base_rate":"1.000000"
                },

                {
                    "id":"c18e0448-c7f2-11e8-86b0-7831c1c14620",

                    "name":"Arla Gadget",

                    "date_entered":"2018-10-04 16:28:37",

                    "date_modified":"2018-10-04 16:28:37",

                    "modified_user_id":"1",

                    "created_by":"1",

                    "description":null,

                    "deleted":0,

                    "type_id":null,

                    "manufacturer_id":"c03f952a-
c7f2-11e8-b410-7831c1c14620",

                    "category_id":"c18b8b64-c7f2-11e8-8c1a-7831c1c1462
0",

                    "mft_part_num":"24\/7 Couriers 795391XYZ987",

                    "vendor_part_num":null,

                    "date_cost_price":null,

                    "cost_price":"398.000000",

                    "discount_price":"402.020202",

                    "list_price":"941.000000",
```

```
        "cost_usdollar":"398.000000",
        "discount_usdollar":"402.020202",
        "list_usdollar":"941.000000",
        "status":"Available",
        "tax_class":"Taxable",
        "date_available":"2004-10-15",
        "website":null,
        "weight":"22.00",
        "qty_in_stock":145,
        "support_name":null,
        "support_description":null,
        "support_contact":null,
        "support_term":null,
        "pricing_formula":"ProfitMargin",
        "pricing_factor":"1.00",
        "assigned_user_id":null,
        "currency_id":"-99",
        "Base_rate":"1.000000"
    },
    {
        "id":"c181bd32-c7f2-11e8-b4eb-7831c1c14620",
        "name":"Allyson Gadget",
        "date_entered":"2018-10-04 16:28:37",
        "date_modified":"2018-10-04 16:28:37",
```

```
                    "modified_user_id":"1",

                    "created_by":"1",

                    "description":null,

                    "deleted":0,

                    "type_id":null,

                    "manufacturer_id":"c03f952a-
c7f2-11e8-b410-7831c1c14620",

                    "category_id":"c18112d8-c7f2-11e8-b753-7831c1c1462
0",

                    "mft_part_num":"Ink Conglomerate Inc 921535XYZ987"
,

                    "vendor_part_num":null,

                    "date_cost_price":null,

                    "cost_price":"534.000000",

                    "discount_price":"842.000000",

                    "list_price":"905.000000",

                    "cost_usdollar":"534.000000",

                    "discount_usdollar":"842.000000",

                    "list_usdollar":"905.000000",

                    "status":"Available",

                    "tax_class":"Taxable",

                    "date_available":"2004-10-15",

                    "website":null,

                    "weight":"20.00",

                    "qty_in_stock":26,
```

```
                    "support_name":null,

                    "support_description":null,

                    "support_contact":null,

                    "support_term":null,

                    "pricing_formula":"Fixed",

                    "pricing_factor":"1.00",

                    "assigned_user_id":null,

                    "currency_id":"-99",

                    "base_rate":"1.000000"
                }
            ]
}
```


Change Log

Version	Change
v10	Added /Opportunities/recent_product POST endpoint.

//record_list POST

Overview

An API to create and save lists of records.

Summary

Create record list. Only POST request allowed.

Request Arguments

Name	Type	Description	Required
module	string	Module name	True
records	array	Array of records	True

Request Example

```
POST /rest/v10/:module/record_list


{records: ["e5c8bb3b-5eea-3d8a-
c278-56c6affa6212", "e49395b4-d3b0-1e3f-600a-56c6afe7e34f"]}
```

Response Arguments

Name	Type	Description
id	guid	record list id
assigned_user_id	guid	user id
module_name	string	Module name
records	array	record ids
date modified	date	date modified

Output Example

```
{
    "id": "39d2c781-c0b7-446f-1d83-56c6e3cda510",
    "assigned_user_id": "1",
    "module_name": "Accounts",
    "records": [
        "e5c8bb3b-5eea-3d8a-c278-56c6affa6212",
        "e49395b4-d3b0-1e3f-600a-56c6afe7e34f",
        "c03ed8ee-60d3-c6a6-55c3-56c6af95e696",
        "d65573e3-c25a-f9b4-33f2-56c6afb8d856"
    ],
    "date_modified": "2016-02-19 09:42:44"
}
```

//record_list/:record_list_id DELETE

Overview

An API to delete lists of records

Summary

Delete record list. Only DELETE request allowed.

Request Arguments

Name	Type	Description	Required
module	string	Module name	True
record_list_id	guid	record list id	True

Request Example

```
DELETE /rest/v10/:module/record_list/:id
```

Response Arguments

Return boolean true.

//record_list/:record_list_id GET

Overview

An API to return record list data

Summary

Get record list data. Only GET request allowed.

Request Arguments

Name	Type	Description	Required
module	string	Module name	True
record_list_id	guid	record list id	True

Request Example

```
GET /rest/v10/:module/record_list/:id
```

Response Arguments

Name	Type	Description
id	guid	record list id
assigned_user_id	guid	user id
module_name	string	Module name
records	array	record ids
date modified	date	date modified

Output Example

```
{
    "id": "39d2c781-c0b7-446f-1d83-56c6e3cda510",
    "assigned_user_id": "1",
    "module_name": "Accounts",
    "records": [
        "e5c8bb3b-5eea-3d8a-c278-56c6affa6212",
        "e49395b4-d3b0-1e3f-600a-56c6afe7e34f",
        "c03ed8ee-60d3-c6a6-55c3-56c6af95e696",
        "d65573e3-c25a-f9b4-33f2-56c6afb8d856"
    ],
    "date_modified": "2016-02-19 09:42:44"
}
```

//sync_key/:sync_key_field_value DELETE

Overview

Deletes the record with the given sync_key.

Request Arguments

Name	**Type**	**Description**	**Required**
module	String	The module the record belongs to	True
sync_key_field_valu	String	A unique ID for the	True

Name	Type	Description	Required
e		record identifying it in an external system	

Request

```
/<module>/sync_key/:sync_key_field_value
```

Response Arguments

Name	Type	Description
record	String	The ID of the record that was deleted.

Response

Status 200

```
{
    "id": "a0328573-a252-a27c-3530-4e4297d4c9e1"
}
```

Change Log

Version	Change
v11_10	Added /<module>/sync_key/:sync_key_field_value DELETE endpoint.

//sync_key/:sync_key_field_value GET

Overview

Retrieves the record with the given sync_key.

Request Arguments

Name	Type	Description	Required
module	String	The module the record belongs to	True
sync_key_field_value	String	A unique ID for the record identifying it in an external system	True

Request

```
/<module>/sync_key/:sync_key_field_value
```

Response Arguments

Body: formatted JSON object.

Response

Successful Response

Status 200

Failed Response

Status 422

```
  {
    "error": "invalid_parameter",
    "error_message": "Could not find record with :sync_key_field_value
 in module: <module>"
}
```

Change Log

Version	Change
v11_10	Added /<module>/sync_key/:sync_key_field_value GET endpoint.

//sync_key/:sync_key_field_value PATCH

Overview

Upserts based on sync_key. If a record can be found with sync_key, then update. If the record does not exist, then create it.

Note: This endpoint cannot be used to set the sync_key for an existing Sugar record. To do so, it is recommended to use the set sync key endpoint to update the matching Sugar record ID. Once the sync_key is set for the record, then it is possible to leverage this endpoint.

Request Arguments

Name	Type	Description	Required
module	String	The module the record belongs to	True
sync_key_field_value	String	A unique ID for the record identifying it in an external system	True

Request

```
/<module>/sync_key/:sync_key_field_value
```

Response Arguments

Name	Type	Description
record	String	The ID of the record that was upserted.

Response

- Status 201: For a created record
- Status 200: For an updated record

```
{
    record: "a0328573-a252-a27c-3530-4e4297d4c9e1"
}
```

Change Log

Version	Change
v11_10	Added /<module>/sync_key/:sync_key_field_value PATCH endpoint.

//sync_key/:sync_key_field_value PUT

Overview

Upserts based on sync_key. If a record can be found with sync_key, then update. If the record does not exist, then create it. Note that the PATCH method is recommended over the PUT method.

Note: This endpoint cannot be used to set the sync_key for an existing Sugar record. To do so, it is recommended to use the set sync key endpoint to update the matching Sugar record ID. Once the sync_key is set for the record, then it is possible to leverage this endpoint.

Request Arguments

Name	Type	Description	Required
module	String	The module the record belongs to	True
sync_key_field_value	String	A unique ID for the record identifying it in an external system	True

Request

```
/<module>/sync_key/:sync_key_field_value
```

Response Arguments

Name	Type	Description
record	String	The ID of the record that was upserted.

Response

- Status 201: For a created record
- Status 200: For an updated record

```
{
    record: "a0328573-a252-a27c-3530-4e4297d4c9e1"
}
```

Change Log

Version	Change
v11_10	Added /<module>/sync_key/:sync_key_field_value PUT endpoint.

//temp/file/:field POST

Overview

Saves an image to a temporary folder.

Request Arguments

Name	Type	Description	Required
format	String	The data format. Currently accepts 'sugar-html-json'.	True
delete_if_fails	Boolean	Indicates whether the API is to mark related record deleted if the file upload fails.	False
oauth_token	String	The oauth_token value.	False - Required if only if delete_if_fails is true.
<image field>	String	The field and file to populate. Example: {"":"@\/path\/to\/ExampleImage.png"}	True

Request

```
{
   "format":"sugar-html-json",
   "delete_if_fails":true,
   "oauth_token":"8d240d9d-04ea-571b-35ea-513037ed5857",
   "<image field>":"@\/path\/to\/\/ExampleImage.png"
}
```

Response Arguments

Name	Type	Description
guid	String	The temp ID of the image.

Response

```
{
   "picture":{
      "guid":"50a8760d-966f-9d7e-f45c-513037fac8fc"
   }
}
```

Change Log

Version	Change
v10	Added /<module>/temp/file/:field POST endpoint.

//temp/file/:field/:temp_id GET

Overview

Reads a temporary image and deletes it.

Request Arguments

This endpoint does not accept any arguments.

Response Arguments

Name	Type	Description
<image file>	String;	Returns the image content with the content headers.

Response

```
result: HTTP/1.1 200 OK
Date: Fri, 01 Mar 2013 05:36:24 GMT
Server: Apache/2.2.22 (Unix) DAV/2 PHP/5.3.17
X-Powered-By: PHP/5.3.17
Expires: Sun, 31 Mar 2013 05:36:24 GMT
Cache-Control: max-age=1, post-check=0, pre-check=0
Pragma: public
X-Content-Type-Options: nosniff
Content-Length: 2072
ETag: d41d8cd98f00b204e9800998ecf8427e
Connection: close
Content-Type: image/png

‰PNG


IHDRIIqsÜßIDATxœíœ}hUçÇ?¿›ëiÐl‹Ñ¹Ðë,¤©™šS'âD,ÉÈ4+ˆÖv¸±Mö['vl°¹wêÝ–mŒMŠ
Ô—¦jBWDD$Ý,ˆdR²L$ˆˆ/!^sï9çùíç^st7×crßÚž\î%Üœû¾çy¹y~˜˜˜˜êA*Ý€x"÷r!°p€p
H¬ÿwEÚ–£b'Brj6`°h®ÿŽçi¨œ¬²J
‰IMX)ŸË>?
$Co÷ àÐŸ}ž"WXY$ÝwKµcSÓ ¬&éA¤€Ødà¦«l·bI%eå$€åÀF X4fþ°à
p›®"À%À"RVÑ%…RS¬ÆŠéZ‰–š¨L£Ì¤ë0ŠOWÑ$…RÓlÊY‡±æ'š¨à*p+ë`Œ"¦k^'B©ù(v¬éº•75
Q™Þà¦«›®I˜_ºæ$)+'‰MM'Ð¸">5Q1ØuÖàïÀ6]þ\dE–JM=v¬é¶`Sã6]CÀQ`¸ ÑÓõ@I¡Ô´`¥ä
RÓ@u¤&*»HÆŽ]ƒDLWAI#·$ùº
ßXÁ½¾÷+`Õ—WÍ/Îúæ,IX PµŸX®JR‹ÛØÊ êÍOâ{IÔ|û¬IrÝ œW®M5|Ìh§'ôXJ#BB] Ú°5¨ÚŠ
¨ï÷éÇ—é¡ß_—ŒŸÁÖºOåýÕ¼}tÝ°)ÛŠòt ü'€ÁÝ·ÓãÛS™åiè '6EkAªV–ª"vscŽúìÎq}vg3¾ß
E4 AdÈ+ª¤ÕÀ€µj½ UNgÐ£µÊ™7n'–u=è5ÈzÄnP«"]¶>ªÀi'¬Ù@&Ýƒê:±›l°K…/£ù$„e'Ýµ‹
èŽG"Ï«0ºwIÝqý»ngöö¦ÒMµé'vUæL•K˜¢šÎ˜~s\¿üÜ„îøB+~ð5 ¼n¯UTäž–5a'¥Ñ|×.$Éd!
jÕIhWØsdá,S¯-\p"©~éfêµOÆõ…ÞY4iö3J,ËÕK 'ñü~mþäˆyù¥„ÔÔlÀKÿP2™
¨.›éÆÿµ&O_g˜ÏtžxTDv8ªÛ.~¯aÑqúzîd¾ûÜTºÑƒÎ@Ø¬V¤înçñ¡9lrt
!ú@Í¾s•ÏnlÁóöJàux+ŠÎdùÄD¦(kq€¶¬r"=µÎéþZ§/©úÖ/n¥7ùÆ
Ðž@d°|^éRõ1Q=EôëÊæasà7u'LlÄËôN¯´Í.oæ!'GQ† „eÛÕn„±}‹
zðÆgÒÞ¾oMN7ØÀÓÙ™±>§jÖ®äfÕ›Ày,àˆ&j†ôÇ/^ç©5mxÞób¼-dh¾›(Šœ%[=gÓÕ*J‹ƒ>sÖIž
Ýµô#}5ª?¿•zýqß´ûHo lš'án©„}1Œ¾ƒïÕÖ'Î™¿jas65.ª
```

2ó¡%éKÉ·*$Y*Ðí¨nA{añ¢Áô¯Nû?z~òN½V]F¤I²-tœ 8¡ÉäÛúËý7ùTk™ô>ñÒ[¢jÛ^"1aJ.)
Ü…»é,–GT¿xÁ©9»{iÝ‰¤2øÓ[©?7ùf¡/R+ž?Í"-Sæ×?k"'èÆ÷ºHO¯ÕúÐŠJÝô»Td³ª'zNÝ¬ð/
ÚCäÐWvî4]Ý]IRwöøß_›ï¶³ŒbTä«Ž{»*IiIÀþc6¥oÜ@n§¶ˆ1?ÕD'ˆTLT×÷AMªºÛqœ:`"•¦š
$¬,`v‹P5T•$µªjTaƒª'XRbIˆ%E –XRbIˆ%E –XRbIˆ%E –XRbIˆ%E –XRbI($éà&pÖþáSà
ÌÎŸlM]>
IŠžsþ0ð¶T,/…$]b‹[>È\^¡@?óJ~7÷òoÀXË™"7®ÒLctöoBþ"·P IYQ>ðGlÉÞìŠ®µ©å%WTø&
ðMlçŸ6Að€S%YQÆu;.àÌúuc«"*Q²õ°LcÛÿ¶ji„l‰W!99"½ Ý~"®Ûq[Þy[2Ú‹-È™kÉh©—¢öa
KQ/&Š˜0}>)wºnÇvöû+÷·`'+ŠvfŽ-j~X99æ|ˆ+"®)×í8=Uÿ;f
'7bË—#]['ûÊãç*&LQNº…Æ®KÀ!ì¬ØŽùÏ)×Íz¹"KÍö–ºÀ

Change Log

Version	Change
v10	Added /<module>/enum/:field GET endpoint.

/Accounts/:record/link/:link_name/filter GET

Overview

Lists related filtered records.

Summary

This endpoint will return a set of related records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Care will need to be taken to make sure this filter has appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

Name	Type	Description	Required
filter	String	The filter expression. Filter expressions are explained below.	False
max_num	Integer	A maximum number of records	False

Name	Type	Description	Required
		to return. Default is 20.	
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
fields	String	Comma delimited list of fields to return. The field date_modified will always be returned. This argument can be combined with the view argument. Example: name,account_type,description	False
view	String	Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list". Example: record	False
order_by	String	How to sort the returned records, in a comma delimited list with the direction appended to the	False

Name	Type	Description	Required
		column name after a colon. Example: name:DESC,account_type:DESC,date_modified:ASC	

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
   "filter":[
      {
         "name":"Nelson Inc"
      }
   ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
   "filter":[
      {
         "name":{
            "$starts":"Nelson"
         }
      }
   ]
}
```

Below is a list of operation types:

Operation	Description
$equals	Performs an exact match on that field.
$not_equals	Performs an exact match on that field.
$not_equals	Matches on non-matching values.
$starts	Matches on anything that starts with the value.
$ends	Matches anything that ends with the value.
$contains	Matches anything that contains the value
$in	Finds anything where field matches one of the values as specified as an array.
$not_in	Finds anything where field does not matches any of the values as specified as an array.
$is_null	Checks if the field is null. This operation does not need a value specified.
$not_null	Checks if the field is not null. This operation does not need a value specified.
$lt	Matches when the field is less than the value.
$lte	Matches when the field is less than or equal to the value.
$gt	Matches when the

Operation		Description	
			field is greater than the value.
	$gte		Matches when the field is greater than or equal to the value.

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "$and" and "$or".

Example

```
{
   "filter":[
      {
         "$or":[
            {
               "name":"Nelson Inc"
            },
            {
               "name":"Nelson LLC"
            }
         ]
      }
   ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "$favorite" and "$owner".

Example

```
{
```

```
   "filter":[
      {
         "$favorite":"_this"
      }
   ]
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Repsonse

```
{
   "next_offset":-1,
   "records":[
      {
         "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
         "name":"Dale Spivey",
         "date_entered":"2013-02-26T19:12:00+00:00",
         "date_modified":"2013-02-28T05:03:00+00:00",
         "modified_user_id":"1",
         "modified_by_name":"Administrator",
         "created_by":"1",
         "created_by_name":"Administrator",
         "description":"",
         "img":"",
         "deleted":false,
         "assigned_user_id":"seed_sally_id",
         "assigned_user_name":"Sally Bronsen",
         "team_name":[
            {
               "id":"East",
```

```
            "name":"East",
            "name_2":"",
            "primary":false
        },
        {
            "id":1,
            "name":"Global",
            "name_2":"",
            "primary":false
        },
        {
            "id":"West",
            "name":"West",
            "name_2":"",
            "primary":true
        }
    ],
    "salutation":"",
    "first_name":"Dale",
    "last_name":"Spivey",
    "full_name":"Dale Spivey",
    "title":"VP Operations",
    "linkedin":"",
    "facebook":"",
    "twitter":"",
    "googleplus":"",
    "department":"",
    "do_not_call":false,
    "phone_home":"(523) 825-4311",
    "email":[
        {
            "email_address":"sugar.dev.sugar@example.co.jp",
            "opt_out":"0",
            "invalid_email":"0",
            "primary_address":"1"
        },
        {
            "email_address":"the.support@example.biz",
            "opt_out":"0",
            "invalid_email":"0",
            "primary_address":"0"
        }
    ],
    "phone_mobile":"(373) 861-0757",
    "phone_work":"(212) 542-9596",
    "phone_other":"",
```

```
"phone_fax":"",
"email1":"sugar.dev.sugar@example.co.jp",
"email2":"the.support@example.biz",
"invalid_email":false,
"email_opt_out":false,
"primary_address_street":"345 Sugar Blvd.",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Denver",
"primary_address_state":"CA",
"primary_address_postalcode":"87261",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
"assistant":"",
"assistant_phone":"",
"picture":"",
"email_and_name1":"",
"lead_source":"Campaign",
"account_name":"Smallville Resources Inc",
"account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",
"opportunity_role_fields":"",
"opportunity_role_id":"",
"opportunity_role":"",
"reports_to_id":"",
"report_to_name":"",
"portal_name":"DaleSpivey97",
"portal_active":true,
"portal_password":"$1$yKMAONHM$Y5S.8CY.WZCZCwfGD1a1Q\/",
"portal_password1":"",
"portal_app":"",
"preferred_language":"en_us",
"campaign_id":"",
"campaign_name":"",
"c_accept_status_fields":"",
"m_accept_status_fields":"",
"accept_status_id":"",
"accept_status_name":"",
"sync_contact":"",
"my_favorite":false,
"_acl":{
```

```
            "fields":{

            }
        }
    },
    {
        "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
        "name":"Florence Haddock",
        "date_entered":"2013-02-26T19:12:00+00:00",
        "date_modified":"2013-02-26T19:12:00+00:00",
        "modified_user_id":"1",
        "modified_by_name":"Administrator",
        "created_by":"1",
        "created_by_name":"Administrator",
        "description":"",
        "img":"",
        "deleted":false,
        "assigned_user_id":"seed_sally_id",
        "assigned_user_name":"Sally Bronsen",
        "team_name":[
            {
                "id":"East",
                "name":"East",
                "name_2":"",
                "primary":false
            },
            {
                "id":1,
                "name":"Global",
                "name_2":"",
                "primary":false
            },
            {
                "id":"West",
                "name":"West",
                "name_2":"",
                "primary":true
            }
        ],
        "salutation":"",
        "first_name":"Florence",
        "last_name":"Haddock",
        "full_name":"Florence Haddock",
        "title":"Director Sales",
        "linkedin":"",
        "facebook":"",
```

```
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(729) 845-3137",
"email":[
   {
      "email_address":"dev.vegan@example.de",
      "opt_out":"1",
      "invalid_email":"0",
      "primary_address":"0"
   },
   {
      "email_address":"section71@example.it",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"1"
   }
],
"phone_mobile":"(246) 233-1382",
"phone_work":"(565) 696-6981",
"phone_other":"",
"phone_fax":"",
"email1":"section71@example.it",
"email2":"dev.vegan@example.de",
"invalid_email":false,
"email_opt_out":false,
"primary_address_street":"111 Silicon Valley Road",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Denver",
"primary_address_state":"CA",
"primary_address_postalcode":"79900",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
"assistant":"",
"assistant_phone":"",
"picture":"",
"email_and_name1":"",
"lead_source":"Support Portal User Registration",
```

```
         "account_name":"Smallville Resources Inc",
         "account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",
         "opportunity_role_fields":"",
         "opportunity_role_id":"",
         "opportunity_role":"",
         "reports_to_id":"",
         "report_to_name":"",
         "portal_name":"FlorenceHaddock169",
         "portal_active":true,
         "portal_password":"$1$nWFhTbK6$JF9BCGSqL\/NCrbhueX5ia0",
         "portal_password1":"",
         "portal_app":"",
         "preferred_language":"en_us",
         "campaign_id":"",
         "campaign_name":"",
         "c_accept_status_fields":"",
         "m_accept_status_fields":"",
         "accept_status_id":"",
         "accept_status_name":"",
         "sync_contact":"",
         "my_favorite":false,
         "_acl":{
            "fields":{

            }
         }
      }
   ]
}
```


Change Log

Version	Change
v10	Added /<module>/:record/link/:link_name/filter GET endpoint.

/Activities GET

Activities on the home page

Summary:

This endpoint lists activities across the entire sytstem. It uses a subscription model, and can be queried like a normal bean collection, but without search, ordering or filtering.

Query Parameters:

Param	Description	Optional
max_num	A maximum number of records to return	Optional
offset	How many records to skip over before records are returned	Optional

Input Example:

This endpoint does not accept any input

Output Example:

```
{
    "next_offset": 20,
This will be set to -1 when there are no more records after this "page
".
    "records": [{
        "id": "22fb8b16-de1d-f1dc-b15b-51240efde177",
        "date_entered": "2013-02-19T23:47:11+00:00",
        "date_modified": "2013-02-19T23:47:11+00:00",
        "created_by": "1",
        "deleted": "0",
        "parent_id": "f5bb0331-2c0f-5c7c-b4db-5123caac0056",
        "parent_type": "Contacts",
        "activity_type": "post",
This will be the type of activity performed.
        "data": {
```

```
            "value": "This is a test post on a contact I'm subscribed
to."
        },
        "comment_count": 0,
This will be set to the total number of comments on the post.
        "last_comment": {
This will be the last comment on the post, which can be used to create
 a comment model on the frontend.
            "deleted": 0,
            "data": []
        },
        "fields": [],
        "first_name": null,
        "last_name": "Administrator",
        "created_by_name":
 " Administrator" This will be the locale-
formatted full name of the user.
    }, ... ]
}
```

/Activities/filter GET

Activities on the home page

Summary:

This endpoint lists activities across the entire sytstem. It uses a subscription model, and can be queried like a normal bean collection, but without search, ordering or filtering.

Query Parameters:

Param	Description	Optional
max_num	A maximum number of records to return	Optional
offset	How many records to skip over before records are returned	Optional

Input Example:

This endpoint does not accept any input

Output Example:

```
{
    "next_offset": 20,
This will be set to -1 when there are no more records after this "page
".
    "records": [{
        "id": "22fb8b16-de1d-f1dc-b15b-51240efde177",
        "date_entered": "2013-02-19T23:47:11+00:00",
        "date_modified": "2013-02-19T23:47:11+00:00",
        "created_by": "1",
        "deleted": "0",
        "parent_id": "f5bb0331-2c0f-5c7c-b4db-5123caac0056",
        "parent_type": "Contacts",
        "activity_type": "post",
This will be the type of activity performed.
        "data": {
            "value": "This is a test post on a contact I'm subscribed
to."
        },
        "comment_count": 0,
This will be set to the total number of comments on the post.
        "last_comment": {
This will be the last comment on the post, which can be used to create
 a comment model on the frontend.
            "deleted": 0,
            "data": []
        },
        "fields": [],
        "first_name": null,
        "last_name": "Administrator",
        "created_by_name":
 " Administrator" This will be the locale-
formatted full name of the user.
    }, ... ]
}
```

/Administration/aws GET

Overview

[ADMIN] Gets Amazon Web Services configs from Sugar Serve.

Summary

This endpoint gets configuration values for Amazon Web Services. This endpoint is only available to administrators of Sugar Serve.

Request Arguments

This endpoint does not accept any arguments.

Response

```
  {
    "aws_connect_instance_name": "my-aws-instance",
    "aws_connect_region": "us-east-2"
}
```

Change Log

Version	Change
v11_10	Added /Administration/aws GET endpoint.

/Administration/aws POST

Overview

[ADMIN] Set Amazon Web Services configs in Sugar Serve.

Summary

This endpoint sets configuration values for Amazon Web Services. This endpoint is

only available to administrators of Sugar Serve.

Request Arguments

Name	Type	Description	Required
aws_connect_instance_name	String	The name of the AWS Connect instance	False
aws_connect_region	String	The region assigned to the AWS Connect instance	False

Response

```
  {
    "aws_connect_instance_name": "my-aws-instance",
    "aws_connect_region": "us-east-2"
}
```

Change Log

Version	Change
v11_10	Added /Administration/aws POST endpoint.

/Administration/config/:category GET

Overview

[ADMIN] Get Configuration Settings for a Category.

Summary

This endpoint gets configuration values for a given category. This endpoint is only available to administrators.

Request

```
  GET /Administration/config/:category
```

Response Example

```
  {
    "a_config": "good"
}
```

Change Log

Version	Change
v11.13	Added Administration/config/:category GET endpoint.

/Administration/config/:category POST

Overview

[ADMIN] Set Configuration Settings for a Category.

Summary

This endpoint sets configuration values for a given category. This endpoint is only available to administrators.

Request

```
  POST /Administration/config/:category
```

Response Example

```
  {
    "a_config": "good"
}
```

Change Log

Version	Change
v11.13	Added Administration/config/:category POST endpoint.

/Administration/elasticsearch/indices GET

Overview

[ADMIN] Elasticsearch index statistics

Summary

This endpoint returns the index statistics for the Elasticsearch backend. This endpoint is only available to administrators.

Response


```
{
    "autobr2583_shared":{
        "_shards":{
            "total":1,
            "successful":1,
            "failed":0
        },
        "indices":{
            "autobr2583_shared":{
                "index":{
                    "primary_size_in_bytes":1134148,
                    "size_in_bytes":1134148
                },
                "translog":{
                    "operations":1100
                },
                "docs":{
                    "num_docs":1100,
                    "max_doc":1100,
                    "deleted_docs":0
                },
                "merges":{
                    "current":0,
                    "current_docs":0,
                    "current_size_in_bytes":0,
                    "total":1,
```

```
        "total_time_in_millis":103,
        "total_docs":1000,
        "total_size_in_bytes":1138070
    },
    "refresh":{
        "total":13,
        "total_time_in_millis":177
    },
    "flush":{
        "total":0,
        "total_time_in_millis":0
    },
    "shards":[[{
        "routing":{
            "state":"STARTED",
            "primary":true,
            "node":"4rjVEVuYQQqKl4sT1FUxNA",
            "relocating_node":null,
            "shard":0,
            "index":"autobr2583_shared"
        },
        "state":"STARTED",
        "index":{
            "size_in_bytes":1134148
        },
        "translog":{
            "id":1427003304006,
            "operations":1100
        },
        "docs":{
            "num_docs":1100,
            "max_doc":1100,
            "deleted_docs":0
        },
        "merges":{
            "current":0,
            "current_docs":0,
            "current_size_in_bytes":0,
            "total":1,
            "total_time_in_millis":103,
            "total_docs":1000,
            "total_size_in_bytes":1138070
        },
        "refresh":{
            "total":13,
            "total_time_in_millis":177
```

```
                    },
                    "flush":{
                        "total":0,
                        "total_time_in_millis":0
                    }
                }]]
            }
        }
    }
}
```

Change Log

Version	Change
v10	Added /Administration/elasticsearch/indices GET endpoint.

/Administration/elasticsearch/mapping GET

Overview

[ADMIN] Elasticsearch mapping

Summary

This endpoint returns the mapping for every available index. This endpoint is only available to administrators.

Response

```
{
    "autobr2583_shared":{
        "KBDocuments":{
            "dynamic":"false",
            "_all":{
                "enabled":false
            },
            "properties":{
                "kbdocument_name":{
                    "type":"string",
                    "index":"not_analyzed",
```

```
                "fields":{
                    "gs_string_default":{
                        "type":"string",
                        "analyzer":"gs_analyzer_default"
                    },
                    "gs_string_ngram":{
                        "type":"string",
                        "index_analyzer":"gs_analyzer_ngram",
                        "search_analyzer":"gs_analyzer_default"
                    }
                }
            }
        }
    },
    "RevenueLineItems":{
        "dynamic":"false",
        "_all":{
            "enabled":false
        },
        "properties":{
            "date_modified":{
                "type":"string",
                "index":"not_analyzed",
                "fields":{
                    "gs_datetime":{
                        "type":"date",
                        "index":"no",
                        "format":"YYYY-MM-dd HH:mm:ss"
                    }
                }
            },
            "description":{
                "type":"string",
                "index":"not_analyzed",
                "fields":{
                    "gs_string_default":{
                        "type":"string",
                        "analyzer":"gs_analyzer_default"
                    },
                    "gs_string_ngram":{
                        "type":"string",
                        "index_analyzer":"gs_analyzer_ngram",
                        "search_analyzer":"gs_analyzer_default"
                    }
                }
            }
```

```
            }
        }
    }
}
```

Change Log

Version	Change
v10	Added /Administration/elasticsearch/mapping GET endpoint.

/Administration/elasticsearch/queue GET

Overview

[ADMIN] Elasticsearch queue statistics

Summary

This endpoint returns the queue statistics for the Elasticsearch backend. This endpoint is only available to administrators.

Response


```
{
    "total":2238,
    "queued":{
        "Cases":"250",
        "KBDocuments":"5",
        "Notes":"50",
        "Quotes":"2",
        "Accounts":"51",
        "Contacts":"201",
        "Leads":"201",
        "Opportunities":"150",
        "RevenueLineItems":"578",
        "Bugs":"50",
        "Contracts":"2",
        "Manufacturers":"2",
        "ProductCategories":"43",
        "Tasks":"200",
```

```
        "Calls":"50",
        "Emails":"200",
        "Meetings":"200",
        "Products":"3"
    }
}
```

Change Log

Version	Change
v10	Added /Administration/elasticsearch/queue GET endpoint.

/Administration/elasticsearch/refresh/enable POST

Overview

[ADMIN] Elasticsearch enable refresh_interval

Summary

Enable refresh_interval for all indices managed by SugarCRM. This endpoint is only available to administrators.

Response

```
{
  "aabbcc_contactsleads": 200,
  "aabbcc_accountsonly": 200,
  "aabbcc_shared": 200
}
```

Change Log

Version	Change
v10	Added /Administration/elasticsearch/re

	fresh/enable POST endpoint.

/Administration/elasticsearch/refresh/status GET

Overview

[ADMIN] Elasticsearch index refresh interval status

Summary

This endpoint returns the current refresh_interval for every index managed by SugarCRM. This endpoint is only available to administrators.

Response

```
{
  "aabbcc_contactsleads": "1s",
  "aabbcc_accountsonly": "1s",
  "aabbcc_shared": "1s"
}
```

Change Log

Version	Change
v10	Added /Administration/elasticsearch/refresh/status GET endpoint.

/Administration/elasticsearch/refresh/trigger POST

Overview

[ADMIN] Elasticsearch trigger explicit index refresh on all indices

Summary

Trigger an explicit index refresh for all indices managed by SugarCRM. This endpoint is only available to administrators.

Response

```
{
  "aabbcc_contactsleads": 200,
  "aabbcc_accountsonly": 200,
  "aabbcc_shared": 200
}
```


Change Log

Version	Change
v10	Added /Administration/elasticsearch/refresh/trigger POST endpoint.

/Administration/elasticsearch/replicas/enable POST

Overview

[ADMIN] Elasticsearch enable replicas

Summary

Enable replicas for all indices managed by SugarCRM. This endpoint is only available to administrators. This requires reindex_zero_replica to be enabled.

Response


```
{
  "aabbcc_contactsleads": 200,
  "aabbcc_accountsonly": 200,
  "aabbcc_shared": 200
}
```


Change Log

Version	Change
v10	Added /Administration/elasticsearch/replicas/enable POST endpoint.

/Administration/elasticsearch/replicas/status GET

Overview

[ADMIN] Elasticsearch index replica status

Summary

This endpoint returns the number of replicas for every index managed by SugarCRM. This endpoint is only available to administrators.

Response

```
{
  "aabbcc_contactsleads": "4",
  "aabbcc_accountsonly": "2",
  "aabbcc_shared": "1"
}
```

Change Log

Version	Change
v10	Added /Administration/elasticsearch/replicas/status GET endpoint.

/Administration/elasticsearch/routing GET

Overview

[ADMIN] Elasticsearch index routing

Summary

This endpoint returns an overview of the current read and write indices per module. This routing information is based on the configured index routing strategies and the index configuration per module. Note that this output shows the default routing only as no context is available to determine dynamic routing strategies. This endpoint is only available to administrators.

Response

```
{
    "Contacts":{
        "strategy":"static",
        "routing":{
            "write_index":"autobr2583_contacts",
            "read_indices":[
                "autobr2583_contacts"
            ]
        }
    },
    "Accounts":{
        "strategy":"static",
        "routing":{
            "write_index":"autobr2583_shared",
            "read_indices":[
                "autobr2583_shared"
            ]
        }
    },
    "Emails":{
        "strategy":"rolling",
        "routing":{
            "write_index":"autobr2583_emails_201503",
            "read_indices":[
                "autobr2583_emails_201503",
                "autobr2583_emails_201502",
                "autobr2583_emails_201501"
            ]
        }
    }
}
```

Change Log

Version	Change
v10	Added /Administration/elasticsearch/routing GET endpoint.

/Administration/idm/migration/disable POST

Overview

[ADMIN] Disable Idm migrations

Summary

This endpoint disables Idm migrations. This endpoint is only available to administrators.

Request Arguments

This endpoint does not accept any arguments.

Response

```
{
    "success": true
}
```

Change Log

Version	Change
v11_2	Added Administration/idm/migration/disable POST endpoint.

/Administration/idm/migration/enable POST

Overview

[ADMIN] Enable Idm migrations

Summary

This endpoint enables Idm migrations. This endpoint is only available to administrators.

Request Arguments

This endpoint does not accept any arguments.

Response

```
{
    "success": true
}
```

Change Log

Version	Change
v11_2	Added Administration/idm/migration/enable POST endpoint.

/Administration/idm/users GET

Overview

Lists filtered user records.

Summary

This endpoint will return a set of user records with raw password hashes filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

Name	Type	Description	Required
filter	String	The filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query parameters in one of two ways	False

Name	Type	Description	Required
		for GET requests: 1. By specifying individual filter arguments as distinct parameters. Example: filter[0][id]=1. 2. By specifying the whole filter as a single JSON-encoded string. Note that this syntax is currently not supported on certain modules. Example: filter=[{"id":"1"}].	
filter_id	String	Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND.	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is	False

Name	Type	Description	Required
		0.	
fields	String	Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters (applicable to link and collection fields). The fields id and date_modified will always be returned. Example: name,account_type,description,{"name":"opportunities","fields":["id","name","sales_status"],"order_by":"date_closed:desc"} For more details on additional field parameters, see Relate API and Collection API.	False
view	String	Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common	False

Name	Type	Description	Required
		views are "record" and "list". Example: record	
order_by	String	How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon. Example: name:DESC,account_type:DESC,date_modified:ASC	False
q	String	A search expression, will search on this module. Cannot be used at the same time as a filter expression or id.	False
deleted	Boolean	Boolean to show deleted records in the result set.	False

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
   "filter":[
      {
         "name":"Nelson Inc"
      }
   ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
   "filter":[
      {
         "name":{
            "$starts":"Nelson"
         }
      }
   ]
}
```

Below is a list of operation types:

Operation	Description
$equals	Performs an exact match on that field.
$not_equals	Performs an exact match on that field.
$not_equals	Matches on non-matching values.
$starts	Matches on anything that starts with the value.
$ends	Matches anything that ends with the value.
$contains	Matches anything that contains the value
$in	Finds anything where field matches one of the values as specified as an array.
$not_in	Finds anything where field does not

Operation		Description
		matches any of the values as specified as an array.
	$is_null	Checks if the field is null. This operation does not need a value specified.
	$not_null	Checks if the field is not null. This operation does not need a value specified.
	$lt	Matches when the field is less than the value.
	$lte	Matches when the field is less than or equal to the value.
	$gt	Matches when the field is greater than the value.
	$gte	Matches when the field is greater than or equal to the value.

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "$and" and "$or".

Example

```
{
   "filter":[
      {
         "$or":[
            {
               "name":"Nelson Inc"
            },
            {
```

```
               "name":"Nelson LLC"
            }
         ]
      }
   ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "$favorite" and "$owner".

Example

```
{
   "filter":[
      {
         "$favorite":"_this"
      }
   ]
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Response

```
{
    "next_offset": -1,
    "records": [
        {
            "id": "1",
            "user_name": "admin",
            "user_hash": "$2y$10$PXY1UPZHjcm.t6ZArbia0uVzjCNEDm0XcGu/w
hGGk2xaPzAEKrKLa",
            "system_generated_password": false,
            "pwd_last_changed": "",
            "authenticate_id": "",
            "sugar_login": true,
            "picture": "7772d3b4-5779-11e8-835a-6a00025a7f70",
            "first_name": "",
            "last_name": "Administrator",
            "full_name": "Administrator",
            "name": "Administrator",
            "is_admin": true,
            "external_auth_only": false,
            "receive_notifications": true,
            "description": "",
            "date_entered": "2018-05-14T16:19:36+03:00",
            "date_modified": "2018-05-14T16:27:43+03:00",
            "last_login": "2018-05-16T12:12:43+03:00",
            "modified_user_id": "1",
            "modified_by_name": "",
            "created_by": "",
            "created_by_name": "",
            "created_by_link": {
                "full_name": "",
                "id": "",
                "_acl": {
                    "fields": [],
                    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
                }
            },
            "title": "Administrator",
            "department": "",
            "phone_home": "",
            "phone_mobile": "",
            "phone_work": "",
            "phone_other": "",
            "phone_fax": "",
            "status": "Active",
            "address_street": "",
```

```
"address_city": "",
"address_state": "",
"address_country": "",
"address_postalcode": "",
"UserType": "",
"default_team": "1",
"team_count": "",
"team_count_link": {
    "team_count": "",
    "id": "",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"team_name": [
    {
        "id": "1",
        "name": "Global",
        "name_2": "",
        "primary": false,
        "selected": false
    }
],
"deleted": false,
"portal_only": false,
"show_on_employees": true,
"employee_status": "Active",
"messenger_id": "",
"messenger_type": "",
"reports_to_id": "",
"reports_to_name": "",
"reports_to_link": {
    "name": "",
    "id": "",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"email1": "",
"email": [
    {
        "email_address": "admin@ex.com",
        "primary_address": true,
        "reply_to_address": false,
```

```
                    "invalid_email": false,
                    "opt_out": false,
                    "email_address_id": "93634e90-577a-11e8-a2f0-6a000
25a7f70"
                }
            ],
            "email_link_type": "",
            "is_group": false,
            "c_accept_status_fields": "",
            "calls": {
                "id": ""
            },
            "m_accept_status_fields": "",
            "meetings": {
                "id": ""
            },
            "accept_status_id": "",
            "accept_status_name": "",
            "accept_status_calls": "",
            "accept_status_meetings": "",
            "preferred_language": "",
            "acl_role_set_id": "",
            "my_favorite": false,
            "_acl": {
                "delete": "no",
                "fields": {
                    "pwd_last_changed": {
                        "write": "no",
                        "create": "no"
                    },
                    "last_login": {
                        "write": "no",
                        "create": "no"
                    }
                }
            },
            "_module": "Users"
        }
    ]
}
```


Change Log

Version	Change
v11_2	Added /Administration/idm/users GET endpoint.

/Administration/license/limits GET

Overview

[ADMIN] Get License Limits.

Summary

This endpoint gets the license limits, the number of unused licensed seats for each license type. This endpoint is only available to administrators.

Request Arguments

This endpoint does not accept any arguments.

Response

```
  {
  "default_limit": 100,
  "limit_enforced": 1,
  "default_license_type" : "CURRENT",
  "seats": {
    "CURRENT": 100,
    "SUGAR_SERVE": 10
  },
  "available_seats": {
    "CURRENT": 25,
    "SUGAR_SERVE": 1,
    "SUGAR_SELL": 0
  }
}
```

Change Log

Version	Change
v11_2	Added /Administration/license/limits GET endpoint.

/Administration/packages GET

Overview

[ADMIN] PackageManager list packages.

Summary

Lists all packages in the system.

Response


```
  {
    "packages": [
        {
            "name": "project_P3",
            "version": "1580885734",
            "type": "module",
            "published_date": "2020-02-06 12:51:17",
            "description": "",
            "uninstallable": true,
            "file_install": "cd590eb069a73b2eeb6c107776fd60dc",
            "file": "cd590eb069a73b2eeb6c107776fd60dc",
            "enabled": "ENABLED",
            "id": "66172e0e-48df-11ea-9f1f-acde48001122",
            "installed": true
        },
        {
            "name": "project_P1",
            "version": 1580885419,
            "published_date": "2020-02-05 06:50:19",
            "description": "None",
            "uninstallable": "Yes",
            "type": "Module",
            "file": "0ef4dd8b237f4358bdb246a744eb2b38",
            "file_install": "5d4b4614421586fd2111714abf622392",
            "unFile": "5d4b4614421586fd2111714abf622392",
            "installed": false
        }
    ]
}
```

Change Log

Version	Change
v11.8	Added /Administration/packages GET endpoint.

/Administration/packages POST

Overview

[ADMIN] PackageManager upload package.

Summary

Uploads a module loadable package via a multi-part form request. The only parameter is 'upgrade_zip' which must contain a valid module loadable zip file.

Sample Request

```
  POST /rest/v11_1/Administration/packages HTTP/1.1
Host: null
Content-Type: multipart/form-data
Accept: application/json
OAuth-Token: 3bbd6227-cfd0-45c7-8a6e-3c737e3c73ff
Content-Type: multipart/form-data; boundary=--------------------------
-905191404154484336597275426 6
Content-Length: 554

------------------------------905191404154484336597275426 6
Content-Disposition: form-
data; name="upgrade_zip"; filename="module.zip"
Content-Type: application/zip


{content-data}
------------------------------905191404154484336597275426 6--
```

Response

Returns status 200 OK and file hashes for install and delete package if request is successful.

```
  {
"file_install": "5d4b4614421586fd2111714abf622392",
"unFile": "5d4b4614421586fd2111714abf622392"
}
```

Otherwise returns status 4xx with errors.

```
  {
"error": "upload_package_error",
"error_message": "Issue with the manifest",
"error_description": "The uploaded file is not compatible with this ve
rsion of Sugar: 11.3.0"
}
```

Change Log

Version	Change
v11.8	Added /Administration/packages POST endpoint.

/Administration/packages/:file_install/install GET

Overview

[ADMIN] PackageManager install a package.

Summary

Installs the specified package. Package 'file_install' hash must be provided.
file_install hash could be found in staging package list.

Response

Returns http status 200 OK and newly installed package ID if request is successful.

```
  {
"id": "56365210-4cce-11ea-a7bf-acde48001122"
}
```

Otherwise returns http status 4xx with errors.

```
  {
"error": "not_found",
"error_message": "Could not find package by ID."
}
```

Change Log

Version	Change
v11.8	Added /Administration/packages/:file_install/install GET endpoint.

/Administration/packages/:id/disable GET

Overview

[ADMIN] PackageManager disable a package.

Summary

Disable the specified package. Package id hash must be provided. Package id could be found in installed package list.

Response

Returns http status 200 OK and disabled package ID if request is successful.

```
  {
"id": "56365210-4cce-11ea-a7bf-acde48001122"
}
```

Otherwise returns http status 4xx with errors.

```
  {
"error": "not_found",
"error_message": "Could not find package by ID."
}
```

Change Log

Version	Change
v11.8	Added /Administration/packages/:id/disable GET endpoint.

/Administration/packages/:id/enable GET

Overview

[ADMIN] PackageManager enable a package.

Summary

Enable the specified package. Package id hash must be provided. Package id could be found in installed package list.

Response

Returns http status 200 OK and enabled package ID if request is successful.

```
  {
"id": "56365210-4cce-11ea-a7bf-acde48001122"
}
```

Otherwise returns http status 4xx with errors.

```
  {
"error": "not_found",
"error_message": "Could not find package by ID."
}
```

Change Log

Version	Change
v11.8	Added /Administration/packages/:id/enable GET endpoint.

/Administration/packages/:id/uninstall GET

Overview

[ADMIN] PackageManager uninstall a package.

Summary

Uninstalls the specified package. Package id hash must be provided. Package id could be found in installed package list.

Response

Returns http status 200 OK and newly installed package ID if request is successful. Otherwise returns http status 4xx with errors.

```
  {
"error": "not_found",
"error_message": "Could not find package by ID."
}
```

Change Log

Version	Change
v11.8	Added /Administration/packages/:id/uninstall GET endpoint.

/Administration/packages/:unFile DELETE

Overview

[ADMIN] PackageManager delete a package.

Summary

Delete the specified package. Package 'unFile' hash must be provided. Package 'unFile' hash could be found in staging package list.

Response

Returns http status 200 OK if request is successful. Otherwise returns http status

4xx with errors.

```
  {
"error": "delete_package_error",
"error_message": "Could not find package file."
}
```

Change Log

Version	Change
v11.8	Added /Administration/packages/:unFile DELETE endpoint.

/Administration/packages/installed GET

Overview

[ADMIN] PackageManager lists installed packages.

Summary

Lists installed packages in the system.

Response

```
  {
    "packages": [
        {
            "name": "project_P3",
            "version": "1580885734",
            "type": "module",
            "published_date": "2020-02-06 12:51:17",
            "description": "",
            "uninstallable": true,
            "file_install": "cd590eb069a73b2eeb6c107776fd60dc",
            "file": "cd590eb069a73b2eeb6c107776fd60dc",
            "enabled": "ENABLED",
            "id": "66172e0e-48df-11ea-9f1f-acde48001122"
        }
    ]
}
```

Change Log

Version	Change
v11.8	Added /Administration/packages/installed GET endpoint.

/Administration/packages/staged GET

Overview

[ADMIN] PackageManager lists staged packages.

Summary

Lists staged packages in the system.

Response


```
  {
    "packages": [
        {
            "name": "project_P1",
            "version": 1580885419,
            "published_date": "2020-02-05 06:50:19",
            "description": "None",
            "uninstallable": "Yes",
            "type": "Module",
            "file": "0ef4dd8b237f4358bdb246a744eb2b38",
            "file_install": "5d4b4614421586fd2111714abf622392",
            "unFile": "5d4b4614421586fd2111714abf622392"
        }
    ]
}
```


Change Log

Version	Change
v11.8	Added /Administration/packages/staged GET endpoint.

/Administration/portalmodules GET

Overview

[ADMIN] Gets an array of modules currently enabled for Sugar Portal.

Summary

This endpoint gets an array of modules currently enabled for Sugar Portal. This endpoint is only available to administrators.

Request Arguments

This endpoint does not accept any arguments.

Response Example

```
  [
    'Home',
    'Bugs',
    'Cases',
    'KBContents',
]
```

Change Log

Version	Change
v11.13	Added Administration/portalmodules GET endpoint

/Administration/search/fields GET

Overview

[ADMIN] Search field configuration

Summary

This endpoint lists the full text search configuration of all fields for the full text search enabled modules. This endpoint is only available to administrators.

Request Arguments

Name	Type	Description	Required
module_list	String	Comma delimited list of modules to return. If omitted, all search enabled modules will be returned. Example: Accounts,Contacts	False
search_only	Boolean	When set, only searchable fields are returned. Defaults to false.	False
order_by_boost	Boolean	When set, a flat list of searchable fields is returned ordered by boost value.	False

Response

```
{
    "Accounts":{
        "name":{
            "name":"name",
            "type":"name",
            "searchable":true,
            "boost":1
        },
        "date_modified":{
            "name":"date_modified",
            "type":"datetime",
            "searchable":false
        }
    }
}
```

Response using order_by_boost

```
{
    "Quotes.quote_num":1.5,
    "Manufacturers.name":1,
    "Bugs.name":0.9,
    "ProjectTask.name":0.5,
}
```

Change Log

Version	Change
v10	Added /Administration/search/fields GET endpoint.

/Administration/search/reindex POST

Overview

[ADMIN] Search schedule reindex

Summary

This endpoint schedules a reindex. This endpoint is only available to administrators.

Request Arguments

Name	Type	Description	Required
module_list	String	Comma delimited list of modules to be reindexed. If omitted, all search enabled modules will be reindexed. Example: Accounts,Contacts	False
clear_data	Boolean	If set the records of the selected modules will be removed from the search backend and the schema will	False

Name	Type	Description	Required
		be recreated. All records from given modules will be queued to be reindexed which is orchestrated by the job queue. If not set, nothing is dropped from the search backend and all records of the selected modules are queued for reindexing. This is the default behavior. Example: Accounts,Contacts	

Response

```
{
    "success": true
}
```

Change Log

Version	Change
v10	Added /Administration/search/reindex POST endpoint.

/Administration/search/status GET

Overview

[ADMIN] Search status

Summary

This endpoint returns the status of the search backend including a list of the activated full text search modules. This endpoint is only available to administrators.

Response

```
{
    "available":true,
    "enabled_modules":
    [
        "Accounts",
        "Bugs",
        "Calls",
        "Campaigns",
    ]
}
```

Change Log

Version	Change
v10	Added /Administration/search/status GET endpoint.

/Administration/settings/auth GET

Overview

[ADMIN] Fetch auth settings.

Summary

This endpoint will return auth settings. This endpoint is only available to administrators when IDM migration is turned on.

Response if enabled only local auth provider

```
{
  "enabledProviders": [
    "local"
  ],
  "local": {
        "password_requirements": {
```

```
            "minimum_length": 6,
            "maximum_length": 0,
            "require_upper": true,
            "require_lower": true,
            "require_number": true,
            "require_special": false,
            "password_regex": "",
            "regex_description": ""
        },
        "password_expiration": {
            "time": 0,
            "attempt": 0
        }
    }
}
```


Response if enabled local and ldap auth providers


```
{
    "enabledProviders": [
        "local",
        "ldap"
    ],
    "local": {
        "password_requirements": {
            "minimum_length": 6,
            "maximum_length": 0,
            "require_upper": true,
            "require_lower": true,
            "require_number": true,
            "require_special": false,
            "password_regex": "",
            "regex_description": ""
        },
        "password_expiration": {
            "time": 0,
            "attempt": 0
        }
    },
    "ldap": {
        "user": {
            "mapping": {
                "givenName": "first_name",
```

```
                "sn": "last_name",
                "mail": "email1",
                "telephoneNumber": "phone_work",
                "facsimileTelephoneNumber": "phone_fax",
                "mobile": "phone_mobile",
                "street": "address_street",
                "l": "address_city",
                "st": "address_state",
                "postalCode": "address_postalcode",
                "c": "address_country"
            }
        },
        "adapter_config": {
            "host": "127.0.0.1",
            "port": "389",
            "options": {
                "network_timeout": 60,
                "timelimit": 60
            },
            "encryption": "none"
        },
        "adapter_connection_protocol_version": 3,
        "baseDn": "dc=openldap,dc=com",
        "uidKey": "uid",
        "filter": "({uid_key}={username})",
        "dnString": null,
        "entryAttribute": null,
        "autoCreateUser": "1",
        "searchDn": "cn=admin,ou=admins,dc=openldap,dc=com",
        "searchPassword": "admin&password"
    }
}
```

Response if enabled local and saml auth providers

```
{
    "enabledProviders": [
        "local",
        "saml"
    ],
    "local": {
        "password_requirements": {
            "minimum_length": 6,
```

```
            "maximum_length": 0,
            "require_upper": true,
            "require_lower": true,
            "require_number": true,
            "require_special": false,
            "password_regex": "",
            "regex_description": ""
        },
        "password_expiration": {
            "time": 0,
            "attempt": 0
        }
    },
    "saml": {
        "strict": true,
        "debug": false,
        "sp": {
            "entityId": "logoutFlowWithRedirectBinding",
            "assertionConsumerService": {
                "url": "http://localhost/index.php?module=Users&action
=Authenticate",
                "binding": "urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
POST"
            },
            "singleLogoutService": {
                "url": "http://localhost/index.php?module=Users&action
=Logout",
                "binding": "urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
Redirect"
            },
            "NameIDFormat": "urn:oasis:names:tc:SAML:1.1:nameid-
format:emailAddress",
            "x509cert": "",
            "privateKey": "",
            "provisionUser": true,
            "sugarCustom": []
        },
        "idp": {
            "entityId": "http://localhost:8080/simplesaml/saml2/idp/me
tadata.php",
            "singleSignOnService": {
                "url": "http://localhost:8080/simplesaml/saml2/idp/SSO
Service.php",
                "binding": "urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
Redirect"
            },
```

```
            "singleLogoutService": {
                "url": "http://localhost:8080/simplesaml/saml2/idp/Sin
gleLogoutService.php",
                "binding": "urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
Redirect"
            },
            "x509cert": "-----BEGIN CERTIFICATE-----\n --x509cert-- \n
-----END CERTIFICATE-----"
        },
        "security": {
            "authnRequestsSigned": false,
            "logoutRequestSigned": false,
            "logoutResponseSigned": false,
            "signatureAlgorithm": "http://www.w3.org/2001/04/xmldsig-
more#rsa-sha256",
            "validateRequestId": false
        }
    }
}
```


Change Log

Version	Change
v11_2	Added /Administration/settings/auth GET endpoint.

/Administration/settings/idmMode DELETE

Overview

[ADMIN] Turn IDM-mode off.

Summary

This endpoint switches IDM mode off. This is only available to administrators when IDM migration is turned on and intended for internal use for IDM integration.

Response when IDM mode has been disabled


```
[]
```

Change Log

Version	Change
v11_2	Added /Administration/settings/idmMode DELETE endpoint.

/Administration/settings/idmMode POST

Overview

[ADMIN] Turn IDM-mode on.

Summary

This endpoint switches IDM mode on. This endpoint is only available to administrators when IDM migration is turned on and intended for internal use for IDM integration.

Request to enable IDM-mode.

```
{"idmMode":
    {
        "enabled": true,
        "clientId": "mangoOIDCClientId",
        "clientSecret": "mangoOIDCClientSecret",
        "stsUrl": "http://sts.sugarcrm.local",
        "idpUrl": "http://login.sugarcrm.local",
        "stsKeySetId": "mangoOIDCKeySet",
        "tid": "srn:cloud:idp:eu:0000000001:tenant",
        "crmOAuthScope": "https://apis.sugarcrm.com/auth/crm",
        "requestedOAuthScopes": ["offline", "https://apis.sugarcrm.com
/auth/crm", "profile", "email", "address", "phone"],
        "cloudConsoleUrl": "http://console.sugarcrm.local",
        "allowedSAs": ["srn:cloud:iam:us-west-2:9999999999:sa:user-
sync"]
    }
}
```

Refer to multiverse documenation for parameters explanation.

Response when IDM mode has been enabled


```
{
    "tid": "srn:cloud:idp:eu:0000000001:tenant",
    "clientId": "mangoOIDCClientId",
    "clientSecret": "mangoOIDCClientSecret",
    "stsUrl": "http://sts.sugarcrm.local",
    "idpUrl": "http://login.sugarcrm.local",
    "redirectUri": "http://sugar",
    "urlAuthorize": "http://sts.sugarcrm.local/oauth2/auth",
    "urlAccessToken": "http://sts.sugarcrm.local/oauth2/token",
    "urlResourceOwnerDetails": "http://sts.sugarcrm.local/oauth2/intro
spect",
    "urlUserInfo": "http://sts.sugarcrm.local/userinfo",
    "http_client": [],
    "cloudConsoleUrl": "http://console.sugarcrm.local",
    "cloudConsoleRoutes": [],
    "caching": [],
    "crmOAuthScope": "https://apis.sugarcrm.com/auth/crm",
    "requestedOAuthScopes": [
        "offline",
        "https://apis.sugarcrm.com/auth/crm",
        "profile",
        "email",
        "address",
        "phone"
    ],
    "keySetId": "mangoOIDCKeySet",
    "urlKeys": "http://sts.sugarcrm.local/keys/mangoOIDCKeySet",
    "allowedSAs": ["srn:cloud:iam:us-west-2:9999999999:sa:user-sync"]
}
```


Change Log

Version	Change
v11_2	Added /Administration/settings/idmMode POST endpoint.

/Calendar/invitee_search GET

Overview

Temporary API - Do Not Use! This endpoint will be removed in an upcoming release. Use /search endpoint instead. Searches for people to invite to an event (meeting or call).

Request Arguments

Name	Type	Description	Required
q	String	The search text to match records on.	True
module_list	String	Comma delimited list of modules to search.	True
search_fields	String	Comma delimited list of fields to search.	True
fields	String	Comma delimited list of fields to return. Search fields will automatically be added to return list.	True

Response Arguments

Name	Type	Description
next_offset	Integer	Currently only returns -1 as paging is not supported yet.
records	Array	An array of results containing matched records.

Response


```
{
    "next_offset": -1,
    "records": [
        {
            "id": "17283aad-8d15-c545-fc5a-5422fe3d8ef8",
            "date_modified": "2014-09-24T13:25:28-04:00",
```

```
            "email": [
                {
                    "email_address": "hr.phone.support@example.cn",
                    "invalid_email": false,
                    "opt_out": true,
                    "primary_address": false,
                    "reply_to_address": false
                },
                {
                    "email_address": "section.section.phone@example.ed
u",
                    "invalid_email": false,
                    "opt_out": false,
                    "primary_address": true,
                    "reply_to_address": true
                }
            ],
            "full_name": "Renaldo Cuffee",
            "account_name": "EEE Endowments LTD",
            "_acl": {
                "fields": {}
            },
            "_module": "Contacts",
            "_search": {
                "highlighted": {
                    "account_name": {
                        "text": "EEE Endowments LTD",
                        "module": "Contacts",
                        "label": "LBL_ACCOUNT_NAME"
                    }
                }
            }
        },
        {
            "id": "19c0fa4a-a6c0-940f-7ad6-5422fe62a00a",
            "date_modified": "2014-09-24T13:25:28-04:00",
            "email": [
                {
                    "email_address": "qa.beans@example.com",
                    "invalid_email": false,
                    "opt_out": true,
                    "primary_address": false,
                    "reply_to_address": false
                },
                {
                    "email_address": "the.phone.sales@example.de",
```

```
                    "invalid_email": false,
                    "opt_out": false,
                    "primary_address": true,
                    "reply_to_address": true
                }
            ],
            "full_name": "Noble Canto",
            "account_name": "EEE Endowments LTD",
            "_acl": {
                "fields": {}
            },
            "_module": "Contacts",
            "_search": {
                "highlighted": {
                    "account_name": {
                        "text": "EEE Endowments LTD",
                        "module": "Contacts",
                        "label": "LBL_ACCOUNT_NAME"
                    }
                }
            }
        }
    ]
}
```


Change Log

Version	Change
v10	Added /<module>/send_invites GET endpoint.

/Calls POST

Overview

Create a single event or a series of event records.

Request Arguments

Name	Type	Description	Required
<record field>	<record field	The name value	True

Name	Type	Description	Required
	type>	list of fields to populate.	

Request

```
{
    "name":"Department Meeting",
    "description":"Weekly Department Meeting",
    "date_start":"yyyy-mm-ddThh:mm:ss-00:00",
    "date_end":"yyyy-mm-ddThh:mm:ss-00:00",
    "repeat_type":"Weekly",
    "repeat_interval":"1",  /* Every Week */
    "repeat_dow":"25",      /* Tue and Fri */
    "repeat_count":"4",     /* 4 Recurrences */
    "repeat_until":"",
}
```

Response Arguments

Name	Description	Start Date	End Date
<record field>	Returns the fields for the newly created record.		

Response

```
{

}
```

Change Log

Version	Change
v10	Added /<module> POST endpoint.

/Calls/:record DELETE

Overview

Deletes either a single event record or a series of event records

Request Arguments

Name	Type	Description	Required
all_recurrences	Boolean	Flag to delete all events in a series.	False

Request

```
http://{site_url}/rest/v10/Meetings/2c9e5c09-6824-0d14-f5cb-5130321ac3cf?all_recurrences=true
```

Response Arguments

Name	Type	Description
id	String	Returns the ID of the deleted record.

Response

```
{
    "id": "11cf0d0a-40af-8cb1-9da0-5057a5f511f9"
}
```

Change Log

Version	Change
v10	Added /<module>/:record DELETE endpoint.

/Calls/:record PUT

Overview

Update a calendar event record of the specified type.

Request Arguments

Name	Type	Description	Required
all_recurrences	Boolean	Flag to update all events in a series.	False

Request

```
http://{site_url}/rest/v10/Meetings/2c9e5c09-6824-0d14-f5cb-5130321ac3
cf?all_recurrences=true
{
    "name":"Department Meeting",
    "description":"Weekly Department Meeting",
    "date_start":"yyyy-mm-ddThh:mm:ss-00:00",
    "date_end":"yyyy-mm-ddThh:mm:ss-00:00",
    "repeat_type":"Weekly",
    "repeat_interval":"1",  /* Every Week */
    "repeat_dow":"25",      /* Tue and Fri */
    "repeat_count":"4",     /* 4 Recurrences */
    "repeat_until":"",
}
```

Response Arguments

Name	Description	Start Date	End Date
<record field>	Returns the fields for the updated record.		

Response

```
{

}
```

Change Log

Version	Change
v10	Added /<module>/:record PUT endpoint.

/Cases/clients/portal PUT

Overview

[ADMIN] Set a case as 'Requested For Close'

Summary

This endpoint sets a case as 'Requested for Close' in the portal on a Sugar Serve instance.

Request Arguments

This endpoint does not accept any arguments.

Response Arguments

Name	Type	Description
<record field>	<record field type>	Returns the fields for the updated Cases record.

Response


```
{
  "id":"d6582610-d69b-11ea-afe4-acde48001122",
  "name":"Need to purchase additional licenses",
  "date_entered":"2020-08-04T21:44:16+00:00",
  "date_modified":"2020-08-04T21:56:03+00:00",
  "modified_user_id":"19dc0582-d69c-11ea-95e6-acde48001122",
  "modified_by_name":"Sugar Customer Support Portal ",
  "created_by":"1",
  "created_by_name":"Administrator",
  "description":"",
  "deleted":false,
  "case_number":150,
  "type":"Product",
  "status":"Pending Input",
  "priority":"P2",
  "resolution":"",
  "work_log":"",
```

```
    "follow_up_datetime":"",
    "widget_follow_up_datetime":"",
    "resolved_datetime":"",
    "hours_to_resolution":null,
    "business_hours_to_resolution":null,
    "pending_processing":false,
    "account_name":"Gifted Holdings AG",
    "business_center_id":"",
    "source":"",
    "request_close":true,
    "request_close_date":"2020-08-04T21:56:03+00:00",
    "portal_viewable":true,
    "widget_status":"",
    "primary_contact_name":"",
    "team_count":"",
    "team_name":[{"id":"West","name":"West","name_2":"","primary":true
,"selected":false}],
    "first_response_target_datetime":"",
    "first_response_actual_datetime":"",
    "hours_to_first_response":null,
    "business_hrs_to_first_response":null,
    "first_response_var_from_target":null,
    "first_response_sla_met":"",
    "first_response_user_id":"",
    "first_response_user_name":"",
    "first_response_sent":false,
    "_acl":{"edit":"no","create":"no","fields":{}},"_module":"Cases"
}
```


Change Log

Version	Change
v11_10	Added /Cases/clients/portal PUT endpoint.

/CommentLog/:record GET

Overview

Retrieves a record.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

Name	Type	Description
<record field>	<record field type>	Returns the fields for the selected record.

Response


```
{
   "id":"bdd59d85-687b-1739-b00a-512d09f6db9e",
   "name":"Insight Marketing Inc",
   "date_entered":"2013-02-26T19:12:00+00:00",
   "date_modified":"2013-02-26T19:12:00+00:00",
   "modified_user_id":"1",
   "modified_by_name":"Administrator",
   "created_by":"1",
   "created_by_name":"Administrator",
   "description":"",
   "img":"",
   "last_activity_date":"2013-02-26T19:12:00+00:00",
   "deleted":false,
   "assigned_user_id":"seed_max_id",
   "assigned_user_name":"Max Jensen",
   "team_name":[
      {
         "id":"East",
         "name":"East",
         "name_2":"",
         "primary":false
      },
      {
         "id":1,
         "name":"Global",
         "name_2":"",
         "primary":false
      },
      {
         "id":"West",
         "name":"West",
         "name_2":"",
         "primary":true
      }
   ],
```

```
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"account_type":"Customer",
"industry":"Electronics",
"annual_revenue":"",
"phone_fax":"",
"billing_address_street":"345 Sugar Blvd.",
"billing_address_street_2":"",
"billing_address_street_3":"",
"billing_address_street_4":"",
"billing_address_city":"San Mateo",
"billing_address_state":"CA",
"billing_address_postalcode":"56019",
"billing_address_country":"USA",
"rating":"",
"phone_office":"(927) 136-9572",
"phone_alternate":"",
"website":"www.sectionvegan.de",
"ownership":"",
"employees":"",
"ticker_symbol":"",
"shipping_address_street":"345 Sugar Blvd.",
"shipping_address_street_2":"",
"shipping_address_street_3":"",
"shipping_address_street_4":"",
"shipping_address_city":"San Mateo",
"shipping_address_state":"CA",
"shipping_address_postalcode":"56019",
"shipping_address_country":"USA",
"email1":"kid.support.vegan@example.info",
"parent_id":"",
"sic_code":"",
"parent_name":"",
"email_opt_out":false,
"invalid_email":false,
"email":[
   {
      "email_address":"kid.support.vegan@example.info",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"1"
   },
   {
      "email_address":"phone.kid@example.cn",
```

```
         "opt_out":"0",
         "invalid_email":"0",
         "primary_address":"0"
      }
   ],
   "campaign_id":"",
   "campaign_name":"",
   "my_favorite":false,
   "_acl":{
      "fields":{

      }
   }
}
```

Change Log

Version	Change
v10	Added /<module>/:record GET endpoint.

/ConsoleConfiguration/config POST

Overview

Saves configuration changes in the database.

Request Arguments

This endpoint does not accept any request arguments.

URL Example

```
/rest/v11_9/ConsoleConfiguration/config
```

Request Payload Example

```
{
 "is_setup":true,
```

```
"enabled_modules":{
"c108bb4a-775a-11e9-b570-f218983a1c3e":[
       "Cases"
],
"da438c86-df5e-11e9-9801-3c15c2c53980":[
       "Accounts",
       "Opportunities"
]
},
"order_by_primary":{
"c108bb4a-775a-11e9-b570-f218983a1c3e":{
       "Cases":"follow_up_datetime:asc"
},
"da438c86-df5e-11e9-9801-3c15c2c53980":{
       "Accounts":"next_renewal_date:asc",
       "Opportunities":"date_closed:asc"
}
},
"order_by_secondary":{
"c108bb4a-775a-11e9-b570-f218983a1c3e":{
       "Cases":""
},
"da438c86-df5e-11e9-9801-3c15c2c53980":{
       "Accounts":"",
   "Opportunities":""
}
},
"filter_def":{
"c108bb4a-775a-11e9-b570-f218983a1c3e":{
       "Cases":[
              {
                     "status":{
                "$not_in":[
                       "Closed",
                       "Rejected",
                       "Duplicate"
                ]
                     }
              },
              {
         "$owner":""
              }
       ]
},
"da438c86-df5e-11e9-9801-3c15c2c53980":{
  "Accounts":[
```

```
        {
               "$owner":""
        }
    ],
    "Opportunities":[
        {
               "sales_status":{
              "$not_in":[
                     "Closed Won",
                     "Closed Lost"
              ]
               }
        },
        {
           "$owner":""
        }
    ]
}
},
"defaults":{
"is_setup":0,
"enabled_modules":{
   "c108bb4a-775a-11e9-b570-f218983a1c3e":[
          "Cases"
     ],
   "da438c86-df5e-11e9-9801-3c15c2c53980":[
          "Accounts",
          "Opportunities"
   ]
},
"order_by_primary":{
   "c108bb4a-775a-11e9-b570-f218983a1c3e":{
          "Cases":"follow_up_datetime:asc"
   },
   "da438c86-df5e-11e9-9801-3c15c2c53980":{
          "Accounts":"next_renewal_date:asc",
          "Opportunities":"date_closed:asc"
   }
},
"order_by_secondary":{
   "c108bb4a-775a-11e9-b570-f218983a1c3e":{
          "Cases":""
   },
   "da438c86-df5e-11e9-9801-3c15c2c53980":{
          "Accounts":"",
          "Opportunities":""
```

```
   }
},
"filter_def":{
       "c108bb4a-775a-11e9-b570-f218983a1c3e":{
      "Cases":[
             {
            "status":{
                  "$not_in":[
                        "Closed",
                        "Rejected",
                        "Duplicate"
                  ]
            }
             },
             {
            "$owner":""
             }
      ]
       },
       "da438c86-df5e-11e9-9801-3c15c2c53980":{
              "Accounts":[
                    {
              "$owner":""
                    }
              ],
              "Opportunities":[
                    {
              "sales_status":{
                     "$not_in":[
                           "Closed Won",
                           "Closed Lost"
                     ]
              }
                    },
                    {
              "$owner":""
                    }
              ]
       }
       }
},
"labels":{
"Cases":[
       {
      "label":"LBL_LIST_PRIORITY/LBL_STATUS",
      "labelValue":"Priority/Status"
```

```
        },
        {
               "label":"LBL_LIST_SUBJECT/LBL_SERVICE_LEVEL",
               "labelValue":"Subject/Service Level"
        }
   ]
   },
   "viewdefs":{
      "Cases":{
      "base":{
             "view":{
                    "multi-line-list":{
               "panels":[
                      {
                             "label":"LBL_LABEL_1",
                             "fields":[
                                    {
                               "name":"case_number",
                               "label":"LBL_AGENT_WORKBENCH_NUMBER",
                               "subfields":[
                                      {
                                             "default":true,
                                             "enabled":true,
                                              "name":"case_number",
                                            "label":"LBL_AGENT_WORKBENCH_NUMB
ER"
                                      }
                               ]
                                    },
                                    {
                               "name":"Priority/Status",
                               "label":"LBL_LIST_PRIORITY/LBL_STATUS",
                               "subfields":[
                                      {
                                             "default":true,
                                             "enabled":true,
                                             "name":"priority",
                                             "label":"LBL_LIST_PRIORIT
Y",
                                             "type":"enum"
                                      },
                                      {
                                             "default":true,
                                             "enabled":true,
                                             "name":"status",
                                             "label":"LBL_STATUS",
```

```
                                        "type":"case-status",
                                        "widget_name":"widget_sta
tus"
                                    }
                                ]
                                },
                                {
                                "name":"Subject/Service Level",
                                "label":"LBL_LIST_SUBJECT/LBL_SERVICE_L
EVEL",
                                "subfields":[
                                    {
                                        "default":true,
                                        "enabled":true,
                                        "name":"name",
                                        "label":"LBL_LIST_SUBJECT
",
                                        "link":false
                                    },
                                    {
                                        "default":true,
                                        "enabled":true,
                                        "name":"service_level",
                                        "label":"LBL_SERVICE_LEVE
L",
                                        "type":"enum",
                                        "enum_module":"Accounts",
                                        "link":false
                                    }
                                ]
                                }
                            ]
                        }
                    ],
                    "collectionOptions":{
                "limit":100,
                "params":{
                        "max_num":100,
                        "order_by":"follow_up_datetime",
                        "nulls_last":true
                }
                    },
                    "filterDef":[
                {
                        "status":{
                                "$not_in":[
```

```
                              "Closed",
                              "Rejected",
                              "Duplicate"
                                   ]
                             },
                             "$owner":""
                    }
                      ]
                    }
              }
       }
       }
   }
}
```

Response Arguments

Response

```
  {
   "is_setup":1,
   "enabled_modules":{
   "c108bb4a-775a-11e9-b570-f218983a1c3e":[
      "Cases"
   ],
   "da438c86-df5e-11e9-9801-3c15c2c53980":[
      "Accounts",
      "Opportunities"
   ]
   },
   "order_by_primary":{
   "c108bb4a-775a-11e9-b570-f218983a1c3e":{
      "Cases":"follow_up_datetime:asc"
   },
   "da438c86-df5e-11e9-9801-3c15c2c53980":{
      "Accounts":"next_renewal_date:asc",
      "Opportunities":"date_closed:asc"
   }
   },
   "order_by_secondary":{
   "c108bb4a-775a-11e9-b570-f218983a1c3e":{
      "Cases":""
   },
   "da438c86-df5e-11e9-9801-3c15c2c53980":{
```

```
   "Accounts":"",
   "Opportunities":""
}
},
"filter_def":{
"c108bb4a-775a-11e9-b570-f218983a1c3e":{
   "Cases":[
      {
         "status":{
            "$not_in":[
               "Closed",
               "Rejected",
               "Duplicate"
            ]
         }
      },
      {
         "$owner":""
      }
   ]
},
"da438c86-df5e-11e9-9801-3c15c2c53980":{
   "Accounts":[
      {
         "$owner":""
      }
   ],
   "Opportunities":[
      {
         "sales_status":{
            "$not_in":[
               "Closed Won",
               "Closed Lost"
            ]
         }
      },
      {
         "$owner":""
      }
   ]
}
},
"defaults":{
"is_setup":0,
"enabled_modules":{
   "c108bb4a-775a-11e9-b570-f218983a1c3e":[
```

```
      "Cases"
   ],
   "da438c86-df5e-11e9-9801-3c15c2c53980":[
      "Accounts",
      "Opportunities"
   ]
},
"order_by_primary":{
   "c108bb4a-775a-11e9-b570-f218983a1c3e":{
      "Cases":"follow_up_datetime:asc"
   },
   "da438c86-df5e-11e9-9801-3c15c2c53980":{
      "Accounts":"next_renewal_date:asc",
      "Opportunities":"date_closed:asc"
   }
},
"order_by_secondary":{
   "c108bb4a-775a-11e9-b570-f218983a1c3e":{
      "Cases":""
   },
   "da438c86-df5e-11e9-9801-3c15c2c53980":{
      "Accounts":"",
      "Opportunities":""
   }
},
"filter_def":{
   "c108bb4a-775a-11e9-b570-f218983a1c3e":{
      "Cases":[
         {
            "status":{
               "$not_in":[
                  "Closed",
                  "Rejected",
                  "Duplicate"
               ]
            }
         },
         {
            "$owner":""
         }
      ]
   },
   "da438c86-df5e-11e9-9801-3c15c2c53980":{
      "Accounts":[
         {
            "$owner":""
```

```
            }
         ],
         "Opportunities":[
            {
               "sales_status":{
                  "$not_in":[
                     "Closed Won",
                     "Closed Lost"
                  ]
               }
            },
            {
               "$owner":""
            }
         ]
      }
   }
   }
}
```

Response Codes and Error Messages

This API does not return new errors other than the ones that may be returned by its parent API - ConfigModule API. The following error responses are possible:

- **403 (SugarAPIExceptionNotAuthorized)** : Sent if the user attempts to access a module that they are forbidden to access.

Change Log

Version	Change
v11.9	Added /ConsoleConfiguration/config POST endpoint.

/ConsoleConfiguration/default-metadata GET

Overview

Gets the request metadata in JSON format.

Request Arguments

This endpoint does not accept any request arguments.

URL Parameters

Parameter	Description	Required?
modules	String. Comma-separated module names. Example: 'Accounts,Opportunities'.	Yes
type	String. The type of the desired metadata. The following types are supported: 'filter', 'layout', 'menu', 'view'.	Yes
name	String. The name of the metadata. Examples: 'list', 'record', or 'multi-line-list'.	Yes

URL Example

```
  /rest/v11_9/ConsoleConfiguration/default-
metadata?modules=Cases&type=view&name=multi-line-list
```

Result Arguments

Response

```
  {
   "Cases":{
   "panels":[
      {
         "label":"LBL_PANEL_1",
         "fields":[
            {
               "name":"case_number",
               "label":"LBL_AGENT_WORKBENCH_NUMBER",
               "width":"xsmall",
               "subfields":[
                  {
                     "name":"case_number",
                     "label":"LBL_AGENT_WORKBENCH_NUMBER",
                     "default":true,
                     "enabled":true,
```

```
            "readonly":true
         }
      ]
   },
   {
      "name":"status",
      "label":"LBL_AGENT_WORKBENCH_PRIORITY_STATUS",
      "width":"small",
      "subfields":[
         {
            "name":"priority",
            "label":"LBL_LIST_PRIORITY",
            "default":true,
            "enabled":true,
            "type":"enum"
         },
         {
            "name":"status",
            "label":"LBL_STATUS",
            "default":true,
            "enabled":true,
            "type":"case-status",
            "widget_name":"widget_status"
         }
      ]
   },
   {
      "name":"follow_up_datetime",
      "label":"LBL_AGENT_WORKBENCH_FOLLOW_UP",
      "width":"medium",
      "subfields":[
         {
            "name":"follow_up_datetime",
            "label":"LBL_FOLLOW_UP_DATETIME",
            "default":true,
            "enabled":true,
            "readonly":true,
            "type":"follow-up-datetime-colorcoded",
            "widget_name":"widget_follow_up_datetime",
            "color_code_classes":{
               "overdue":"expired",
               "in_a_day":"soon-expired",
               "more_than_a_day":"white black-text"
            }
         },
         {
```

```
            "name":"follow_up_datetime",
            "label":"LBL_FOLLOW_UP_DATETIME",
            "default":true,
            "enabled":true,
            "readonly":true,
            "type":"datetimecombo"
         }
      ]
   },
   {
      "name":"name",
      "label":"LBL_AGENT_WORKBENCH_SUBJECT_DESCRIPTION",
      "width":"xlarge",
      "subfields":[
         {
            "name":"name",
            "label":"LBL_LIST_SUBJECT",
            "link":false,
            "default":true,
            "enabled":true,
            "readonly":true
         },
         {
            "name":"description",
            "default":true,
            "enabled":true,
            "readonly":true,
            "sortable":false
         }
      ]
   },
   {
      "name":"business_center",
      "label":"LBL_BUSINESS_CENTER",
      "width":"small",
      "subfields":[
         {
            "name":"business_center_name",
            "label":"LBL_BUSINESS_CENTER",
            "link":false,
            "default":true,
            "enabled":true,
            "readonly":true
         }
      ]
   },
```

```
    {
       "name":"account_name",
       "label":"LBL_ACCOUNT",
       "width":"medium",
       "subfields":[
          {
             "name":"account_name",
             "label":"LBL_LIST_ACCOUNT_NAME",
             "module":"Accounts",
             "id":"ACCOUNT_ID",
             "ACLTag":"ACCOUNT",
             "related_fields":[
                "account_id"
             ],
             "link":false,
             "default":true,
             "enabled":true
          },
          {
             "name":"service_level",
             "label":"LBL_SERVICE_LEVEL",
             "type":"enum",
             "enum_module":"Accounts",
             "link":false,
             "default":true,
             "enabled":true,
             "readonly":true
          }
       ]
    },
    {
       "name":"assigned_user_name",
       "label":"LBL_ASSIGNED_TO_NAME",
       "width":"small",
       "subfields":[
          {
             "name":"assigned_user_name",
             "label":"LBL_ASSIGNED_TO_NAME",
             "id":"ASSIGNED_USER_ID",
             "link":false,
             "default":true,
             "enabled":true
          }
       ]
    }
]
```

```
      }
   ],
   "collectionOptions":{
      "limit":100,
      "params":{
         "max_num":100,
         "order_by":"follow_up_datetime",
         "nulls_last":true
      }
   },
   "filterDef":[
      {
         "status":{
            "$not_in":[
               "Closed",
               "Rejected",
               "Duplicate"
            ]
         },
         "$owner":""
      }
   ]
   }
}
```

Response Codes and Error Messages

All successful responses return 200.

The following error responses are possible:

- **403 (SugarAPIExceptionNotAuthorized)** : Sent if the user attempts to access a module that they are forbidden to access.
- **422 (SugarApiExceptionInvalidParameter)** : Sent if the user specifies an invalid type or module.

Change Log

Version	Change
v11.9	Added /ConsoleConfiguration/default-metadata GET endpoint.

/Contact/:record/Cases GET

Overview

Get a contact's related cases, filtered by an expression, which are accessible to a contact with portal visibility.

Summary

This endpoint will return a set of cases filtered by an expression which are accessible to a contact with portal visibility. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its ID. If both a filter definition and a filter ID are passed, the two filters will be joined with an
AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side, otherwise the runtime of the endpoint will be very long.

Request

GET /Contact/:id/Cases

Request Arguments

Name	Type	Description	Required
filter	String	See the Filter API for details.	False
filter_id	String	Identifier for a pre-existing filter. See the Filter API for details.	False
max_num	Integer	Max number of records that can be returned. Default is 20, unless overruled by your Sugar configuration.	False
offset	Integer	The number of records to skip over before records are returned. Default is	False

		0.	
fields	String	Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters (applicable to link and collection fields). The field's ID and date_modified will always be returned. For more details on additional field parameters, see the Relate API and Collection API.	False
order_by	String	See the Filter API.	False
deleted	Boolean	Boolean to show deleted records in the result set.	False

Response


```
{
  "next_offset": -1,
  "records": [
    {
      "id": "d509b3da-29f6-7b23-7cce-56fab1a6335b",
      "name": "Need help",
      "date_modified": "2016-03-29T09:46:00-07:00",
      "_acl": {
        "fields": {}
      },
      "_module": "Cases"
    },
    {
      "id": "cc4982cd-0bdf-bad4-d079-56fab4bdddf1",
      "name": "Having trouble",
```

```
      "date_modified": "2016-03-29T09:57:58-07:00",
      "_acl": {
        "fields": {}
      },
      "_module": "Cases"
    }
  ]
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Change Log

Version	Change
v11	Added /Contact/:record/Cases GET endpoint.

/Contacts/:record/freebusy GET

Get a contact's FreeBusy schedule

Summary:

This endpoint returns a list of time slots for which the specified person is busy.

Request

GET /Contacts/:id/freebusy

Response

```
{
  "id": "foo"
  "module": "Users",
  "freebusy": [
                {
                  "start": "2014-08-24T08:45:00-04:00",
                  "end": "2014-08-24T09:15:00-04:00"
                },
                {
                  "start": "2014-08-30T05:45:00-04:00",
                  "end": "2014-08-30T06:15:00-04:00"
                },
                {
                  "start": "2014-09-12T15:45:00-04:00",
                  "end": "2014-09-12T16:15:00-04:00"
                }
              ]
}
```

/Currencies GET

Returns a collection of Currency models

Summary:

This end point is used to return the Currencies defined in the application

Url Parameters:

Param	Description	Optional

Possible Errors

Error	Description

Url Example:

/rest/v10/Currencies

Output Example:

{ "next_offset": -1, "records": [{ "id": "-99", "name": "US Dollars", "symbol": "$", "iso4217": "USD", "conversion_rate": 1, "status": "Active", "_acl": { "fields": {} }, "_module": "Currencies" }, { "id": "a3cba348-756c-935c-66ad-55d772c7960f", "name": "Euro", "symbol": "€", "iso4217": "EUR", "conversion_rate": 0.9, "status": "Active", "date_modified": "2015-08-21T11:47:51-07:00", "created_by": "1", "_acl": { "fields": {} }, "_module": "Currencies" }] }

/Dashboards GET

Overview

List current user's filtered Home dashboards.

Summary

This endpoint will return a set of Home dashboards filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long.

Notice

The behavior of this endpoint has changed in v11. For a description of behavior in v11, refer to GET /<module>. If you continue to use v10 of the API, your REST calls should behave as before. But, it is advisable to upgrade to v11. Slight differences exist in the GET method of v10 vs v11.

Migrating to v11:
Recommended Replacement: GET /Dashboards

The Home module is no longer the default dashboard_module. To get the Home module dashboard, include the following filter parameter.
v11: GET /Dashboards?filter[0][dashboard_module]=Home

name, id. etc. are no longer specified as default fields. To retain the default fields, use
v11: GET /Dashboards?filter[0][dashboard_module]=Home&fields=id,name,view_name

Request Arguments

Name	Type	Description	Required
filter	String	See Filter API.	False
filter_id	String	Identifier for a preexisting filter. See the Filter API for details.	False
max_num	Integer	Max number of records that can be returned. Default is 20, unless overruled by your Sugar configuration.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
fields	String	Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters (applicable to link and collection fields). The fields id	False

Name	Type	Description	Required
		and date_modified will always be returned. For more details on additional field parameters, see Relate API and Collection API.	
order_by	String	See Filter API.	False
deleted	Boolean	Boolean to show deleted records in the result set.	False

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Response


```
{
  "next_offset": -1,
  "records": [
    {
      "id": "c630c772-faca-2051-6a42-56fab2e73e42",
      "name": "New Home Dashboard",
      "date_modified": "2016-03-29T09:49:06-07:00",
      "view_name": "",
      "_acl": {
        "fields": {}
      },
      "view": "",
      "_module": "Dashboards"
    },
    {
      "id": "15677f2c-f00d-c59a-6fbe-56fab1f33732",
```

```
      "name": "Home Dashboard2",
      "date_modified": "2016-03-29T09:47:25-07:00",
      "view_name": "",
      "_acl": {
        "fields": {}
      },
      "view": "",
      "_module": "Dashboards"
    },
    {
      "id": "d509b3da-29f6-7b23-7cce-56fab1a6335b",
      "name": "Accounts Dashboard",
      "date_modified": "2016-03-29T09:46:00-07:00",
      "view_name": "records",
      "_acl": {
        "fields": {}
      },
      "view": "records",
      "_module": "Dashboards"
    },
    {
      "id": "cc4982cd-0bdf-bad4-d079-56fab4bdddf1",
      "name": "Activity Stream Dashboard",
      "date_modified": "2016-03-29T09:57:58-07:00",
      "view_name": "activities",
      "_acl": {
        "fields": {}
      },
      "view": "activities",
      "_module": "Dashboards"
    }
  ]
}
```


Change Log

Version	Change
v11	Behavior has changed. Please see GET /Dashboards for the designated replacement.
v10	Added /Dashboards GET endpoint.

/Dashboards POST

Create a new Home dashboard.

Notice

The behavior of this endpoint has changed in v11. For a description of behavior in v11, refer to POST /<module>. If you continue to use v10 of the API, your REST calls should behave as below. But, it is advisable to upgrade to v11.
Migrating to v11:
Recommended Replacement: POST /Dashboards
The Home module is no longer the default dashboard_module. To specify the Home module, include it in the Request Body.
v11: POST /Dashboards with the following in the Request Body:
{"dashboard_module":"Home"}

Request Arguments

Name	Type	Description	Required
<record field>	<record field type>	The name value list of fields to populate.	True

Request

```
{
    "name": "New Home Dashboard",
    "dashboard_module": "Home",
    "view_name": "",
    "metadata": {
        "components": [{
            "rows": [[{
                "width": 12,
                "view": {
                    "limit": 10,
                    "visibility": "user",
                    "label": "Active Tasks",
                    "type": "active-tasks",
                    "module": null,
                    "template": "tabbed-dashlet"
                }
            }]],
            "width": 12
        }]
    }
```

```
}
```

Response Arguments

Name	Type	Description
<record field>	<record field type>	Returns the fields for the newly created dashboard.

Response


```
{
  "id": "4ccda6fa-fa02-9c8e-215f-56cf72f98fcb",
  "name": "New Home Dashboard",
  "date_entered": "2016-02-25T13:30:51-08:00",
  "date_modified": "2016-02-25T13:30:51-08:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
  "description": "",
  "deleted": false,
  "dashboard_module": "Home",
  "view_name": "",
  "metadata": {
    "components": [
      {
        "rows": [
          [
            {
              "width": 12,
              "view": {
                "limit": 10,
                "visibility": "user",
                "label": "Active Tasks",
                "type": "active-tasks",
                "module": null,
                "template": "tabbed-dashlet"
              }
            }
          ]
        ],
        "width": 12
```

```
      }
    ]
  },
  "following": "",
  "my_favorite": false,
  "tag": [],
  "assigned_user_id": "1",
  "assigned_user_name": "Administrator",
  "_acl": {
    "fields": {}
  },
  "view": "",
  "_module": "Dashboards"
}
```


Change Log

Version	Change
v11	Behavior has changed. Please see POST /Dashboards for the designated replacement.
v10	Added /Dashboards POST endpoint.

/Dashboards/:id/restore-metadata PUT

Overview

Restores the metadata for a module's dashboard to the default metadata.

Summary

This endpoint is used to restore the default metadata for a module's dashboard.

Request Arguments

Name	Type	Description	Required
dashboard_module	string	The module dashboard belongs to (e.g. Home, Cases, Accounts)	True

Name	Type	Description	Required
dashboard	string	The dashboard name (e.g. portal-home)	True

Request

```
  {
    "dashboard_module":"Home",
    "dashboard":"portal-home"
}
```

Response Example

```
  {
    "id":"123",
    "name":"LBL_PORTAL_HOME",
    "dashboard_module": "Home",
    "view_name": "home",
    "metadata": {
        "css_class": "portal-home-dashboard"
        "components": {...},
    },
    ...
}
```

Change Log

Version	Change
v11.13	Added /Dashboards/<id>/restore-metadata PUT endpoint.

/Dashboards/ GET

Overview

List current user's filtered dashboards.

Summary

This endpoint will return a set of dashboards filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, currently the definition will override the id, **but please do not rely on this behavior as it is subject to change**. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long.

Notice

This endpoint is not supported in v11. If you continue to use v10 of the API, your REST calls should behave as below. But, it is advisable to upgrade to v11.
Migrating to v11:
For more information on behavior in v11, refer to GET /<module>.
Recommended Replacement: GET /Dashboards

The module name is no longer part of the path. To specify a module name, include it as a filter parameter. For example, to retrieve all dashboards from the Accounts module, use
v11: GET /Dashboards?filter[0][dashboard_module]=Accounts

name, id. etc. are no longer specified as default fields. To retain the default fields, use
v11: GET /Dashboards?filter[0][dashboard_module]=Accounts&fields=id,name,view_name

Request Arguments

Name	Type	Description	Required
filter	String	See Filter API.	False
filter_id	String	Identifier for a preexisting filter. See the Filter API for details.	False
max_num	Integer	Max number of records that can be returned. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is	False

Name	Type	Description	Required
		0.	
fields	String	Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters (applicable to link and collection fields). The fields id and date_modified will always be returned. For more details on additional field parameters, see Relate API and Collection API.	False
order_by	String	See Filter API.	False
deleted	Boolean	Boolean to show deleted records in the result set.	False

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Response

```
{
  "next_offset": -1,
```

```
  "records": [
    {
      "id": "d509b3da-29f6-7b23-7cce-56fab1a6335b",
      "name": "Accounts Dashboard",
      "date_modified": "2016-03-29T09:46:00-07:00",
      "view_name": "records",
      "_acl": {
        "fields": {}
      },
      "view": "records",
      "_module": "Dashboards"
    },
    {
      "id": "cc4982cd-0bdf-bad4-d079-56fab4bdddf1",
      "name": "Contacts Dashboard",
      "date_modified": "2016-03-29T09:57:58-07:00",
      "view_name": "records",
      "_acl": {
        "fields": {}
      },
      "view": "records",
      "_module": "Dashboards"
    }
  ]
}
```


Change Log

Version	Change
v11	No longer supported. Please use GET /Dashboards instead.
v10	Added /Dashboards/<module> GET endpoint.

/Dashboards/ POST

Create a new dashboard.

Notice

This endpoint is not supported in v11. If you continue to use v10 of the API,

your REST calls should behave as below. But, it is advisable to upgrade to v11.
Migrating to v11:
For more information on behavior in v11, refer to POST /<module>.
Recommended Replacement: POST /Dashboards
The module name is no longer part of the path. To specify a module name, include it in the Request Body. For example, to create an Accounts module dashboard, use **v11:** POST /Dashboards with the following in the Request Body:
{"dashboard_module":"Accounts"}

Request Arguments

Name	Type	Description	Required
<record field>	<record field type>	The name value list of fields to populate.	True

Request

```
{
    "name": "New Dashboard for Accounts Module",
    "dashboard_module": "Accounts",
    "view_name": "records",
    "metadata": {
        "components": [{
            "rows": [[{
                "view": {
                    "type": "dashablelist",
                    "label": "TPL_DASHLET_MY_MODULE",
                    "display_columns": ["name","billing_address_city"]
                },
                "context": {"module":"Accounts"},
                "width": 12
            }]],
            "width": 12
        }]
    }
}
```

Response Arguments

Name	Type	Description
<record field>	<record field type>	Returns the fields for the

Name	Type	Description
		newly created dashboard.

Response


```
{
  "id": "8f9cd3d1-ce5d-e8d4-12d8-56cf674d2600",
  "name": "New Dashboard for Accounts Module",
  "date_entered": "2016-02-25T12:44:27-08:00",
  "date_modified": "2016-02-25T12:44:27-08:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
  "description": "",
  "deleted": false,
  "dashboard_module": "Accounts",
  "view_name": "records",
  "metadata": {
    "components": [
      {
        "rows": [
          [
            {
              "view": {
                "type": "dashablelist",
                "label": "TPL_DASHLET_MY_MODULE",
                "display_columns": [
                  "name",
                  "billing_address_city"
                ]
              },
              "context": {
                "module": "Accounts"
              },
              "width": 12
            }
          ]
        ],
        "width": 12
      }
    ]
  },
  "following": "",
  "my_favorite": false,
```

```
  "tag": [],
  "assigned_user_id": "1",
  "assigned_user_name": "Administrator",
  "_acl": {
    "fields": {}
  },
  "view": "records",
  "_module": "Dashboards"
}
```


Change Log

Version	Change
v11	No longer supported. Please use POST /Dashboards instead.
v10	Added /Dashboards/<module> POST endpoint.

/Dashboards/Activities GET

Overview

List current user's filtered Activity Stream dashboards.

Summary

This endpoint will return a set of Activity Stream dashboards filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long.

Notice

The behavior of this endpoint has changed in v11. If you continue to use v10 of the API, your REST calls should behave as below. But, it is advisable to upgrade to v11.
Migrating to v11:
For more information on behavior in v11, refer to GET /<module>.

Recommended Replacement: GET /Dashboards

The module name is no longer part of the path. To specify a module name, include it as a filter parameter, use
v11: GET /Dashboards?filter[0][dashboard_module]=Activities

name, id. etc. are no longer specified as default fields. To retain the default fields, use
v11: GET /Dashboards?filter[0][dashboard_module]=Activities&fields=id,name,view_name

Request Arguments

Name	Type	Description	Required
filter	String	See Filter API.	False
filter_id	String	Identifier for a preexisting filter. See the Filter API for details.	False
max_num	Integer	Max number of records that can be returned. Default is 20, unless overruled by your Sugar configuration.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
fields	String	Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters (applicable to link and collection fields). The fields id and date_modified	False

Name	Type	Description	Required
		will always be returned. For more details on additional field parameters, see Relate API and Collection API.	
order_by	String	See Filter API.	False
deleted	Boolean	Boolean to show deleted records in the result set.	False

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Response


```
{
  "next_offset": -1,
  "records": [
    {
      "id": "d509b3da-29f6-7b23-7cce-56fab1a6335b",
      "name": "Activity Stream Dashboard",
      "date_modified": "2016-03-29T09:46:00-07:00",
      "view_name": "activities",
      "_acl": {
        "fields": {}
      },
      "view": "activities",
      "_module": "Dashboards"
    },
    {
      "id": "cc4982cd-0bdf-bad4-d079-56fab4bdddf1",
      "name": "Activity Stream Dashboard 2",
```

```
      "date_modified": "2016-03-29T09:57:58-07:00",
      "view_name": "activities",
      "_acl": {
        "fields": {}
      },
      "view": "activities",
      "_module": "Dashboards"
    }
  ]
}
```


Change Log

Version	Change
v11	No longer supported. Please use GET /Dashboards instead.
v10	Added /Dashboards/Activities GET endpoint.

/DataArchiver/:id/run POST

Overview

[ADMIN] Performs the archiving process.

Summary

Performs the archiving process for the given archive definition. This endpoint is only available to administrators.

Request Arguments

This endpoint does not accept any request arguments.

Response

This API returns the ID of the created ArchiveRun bean.

Change Log

Version	Change
v11_11	Added /DataArchiver/:id/run POST endpoint.

/Documents/:record/file/:field POST

Overview

Attaches a file to a field on a record.

Request Arguments

Name	Type	Description	Required
format	String	The data format. Currently accepts 'sugar-html-json'.	True
delete_if_fails	Boolean	Indicates whether the API is to mark related record deleted if the file upload fails.	False
oauth_token	String	The oauth_token value.	False - Required if only if delete_if_fails is true.
<attachment field>	String	The field and file to populate. Example: {"":"@\/path\/to\/ExampleDocument.txt"}	True

Request

```
{
   "format":"sugar-html-json",
   "delete_if_fails":true,
   "oauth_token":"43b6b327-cc70-c301-3299-512ffb99ad97",
   "<attachment field>":"@\/path\/to\/ExampleDocument.txt"
}
```

Response Arguments

Name	Type	Description
content-type	String	The files content type.
content-length	Integer	The files content length.
name	String	The files name.
uri	String	The URI of the file.

Response


```
{
   "content-type":"text\/plain",
   "content-length":16,
   "name":"ExampleDocument.txt",
   "uri":"http:\/\/sugarcrm\/rest\/v10\/Notes\/ca66c92f-5a8b-28b4-4fc8
-512d099b790b\/file\/<attachment field>?format=sugar-html-json&delete_
if_fails=1&oauth_token=6ec91cf3-1f97-25b9-e0b1-512f8971b2d4"
}
```


Change Log

Version	Change
v10	Added /<module>/:record/file/:field POST endpoint.

/Documents/:record/file/:field PUT

Overview

This endpoint takes a file or image and saves it to a record that already contains an attachment in the specified field. The PUT method is very similar to the POST method but differs slightly in how the request is constructed. PUT requests should send the file data as the body of the request. Optionally, a filename query parameter can be sent with the request to assign a name. Additionally, the PUT method can accept base64 encoded file data which will be decoded by the endpoint if the content_transfer_encoding parameter is set to 'base64'.

NOTE: In lieu of the content_transfer_encoding parameter, a request header of X-

Content-Transfer-Encoding can also be sent with a value of 'base64'. In the event both the content transfer encoding header and request parameter are sent, the header will be used.

Request Arguments

Name	Type	Description	Required
filename	String	Filename to save the document as	False
content_transfer_encoding	String	When set to 'base64', indicates the file contents are base64 encoded and will result in the file contents being base64 decoded before being saved	False

Request

PUT /rest/v10/Notes/abcd-1234/file/field/ HTTP/1.1 Host: localhost Connection: keep-alive Content-Length: 23456 Content-Type: application/document-doc ...This is where the bin data would be

Response Arguments

Name	Type	Description
field	Array	The fields containing file properties.
field.content-type	String	The files content type.
field.content-length	Integer	The files content length.
field.name	String	The files name.
field.width	Integer	The width of the image.
field.height	Integer	The height of the image.
field.uri	String	The URI of the file.

Response

```
{
   "picture":{
      "content-type":"image\/png",
```

```
      "content-length":72512,
      "name":"1a7b8f5c-b11c-0094-c8d8-512e9daaa983",
      "width":933,
      "height":519,
      "uri":"http:\/\/sugarcrm\/rest\/v10\/Contacts\/fa300a0e-0ad1-b32
2-9601-512d0983c19a\/file\/picture"
   }
    "attachment":{
        "content-type":"application/document-doc",
        "content-length":"873921",
        "name":"myFile.doc",
        "uri": "http:\/\/sugarcrm\/rest\/v10\/Contacts\/f2f9aa4d-99a8-
e86e-f4d5-512d0986effa\/file\/attachment"
    }
}
```

Change Log

Version	Change
v10	Added /<module>/:record/file/:field PUT endpoint.

/EmailAddresses POST

Overview

Create a new email address.

Summary

In the event that the email address already exists, that record will be returned without making any changes.

Request Arguments

Name	Type	Description	Required
email_address	String	The email address.	True
invalid_email	Boolean	**true** if the email address is invalid. **false** is the default.	False

Name	Type	Description	Required
opt_out	Boolean	**true** if the email address is opted out. **false** is the default.	False

Request

```
{
  "email_address": "eharmon@example.com",
  "invalid_email": false,
  "opt_out": false
}
```

Response Arguments

Name	Type	Description
<record field>	<record field type>	Returns the fields for the newly created record.

Response

```
{
  "id": "a028341c-ba92-9343-55e7-56cf5beda121",
  "date_created": "2016-02-25T11:53:07-08:00",
  "date_modified": "2016-02-25T11:53:07-08:00",
  "deleted": false,
  "email_address": "eharmon@example.com",
  "email_address_caps": "EHARMON@EXAMPLE.COM",
  "invalid_email": false,
  "opt_out": false,
  "_acl": {
    "fields": {}
  },
  "_module": "EmailAddresses"
}
```

Change Log

Version	Change
v10	Added /EmailAddresses POST endpoint.

/EmailAddresses/:record PUT

Overview

Update an existing email address.

Summary

The **email_address** parameter is not supported as email addresses are immutable.

Request Arguments

Name	Type	Description	Required
invalid_email	Boolean	**true** if the email address is invalid.	False
opt_out	Boolean	**true** if the email address is opted out.	False

Request

```
{
  "opt_out": true
}
```

Response Arguments

Name	Type	Description
<record field>	<record field type>	Returns the fields for the updated record.

Response

```
{
  "id": "a028341c-ba92-9343-55e7-56cf5beda121",
```

```
  "date_created": "2016-02-25T11:53:07-08:00",
  "date_modified": "2016-02-25T11:53:07-08:00",
  "deleted": false,
  "email_address": "eharmon@example.com",
  "email_address_caps": "EHARMON@EXAMPLE.COM",
  "invalid_email": false,
  "opt_out": true,
  "_acl": {
    "fields": {}
  },
  "_module": "EmailAddresses"
}
```

Change Log

Version	Change
v10	Added /EmailAddresses/:record PUT endpoint.

/Emails GET

Overview

Lists filtered emails.

Summary

Adds additional operators to Filter API that are specific to Emails. A special macro, **$current_user_id**, is a convenience that allows for finding emails involving the current user.

Filter By Sender

This will return all emails sent by the current user, by the contact whose ID is fa300a0e-0ad1-b322-9601-512d0983c19a, using the email address sam@example.com, which is referenced by the ID b0701501-1fab-8ae7-3942-540da93f5017, or by the lead whose ID is 73b1087e-4bb6-11e7-acaa-3c15c2d582c6 using the email address wally@example.com, which is referenced by the ID b651d834-4bb6-11e7-bfcf-3c15c2d582c6.

```
{
```

```
  "filter": [{
    "$from": [{
      "parent_type": "Users",
      "parent_id": "$current_user_id"
    }, {
      "parent_type": "Contacts",
      "parent_id": "fa300a0e-0ad1-b322-9601-512d0983c19a"
    }, {
      "email_address_id": "b0701501-1fab-8ae7-3942-540da93f5017"
    }, {
      "parent_type": "Leads",
      "parent_id": "73b1087e-4bb6-11e7-acaa-3c15c2d582c6",
      "email_address_id": "b651d834-4bb6-11e7-bfcf-3c15c2d582c6"
    }]
  }]
}
```

Equivalent to:

```
{
  "filter": [{
    "from_collection": {
      "$in": [{
        "parent_type": "Users",
        "parent_id": "$current_user_id"
      }, {
        "parent_type": "Contacts",
        "parent_id": "fa300a0e-0ad1-b322-9601-512d0983c19a"
      }, {
        "email_address_id": "b0701501-1fab-8ae7-3942-540da93f5017"
      }, {
        "parent_type": "Leads",
        "parent_id": "73b1087e-4bb6-11e7-acaa-3c15c2d582c6",
        "email_address_id": "b651d834-4bb6-11e7-bfcf-3c15c2d582c6"
      }]
    }
  }]
}
```

Filter By Recipients

This would return all archived emails received by the current user.

```
{
  "filter": [{
    "$or": [{
      "$to": [{
        "parent_type": "Users",
        "parent_id": "$current_user_id"
      }],
      "$cc": [{
        "parent_type": "Users",
        "parent_id": "$current_user_id"
      }],
      "$bcc": [{
        "parent_type": "Users",
        "parent_id": "$current_user_id"
      }]
    }]
  }, {
    "state": {
      "$in": [
        "Archived"
      ]
    }
  }]
}
```

Equivalent to:

```
{
  "filter": [{
    "$or": [{
      "to_collection": {
        "$in": [{
          "parent_type": "Users",
          "parent_id": "$current_user_id"
        }]
      },
      "cc_collection": {
        "$in": [{
          "parent_type": "Users",
          "parent_id": "$current_user_id"
        }]
      },
      "bcc_collection": {
        "$in": [{
          "parent_type": "Users",
```

```
          "parent_id": "$current_user_id"
        }]
      }
    }]
  }, {
    "state": {
      "$in": [
        "Archived"
      ]
    }
  }]
}
```

Filter By Sender and Recipients

This would return all archived emails sent by a contact to the current user.

```
{
  "filter": [{
    "$from": [{
      "parent_type": "Contacts",
      "parent_id": "fa300a0e-0ad1-b322-9601-512d0983c19a"
    }]
  }, {
    "$or": [{
      "$to": [{
        "parent_type": "Users",
        "parent_id": "$current_user_id"
      }],
      "$cc": [{
        "parent_type": "Users",
        "parent_id": "$current_user_id"
      }],
      "$bcc": [{
        "parent_type": "Users",
        "parent_id": "$current_user_id"
      }]
    }]
  }, {
    "state": {
      "$in": [
        "Archived"
      ]
```

```
    }
  }]
}
```

Equivalent to:

```
{
  "filter": [{
    "from_collection": {
      "$in": [{
        "parent_type": "Contacts",
        "parent_id": "fa300a0e-0ad1-b322-9601-512d0983c19a"
      }]
    }
  }, {
    "$or": [{
      "to_collection": {
        "$in": [{
          "parent_type": "Users",
          "parent_id": "$current_user_id"
        }]
      },
      "cc_collection": {
        "$in": [{
          "parent_type": "Users",
          "parent_id": "$current_user_id"
        }]
      },
      "bcc_collection": {
        "$in": [{
          "parent_type": "Users",
          "parent_id": "$current_user_id"
        }]
      }
    }]
  }, {
    "state": {
      "$in": [
        "Archived"
      ]
    }
  }]
}
```

Change Log

Version	Change
v10	Added the $from, $to, $cc, and $bcc operators to /Emails/filter GET endpoint.

/Emails POST

Overview

Create a new Emails record.

Summary

Used for creating an archived email, creating a draft to send later, or creating and sending an email.

Creating an Archived Email

Request Arguments

Name	Type	Description	Required
state	String	Use "Archived" to create an archived email. "Archived" is the default value if this argument is not provided.	False
name	String	The email's subject.	False
description_html	String	The email's HTML body.	False
description	String	The email's plain text body. This field is not necessary unless the text body is different from the HTML body.	False
raw_source	String	The complete raw source of the email showing the headers and	False

Name	Type	Description	Required
		content in one document.	
date_sent	String	The date that the email was sent/received.	False
message_id	String	The email's unique identifier.	False
message_uid	String	Only use this field for importing emails downloaded from an IMAP server, as this is the email's IMAP UID.	False
assigned_user_id	String	The assigned user's ID.	False
parent_type	String	The parent record's module.	False
parent_id	String	The parent record's ID.	False
team_name	List	List of teams that can access the email.	False
from	Object	The email's sender. Existing EmailParticipants records cannot be used; only the **create** action is supported. Metadata about the sender is required to link the record. • **parent_type** and **parent_id** can be Accounts, Contacts,	True

Name	Type	Description	Required
		Employees, Leads, Prospects, or Users, out out of the box. These fields are not necessary if only an email address is known for the sender. • **email_address_id** is the ID of the email address the sender used to send the email. To get the ID of the email address, first make a request to /EmailAddresses POST. If this argument is not provided, then the primary email address of the parent record is used.	
to	Object	The email's recipients found in the TO field.	False

Name	Type	Description	Required
		Existing EmailParticipants records cannot be used; only the **create** action is supported. Metadata about each recipient is required to link the records. • **parent_type** and **parent_id** can be Accounts, Contacts, Employees, Leads, Prospects, or Users, out out of the box. These fields are not necessary if only an email address is known for the recipient. • **email_address_id** is the ID of the email address the recipient used in the email. To get the ID of the email	

Name	Type	Description	Required
		address, first make a request to /EmailAddresses POST. If this argument is not provided, then the primary email address of the parent record is used.	
cc	Object	The email's recipients found in the CC field. Existing EmailParticipants records cannot be used; only the **create** action is supported. Metadata about each recipient is required to link the records. • **parent_type** and **parent_id** can be Accounts, Contacts, Employees, Leads, Prospects, or Users, out out of the box. These fields	False

Name	Type	Description	Required
		are not necessary if only an email address is known for the recipient. • **email_address_id** is the ID of the email address the recipient used in the email. To get the ID of the email address, first make a request to /EmailAddresses POST. If this argument is not provided, then the primary email address of the parent record is used.	
bcc	Object	The email's recipients found in the BCC field. Existing EmailParticipants records cannot be used; only the **create** action is supported.	False

Name	Type	Description	Required
		Metadata about each recipient is required to link the records. • **parent_type** and **parent_id** can be Accounts, Contacts, Employees, Leads, Prospects, or Users, out out of the box. These fields are not necessary if only an email address is known for the recipient. • **email_address_id** is the ID of the email address the recipient used in the email. To get the ID of the email address, first make a request to /EmailAddresses POST. If this	

Name	Type	Description	Required
		argument is not provided, then the primary email address of the parent record is used.	
attachments	Object	The email's attachments. Existing Notes records cannot be used as email attachments; only the **create** action is supported. Metadata about each attachment is required to link the records. • **filename_guid** is the temporary file ID for an uploaded file to attach as a Notes record. • **upload_id** is the ID of an existing file that the Notes record should reference as the attachment. This allows	False

Name	Type	Description	Required
		files to be shared by multiple Notes records, to preserve space. • When using **upload_id**, **file_source** should be provided to indicate from where the file originated. The value should be the module of the record that shares its ID with the file. So when an attachment comes from an email template, **file_source** should be **EmailTemplates**. And when an attachment comes from a document, **file_source** should be **Document Revisions**.	

Request

```
{
  "state": "Archived",
  "name": "Re: Discuss Proposal",
  "description": "Now is a good time",
  "description_html": "<p>Now is a good time</p>",
  "date_sent": "2012-02-17T06:53:00-08:00",
  "message_id": "<d9c165d0-8863-ba61-dc85-51ed8016c476@example.com>",
  "assigned_user_id": "9c61c46a-a7c5-df71-481c-51d48232f820",
  "parent_type": "Leads",
  "parent_id": "f6a0611a-7d27-11e7-9d70-3c15c2d582c6",
  "team_name": [{
    "id": "1",
    "display_name": "Global",
    "name": "Global",
    "name_2": "",
    "primary": true,
    "selected": false
  }],
  "from": {
    "create": [{
      "parent_type": "Leads",
      "parent_id": "f6a0611a-7d27-11e7-9d70-3c15c2d582c6"
    }]
  },
  "to": {
    "create": [{
      "parent_type": "Users",
      "parent_id": "9c61c46a-a7c5-df71-481c-51d48232f820"
    }]
  },
  "cc": {
    "create": [{
      "parent_type": "Contacts",
      "parent_id": "79b9c194-7d79-11e7-8fc5-3c15c2d582c6"
    }, {
      "parent_type": "Users",
      "parent_id": "79c2e800-7d79-11e7-84af-3c15c2d582c6"
      "email_address_id": "79cc341e-7d79-11e7-8748-3c15c2d582c6"
    }]
  },
  "bcc": {
    "create": [{
      "email_address_id": "ac804842-7d78-11e7-a809-3c15c2d582c6"
    }]
  },
```

```
  "attachments": {
    "create": [{
      "filename_guid": "afe9702e-53a3-0efb-6bbe-56c3580885ef",
      "name": "Quote.pdf",
      "filename": "Quote.pdf"
    }]
  }
}
```

Response

```
{
  "id": "a028341c-ba92-9343-55e7-56cf5beda121",
  "date_entered": "2016-02-25T11:53:07-08:00",
  "created_by": "9c61c46a-a7c5-df71-481c-51d48232f820",
  "created_by_name": "Sarah Smith",
  "date_modified": "2016-02-25T11:53:07-08:00",
  "modified_by_name": "Sarah Smith",
  "modified_user_id": "9c61c46a-a7c5-df71-481c-51d48232f820",
  "deleted": false,
  "assigned_user_id": "9c61c46a-a7c5-df71-481c-51d48232f820",
  "assigned_user_name": "Sarah Smith",
  "state": "Archived",
  "name": "Re: Discuss Proposal",
  "description": "Now is a good time",
  "description_html": "<p>Now is a good time</p>",
  "date_sent": "2012-02-17T06:53:00-08:00",
  "message_id": "",
  "message_uid": "",
  "reply_to_status": false,
  "total_attachments": 1,
  "parent_name": "Tom Davis",
  "parent_type": "Leads",
  "parent_id": "f6a0611a-7d27-11e7-9d70-3c15c2d582c6"
  "team_count": "",
  "team_name": [{
    "id": 1,
    "name": "Global",
    "name_2": "",
    "primary": true,
    "selected": false
  }],
  "my_favorite": false,
```

```
  "_acl": {
    "fields": {}
  },
  "_module": "Emails"
}
```

Creating a Draft

Request Arguments

Name	Type	Description	Required
state	String	Use "Draft" to save a draft.	True
outbound_email_id	String	The ID of the chosen SMTP configuration for sending an email. Only use this field when saving a draft or sending an email. The user's default SMTP configuration is used if one hasn't been chosen prior to sending.	False
name	String	The email's subject.	False
description_html	String	The email's HTML body.	False
description	String	The email's plain text body. This field is not necessary unless the text body is different from the HTML body.	False
reply_to_id	String	Only use this field when saving a draft or sending an email that is a reply to	False

Name	Type	Description	Required
		another email. This is the ID of that other email.	
parent_type	String	The parent record's module.	False
parent_id	String	The parent record's ID.	False
to	Object	The email's recipients found in the TO field. Existing EmailParticipants records cannot be used; only the **create** action is supported. Metadata about each recipient is required to link the records. • **parent_type** and **parent_id** can be Accounts, Contacts, Employees, Leads, Prospects, or Users, out out of the box. These fields are not necessary if only an email address is known for the recipient.	False

Name	Type	Description	Required
		• **email_address_id** is the ID of the email address the recipient used in the email. To get the ID of the email address, first make a request to /EmailAddresses POST. If this argument is not provided, then the primary email address of the parent record is used.	
cc	Object	The email's recipients found in the CC field. Existing EmailParticipants records cannot be used; only the **create** action is supported. Metadata about each recipient is required to link the records. • **parent_type** and	False

Name	Type	Description	Required
		parent_id can be Accounts, Contacts, Employees, Leads, Prospects, or Users, out out of the box. These fields are not necessary if only an email address is known for the recipient. • **email_address_id** is the ID of the email address the recipient used in the email. To get the ID of the email address, first make a request to /EmailAddresses POST. If this argument is not provided, then the primary email address of the parent	

Name	Type	Description	Required
		record is used.	
bcc	Object	The email's recipients found in the BCC field. Existing EmailParticipants records cannot be used; only the **create** action is supported. Metadata about each recipient is required to link the records. • **parent_type** and **parent_id** can be Accounts, Contacts, Employees, Leads, Prospects, or Users, out out of the box. These fields are not necessary if only an email address is known for the recipient. • **email_address_id** is the ID of the email address the	False

Name	Type	Description	Required
		recipient used in the email. To get the ID of the email address, first make a request to /EmailAddresses POST. If this argument is not provided, then the primary email address of the parent record is used.	
attachments	Object	The email's attachments. Existing Notes records cannot be used as email attachments; only the **create** action is supported. Metadata about each attachment is required to link the records. • **filename_guid** is the temporary file ID for an uploaded file to attach as a Notes record.	False

Name	Type	Description	Required
		• **upload_id** is the ID of an existing file that the Notes record should reference as the attachment. This allows files to be shared by multiple Notes records, to preserve space. • When using **upload_id**, **file_source** should be provided to indicate from where the file originated. The value should be the module of the record that shares its ID with the file. So when an attachment comes from an email template, **file_source** should be **EmailTem**	

Name	Type	Description	Required
		plates. And when an attachment comes from a document, **file_source** should be **Document Revisions**.	

Request

```
{
  "state": "Draft",
  "outbound_email_id": "b80adf1a-7d78-11e7-855a-3c15c2d582c6",
  "name": "Re: Discuss Proposal",
  "description_html": "<p>Calling you now.</p>",
  "reply_to_id": "a028341c-ba92-9343-55e7-56cf5beda121",
  "parent_type": "Leads",
  "parent_id": "f6a0611a-7d27-11e7-9d70-3c15c2d582c6",
  "to": {
    "create": [{
      "parent_type": "Leads",
      "parent_id": "f6a0611a-7d27-11e7-9d70-3c15c2d582c6"
    }]
  },
  "cc": {
    "create": [{
      "parent_type": "Contacts",
      "parent_id": "79b9c194-7d79-11e7-8fc5-3c15c2d582c6"
    }, {
      "parent_type": "Users",
      "parent_id": "79c2e800-7d79-11e7-84af-3c15c2d582c6"
      "email_address_id": "79cc341e-7d79-11e7-8748-3c15c2d582c6"
    }]
  },
  "bcc": {
    "create": [{
      "email_address_id": "ac804842-7d78-11e7-a809-3c15c2d582c6"
    }]
  },
  "attachments": {
    "create": [{
      "upload_id": "a028341c-ba92-9343-55e7-56cf5beda121",
      "name": "Contract.pdf",
```

```
      "filename": "Contract.pdf",
      "file_source": "DocumentRevisions"
    }, {
      "upload_id": "18a72782-4514-11e6-a5f7-3c15c2d582c6",
      "name": "survey.doc",
      "filename": "survey.doc",
      "file_source": "EmailTemplates"
    }]
  }
}
```


Response


```
{
  "id": "f3c85966-7d27-11e7-9e9e-3c15c2d582c6",
  "date_entered": "2016-02-25T11:53:07-08:00",
  "created_by": "9c61c46a-a7c5-df71-481c-51d48232f820",
  "created_by_name": "Sarah Smith",
  "date_modified": "2016-02-25T11:53:07-08:00",
  "modified_by_name": "Sarah Smith",
  "modified_user_id": "9c61c46a-a7c5-df71-481c-51d48232f820",
  "deleted": false,
  "assigned_user_id": "9c61c46a-a7c5-df71-481c-51d48232f820",
  "assigned_user_name": "Sarah Smith",
  "state": "Draft",
  "outbound_email_id": "b80adf1a-7d78-11e7-855a-3c15c2d582c6",
  "name": "Re: Discuss Proposal",
  "description": "Now is a good time",
  "description_html": "<p>Now is a good time</p>",
  "date_sent": "2012-02-17T11:53:07-08:00",
  "message_id": "",
  "message_uid": "",
  "reply_to_id": "a028341c-ba92-9343-55e7-56cf5beda121",
  "reply_to_status": false,
  "total_attachments": 2,
  "parent_name": "Tom Davis",
  "parent_type": "Leads",
  "parent_id": "f6a0611a-7d27-11e7-9d70-3c15c2d582c6"
  "team_count": "",
  "team_name": [{
    "id": 1,
    "name": "Global",
    "name_2": "",
```

```
    "primary": true,
    "selected": false
  }],
  "my_favorite": false,
  "_acl": {
    "fields": {}
  },
  "_module": "Emails"
}
```

Sending an Email

Request Arguments

Name	Type	Description	Required
state	String	Use "Ready" to send an email.	True
outbound_email_id	String	The ID of the chosen SMTP configuration for sending an email. Only use this field when saving a draft or sending an email. The user's default SMTP configuration is used if one hasn't been chosen prior to sending.	False
name	String	The email's subject.	False
description_html	String	The email's HTML body.	False
description	String	The email's plain text body. This field is not necessary unless the text body is different from the HTML body.	False

Name	Type	Description	Required
reply_to_id	String	Only use this field when saving a draft or sending an email that is a reply to another email. This is the ID of that other email.	False
parent_type	String	The parent record's module.	False
parent_id	String	The parent record's ID.	False
to	Object	The email's recipients found in the TO field. Existing EmailParticipants records cannot be used; only the **create** action is supported. Metadata about each recipient is required to link the records. • **parent_type** and **parent_id** can be Accounts, Contacts, Employees, Leads, Prospects, or Users, out out of the box. These fields are not necessary if only an email	False

Name	Type	Description	Required
		address is known for the recipient. • **email_address_id** is the ID of the email address the recipient used in the email. To get the ID of the email address, first make a request to /EmailAddresses POST. If this argument is not provided, then the primary email address of the parent record is used.	
cc	Object	The email's recipients found in the CC field. Existing EmailParticipants records cannot be used; only the **create** action is supported. Metadata about each recipient is required to link the records.	False

Name	Type	Description	Required
		• **parent_type** and **parent_id** can be Accounts, Contacts, Employees, Leads, Prospects, or Users, out out of the box. These fields are not necessary if only an email address is known for the recipient. • **email_address_id** is the ID of the email address the recipient used in the email. To get the ID of the email address, first make a request to /EmailAddresses POST. If this argument is not provided, then the primary	

Name	Type	Description	Required
		email address of the parent record is used.	
bcc	Object	The email's recipients found in the BCC field. Existing EmailParticipants records cannot be used; only the **create** action is supported. Metadata about each recipient is required to link the records. • **parent_type** and **parent_id** can be Accounts, Contacts, Employees, Leads, Prospects, or Users, out out of the box. These fields are not necessary if only an email address is known for the recipient. • **email_address_id** is	False

Name	Type	Description	Required
		the ID of the email address the recipient used in the email. To get the ID of the email address, first make a request to /EmailAddresses POST. If this argument is not provided, then the primary email address of the parent record is used.	
attachments	Object	The email's attachments. Existing Notes records cannot be used as email attachments; only the **create** action is supported. Metadata about each attachment is required to link the records. • **filename_guid** is the temporary file ID for an uploaded file to	False

Name	Type	Description	Required
		attach as a Notes record. • **upload_id** is the ID of an existing file that the Notes record should reference as the attachment. This allows files to be shared by multiple Notes records, to preserve space. • When using **upload_id**, **file_source** should be provided to indicate from where the file originated. The value should be the module of the record that shares its ID with the file. So when an attachment comes from an email template,	

Name	Type	Description	Required
		file_source should be **EmailTemplates**. And when an attachment comes from a document, **file_source** should be **Document Revisions**.	

Request

```
{
  "state": "Ready",
  "outbound_email_id": "b80adf1a-7d78-11e7-855a-3c15c2d582c6",
  "name": "Discuss Proposal",
  "description_html": "<p>When is a good time for us to chat?</p>",
  "parent_type": "Leads",
  "parent_id": "f6a0611a-7d27-11e7-9d70-3c15c2d582c6",
  "to": {
    "create": [{
      "parent_type": "Leads",
      "parent_id": "f6a0611a-7d27-11e7-9d70-3c15c2d582c6"
    }]
  },
  "cc": {
    "create": [{
      "parent_type": "Users",
      "parent_id": "79c2e800-7d79-11e7-84af-3c15c2d582c6"
      "email_address_id": "79cc341e-7d79-11e7-8748-3c15c2d582c6"
    }]
  },
  "attachments": {
    "create": [{
      "upload_id": "a028341c-ba92-9343-55e7-56cf5beda121",
      "name": "Contract.pdf",
      "filename": "Contract.pdf",
      "file_source": "DocumentRevisions"
    }, {
      "upload_id": "18a72782-4514-11e6-a5f7-3c15c2d582c6",
      "name": "survey.doc",
      "filename": "survey.doc",
```

```
      "file_source": "EmailTemplates"
    }]
  }
}
```

Response

```
{
  "id": "a7795664-7def-11e7-8add-3c15c2d582c6",
  "date_entered": "2016-02-25T08:53:07-08:00",
  "created_by": "9c61c46a-a7c5-df71-481c-51d48232f820",
  "created_by_name": "Sarah Smith",
  "date_modified": "2016-02-25T08:53:07-08:00",
  "modified_by_name": "Sarah Smith",
  "modified_user_id": "9c61c46a-a7c5-df71-481c-51d48232f820",
  "deleted": false,
  "assigned_user_id": "9c61c46a-a7c5-df71-481c-51d48232f820",
  "assigned_user_name": "Sarah Smith",
  "state": "Archived",
  "outbound_email_id": "b80adf1a-7d78-11e7-855a-3c15c2d582c6",
  "name": "Discuss Proposal",
  "description": "When is a good time for us to chat?",
  "description_html": "<p>When is a good time for us to chat?</p>",
  "date_sent": "2012-02-17T08:53:07-08:00",
  "message_id": "<a7795664-7def-11e7-8add-3c15c2d582c6@example.com>",
  "message_uid": "",
  "reply_to_id": "",
  "reply_to_status": false,
  "total_attachments": 2,
  "parent_name": "Tom Davis",
  "parent_type": "Leads",
  "parent_id": "f6a0611a-7d27-11e7-9d70-3c15c2d582c6"
  "team_count": "",
  "team_name": [{
    "id": 1,
    "name": "Global",
    "name_2": "",
    "primary": true,
    "selected": false
  }],
  "my_favorite": false,
  "_acl": {
    "fields": {}
```

```
  },
  "_module": "Emails"
}
```

Change Log

Version	Change
v10	Added /Emails POST endpoint.

/Emails/:record GET

Overview

Retrieves an Emails record.

Fields

Name	Type	Description
<record field>	<record field type>	Returns the fields for the selected record.
from_collection	List	The email's sender.
to_collection	List	The email's recipients found in the TO field.
cc_collection	List	The email's recipients found in the CC field.
bcc_collection	List	The email's recipients found in the BCC field.
attachments_collection	List	The email's attachments.

Request

```
GET /Emails/a028341c-ba92-9343-55e7-56cf5beda121?fields=name,state,
{"name":"from_collection","fields":["email_address","email_address_id"
,
"parent_type","parent_id","parent_name"]},
{"name":"to_collection","fields":["email_address","email_address_id",
"parent_type","parent_id","parent_name"]},
{"name":"cc_collection","fields":["email_address","email_address_id",
"parent_type","parent_id","parent_name"]},
{"name":"bcc_collection","fields":["email_address","email_address_id",
```

```
"parent_type","parent_id","parent_name"]},
{"name":"attachments_collection","fields":["name","filename","file_siz
e",
"file_source","file_mime_type","file_ext","upload_id"]}
```

Response


```
{
  "id": "a028341c-ba92-9343-55e7-56cf5beda121",
  "state": "Archived",
  "name": "Re: Discuss Proposal",
  "from_collection": {
    "next_offset": {
      "from": -1
    },
    "records": [{
      "id": "ec113d14-7d27-11e7-a951-3c15c2d582c6",
      "email_address": "tdavis@example.com",
      "email_address_id": "ec05a896-7d27-11e7-a51e-3c15c2d582c6",
      "parent_type": "Leads",
      "parent_id": "f6a0611a-7d27-11e7-9d70-3c15c2d582c6",
      "parent_name": "Tom Davis",
      "_acl": {
        "fields": {}
      },
      "_link": "from",
      "_module": "EmailParticipants"
    }]
  },
  "to_collection": {
    "next_offset": {
      "to": -1
    },
    "records": [{
      "id": "ec0f1278-7d27-11e7-8d1f-3c15c2d582c6",
      "email_address": "ssmith@example.com",
      "email_address_id": "ec0fb516-7d27-11e7-b7ad-3c15c2d582c6",
      "parent_type": "Users",
      "parent_id": "9c61c46a-a7c5-df71-481c-51d48232f820",
      "parent_name": "Sarah Smith",
      "_acl": {
        "fields": {}
      },
```

```
      "_link": "to",
      "_module": "EmailParticipants"
    }]
  },
  "cc_collection": {
    "next_offset": {
      "cc": -1
    },
    "records": [{
      "id": "eafa7e9a-7d27-11e7-aa2b-3c15c2d582c6",
      "email_address": "broland@example.com",
      "email_address_id": "ea1cec7e-7d27-11e7-9845-3c15c2d582c6",
      "parent_type": "Users",
      "parent_id": "e087e79a-7d27-11e7-b02c-3c15c2d582c6",
      "parent_name": "Brian Roland",
      "_acl": {
        "fields": {}
      },
      "_link": "cc",
      "_module": "EmailParticipants"
    }, {
      "id": "df5eb204-7d27-11e7-b363-3c15c2d582c6",
      "email_address": "twallace@example.com",
      "email_address_id": "ddf1d9e6-7d27-11e7-bb17-3c15c2d582c6",
      "parent_type": "",
      "parent_id": "",
      "parent_name": "",
      "_acl": {
        "fields": {}
      },
      "_link": "cc",
      "_module": "EmailParticipants"
    }]
  },
  "bcc_collection": {
    "next_offset": {
      "bcc": -1
    },
    "records": []
  },
  "attachments_collection": {
    "next_offset": {
      "attachments": -1
    },
    "records": [{
      "id": "afe9702e-53a3-0efb-6bbe-56c3580885ef",
```

```
      "name": "Quote.pdf",
      "filename": "Quote.pdf",
      "upload_id": "",
      "file_mime_type": "application/pdf",
      "file_size": 158589,
      "file_source": "DocumentRevisions",
      "file_ext": "pdf"
      "_acl": {
        "fields": {}
      },
      "_link": "attachments",
      "_module": "Notes"
    }]
  }
  "_acl": {
    "fields": [
    ]
  },
  "_module": "Emails"
}
```

Change Log

Version	Change
v10	Added /Emails/:record GET endpoint.

/Emails/:record PUT

Overview

Update an existing Emails record.

Summary

Used for updating an archived email, updating a draft to send later, or sending a draft.

Updating an Archived Email

Request Arguments

Name	Type	Description	Required
message_uid	String	Only use this field for updating emails imported from an IMAP server, as this is the email's IMAP UID.	False
assigned_user_id	String	The assigned user's ID.	False
parent_type	String	The parent record's module.	False
parent_id	String	The parent record's ID.	False
team_name	List	List of teams that can access the email.	False

Request

```
{
  "assigned_user_id": "9c61c46a-a7c5-df71-481c-51d48232f820",
  "parent_type": "Leads",
  "parent_id": "f6a0611a-7d27-11e7-9d70-3c15c2d582c6",
  "team_name": [{
    "id": "1",
    "display_name": "Global",
    "name": "Global",
    "name_2": "",
    "primary": true,
    "selected": false
  }]
}
```

Response

```
{
  "id": "a028341c-ba92-9343-55e7-56cf5beda121",
  "date_entered": "2016-02-25T11:53:07-08:00",
  "created_by": "9c61c46a-a7c5-df71-481c-51d48232f820",
  "created_by_name": "Sarah Smith",
  "date_modified": "2016-02-25T11:53:07-08:00",
  "modified_by_name": "Sarah Smith",
```

```
  "modified_user_id": "9c61c46a-a7c5-df71-481c-51d48232f820",
  "deleted": false,
  "assigned_user_id": "9c61c46a-a7c5-df71-481c-51d48232f820",
  "assigned_user_name": "Sarah Smith",
  "state": "Archived",
  "name": "Re: Discuss Proposal",
  "description": "Now is a good time",
  "description_html": "<p>Now is a good time</p>",
  "date_sent": "2012-02-17T06:53:00-08:00",
  "message_id": "",
  "message_uid": "",
  "reply_to_status": false,
  "total_attachments": 1,
  "parent_name": "Tom Davis",
  "parent_type": "Leads",
  "parent_id": "f6a0611a-7d27-11e7-9d70-3c15c2d582c6"
  "team_count": "",
  "team_name": [{
    "id": 1,
    "name": "Global",
    "name_2": "",
    "primary": true,
    "selected": false
  }],
  "my_favorite": false,
  "_acl": {
    "fields": {}
  },
  "_module": "Emails"
}
```


Updating a Draft

Request Arguments

Name	Type	Description	Required
outbound_email_id	String	The ID of the chosen SMTP configuration for sending an email. Only use this field when saving a draft	False

Name	Type	Description	Required
		or sending an email. The user's default SMTP configuration is used if one hasn't been chosen prior to sending.	
name	String	The email's subject.	False
description_html	String	The email's HTML body.	False
description	String	The email's plain text body. This field is not necessary unless the text body is different from the HTML body.	False
reply_to_id	String	Only use this field when saving a draft or sending an email that is a reply to another email. This is the ID of that other email.	False
parent_type	String	The parent record's module.	False
parent_id	String	The parent record's ID.	False
to	Object	The email's recipients found in the TO field. Existing EmailParticipants records cannot be used; only the **create** and **delete** actions are supported. Metadata about each recipient is required to link the	False

Name	Type	Description	Required
		records. • **parent_type** and **parent_id** can be Accounts, Contacts, Employees, Leads, Prospects, or Users, out out of the box. These fields are not necessary if only an email address is known for the recipient. • **email_address_id** is the ID of the email address the recipient used in the email. To get the ID of the email address, first make a request to /EmailAddresses POST. If this argument is not provided,	

Name	Type	Description	Required
		then the primary email address of the parent record is used.	
cc	Object	The email's recipients found in the CC field. Existing EmailParticipants records cannot be used; only the **create** and **delete** actions are supported. Metadata about each recipient is required to link the records. • **parent_type** and **parent_id** can be Accounts, Contacts, Employees, Leads, Prospects, or Users, out out of the box. These fields are not necessary if only an email address is known for the recipient.	False

Name	Type	Description	Required
		• **email_address_id** is the ID of the email address the recipient used in the email. To get the ID of the email address, first make a request to /EmailAddresses POST. If this argument is not provided, then the primary email address of the parent record is used.	
bcc	Object	The email's recipients found in the BCC field. Existing EmailParticipants records cannot be used; only the **create** and **delete** actions are supported. Metadata about each recipient is required to link the records. • **parent_typ**	False

Name	Type	Description	Required
		e and **parent_id** can be Accounts, Contacts, Employees, Leads, Prospects, or Users, out out of the box. These fields are not necessary if only an email address is known for the recipient. • **email_address_id** is the ID of the email address the recipient used in the email. To get the ID of the email address, first make a request to /EmailAddresses POST. If this argument is not provided, then the primary email address of	

Name	Type	Description	Required
		the parent record is used.	
attachments	Object	The email's attachments. Existing Notes records cannot be used as email attachments; only the **create** and **delete** actions are supported. Metadata about each attachment is required to link the records. • **filename_guid** is the temporary file ID for an uploaded file to attach as a Notes record. • **upload_id** is the ID of an existing file that the Notes record should reference as the attachment. This allows files to be shared by multiple Notes records, to preserve	False

Name	Type	Description	Required
		space. • When using **upload_id**, **file_source** should be provided to indicate from where the file originated. The value should be the module of the record that shares its ID with the file. So when an attachment comes from an email template, **file_source** should be **EmailTemplates**. And when an attachment comes from a document, **file_source** should be **Document Revisions**.	

Request

```
{
  "outbound_email_id": "b80adf1a-7d78-11e7-855a-3c15c2d582c6",
  "name": "Re: Discuss Proposal (new time)",
  "description_html": "<p>I will call you in 10 minutes.</p>",
  "parent_type": "Contacts",
  "parent_id": "79b9c194-7d79-11e7-8fc5-3c15c2d582c6",
```

```
  "to": {
    "create": [{
      "parent_type": "Contacts",
      "parent_id": "79b9c194-7d79-11e7-8fc5-3c15c2d582c6"
    }],
    "delete": [
      "ea1cec7e-7d27-11e7-9845-3c15c2d582c6"
    ]
  },
  "cc": {
    "delete": [
      "eafa7e9a-7d27-11e7-aa2b-3c15c2d582c6"
    ]
  },
  "attachments": {
    "create": [{
      "upload_id": "a028341c-ba92-9343-55e7-56cf5beda121",
      "name": "Contract.pdf",
      "filename": "Contract.pdf",
      "file_source": "DocumentRevisions"
    }],
    "delete": [
      "e087e79a-7d27-11e7-b02c-3c15c2d582c6"
    ]
  }
}
```

Response

```
{
  "id": "f3c85966-7d27-11e7-9e9e-3c15c2d582c6",
  "date_entered": "2016-02-25T11:53:07-08:00",
  "created_by": "9c61c46a-a7c5-df71-481c-51d48232f820",
  "created_by_name": "Sarah Smith",
  "date_modified": "2016-02-25T12:53:07-08:00",
  "modified_by_name": "Sarah Smith",
  "modified_user_id": "9c61c46a-a7c5-df71-481c-51d48232f820",
  "deleted": false,
  "assigned_user_id": "9c61c46a-a7c5-df71-481c-51d48232f820",
  "assigned_user_name": "Sarah Smith",
  "state": "Draft",
  "outbound_email_id": "b80adf1a-7d78-11e7-855a-3c15c2d582c6",
  "name": "Re: Discuss Proposal (new time)",
```

```
  "description": "I will call you in 10 minutes.",
  "description_html": "<p>I will call you in 10 minutes.</p>",
  "date_sent": "2012-02-17T12:53:07-08:00",
  "message_id": "",
  "message_uid": "",
  "reply_to_id": "a028341c-ba92-9343-55e7-56cf5beda121",
  "reply_to_status": false,
  "total_attachments": 1,
  "parent_name": "Jack Horner",
  "parent_type": "Contacts",
  "parent_id": "79b9c194-7d79-11e7-8fc5-3c15c2d582c6"
  "team_count": "",
  "team_name": [{
    "id": 1,
    "name": "Global",
    "name_2": "",
    "primary": true,
    "selected": false
  }],
  "my_favorite": false,
  "_acl": {
    "fields": {}
  },
  "_module": "Emails"
}
```


Sending an Email

Request Arguments

Name	Type	Description	Required
state	String	Use "Ready" to send an email.	True
outbound_email_id	String	The ID of the chosen SMTP configuration for sending an email. Only use this field when saving a draft or sending an email. The user's default SMTP	False

Name	Type	Description	Required
		configuration is used if one hasn't been chosen prior to sending.	
name	String	The email's subject.	False
description_html	String	The email's HTML body.	False
description	String	The email's plain text body. This field is not necessary unless the text body is different from the HTML body.	False
reply_to_id	String	Only use this field when saving a draft or sending an email that is a reply to another email. This is the ID of that other email.	False
parent_type	String	The parent record's module.	False
parent_id	String	The parent record's ID.	False
to	Object	The email's recipients found in the TO field. Existing EmailParticipants records cannot be used; only the **create** and **delete** actions are supported. Metadata about each recipient is required to link the records. •	False

Name	Type	Description	Required
		parent_type and **parent_id** can be Accounts, Contacts, Employees, Leads, Prospects, or Users, out out of the box. These fields are not necessary if only an email address is known for the recipient. • **email_address_id** is the ID of the email address the recipient used in the email. To get the ID of the email address, first make a request to /EmailAddresses POST. If this argument is not provided, then the primary email	

Name	Type	Description	Required
		address of the parent record is used.	
cc	Object	The email's recipients found in the CC field. Existing EmailParticipants records cannot be used; only the **create** and **delete** actions are supported. Metadata about each recipient is required to link the records. • **parent_type** and **parent_id** can be Accounts, Contacts, Employees, Leads, Prospects, or Users, out out of the box. These fields are not necessary if only an email address is known for the recipient. • **email_address_id** is	False

Name	Type	Description	Required
		the ID of the email address the recipient used in the email. To get the ID of the email address, first make a request to /EmailAddresses POST. If this argument is not provided, then the primary email address of the parent record is used.	
bcc	Object	The email's recipients found in the BCC field. Existing EmailParticipants records cannot be used; only the **create** and **delete** actions are supported. Metadata about each recipient is required to link the records. • **parent_type** and **parent_id** can be	False

Name	Type	Description	Required
		Accounts, Contacts, Employees, Leads, Prospects, or Users, out out of the box. These fields are not necessary if only an email address is known for the recipient. • **email_address_id** is the ID of the email address the recipient used in the email. To get the ID of the email address, first make a request to /EmailAddresses POST. If this argument is not provided, then the primary email address of the parent record is used.	

Name	Type	Description	Required
attachments	Object	The email's attachments. Existing Notes records cannot be used as email attachments; only the **create** and **delete** actions are supported. Metadata about each attachment is required to link the records. • **filename_guid** is the temporary file ID for an uploaded file to attach as a Notes record. • **upload_id** is the ID of an existing file that the Notes record should reference as the attachment. This allows files to be shared by multiple Notes records, to preserve space. • When using	False

Name	Type	Description	Required
		upload_id, **file_source** should be provided to indicate from where the file originated. The value should be the module of the record that shares its ID with the file. So when an attachment comes from an email template, **file_source** should be **EmailTemplates**. And when an attachment comes from a document, **file_source** should be **Document Revisions**.	

Request

```
{
  "state": "Ready"
}
```

Response

```
{
```

```
  "id": "a7795664-7def-11e7-8add-3c15c2d582c6",
  "date_entered": "2016-02-25T08:53:07-08:00",
  "created_by": "9c61c46a-a7c5-df71-481c-51d48232f820",
  "created_by_name": "Sarah Smith",
  "date_modified": "2016-02-25T08:53:07-08:00",
  "modified_by_name": "Sarah Smith",
  "modified_user_id": "9c61c46a-a7c5-df71-481c-51d48232f820",
  "deleted": false,
  "assigned_user_id": "9c61c46a-a7c5-df71-481c-51d48232f820",
  "assigned_user_name": "Sarah Smith",
  "state": "Archived",
  "outbound_email_id": "b80adf1a-7d78-11e7-855a-3c15c2d582c6",
  "name": "Discuss Proposal",
  "description": "When is a good time for us to chat?",
  "description_html": "<p>When is a good time for us to chat?</p>",
  "date_sent": "2012-02-17T08:53:07-08:00",
  "message_id": "<a7795664-7def-11e7-8add-3c15c2d582c6@example.com>",
  "message_uid": "",
  "reply_to_id": "",
  "reply_to_status": false,
  "total_attachments": 2,
  "parent_name": "Tom Davis",
  "parent_type": "Leads",
  "parent_id": "f6a0611a-7d27-11e7-9d70-3c15c2d582c6"
  "team_count": "",
  "team_name": [{
    "id": 1,
    "name": "Global",
    "name_2": "",
    "primary": true,
    "selected": false
  }],
  "my_favorite": false,
  "_acl": {
    "fields": {}
  },
  "_module": "Emails"
}
```


Change Log

Version	Change

v10	Added /Emails/:record PUT endpoint.

/Emails/:record/link POST

Overview

Creates a relationship to a pre-existing email.

Summary

Cannot link existing Notes records as attachments (link: **attachments**) and existing EmailParticipants records as a sender (link: **from**) or recipients(links: **to**, **cc**, **bcc**). All other operations described in Relate Record API are supported.

Change Log

Version	Change
v10	Added /Emails/:record/link POST endpoint.

/Emails/:record/link/:link_name/:remote_id DELETE

Overview

Deletes an existing relationship between an email and another record.

Summary

The sender (link: **from**) cannot be removed. Replace the sender with a different sender instead. All other operations described in Relate Record API are supported.

Change Log

Version	Change
v10	Added /Emails/:record/link/:link_name/:remote_id DELETE endpoint.

/Emails/:record/link/:link_name/:remote_id POST

Overview

Creates a relationship to a pre-existing email.

Summary

Cannot link existing Notes records as attachments (link: **attachments**) and existing EmailParticipants records as a sender (link: **from**) or recipients(links: **to**, **cc**, **bcc**). All other operations described in Relate Record API are supported.

Change Log

Version	Change
v10	Added /Emails/:record/link/:link_name/:remote_id POST endpoint.

/Emails/:record/link/:link_name/add_record_list/:remote_id POST

Overview

Creates relationships to a pre-existing email from a record list.

Summary

Cannot link existing Notes records as attachments (link: **attachments**) and existing EmailParticipants records as a sender (link: **from**) or recipients(links: **to**, **cc**, **bcc**). All other operations described in Relate Record API are supported.

Change Log

Version	Change
v10	Added /Emails/:record/link/:link_name/add_record_list/:remote_id POST endpoint.

/Emails/count GET

Overview

Lists filtered emails.

Summary

Adds additional operators to Filter API that are specific to Emails. A special

macro, **$current_user_id**, is a convenience that allows for finding emails involving the current user.

Filter By Sender

This will return all emails sent by the current user, by the contact whose ID is fa300a0e-0ad1-b322-9601-512d0983c19a, using the email address sam@example.com, which is referenced by the ID b0701501-1fab-8ae7-3942-540da93f5017, or by the lead whose ID is 73b1087e-4bb6-11e7-acaa-3c15c2d582c6 using the email address wally@example.com, which is referenced by the ID b651d834-4bb6-11e7-bfcf-3c15c2d582c6.

```
{
  "filter": [{
    "$from": [{
      "parent_type": "Users",
      "parent_id": "$current_user_id"
    }, {
      "parent_type": "Contacts",
      "parent_id": "fa300a0e-0ad1-b322-9601-512d0983c19a"
    }, {
      "email_address_id": "b0701501-1fab-8ae7-3942-540da93f5017"
    }, {
      "parent_type": "Leads",
      "parent_id": "73b1087e-4bb6-11e7-acaa-3c15c2d582c6",
      "email_address_id": "b651d834-4bb6-11e7-bfcf-3c15c2d582c6"
    }]
  }]
}
```

Equivalent to:

```
{
  "filter": [{
    "from_collection": {
      "$in": [{
        "parent_type": "Users",
        "parent_id": "$current_user_id"
      }, {
        "parent_type": "Contacts",
        "parent_id": "fa300a0e-0ad1-b322-9601-512d0983c19a"
      }, {
        "email_address_id": "b0701501-1fab-8ae7-3942-540da93f5017"
      }, {
        "parent_type": "Leads",
```

```
        "parent_id": "73b1087e-4bb6-11e7-acaa-3c15c2d582c6",
        "email_address_id": "b651d834-4bb6-11e7-bfcf-3c15c2d582c6"
      }]
    }
  }]
}
```

Filter By Recipients

This would return all archived emails received by the current user.

```
{
  "filter": [{
    "$or": [{
      "$to": [{
        "parent_type": "Users",
        "parent_id": "$current_user_id"
      }],
      "$cc": [{
        "parent_type": "Users",
        "parent_id": "$current_user_id"
      }],
      "$bcc": [{
        "parent_type": "Users",
        "parent_id": "$current_user_id"
      }]
    }]
  }, {
    "state": {
      "$in": [
        "Archived"
      ]
    }
  }]
}
```

Equivalent to:

```
{
  "filter": [{
    "$or": [{
      "to_collection": {
```

```
        "$in": [{
          "parent_type": "Users",
          "parent_id": "$current_user_id"
        }]
      },
      "cc_collection": {
        "$in": [{
          "parent_type": "Users",
          "parent_id": "$current_user_id"
        }]
      },
      "bcc_collection": {
        "$in": [{
          "parent_type": "Users",
          "parent_id": "$current_user_id"
        }]
      }
    }]
  }, {
    "state": {
      "$in": [
        "Archived"
      ]
    }
  }]
}
```

Filter By Sender and Recipients

This would return all archived emails sent by a contact to the current user.

```
{
  "filter": [{
    "$from": [{
      "parent_type": "Contacts",
      "parent_id": "fa300a0e-0ad1-b322-9601-512d0983c19a"
    }]
  }, {
    "$or": [{
      "$to": [{
        "parent_type": "Users",
        "parent_id": "$current_user_id"
      }],
```

```
      "$cc": [{
        "parent_type": "Users",
        "parent_id": "$current_user_id"
      }],
      "$bcc": [{
        "parent_type": "Users",
        "parent_id": "$current_user_id"
      }]
    }]
  }, {
    "state": {
      "$in": [
        "Archived"
      ]
    }
  }]
}
```

Equivalent to:

```
{
  "filter": [{
    "from_collection": {
      "$in": [{
        "parent_type": "Contacts",
        "parent_id": "fa300a0e-0ad1-b322-9601-512d0983c19a"
      }]
    }
  }, {
    "$or": [{
      "to_collection": {
        "$in": [{
          "parent_type": "Users",
          "parent_id": "$current_user_id"
        }]
      },
      "cc_collection": {
        "$in": [{
          "parent_type": "Users",
          "parent_id": "$current_user_id"
        }]
      },
      "bcc_collection": {
        "$in": [{
          "parent_type": "Users",
```

```
          "parent_id": "$current_user_id"
        }]
      }
    }]
  }, {
    "state": {
      "$in": [
        "Archived"
      ]
    }
  }]
}
```

Change Log

Version	Change
v10	Added the $from, $to, $cc, and $bcc operators to /Emails/filter GET endpoint.

/Emails/filter GET

Overview

Lists filtered emails.

Summary

Adds additional operators to Filter API that are specific to Emails. A special macro, **$current_user_id**, is a convenience that allows for finding emails involving the current user.

Filter By Sender

This will return all emails sent by the current user, by the contact whose ID is fa300a0e-0ad1-b322-9601-512d0983c19a, using the email address sam@example.com, which is referenced by the ID b0701501-1fab-8ae7-3942-540da93f5017, or by the lead whose ID is 73b1087e-4bb6-11e7-acaa-3c15c2d582c6 using the email address wally@example.com, which is referenced by the ID b651d834-4bb6-11e7-bfcf-3c15c2d582c6.

```
{
  "filter": [{
```

```
    "$from": [{
      "parent_type": "Users",
      "parent_id": "$current_user_id"
    }, {
      "parent_type": "Contacts",
      "parent_id": "fa300a0e-0ad1-b322-9601-512d0983c19a"
    }, {
      "email_address_id": "b0701501-1fab-8ae7-3942-540da93f5017"
    }, {
      "parent_type": "Leads",
      "parent_id": "73b1087e-4bb6-11e7-acaa-3c15c2d582c6",
      "email_address_id": "b651d834-4bb6-11e7-bfcf-3c15c2d582c6"
    }]
  }]
}
```

Equivalent to:

```
{
  "filter": [{
    "from_collection": {
      "$in": [{
        "parent_type": "Users",
        "parent_id": "$current_user_id"
      }, {
        "parent_type": "Contacts",
        "parent_id": "fa300a0e-0ad1-b322-9601-512d0983c19a"
      }, {
        "email_address_id": "b0701501-1fab-8ae7-3942-540da93f5017"
      }, {
        "parent_type": "Leads",
        "parent_id": "73b1087e-4bb6-11e7-acaa-3c15c2d582c6",
        "email_address_id": "b651d834-4bb6-11e7-bfcf-3c15c2d582c6"
      }]
    }
  }]
}
```

Filter By Recipients

This would return all archived emails received by the current user.

```
{
  "filter": [{
    "$or": [{
      "$to": [{
        "parent_type": "Users",
        "parent_id": "$current_user_id"
      }],
      "$cc": [{
        "parent_type": "Users",
        "parent_id": "$current_user_id"
      }],
      "$bcc": [{
        "parent_type": "Users",
        "parent_id": "$current_user_id"
      }]
    }]
  }, {
    "state": {
      "$in": [
        "Archived"
      ]
    }
  }]
}
```

Equivalent to:

```
{
  "filter": [{
    "$or": [{
      "to_collection": {
        "$in": [{
          "parent_type": "Users",
          "parent_id": "$current_user_id"
        }]
      },
      "cc_collection": {
        "$in": [{
          "parent_type": "Users",
          "parent_id": "$current_user_id"
        }]
      },
      "bcc_collection": {
        "$in": [{
          "parent_type": "Users",
```

```
          "parent_id": "$current_user_id"
        }]
      }
    }]
  }, {
    "state": {
      "$in": [
        "Archived"
      ]
    }
  }]
}
```

Filter By Sender and Recipients

This would return all archived emails sent by a contact to the current user.

```
{
  "filter": [{
    "$from": [{
      "parent_type": "Contacts",
      "parent_id": "fa300a0e-0ad1-b322-9601-512d0983c19a"
    }]
  }, {
    "$or": [{
      "$to": [{
        "parent_type": "Users",
        "parent_id": "$current_user_id"
      }],
      "$cc": [{
        "parent_type": "Users",
        "parent_id": "$current_user_id"
      }],
      "$bcc": [{
        "parent_type": "Users",
        "parent_id": "$current_user_id"
      }]
    }]
  }, {
    "state": {
      "$in": [
        "Archived"
      ]
```

```
    }
  }]
}
```

Equivalent to:

```
{
  "filter": [{
    "from_collection": {
      "$in": [{
        "parent_type": "Contacts",
        "parent_id": "fa300a0e-0ad1-b322-9601-512d0983c19a"
      }]
    }
  }, {
    "$or": [{
      "to_collection": {
        "$in": [{
          "parent_type": "Users",
          "parent_id": "$current_user_id"
        }]
      },
      "cc_collection": {
        "$in": [{
          "parent_type": "Users",
          "parent_id": "$current_user_id"
        }]
      },
      "bcc_collection": {
        "$in": [{
          "parent_type": "Users",
          "parent_id": "$current_user_id"
        }]
      }
    }]
  }, {
    "state": {
      "$in": [
        "Archived"
      ]
    }
  }]
}
```

Change Log

Version	Change
v10	Added the $from, $to, $cc, and $bcc operators to /Emails/filter GET endpoint.

/Emails/filter POST

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

Name	Type	Description	Required
filter	String	The filter expression. Filter expressions are explained below.	False
filter_id	String	Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND.	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over	False

Name	Type	Description	Required
		before records are returned. Default is 0.	
fields	String	Comma delimited list of fields to return. The field date_modified will always be returned. Example: name,account_type,description	False
view	String	Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list". Example: record	False
order_by	String	How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon. Example: name:DESC,account_type:DESC,date_modified:ASC	False

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
   "filter":[
      {
         "name":"Nelson Inc"
      }
   ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
   "filter":[
      {
         "name":{
            "$starts":"Nelson"
         }
      }
   ]
}
```

Below is a list of operation types:

Operation	Description
$equals	Performs an exact match on that field.
$not_equals	Performs an exact match on that field.
$not_equals	Matches on non-

Operation		Description
		matching values.
	$starts	Matches on anything that starts with the value.
	$ends	Matches anything that ends with the value.
	$contains	Matches anything that contains the value
	$in	Finds anything where field matches one of the values as specified as an array.
	$not_in	Finds anything where field does not matches any of the values as specified as an array.
	$is_null	Checks if the field is null. This operation does not need a value specified.
	$not_null	Checks if the field is not null. This operation does not need a value specified.
	$lt	Matches when the field is less than the value.
	$lte	Matches when the field is less than or equal to the value.
	$gt	Matches when the field is greater than the value.
	$gte	Matches when the field is greater than or equal to the value.

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "$and" and "$or".

Example

```
{
   "filter":[
      {
         "$or":[
            {
               "name":"Nelson Inc"
            },
            {
               "name":"Nelson LLC"
            }
         ]
      }
   ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "$favorite" and "$owner".

Example

```
{
   "filter":[
      {
         "$favorite":"_this"
      }
   ]
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Response


```
{
   "next_offset":-1,
   "records":[
      {
         "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
         "name":"Dale Spivey",
         "date_entered":"2013-02-26T19:12:00+00:00",
         "date_modified":"2013-02-28T05:03:00+00:00",
         "modified_user_id":"1",
         "modified_by_name":"Administrator",
         "created_by":"1",
         "created_by_name":"Administrator",
         "description":"",
         "img":"",
         "deleted":false,
         "assigned_user_id":"seed_sally_id",
         "assigned_user_name":"Sally Bronsen",
         "team_name":[
            {
               "id":"East",
               "name":"East",
               "name_2":"",
               "primary":false
            },
            {
               "id":1,
               "name":"Global",
               "name_2":"",
```

```
            "primary":false
         },
         {
            "id":"West",
            "name":"West",
            "name_2":"",
            "primary":true
         }
      ],
      "salutation":"",
      "first_name":"Dale",
      "last_name":"Spivey",
      "full_name":"Dale Spivey",
      "title":"VP Operations",
      "linkedin":"",
      "facebook":"",
      "twitter":"",
      "googleplus":"",
      "department":"",
      "do_not_call":false,
      "phone_home":"(523) 825-4311",
      "email":[
         {
            "email_address":"sugar.dev.sugar@example.co.jp",
            "opt_out":"0",
            "invalid_email":"0",
            "primary_address":"1"
         },
         {
            "email_address":"the.support@example.biz",
            "opt_out":"0",
            "invalid_email":"0",
            "primary_address":"0"
         }
      ],
      "phone_mobile":"(373) 861-0757",
      "phone_work":"(212) 542-9596",
      "phone_other":"",
      "phone_fax":"",
      "email1":"sugar.dev.sugar@example.co.jp",
      "email2":"the.support@example.biz",
      "invalid_email":false,
      "email_opt_out":false,
      "primary_address_street":"345 Sugar Blvd.",
      "primary_address_street_2":"",
      "primary_address_street_3":"",
```

```
      "primary_address_city":"Denver",
      "primary_address_state":"CA",
      "primary_address_postalcode":"87261",
      "primary_address_country":"USA",
      "alt_address_street":"",
      "alt_address_street_2":"",
      "alt_address_street_3":"",
      "alt_address_city":"",
      "alt_address_state":"",
      "alt_address_postalcode":"",
      "alt_address_country":"",
      "assistant":"",
      "assistant_phone":"",
      "picture":"",
      "email_and_name1":"",
      "lead_source":"Campaign",
      "account_name":"Smallville Resources Inc",
      "account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",
      "opportunity_role_fields":"",
      "opportunity_role_id":"",
      "opportunity_role":"",
      "reports_to_id":"",
      "report_to_name":"",
      "portal_name":"DaleSpivey97",
      "portal_active":true,
      "portal_password":"$1$yKMAONHM$Y5S.8CY.WZCZCwfGD1a1Q\/",
      "portal_password1":"",
      "portal_app":"",
      "preferred_language":"en_us",
      "campaign_id":"",
      "campaign_name":"",
      "c_accept_status_fields":"",
      "m_accept_status_fields":"",
      "accept_status_id":"",
      "accept_status_name":"",
      "sync_contact":"",
      "my_favorite":false,
      "_acl":{
         "fields":{

         }
      }
   },
   {
      "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
      "name":"Florence Haddock",
```

```
"date_entered":"2013-02-26T19:12:00+00:00",
"date_modified":"2013-02-26T19:12:00+00:00",
"modified_user_id":"1",
"modified_by_name":"Administrator",
"created_by":"1",
"created_by_name":"Administrator",
"description":"",
"img":"",
"deleted":false,
"assigned_user_id":"seed_sally_id",
"assigned_user_name":"Sally Bronsen",
"team_name":[
   {
      "id":"East",
      "name":"East",
      "name_2":"",
      "primary":false
   },
   {
      "id":1,
      "name":"Global",
      "name_2":"",
      "primary":false
   },
   {
      "id":"West",
      "name":"West",
      "name_2":"",
      "primary":true
   }
],
"salutation":"",
"first_name":"Florence",
"last_name":"Haddock",
"full_name":"Florence Haddock",
"title":"Director Sales",
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(729) 845-3137",
"email":[
   {
      "email_address":"dev.vegan@example.de",
```

```
            "opt_out":"1",
            "invalid_email":"0",
            "primary_address":"0"
        },
        {
            "email_address":"section71@example.it",
            "opt_out":"0",
            "invalid_email":"0",
            "primary_address":"1"
        }
    ],
    "phone_mobile":"(246) 233-1382",
    "phone_work":"(565) 696-6981",
    "phone_other":"",
    "phone_fax":"",
    "email1":"section71@example.it",
    "email2":"dev.vegan@example.de",
    "invalid_email":false,
    "email_opt_out":false,
    "primary_address_street":"111 Silicon Valley Road",
    "primary_address_street_2":"",
    "primary_address_street_3":"",
    "primary_address_city":"Denver",
    "primary_address_state":"CA",
    "primary_address_postalcode":"79900",
    "primary_address_country":"USA",
    "alt_address_street":"",
    "alt_address_street_2":"",
    "alt_address_street_3":"",
    "alt_address_city":"",
    "alt_address_state":"",
    "alt_address_postalcode":"",
    "alt_address_country":"",
    "assistant":"",
    "assistant_phone":"",
    "picture":"",
    "email_and_name1":"",
    "lead_source":"Support Portal User Registration",
    "account_name":"Smallville Resources Inc",
    "account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",
    "opportunity_role_fields":"",
    "opportunity_role_id":"",
    "opportunity_role":"",
    "reports_to_id":"",
    "report_to_name":"",
    "portal_name":"FlorenceHaddock169",
```

```
            "portal_active":true,
            "portal_password":"$1$nWFhTbK6$JF9BCGSqL\/NCrbhueX5ia0",
            "portal_password1":"",
            "portal_app":"",
            "preferred_language":"en_us",
            "campaign_id":"",
            "campaign_name":"",
            "c_accept_status_fields":"",
            "m_accept_status_fields":"",
            "accept_status_id":"",
            "accept_status_name":"",
            "sync_contact":"",
            "my_favorite":false,
            "_acl":{
               "fields":{

               }
            }
         }
      ]
}
```


Change Log

Version	Change
v10	Added /<module>/filter POST endpoint.

/Emails/filter/count GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long.

Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

Name	Type	Description	Required
filter	String	The filter expression. Filter expressions are explained below.	False
filter_id	String	Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND.	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
fields	String	Comma delimited list of fields to return. The field date_modified will always be returned. Example: name,account_type,description	False
view	String	Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view	False

Name	Type	Description	Required
		definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list". Example: record	
order_by	String	How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon. Example: name:DESC,account_type:DESC,date_modified:ASC	False

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
   "filter":[
      {
         "name":"Nelson Inc"
      }
   ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
   "filter":[
      {
         "name":{
            "$starts":"Nelson"
         }
      }
   ]
}
```

Below is a list of operation types:

Operation	Description
$equals	Performs an exact match on that field.
$not_equals	Performs an exact match on that field.
$not_equals	Matches on non-matching values.
$starts	Matches on anything that starts with the value.
$ends	Matches anything that ends with the value.
$contains	Matches anything that contains the value
$in	Finds anything where field matches one of the values as specified as an array.
$not_in	Finds anything where field does not matches any of the values as specified as an array.

Operation		Description
	$is_null	Checks if the field is null. This operation does not need a value specified.
	$not_null	Checks if the field is not null. This operation does not need a value specified.
	$lt	Matches when the field is less than the value.
	$lte	Matches when the field is less than or equal to the value.
	$gt	Matches when the field is greater than the value.
	$gte	Matches when the field is greater than or equal to the value.

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "$and" and "$or".

Example

```
{
   "filter":[
      {
         "$or":[
            {
               "name":"Nelson Inc"
            },
            {
               "name":"Nelson LLC"
            }
         ]
```

```
      }
   ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "$favorite" and "$owner".

Example

```
{
   "filter":[
      {
         "$favorite":"_this"
      }
   ]
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Response

```
{
   "next_offset":-1,
```

```
"records":[
   {
      "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
      "name":"Dale Spivey",
      "date_entered":"2013-02-26T19:12:00+00:00",
      "date_modified":"2013-02-28T05:03:00+00:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "description":"",
      "img":"",
      "deleted":false,
      "assigned_user_id":"seed_sally_id",
      "assigned_user_name":"Sally Bronsen",
      "team_name":[
         {
            "id":"East",
            "name":"East",
            "name_2":"",
            "primary":false
         },
         {
            "id":1,
            "name":"Global",
            "name_2":"",
            "primary":false
         },
         {
            "id":"West",
            "name":"West",
            "name_2":"",
            "primary":true
         }
      ],
      "salutation":"",
      "first_name":"Dale",
      "last_name":"Spivey",
      "full_name":"Dale Spivey",
      "title":"VP Operations",
      "linkedin":"",
      "facebook":"",
      "twitter":"",
      "googleplus":"",
      "department":"",
      "do_not_call":false,
```

```
"phone_home":"(523) 825-4311",
"email":[
   {
      "email_address":"sugar.dev.sugar@example.co.jp",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"1"
   },
   {
      "email_address":"the.support@example.biz",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"0"
   }
],
"phone_mobile":"(373) 861-0757",
"phone_work":"(212) 542-9596",
"phone_other":"",
"phone_fax":"",
"email1":"sugar.dev.sugar@example.co.jp",
"email2":"the.support@example.biz",
"invalid_email":false,
"email_opt_out":false,
"primary_address_street":"345 Sugar Blvd.",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Denver",
"primary_address_state":"CA",
"primary_address_postalcode":"87261",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
"assistant":"",
"assistant_phone":"",
"picture":"",
"email_and_name1":"",
"lead_source":"Campaign",
"account_name":"Smallville Resources Inc",
"account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",
"opportunity_role_fields":"",
"opportunity_role_id":"",
```

```
    "opportunity_role":"",
    "reports_to_id":"",
    "report_to_name":"",
    "portal_name":"DaleSpivey97",
    "portal_active":true,
    "portal_password":"$1$yKMAONHM$Y5S.8CY.WZCZCwfGD1a1Q\/",
    "portal_password1":"",
    "portal_app":"",
    "preferred_language":"en_us",
    "campaign_id":"",
    "campaign_name":"",
    "c_accept_status_fields":"",
    "m_accept_status_fields":"",
    "accept_status_id":"",
    "accept_status_name":"",
    "sync_contact":"",
    "my_favorite":false,
    "_acl":{
       "fields":{

       }
    }
},
{
    "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
    "name":"Florence Haddock",
    "date_entered":"2013-02-26T19:12:00+00:00",
    "date_modified":"2013-02-26T19:12:00+00:00",
    "modified_user_id":"1",
    "modified_by_name":"Administrator",
    "created_by":"1",
    "created_by_name":"Administrator",
    "description":"",
    "img":"",
    "deleted":false,
    "assigned_user_id":"seed_sally_id",
    "assigned_user_name":"Sally Bronsen",
    "team_name":[
       {
          "id":"East",
          "name":"East",
          "name_2":"",
          "primary":false
       },
       {
          "id":1,
```

```
            "name":"Global",
            "name_2":"",
            "primary":false
        },
        {
            "id":"West",
            "name":"West",
            "name_2":"",
            "primary":true
        }
    ],
    "salutation":"",
    "first_name":"Florence",
    "last_name":"Haddock",
    "full_name":"Florence Haddock",
    "title":"Director Sales",
    "linkedin":"",
    "facebook":"",
    "twitter":"",
    "googleplus":"",
    "department":"",
    "do_not_call":false,
    "phone_home":"(729) 845-3137",
    "email":[
        {
            "email_address":"dev.vegan@example.de",
            "opt_out":"1",
            "invalid_email":"0",
            "primary_address":"0"
        },
        {
            "email_address":"section71@example.it",
            "opt_out":"0",
            "invalid_email":"0",
            "primary_address":"1"
        }
    ],
    "phone_mobile":"(246) 233-1382",
    "phone_work":"(565) 696-6981",
    "phone_other":"",
    "phone_fax":"",
    "email1":"section71@example.it",
    "email2":"dev.vegan@example.de",
    "invalid_email":false,
    "email_opt_out":false,
    "primary_address_street":"111 Silicon Valley Road",
```

```
      "primary_address_street_2":"",
      "primary_address_street_3":"",
      "primary_address_city":"Denver",
      "primary_address_state":"CA",
      "primary_address_postalcode":"79900",
      "primary_address_country":"USA",
      "alt_address_street":"",
      "alt_address_street_2":"",
      "alt_address_street_3":"",
      "alt_address_city":"",
      "alt_address_state":"",
      "alt_address_postalcode":"",
      "alt_address_country":"",
      "assistant":"",
      "assistant_phone":"",
      "picture":"",
      "email_and_name1":"",
      "lead_source":"Support Portal User Registration",
      "account_name":"Smallville Resources Inc",
      "account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",
      "opportunity_role_fields":"",
      "opportunity_role_id":"",
      "opportunity_role":"",
      "reports_to_id":"",
      "report_to_name":"",
      "portal_name":"FlorenceHaddock169",
      "portal_active":true,
      "portal_password":"$1$nWFhTbK6$JF9BCGSqL\/NCrbhueX5ia0",
      "portal_password1":"",
      "portal_app":"",
      "preferred_language":"en_us",
      "campaign_id":"",
      "campaign_name":"",
      "c_accept_status_fields":"",
      "m_accept_status_fields":"",
      "accept_status_id":"",
      "accept_status_name":"",
      "sync_contact":"",
      "my_favorite":false,
      "_acl":{
         "fields":{

         }
      }
   }
]
```

```
}
```

Change Log

Version	Change
v10	Added /<module>/filter POST endpoint.

/Emails/filter/count POST

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

Name	Type	Description	Required
filter	String	The filter expression. Filter expressions are explained below.	False
filter_id	String	Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND.	False
max_num	Integer	A maximum	False

Name	Type	Description	Required
		number of records to return. Default is 20.	
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
fields	String	Comma delimited list of fields to return. The field date_modified will always be returned. Example: name,account_type,description	False
view	String	Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list". Example: record	False
order_by	String	How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon.	False

Name	Type	Description	Required
		Example: name:DESC,account_type:DESC,date_modified:ASC	

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
   "filter":[
      {
         "name":"Nelson Inc"
      }
   ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
   "filter":[
      {
         "name":{
            "$starts":"Nelson"
         }
      }
   ]
}
```

Below is a list of operation types:

Operation	Description
$equals	Performs an exact match on that field.
$not_equals	Performs an exact match on that field.
$not_equals	Matches on non-matching values.
$starts	Matches on anything that starts with the value.
$ends	Matches anything that ends with the value.
$contains	Matches anything that contains the value
$in	Finds anything where field matches one of the values as specified as an array.
$not_in	Finds anything where field does not matches any of the values as specified as an array.
$is_null	Checks if the field is null. This operation does not need a value specified.
$not_null	Checks if the field is not null. This operation does not need a value specified.
$lt	Matches when the field is less than the value.
$lte	Matches when the field is less than or equal to the value.
$gt	Matches when the

Operation		Description	
			field is greater than the value.
	$gte		Matches when the field is greater than or equal to the value.

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "$and" and "$or".

Example

```
{
   "filter":[
      {
         "$or":[
            {
               "name":"Nelson Inc"
            },
            {
               "name":"Nelson LLC"
            }
         ]
      }
   ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "$favorite" and "$owner".

Example

```
{
```

```
   "filter":[
      {
         "$favorite":"_this"
      }
   ]
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Response

```
{
   "next_offset":-1,
   "records":[
      {
         "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
         "name":"Dale Spivey",
         "date_entered":"2013-02-26T19:12:00+00:00",
         "date_modified":"2013-02-28T05:03:00+00:00",
         "modified_user_id":"1",
         "modified_by_name":"Administrator",
         "created_by":"1",
         "created_by_name":"Administrator",
         "description":"",
         "img":"",
         "deleted":false,
         "assigned_user_id":"seed_sally_id",
         "assigned_user_name":"Sally Bronsen",
         "team_name":[
            {
               "id":"East",
```

```
            "name":"East",
            "name_2":"",
            "primary":false
        },
        {
            "id":1,
            "name":"Global",
            "name_2":"",
            "primary":false
        },
        {
            "id":"West",
            "name":"West",
            "name_2":"",
            "primary":true
        }
    ],
    "salutation":"",
    "first_name":"Dale",
    "last_name":"Spivey",
    "full_name":"Dale Spivey",
    "title":"VP Operations",
    "linkedin":"",
    "facebook":"",
    "twitter":"",
    "googleplus":"",
    "department":"",
    "do_not_call":false,
    "phone_home":"(523) 825-4311",
    "email":[
        {
            "email_address":"sugar.dev.sugar@example.co.jp",
            "opt_out":"0",
            "invalid_email":"0",
            "primary_address":"1"
        },
        {
            "email_address":"the.support@example.biz",
            "opt_out":"0",
            "invalid_email":"0",
            "primary_address":"0"
        }
    ],
    "phone_mobile":"(373) 861-0757",
    "phone_work":"(212) 542-9596",
    "phone_other":"",
```

```
"phone_fax":"",
"email1":"sugar.dev.sugar@example.co.jp",
"email2":"the.support@example.biz",
"invalid_email":false,
"email_opt_out":false,
"primary_address_street":"345 Sugar Blvd.",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Denver",
"primary_address_state":"CA",
"primary_address_postalcode":"87261",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
"assistant":"",
"assistant_phone":"",
"picture":"",
"email_and_name1":"",
"lead_source":"Campaign",
"account_name":"Smallville Resources Inc",
"account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",
"opportunity_role_fields":"",
"opportunity_role_id":"",
"opportunity_role":"",
"reports_to_id":"",
"report_to_name":"",
"portal_name":"DaleSpivey97",
"portal_active":true,
"portal_password":"$1$yKMAONHM$Y5S.8CY.WZCZCwfGD1a1Q\/",
"portal_password1":"",
"portal_app":"",
"preferred_language":"en_us",
"campaign_id":"",
"campaign_name":"",
"c_accept_status_fields":"",
"m_accept_status_fields":"",
"accept_status_id":"",
"accept_status_name":"",
"sync_contact":"",
"my_favorite":false,
"_acl":{
```

```
            "fields":{

            }
        }
    },
    {
        "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
        "name":"Florence Haddock",
        "date_entered":"2013-02-26T19:12:00+00:00",
        "date_modified":"2013-02-26T19:12:00+00:00",
        "modified_user_id":"1",
        "modified_by_name":"Administrator",
        "created_by":"1",
        "created_by_name":"Administrator",
        "description":"",
        "img":"",
        "deleted":false,
        "assigned_user_id":"seed_sally_id",
        "assigned_user_name":"Sally Bronsen",
        "team_name":[
            {
                "id":"East",
                "name":"East",
                "name_2":"",
                "primary":false
            },
            {
                "id":1,
                "name":"Global",
                "name_2":"",
                "primary":false
            },
            {
                "id":"West",
                "name":"West",
                "name_2":"",
                "primary":true
            }
        ],
        "salutation":"",
        "first_name":"Florence",
        "last_name":"Haddock",
        "full_name":"Florence Haddock",
        "title":"Director Sales",
        "linkedin":"",
        "facebook":"",
```

```
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(729) 845-3137",
"email":[
   {
      "email_address":"dev.vegan@example.de",
      "opt_out":"1",
      "invalid_email":"0",
      "primary_address":"0"
   },
   {
      "email_address":"section71@example.it",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"1"
   }
],
"phone_mobile":"(246) 233-1382",
"phone_work":"(565) 696-6981",
"phone_other":"",
"phone_fax":"",
"email1":"section71@example.it",
"email2":"dev.vegan@example.de",
"invalid_email":false,
"email_opt_out":false,
"primary_address_street":"111 Silicon Valley Road",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Denver",
"primary_address_state":"CA",
"primary_address_postalcode":"79900",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
"assistant":"",
"assistant_phone":"",
"picture":"",
"email_and_name1":"",
"lead_source":"Support Portal User Registration",
```

```
         "account_name":"Smallville Resources Inc",
         "account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",
         "opportunity_role_fields":"",
         "opportunity_role_id":"",
         "opportunity_role":"",
         "reports_to_id":"",
         "report_to_name":"",
         "portal_name":"FlorenceHaddock169",
         "portal_active":true,
         "portal_password":"$1$nWFhTbK6$JF9BCGSqL\/NCrbhueX5ia0",
         "portal_password1":"",
         "portal_app":"",
         "preferred_language":"en_us",
         "campaign_id":"",
         "campaign_name":"",
         "c_accept_status_fields":"",
         "m_accept_status_fields":"",
         "accept_status_id":"",
         "accept_status_name":"",
         "sync_contact":"",
         "my_favorite":false,
         "_acl":{
            "fields":{

            }
         }
      }
   ]
}
```


Change Log

Version	Change
v10	Added /<module>/filter POST endpoint.

/EmbeddedFiles/:record/file/:field GET

Overview

Retrieves an attached file for a specific field on a record.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

The response from this request is an HTTP response with a forced download. This can be used in rendering images through the API or in offering immediate file downloads.

Response

```
Cache-Control: no-cache, must-revalidate Connection: keep-alive Conten
t-Encoding: gzip Content-Type: application/octet-stream Date: Mon, 30
Jan 2012 17:00:46 GMT Expires: Mon, 30 Jan 2012 17:00:45 GMT Last-Modi
fied: Thu, 10 Nov 2011 19:01:31 GMT Transfer-
Encoding: chunked Vary: Accept-Encoding...
```

Change Log

Version	Change
v10	Added /<module>/:record/file/:field GET endpoint.

/EmbeddedFiles/:record/file/:field POST

Overview

Attaches a file to a field on a record.

Request Arguments

Name	Type	Description	Required
format	String	The data format. Currently accepts 'sugar-html-json'.	True
delete_if_fails	Boolean	Indicates whether the API is to mark related record deleted if the file upload fails.	False

Name	Type	Description	Required
oauth_token	String	The oauth_token value.	False - Required if only if delete_if_fails is true.
<attachment field>	String	The field and file to populate. Example: {"":"@\/path\/to\/ExampleDocument.txt"}	True

Request

```
{
   "format":"sugar-html-json",
   "delete_if_fails":true,
   "oauth_token":"43b6b327-cc70-c301-3299-512ffb99ad97",
   "<attachment field>":"@\/path\/to\/ExampleDocument.txt"
}
```

Response Arguments

Name	Type	Description
content-type	String	The files content type.
content-length	Integer	The files content length.
name	String	The files name.
uri	String	The URI of the file.

Response

```
{
   "content-type":"text\/plain",
   "content-length":16,
   "name":"ExampleDocument.txt",
   "uri":"http:\/\/sugarcrm\/rest\/v10\/Notes\/ca66c92f-5a8b-28b4-4fc8
-512d099b790b\/file\/<attachment field>?format=sugar-html-json&delete_
if_fails=1&oauth_token=6ec91cf3-1f97-25b9-e0b1-512f8971b2d4"
}
```

Change Log

Version	Change
v10	Added /<module>/:record/file/:field POST endpoint.

/EmbeddedFiles/:record/file/:field PUT

Overview

This endpoint takes a file or image and saves it to a record that already contains an attachment in the specified field. The PUT method is very similar to the POST method but differs slightly in how the request is constructed. PUT requests should send the file data as the body of the request. Optionally, a filename query parameter can be sent with the request to assign a name. Additionally, the PUT method can accept base64 encoded file data which will be decoded by the endpoint if the content_transfer_encoding parameter is set to 'base64'.

NOTE: In lieu of the content_transfer_encoding parameter, a request header of X-Content-Transfer-Encoding can also be sent with a value of 'base64'. In the event both the content transfer encoding header and request parameter are sent, the header will be used.

Request Arguments

Name	Type	Description	Required
filename	String	Filename to save the document as	False
content_transfer_encoding	String	When set to 'base64', indicates the file contents are base64 encoded and will result in the file contents being base64 decoded before being saved	False

Request

PUT /rest/v10/Notes/abcd-1234/file/field/ HTTP/1.1 Host: localhost Connection: keep-alive Content-Length: 23456 Content-Type: application/document-doc ...This is where the bin data would be

Response Arguments

Name	Type	Description
field	Array	The fields containing file properties.
field.content-type	String	The files content type.
field.content-length	Integer	The files content length.
field.name	String	The files name.
field.width	Integer	The width of the image.
field.height	Integer	The height of the image.
field.uri	String	The URI of the file.

Response

```
{
   "picture":{
      "content-type":"image\/png",
      "content-length":72512,
      "name":"1a7b8f5c-b11c-0094-c8d8-512e9daaa983",
      "width":933,
      "height":519,
      "uri":"http:\/\/sugarcrm\/rest\/v10\/Contacts\/fa300a0e-0ad1-b32
2-9601-512d0983c19a\/file\/picture"
   }
    "attachment":{
        "content-type":"application/document-doc",
        "content-length":"873921",
        "name":"myFile.doc",
        "uri": "http:\/\/sugarcrm\/rest\/v10\/Contacts\/f2f9aa4d-99a8-
e86e-f4d5-512d0986effa\/file\/attachment"
    }
}
```

Change Log

Version	Change
v10	Added /<module>/:record/file/:field PUT endpoint.

/Employees GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

Name	Type	Description	Required
filter	String	The filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query parameters in one of two ways for GET requests: 1. By specifying individual filter arguments as distinct parameters. Example: filter[0][id]=1. 2. By specifying the whole	False

Name	Type	Description	Required
		filter as a single JSON-encoded string. Note that this syntax is currently not supported on certain modules. Example: filter=[{"id":"1"}].	
filter_id	String	Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND.	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
fields	String	Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters (applicable to link and collection fields). The fields id and date_modified will always be	False

Name	Type	Description	Required
		returned. Example: name,account_type,description,{"name":"opportunities","fields":["id","name","sales_status"],"order_by":"date_closed:desc"} For more details on additional field parameters, see Relate API and Collection API.	
view	String	Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list". Example: record	False
order_by	String	How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon. Example: name:DESC,account_type:DESC,date_modified:ASC	False

Name	Type	Description	Required
q	String	A search expression, will search on this module. Cannot be used at the same time as a filter expression or id.	False
deleted	Boolean	Boolean to show deleted records in the result set.	False
nulls_last	Boolean	Boolean to return records with null values in order_by fields last in the result set.	False

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
   "filter":[
      {
         "name":"Nelson Inc"
      }
   ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
   "filter":[
      {
         "name":{
            "$starts":"Nelson"
         }
      }
   ]
}
```

Below is a list of operation types:

Operation	Description
$equals	Performs an exact match on that field.
$not_equals	Performs an exact match on that field.
$not_equals	Matches on non-matching values.
$starts	Matches on anything that starts with the value.
$ends	Matches anything that ends with the value.
$contains	Matches anything that contains the value
$in	Finds anything where field matches one of the values as specified as an array.
$not_in	Finds anything where field does not matches any of the values as specified as an array.
$is_null	Checks if the field is null. This operation does not need a value specified.
$not_null	Checks if the field is

Operation		Description
		not null. This operation does not need a value specified.
	$lt	Matches when the field is less than the value.
	$lte	Matches when the field is less than or equal to the value.
	$gt	Matches when the field is greater than the value.
	$gte	Matches when the field is greater than or equal to the value.

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "$and" and "$or".

Example

```
{
   "filter":[
      {
         "$or":[
            {
               "name":"Nelson Inc"
            },
            {
               "name":"Nelson LLC"
            }
         ]
      }
   ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "$favorite" and "$owner".

Example

```
{
   "filter":[
      {
         "$favorite":"_this"
      }
   ]
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Response

```
{
   "next_offset":-1,
   "records":[
      {
         "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
         "name":"Dale Spivey",
         "date_modified":"2013-02-28T05:03:00+00:00",
```

```
         "description":"",
         "opportunities": [
            {
               _module: "Opportunities",
               "id": "b0701501-1fab-8ae7-3942-540da93f5017",
               "name": "360 Vacations - 228 Units",
               "date_modified": "2014-09-08T16:05:00+03:00",
               "sales_status": "New"
            },
         ],
         "_acl": {
            "fields": {
            }
         }
      },
      {
         "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
         "name":"Florence Haddock",
         "date_modified":"2013-02-26T19:12:00+00:00",
         "description":"",
         "opportunities": [
            {
               _module: "Opportunities"
               date_modified: "2014-09-08T16:05:00+03:00"
               id: "9ce7c088-8ee4-7cd3-18f1-540da944d4c0"
               name: "360 Vacations - 312 Units"
               sales_status: "New"
            },
         ],
         "_acl": {
            "fields": {
            }
         }
      }
   ]
}
```


Change Log

Version	Change
v10	Added /<module>/filter GET endpoint.

/ExpressionEngine/:record/related GET

Retrieve a Forecasting Information In SugarChart Format

Summary:

This endpoint is used to return the json Data for SugarCharts to use to display the needed chart.

Query Parameters:

Param	Description	Optional
timeperiod_id	Show for a specific time period	
user_id	Show for a specific user	
display_manager	Pipeline or Committed are valid values.	
dataset	Which Forecast dataset to show, valid values are likely, best, worst. Defaults to likely if one is not specified	Optional
group_by	Show Which fields the y-axis shows on the chart. Can be any field in the opportunity module, defaults to Sales Stage	Optional
ranges	Pipeline or Committed are valid values.	Optional

Input Example:

{ 'user_id':'seed_max_id, 'timeperiod_id':'36f7085a-5889-ea75-84c8-50f42bd1a5ba', 'display_manager':'false', 'group_by': 'forecast', 'dataset': 'likely', 'ranges': 'include', }

Output Example:

{ "color" : ["#8c2b2b", "#468c2b", "#2b5d8c", "#cd5200", "#e6bf00", "#7f3acd", "#00a9b8", "#572323", "#004d00", "#000087", "#e48d30", "#9fba09", "#560066", "#009f92", "#b36262", "#38795c", "#3D3D99", "#99623d", "#998a3d", "#994e78", "#3d6899", "#CC0000", "#00CC00", "#0000CC", "#cc5200", "#ccaa00", "#6600cc", "#005fcc"], "label" : ["Include"], "properties" : [{ "gauge_target_list" : "Array", "goal_marker_color" : ["#000000", "#7D12B2"], "goal_marker_label" : ["Quota", "Likely Case"], "goal_marker_type" : ["group", "pareto"], "label_name" : "Include in Forecast", "labels" : "value", "legend" : "on", "print" : "on", "subtitle" : "", "thousands" : "", "title" : null, "type" : "bar chart", "value_name" : "Likely Case" }], "values" : [{ "goalmarkervalue" : ["111000.00", "36000.00"], "goalmarkervaluelabel" : ["$111,000.00", "$36,000.00"], "gvalue" : 36000, "gvaluelabel" : "$36,000.00", "label" : "January 2013", "links" : [""], "valuelabels" : ["$36,000.00"], "values" : [36000] }, { "goalmarkervalue" : ["111000.00", "61500.00"], "goalmarkervaluelabel" : ["$111,000.00", "$61,500.00"], "gvalue" : 25500, "gvaluelabel" : "$25,500.00", "label" : "February 2013", "links" : [""], "valuelabels" : ["$25,500.00"], "values" : [25500] }, { "goalmarkervalue" : ["111000.00", "61500.00"], "goalmarkervaluelabel" : ["$111,000.00", "$61,500.00"], "gvalue" : 0, "gvaluelabel" : "$0.00", "label" : "March 2013", "links" : [""], "valuelabels" : ["$0.00"], "values" : [0] }] }

/ExpressionEngine/:record/related POST

Retrieve a Forecasting Information In SugarChart Format

Summary:

This endpoint is used to return the json Data for SugarCharts to use to display the needed chart.

Query Parameters:

Param	Description	Optional
timeperiod_id	Show for a specific time period	
user_id	Show for a specific user	
display_manager	Pipeline or Committed are	

	valid values.	
dataset	Which Forecast dataset to show, valid values are likely, best, worst. Defaults to likely if one is not specified	Optional
group_by	Show Which fields the y-axis shows on the chart. Can be any field in the opportunity module, defaults to Sales Stage	Optional
ranges	Pipeline or Committed are valid values.	Optional

Input Example:

{ 'user_id':'seed_max_id, 'timeperiod_id':'36f7085a-5889-ea75-84c8-50f42bd1a5ba', 'display_manager':'false', 'group_by': 'forecast', 'dataset': 'likely', 'ranges': 'include', }

Output Example:

{ "color" : ["#8c2b2b", "#468c2b", "#2b5d8c", "#cd5200", "#e6bf00", "#7f3acd", "#00a9b8", "#572323", "#004d00", "#000087", "#e48d30", "#9fba09", "#560066", "#009f92", "#b36262", "#38795c", "#3D3D99", "#99623d", "#998a3d", "#994e78", "#3d6899", "#CC0000", "#00CC00", "#0000CC", "#cc5200", "#ccaa00", "#6600cc", "#005fcc"], "label" : ["Include"], "properties" : [{ "gauge_target_list" : "Array", "goal_marker_color" : ["#000000", "#7D12B2"], "goal_marker_label" : ["Quota", "Likely Case"], "goal_marker_type" : ["group", "pareto"], "label_name" : "Include in Forecast", "labels" : "value", "legend" : "on", "print" : "on", "subtitle" : "", "thousands" : "", "title" : null, "type" : "bar chart", "value_name" : "Likely Case" }], "values" : [{ "goalmarkervalue" : ["111000.00", "36000.00"], "goalmarkervaluelabel" : ["$111,000.00", "$36,000.00"], "gvalue" : 36000, "gvaluelabel" : "$36,000.00", "label" : "January 2013", "links" : [""], "valuelabels" : ["$36,000.00"], "values" : [36000] }, { "goalmarkervalue" : ["111000.00", "61500.00"], "goalmarkervaluelabel" : ["$111,000.00", "$61,500.00"], "gvalue" : 25500, "gvaluelabel" : "$25,500.00", "label" : "February 2013", "links" : [""], "valuelabels" : ["$25,500.00"], "values" : [25500] }, { "goalmarkervalue" : ["111000.00", "61500.00"], "goalmarkervaluelabel" : ["$111,000.00", "$61,500.00"], "gvalue" : 0, "gvaluelabel" : "$0.00", "label" : "March 2013", "links"

: [""], "valuelabels" : ["$0.00"], "values" : [0] }] }

/Filters GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

Name	Type	Description	Required
filter	String	The filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query parameters in one of two ways for GET requests: 1. By specifying individual filter arguments as distinct parameters. Example: filter[0][id]=1. 2. By	False

Name	Type	Description	Required
		specifying the whole filter as a single JSON-encoded string. Note that this syntax is currently not supported on certain modules. Example: filter=[{"id":"1"}].	
filter_id	String	Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND.	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
fields	String	Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters (applicable to link and collection fields). The fields id	False

Name	Type	Description	Required
		and date_modified will always be returned. Example: name,account_type,description,{"name":"opportunities","fields":["id","name","sales_status"],"order_by":"date_closed:desc"} For more details on additional field parameters, see Relate API and Collection API.	
view	String	Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list". Example: record	False
order_by	String	How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon. Example: name:DESC,account_type:D	False

Name	Type	Description	Required
		ESC,date_modified: ASC	
q	String	A search expression, will search on this module. Cannot be used at the same time as a filter expression or id.	False
deleted	Boolean	Boolean to show deleted records in the result set.	False
nulls_last	Boolean	Boolean to return records with null values in order_by fields last in the result set.	False

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
   "filter":[
      {
         "name":"Nelson Inc"
      }
   ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
   "filter":[
      {
         "name":{
            "$starts":"Nelson"
         }
      }
   ]
}
```

Below is a list of operation types:

Operation	Description
$equals	Performs an exact match on that field.
$not_equals	Performs an exact match on that field.
$not_equals	Matches on non-matching values.
$starts	Matches on anything that starts with the value.
$ends	Matches anything that ends with the value.
$contains	Matches anything that contains the value
$in	Finds anything where field matches one of the values as specified as an array.
$not_in	Finds anything where field does not matches any of the values as specified as an array.
$is_null	Checks if the field is null. This operation does not need a value specified.

Operation		Description
	$not_null	Checks if the field is not null. This operation does not need a value specified.
	$lt	Matches when the field is less than the value.
	$lte	Matches when the field is less than or equal to the value.
	$gt	Matches when the field is greater than the value.
	$gte	Matches when the field is greater than or equal to the value.

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "$and" and "$or".

Example

```
{
   "filter":[
      {
         "$or":[
            {
               "name":"Nelson Inc"
            },
            {
               "name":"Nelson LLC"
            }
         ]
      }
   ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "$favorite" and "$owner".

Example

```
{
   "filter":[
      {
         "$favorite":"_this"
      }
   ]
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Response

```
{
   "next_offset":-1,
   "records":[
      {
         "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
         "name":"Dale Spivey",
         "date_modified":"2013-02-28T05:03:00+00:00",
```

```
         "description":"",
         "opportunities": [
            {
               _module: "Opportunities",
               "id": "b0701501-1fab-8ae7-3942-540da93f5017",
               "name": "360 Vacations - 228 Units",
               "date_modified": "2014-09-08T16:05:00+03:00",
               "sales_status": "New"
            },
         ],
         "_acl": {
            "fields": {
            }
         }
      },
      {
         "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
         "name":"Florence Haddock",
         "date_modified":"2013-02-26T19:12:00+00:00",
         "description":"",
         "opportunities": [
            {
               _module: "Opportunities"
               date_modified: "2014-09-08T16:05:00+03:00"
               id: "9ce7c088-8ee4-7cd3-18f1-540da944d4c0"
               name: "360 Vacations - 312 Units"
               sales_status: "New"
            },
         ],
         "_acl": {
            "fields": {
            }
         }
      }
   ]
}
```

Change Log

Version	Change
v10	Added /<module>/filter GET endpoint.

/ForecastManagerWorksheets GET

Returns a collection of ForecastManagerWorksheet models

Summary:

This end point is used to return the ManagerWorksheets for a user at a given timeperiod. If no timeperiod is provide it will use the default timeperiod, likewise if no user is provided, it will default to the logged in user.

Url Parameters:

Param	Description	Optional
timeperiod_id	Show for a specific time period, defaults to the current time period if one is not passed	Optional
user_id	Show for a specific user, defaults to current user if not defined	Optional

Possible Errors

Error	Description
412 - Invalid Parameter	When the passed in timeperiod and/or user can not be found
403 - Not Authorized	If you are not an administrator, but you are trying to view another managers forecast manager worksheet, you will receive a 403 Not Authorized Error

Url Example:

/rest/v10/ForecastManagerWorksheets/:timeperiod_id/:user_id

Output Example:

{ "next_offset":-1, "records":[{ "id":"401594f5-5b46-fb66-1627-55771fe8723e",
"name":"Sally Bronsen", "date_modified":"2015-06-09T13:13:26-04:00",
"created_by":"1", "quota":"1932.444445", "best_case":"29848.000000",
"best_case_adjusted":"37310.000000", "likely_case":"28694.444445",
"likely_case_adjusted":"35868.055556", "worst_case":"27540.888889",
"worst_case_adjusted":"34426.111111", "timeperiod_id":"adb78e81-3fbd-
b4e0-287f-55771fd04a06", "draft":true, "is_manager":false,
"user_id":"seed_sally_id", "opp_count":10, "pipeline_opp_count":3,
"pipeline_amount":"2415.555556", "closed_amount":"26278.888889",
"manager_saved":true, "show_history_log":0, "following":false,
"assigned_user_id":"seed_sarah_id", "assigned_user_name":"Sarah Smith",
"team_name":[{ "id":"1", "name":"Global", "name_2":"", "primary":true }],
"currency_id":"-99", "base_rate":"1.000000", "_acl":{ "fields":{ } },
"_module":"ForecastManagerWorksheets" }, {
"id":"28226f12-a3c9-bd84-041e-55771f064c4e", "name":"Max Jensen",
"date_modified":"2015-06-09T13:13:26-04:00", "created_by":"1",
"quota":"3141.333332", "best_case":"15848.222223",
"best_case_adjusted":"13206.851853", "likely_case":"14928.222222",
"likely_case_adjusted":"12440.185185", "worst_case":"14008.222223",
"worst_case_adjusted":"11673.518519", "timeperiod_id":"adb78e81-3fbd-
b4e0-287f-55771fd04a06", "draft":true, "is_manager":false,
"user_id":"seed_max_id", "opp_count":12, "pipeline_opp_count":3,
"pipeline_amount":"2617.777777", "closed_amount":"12310.444445",
"manager_saved":true, "show_history_log":0, "following":false,
"assigned_user_id":"seed_sarah_id", "assigned_user_name":"Sarah Smith",
"team_name":[{ "id":"1", "name":"Global", "name_2":"", "primary":true }],
"currency_id":"-99", "base_rate":"1.000000", "_acl":{ "fields":{ } },
"_module":"ForecastManagerWorksheets" }, { "id":"1aca66af-
f069-bba4-28a1-55771fe27506", "name":"",
"date_modified":"2015-06-09T13:13:26-04:00", "created_by":"1",
"quota":"10142.333333", "best_case":"37625.000000",
"best_case_adjusted":"37625.000000", "likely_case":"34241.333333",
"likely_case_adjusted":"34241.333333", "worst_case":"30857.666667",
"worst_case_adjusted":"30857.666667", "timeperiod_id":"adb78e81-3fbd-
b4e0-287f-55771fd04a06", "draft":true, "is_manager":true,
"user_id":"seed_sarah_id", "opp_count":13, "pipeline_opp_count":6,
"pipeline_amount":"10142.333333", "closed_amount":"24099.000000",
"manager_saved":true, "show_history_log":0, "following":false,
"assigned_user_id":"seed_sarah_id", "assigned_user_name":"Sarah Smith",
"team_name":[{ "id":"1", "name":"Global", "name_2":"", "primary":true }],
"currency_id":"-99", "base_rate":"1.000000", "_acl":{ "fields":{ } },
"_module":"ForecastManagerWorksheets" }] }

/ForecastManagerWorksheets/:timeperiod_id GET

Returns a collection of ForecastManagerWorksheet models

Summary:

This end point is used to return the ManagerWorksheets for a user at a given timeperiod. If no timeperiod is provide it will use the default timeperiod, likewise if no user is provided, it will default to the logged in user.

Url Parameters:

Param	Description	Optional
timeperiod_id	Show for a specific time period, defaults to the current time period if one is not passed	Optional
user_id	Show for a specific user, defaults to current user if not defined	Optional

Possible Errors

Error	Description
412 - Invalid Parameter	When the passed in timeperiod and/or user can not be found
403 - Not Authorized	If you are not an administrator, but you are trying to view another managers forecast manager worksheet, you will receive a 403 Not Authorized Error

Url Example:

/rest/v10/ForecastManagerWorksheets/:timeperiod_id/:user_id

Output Example:

{ "next_offset":-1, "records":[{ "id":"401594f5-5b46-fb66-1627-55771fe8723e",
"name":"Sally Bronsen", "date_modified":"2015-06-09T13:13:26-04:00",
"created_by":"1", "quota":"1932.444445", "best_case":"29848.000000",
"best_case_adjusted":"37310.000000", "likely_case":"28694.444445",
"likely_case_adjusted":"35868.055556", "worst_case":"27540.888889",
"worst_case_adjusted":"34426.111111", "timeperiod_id":"adb78e81-3fbd-
b4e0-287f-55771fd04a06", "draft":true, "is_manager":false,
"user_id":"seed_sally_id", "opp_count":10, "pipeline_opp_count":3,
"pipeline_amount":"2415.555556", "closed_amount":"26278.888889",
"manager_saved":true, "show_history_log":0, "following":false,
"assigned_user_id":"seed_sarah_id", "assigned_user_name":"Sarah Smith",
"team_name":[{ "id":"1", "name":"Global", "name_2":"", "primary":true }],
"currency_id":"-99", "base_rate":"1.000000", "_acl":{ "fields":{ } },
"_module":"ForecastManagerWorksheets" }, {
"id":"28226f12-a3c9-bd84-041e-55771f064c4e", "name":"Max Jensen",
"date_modified":"2015-06-09T13:13:26-04:00", "created_by":"1",
"quota":"3141.333332", "best_case":"15848.222223",
"best_case_adjusted":"13206.851853", "likely_case":"14928.222222",
"likely_case_adjusted":"12440.185185", "worst_case":"14008.222223",
"worst_case_adjusted":"11673.518519", "timeperiod_id":"adb78e81-3fbd-
b4e0-287f-55771fd04a06", "draft":true, "is_manager":false,
"user_id":"seed_max_id", "opp_count":12, "pipeline_opp_count":3,
"pipeline_amount":"2617.777777", "closed_amount":"12310.444445",
"manager_saved":true, "show_history_log":0, "following":false,
"assigned_user_id":"seed_sarah_id", "assigned_user_name":"Sarah Smith",
"team_name":[{ "id":"1", "name":"Global", "name_2":"", "primary":true }],
"currency_id":"-99", "base_rate":"1.000000", "_acl":{ "fields":{ } },
"_module":"ForecastManagerWorksheets" }, { "id":"1aca66af-
f069-bba4-28a1-55771fe27506", "name":"",
"date_modified":"2015-06-09T13:13:26-04:00", "created_by":"1",
"quota":"10142.333333", "best_case":"37625.000000",
"best_case_adjusted":"37625.000000", "likely_case":"34241.333333",
"likely_case_adjusted":"34241.333333", "worst_case":"30857.666667",
"worst_case_adjusted":"30857.666667", "timeperiod_id":"adb78e81-3fbd-
b4e0-287f-55771fd04a06", "draft":true, "is_manager":true,
"user_id":"seed_sarah_id", "opp_count":13, "pipeline_opp_count":6,
"pipeline_amount":"10142.333333", "closed_amount":"24099.000000",
"manager_saved":true, "show_history_log":0, "following":false,
"assigned_user_id":"seed_sarah_id", "assigned_user_name":"Sarah Smith",
"team_name":[{ "id":"1", "name":"Global", "name_2":"", "primary":true }],
"currency_id":"-99", "base_rate":"1.000000", "_acl":{ "fields":{ } },
"_module":"ForecastManagerWorksheets" }] }

/ForecastManagerWorksheets/:timeperiod_id/:user_id GET

Returns a collection of ForecastManagerWorksheet models

Summary:

This end point is used to return the ManagerWorksheets for a user at a given timeperiod. If no timeperiod is provide it will use the default timeperiod, likewise if no user is provided, it will default to the logged in user.

Url Parameters:

Param	Description	Optional
timeperiod_id	Show for a specific time period, defaults to the current time period if one is not passed	Optional
user_id	Show for a specific user, defaults to current user if not defined	Optional

Possible Errors

Error	Description
412 - Invalid Parameter	When the passed in timeperiod and/or user can not be found
403 - Not Authorized	If you are not an administrator, but you are trying to view another managers forecast manager worksheet, you will receive a 403 Not Authorized Error

Url Example:

/rest/v10/ForecastManagerWorksheets/:timeperiod_id/:user_id

Output Example:

{ "next_offset":-1, "records":[{ "id":"401594f5-5b46-fb66-1627-55771fe8723e", "name":"Sally Bronsen", "date_modified":"2015-06-09T13:13:26-04:00", "created_by":"1", "quota":"1932.444445", "best_case":"29848.000000", "best_case_adjusted":"37310.000000", "likely_case":"28694.444445", "likely_case_adjusted":"35868.055556", "worst_case":"27540.888889", "worst_case_adjusted":"34426.111111", "timeperiod_id":"adb78e81-3fbd-b4e0-287f-55771fd04a06", "draft":true, "is_manager":false, "user_id":"seed_sally_id", "opp_count":10, "pipeline_opp_count":3, "pipeline_amount":"2415.555556", "closed_amount":"26278.888889", "manager_saved":true, "show_history_log":0, "following":false, "assigned_user_id":"seed_sarah_id", "assigned_user_name":"Sarah Smith", "team_name":[{ "id":"1", "name":"Global", "name_2":"", "primary":true }], "currency_id":"-99", "base_rate":"1.000000", "_acl":{ "fields":{ } }, "_module":"ForecastManagerWorksheets" }, { "id":"28226f12-a3c9-bd84-041e-55771f064c4e", "name":"Max Jensen", "date_modified":"2015-06-09T13:13:26-04:00", "created_by":"1", "quota":"3141.333332", "best_case":"15848.222223", "best_case_adjusted":"13206.851853", "likely_case":"14928.222222", "likely_case_adjusted":"12440.185185", "worst_case":"14008.222223", "worst_case_adjusted":"11673.518519", "timeperiod_id":"adb78e81-3fbd-b4e0-287f-55771fd04a06", "draft":true, "is_manager":false, "user_id":"seed_max_id", "opp_count":12, "pipeline_opp_count":3, "pipeline_amount":"2617.777777", "closed_amount":"12310.444445", "manager_saved":true, "show_history_log":0, "following":false, "assigned_user_id":"seed_sarah_id", "assigned_user_name":"Sarah Smith", "team_name":[{ "id":"1", "name":"Global", "name_2":"", "primary":true }], "currency_id":"-99", "base_rate":"1.000000", "_acl":{ "fields":{ } }, "_module":"ForecastManagerWorksheets" }, { "id":"1aca66af-f069-bba4-28a1-55771fe27506", "name":"", "date_modified":"2015-06-09T13:13:26-04:00", "created_by":"1", "quota":"10142.333333", "best_case":"37625.000000", "best_case_adjusted":"37625.000000", "likely_case":"34241.333333", "likely_case_adjusted":"34241.333333", "worst_case":"30857.666667", "worst_case_adjusted":"30857.666667", "timeperiod_id":"adb78e81-3fbd-b4e0-287f-55771fd04a06", "draft":true, "is_manager":true, "user_id":"seed_sarah_id", "opp_count":13, "pipeline_opp_count":6, "pipeline_amount":"10142.333333", "closed_amount":"24099.000000", "manager_saved":true, "show_history_log":0, "following":false, "assigned_user_id":"seed_sarah_id", "assigned_user_name":"Sarah Smith", "team_name":[{ "id":"1", "name":"Global", "name_2":"", "primary":true }],

"currency_id":"-99", "base_rate":"1.000000", "_acl":{ "fields":{ } },
"_module":"ForecastManagerWorksheets" }] }

/ForecastManagerWorksheets/assignQuota POST

Assign Quotas to All Reporting Users for a given timeperiod

Summary:

This endpoint is used to assign out the quota to the reporting users with out having to commit a forecast.

Parameters:

Param	Description	Optional
user_id	The Manager's user id for who is assigning out the quota	false
timeperiod_id	Which timeperiod are we assigning quota for	false

Input Example:

{ "user_id" : "seed_sarah_id", "timeperiod_id" :
"36f7085a-5889-ea75-84c8-50f42bd1a5ba" }

Output Example:

{success: true}

/ForecastManagerWorksheets/export GET

Overview

Returns a record set in CSV format along with HTTP headers to indicate content

type.

Request Arguments

Name	Type	Description	Required
uid	Array	A list of bean ids.	False
filter	Array	The filter expression. More information on filter expressions can be found in /<module>/filter.	False
sample	Array	Indicates whether to download sample data.	False

Request

Exporting Records by Specific IDs

```
{
   "uid":"d43243c6-9b8e-2973-aee2-512d09bc34b4"
}
```

Exporting Records by a List of IDs

```
{
   "uid":[
      "d43243c6-9b8e-2973-aee2-512d09bc34b4",
      "b3e87a3f-cd8f-7b86-467a-512d09e8d240"
   ]
}
```

Exporting Records Using a Filter

```
{
   "filter":[
      {
         "name":"airline"
      }
```

```
   ]
}
```

Exporting a Sample Result Set

```
{
    "sample":true
}
```

Response Arguments

Name	Type	Description
<csv export>	String	Returns the selected records in CSV format with the content headers.

Response

```
result: HTTP/1.1 200 OK
Date: Fri, 01 Mar 2013 04:05:55 GMT
Server: Apache/2.2.22 (Unix) DAV/2 PHP/5.3.17
X-Powered-By: PHP/5.3.17
Expires: Mon, 26 Jul 1997 05:00:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-
check=0
Pragma: cache
Content-Disposition: attachment; filename=Accounts.csv
Content-transfer-encoding: binary
Last-Modified: Fri, 01 Mar 2013 04:05:55 GMT
Cache-Control: post-check=0, pre-check=0
Content-Length: 1080
Connection: close
Content-Type: application/octet-stream; charset=UTF-8

"Name","ID","Website","Email Address","Office Phone","Alternate Phone"
,"Fax","Billing Street","Billing City","Billing State","Billing Postal
 Code","Billing Country","Shipping Street","Shipping City","Shipping S
tate","Shipping Postal Code","Shipping Country","Description","Type","
Industry","Annual Revenue","Employees","SIC Code","Ticker Symbol","Par
ent Account ID","Ownership","Campaign ID","Rating","Assigned User Name
```

","Assigned To","Team ID","Teams","Team Set ID","Date Created","Date M
odified","Modified By","Created By","Deleted","Image","last_activity_d
ate","Linkedin Company ID","Facebook Account","Twitter Account","Googl
e Plus ID"
"Arts & Crafts Inc","d43243c6-9b8e-2973-aee2-512d09bc34b4","","","(052
) 034-1853","","","777 West Filmore Ln","Santa Monica","CA","35354","U
SA","777 West Filmore Ln","Santa Monica","CA","35354","USA","","Custom
er","Transportation","","","","","","","","","sally","seed_sally_id","
West","West","West","02/26/2013 07:12 pm","02/26/2013 07:12 pm","1","1
","0","","02/26/2013 07:12 pm","","","",""

Change Log

Version	Change
v10	Added /<module>/export GET endpoint.

/ForecastManagerWorksheets/filter GET

Returns a collection of ForecastManagerWorksheet models

Summary:

This end point is used to return selective ManagerWorksheets with the optional filter parameter.

Query Parameters:

Param	Description	Optional
filter	What you want to filter on	Optional

Possible Errors

Error	Description

412 - Invalid Parameter	When the passed in timeperiod and/or user can not be found
403 - Not Authorized	If you are not an administrator, but you are trying to view another managers forecast manager worksheet, you will receive a 403 Not Authorized Error

Url Example:

/rest/v10/ForecastManagerWorksheets/filter { "filter":[{"assigned_user_id" : "seed_jim_id"}, {"timeperiod_id":"e546185a-5889-ea75-84c8-511178d1a5ba"}], }

Output Example:

{ "next_offset": -1, "records": [{ "id": "ad3908c7-83a3-f360-c223-51117844c208", "name": "Grow-Fast Inc - 1000 units", "date_entered": "2013-02-05T21:22:00+00:00", "date_modified": "2013-02-05T21:22:00+00:00", "modified_user_id": "1", "modified_by_name": "Administrator", "created_by": "1", "created_by_name": "Administrator", "description": "", "img": "", "deleted": "0", "assigned_user_id": "seed_jim_id", "assigned_user_name": "Jim Brennan", "team_name": [{ "id": "East", "name": "East", "name_2": "", "primary": true }], "parent_id": "50b90565-e748-ed77-d9d7-511178f5acae", "parent_type": "Opportunities", "account_name": "", "account_id": "", "likely_case": "75000", "best_case": "75000", "worst_case": "75000", "base_rate": "1", "currency_id": "-99", "currency_name": "", "currency_symbol": "", "date_closed": "2013-02-10", "date_closed_timestamp": "1360531443", "sales_stage": "Perception Analysis", "probability": "70", "commit_stage": "include", "draft": "1", "my_favorite": false, "_acl": { "fields": {} } }] }

/ForecastManagerWorksheets/filter POST

Returns a collection of ForecastManagerWorksheet models

Summary:

This end point is used to return selective ManagerWorksheets with the optional filter parameter.

Query Parameters:

Param	Description	Optional
filter	What you want to filter on	Optional

Possible Errors

Error	Description
412 - Invalid Parameter	When the passed in timeperiod and/or user can not be found
403 - Not Authorized	If you are not an administrator, but you are trying to view another managers forecast manager worksheet, you will receive a 403 Not Authorized Error

Url Example:

/rest/v10/ForecastManagerWorksheets/filter { "filter":[{"assigned_user_id" : "seed_jim_id"}, {"timeperiod_id":"e546185a-5889-ea75-84c8-511178d1a5ba"}], }

Output Example:

{ "next_offset": -1, "records": [{ "id": "ad3908c7-83a3-f360-c223-51117844c208", "name": "Grow-Fast Inc - 1000 units", "date_entered": "2013-02-05T21:22:00+00:00", "date_modified": "2013-02-05T21:22:00+00:00", "modified_user_id": "1", "modified_by_name": "Administrator", "created_by": "1", "created_by_name": "Administrator", "description": "", "img": "", "deleted": "0", "assigned_user_id": "seed_jim_id", "assigned_user_name": "Jim Brennan", "team_name": [{ "id": "East", "name": "East", "name_2": "", "primary": true }], "parent_id": "50b90565-e748-ed77-d9d7-511178f5acae", "parent_type": "Opportunities", "account_name": "", "account_id": "", "likely_case": "75000", "best_case": "75000", "worst_case": "75000", "base_rate": "1", "currency_id": "-99", "currency_name": "", "currency_symbol": "", "date_closed": "2013-02-10", "date_closed_timestamp": "1360531443", "sales_stage": "Perception Analysis", "probability": "70", "commit_stage": "include", "draft": "1", "my_favorite": false, "_acl": { "fields": {} } }] }

/ForecastWorksheets GET

Returns a collection of ForecastWorksheet models

Summary:

This end point is used to return the Worksheets for a user at a given timeperiod. If no timeperiod is provide it will use the default timeperiod, likewise if no user is provided, it will default to the logged in user.

Also this end points use the visibility requirements set forth by the application, that if the records being requested belong to you, you will get the draft version, If the worksheet is not yours, you will only see the committed version.

Url Parameters:

Param	Description	Optional
timeperiod_id	Show for a specific time period, defaults to the current time period if one is not passed	Optional
user_id	Show for a specific user, defaults to current user if not defined	Optional

Query Parameters:

Param	Description	Optional
type	Which type to return, currently the ones that are supported are, opportunity and product	Optional

Possible Errors

Error	Description
412 - Invalid Parameter	When the passed in timeperiod and/or

	user can not be found
403 - Not Authorized	If you are not a manager, but you are tyring to view another sales rep forecast worksheet, you will receive a 403 Not Authorized Error

Url Example:

/rest/v10/ForecastWorksheets/:timeperiod_id/:user_id

Output Example:

{ "next_offset": -1, "records": [{ "id": "ad3908c7-83a3-f360-c223-51117844c208", "name": "Grow-Fast Inc - 1000 units", "date_entered": "2013-02-05T21:22:00+00:00", "date_modified": "2013-02-05T21:22:00+00:00", "modified_user_id": "1", "modified_by_name": "Administrator", "created_by": "1", "created_by_name": "Administrator", "description": "", "img": "", "deleted": "0", "assigned_user_id": "seed_jim_id", "assigned_user_name": "Jim Brennan", "team_name": [{ "id": "East", "name": "East", "name_2": "", "primary": true }], "parent_id": "50b90565-e748-ed77-d9d7-511178f5acae", "parent_type": "Opportunities", "account_name": "", "account_id": "", "likely_case": "75000", "best_case": "75000", "worst_case": "75000", "base_rate": "1", "currency_id": "-99", "currency_name": "", "currency_symbol": "", "date_closed": "2013-02-10", "date_closed_timestamp": "1360531443", "sales_stage": "Perception Analysis", "probability": "70", "commit_stage": "include", "draft": "1", "my_favorite": false, "_acl": { "fields": {} } }] }

/ForecastWorksheets/:record PUT

Saves a collection of ForecastWorksheet models

Summary:

This endpoint is used to save the json Data for an array of worksheet data array entries

Query Parameters:

Param	Description	Optional
This endpoint does not accept any parameters.		

Input Example:

{ "amount" : "10000.000000", "assigned_user_id" : "seed_max_id", "base_rate" : "1", "best_case" : "25000", "commit_stage" : "include", "currency_id" : "-99", "current_user" : "seed_max_id", "date_closed" : "2013-01-14", "date_modified" : "2013-01-14T10:58:27-05:00", "draft" : 1, "id" : "347beb60-d57c-b3aa-5922-50f42b6c27d4", "likely_case" : "15000", "name" : "RR. Talker Co - 1000 units", "probability" : "90", "product_id" : "34b0d547-adaf-03ea-146c-50f42b3e6f04", "sales_stage" : "Value Proposition", "timeperiod_id" : "36f7085a-5889-ea75-84c8-50f42bd1a5ba", "version" : 1, "w_date_modified" : "2013-01-14T10:58:27-05:00", "worksheet_id" : "e0adb532-7d1c-eb49-b102-50f42b455164", "worst_case" : "10000.000000" }

Output Example:

{ "amount" : "15000.000000", "assigned_user_id" : "seed_max_id", "base_rate" : "1", "best_case" : "25000.000000", "commit_stage" : "include", "currency_id" : "-99", "date_closed" : "2013-01-14", "date_modified" : "2013-01-15T10:52:31-05:00", "id" : "347beb60-d57c-b3aa-5922-50f42b6c27d4", "likely_case" : "15000.000000", "name" : "RR. Talker Co - 1000 units", "probability" : "90", "product_id" : "34b0d547-adaf-03ea-146c-50f42b3e6f04", "sales_stage" : "Value Proposition", "version" : 1, "w_date_modified" : "2013-01-14T10:58:27-05:00", "worksheet_id" : "e0adb532-7d1c-eb49-b102-50f42b455164", "worst_case" : "10000.000000" }

/ForecastWorksheets/:timeperiod_id GET

Returns a collection of ForecastWorksheet models

Summary:

This end point is used to return the Worksheets for a user at a given timeperiod. If no timeperiod is provide it will use the default timeperiod, likewise if no user is

provided, it will default to the logged in user.

Also this end points use the visibility requirements set forth by the application, that if the records being requested belong to you, you will get the draft version, If the worksheet is not yours, you will only see the committed version.

Url Parameters:

Param	Description	Optional
timeperiod_id	Show for a specific time period, defaults to the current time period if one is not passed	Optional
user_id	Show for a specific user, defaults to current user if not defined	Optional

Query Parameters:

Param	Description	Optional
type	Which type to return, currently the ones that are supported are, opportunity and product	Optional

Possible Errors

Error	Description
412 - Invalid Parameter	When the passed in timeperiod and/or user can not be found
403 - Not Authorized	If you are not a manager, but you are tyring to view another sales rep forecast worksheet, you will receive a 403 Not Authorized Error

Url Example:

/rest/v10/ForecastWorksheets/:timeperiod_id/:user_id

Output Example:

{ "next_offset": -1, "records": [{ "id": "ad3908c7-83a3-f360-c223-51117844c208", "name": "Grow-Fast Inc - 1000 units", "date_entered": "2013-02-05T21:22:00+00:00", "date_modified": "2013-02-05T21:22:00+00:00", "modified_user_id": "1", "modified_by_name": "Administrator", "created_by": "1", "created_by_name": "Administrator", "description": "", "img": "", "deleted": "0", "assigned_user_id": "seed_jim_id", "assigned_user_name": "Jim Brennan", "team_name": [{ "id": "East", "name": "East", "name_2": "", "primary": true }], "parent_id": "50b90565-e748-ed77-d9d7-511178f5acae", "parent_type": "Opportunities", "account_name": "", "account_id": "", "likely_case": "75000", "best_case": "75000", "worst_case": "75000", "base_rate": "1", "currency_id": "-99", "currency_name": "", "currency_symbol": "", "date_closed": "2013-02-10", "date_closed_timestamp": "1360531443", "sales_stage": "Perception Analysis", "probability": "70", "commit_stage": "include", "draft": "1", "my_favorite": false, "_acl": { "fields": {} } }] }

/ForecastWorksheets/:timeperiod_id/:user_id GET

Returns a collection of ForecastWorksheet models

Summary:

This end point is used to return the Worksheets for a user at a given timeperiod. If no timeperiod is provide it will use the default timeperiod, likewise if no user is provided, it will default to the logged in user.

Also this end points use the visibility requirements set forth by the application, that if the records being requested belong to you, you will get the draft version, If the worksheet is not yours, you will only see the committed version.

Url Parameters:

Param	Description	Optional
timeperiod_id	Show for a specific time	Optional

	period, defaults to the current time period if one is not passed	
user_id	Show for a specific user, defaults to current user if not defined	Optional

Query Parameters:

Param	Description	Optional
type	Which type to return, currently the ones that are supported are, opportunity and product	Optional

Possible Errors

Error	Description
412 - Invalid Parameter	When the passed in timeperiod and/or user can not be found
403 - Not Authorized	If you are not a manager, but you are tyring to view another sales rep forecast worksheet, you will receive a 403 Not Authorized Error

Url Example:

/rest/v10/ForecastWorksheets/:timeperiod_id/:user_id

Output Example:

{ "next_offset": -1, "records": [{ "id": "ad3908c7-83a3-f360-c223-51117844c208", "name": "Grow-Fast Inc - 1000 units", "date_entered": "2013-02-05T21:22:00+00:00", "date_modified": "2013-02-05T21:22:00+00:00", "modified_user_id": "1", "modified_by_name": "Administrator", "created_by": "1", "created_by_name": "Administrator", "description": "", "img": "", "deleted": "0", "assigned_user_id": "seed_jim_id", "assigned_user_name": "Jim Brennan", "team_name": [{ "id": "East", "name": "East", "name_2": "", "primary": true }], "parent_id": "50b90565-e748-ed77-d9d7-511178f5acae", "parent_type":

"Opportunities", "account_name": "", "account_id": "", "likely_case": "75000", "best_case": "75000", "worst_case": "75000", "base_rate": "1", "currency_id": "-99", "currency_name": "", "currency_symbol": "", "date_closed": "2013-02-10", "date_closed_timestamp": "1360531443", "sales_stage": "Perception Analysis", "probability": "70", "commit_stage": "include", "draft": "1", "my_favorite": false, "_acl": { "fields": {} } }] }

/ForecastWorksheets/export GET

Overview

Returns a record set in CSV format along with HTTP headers to indicate content type.

Request Arguments

Name	Type	Description	Required
uid	Array	A list of bean ids.	False
filter	Array	The filter expression. More information on filter expressions can be found in /<module>/filter.	False
sample	Array	Indicates whether to download sample data.	False

Request

Exporting Records by Specific IDs

```
{
   "uid":"d43243c6-9b8e-2973-aee2-512d09bc34b4"
}
```

Exporting Records by a List of IDs

```
{
   "uid":[
      "d43243c6-9b8e-2973-aee2-512d09bc34b4",
      "b3e87a3f-cd8f-7b86-467a-512d09e8d240"
```

```
   ]
}
```

Exporting Records Using a Filter

```
{
   "filter":[
      {
         "name":"airline"
      }
   ]
}
```

Exporting a Sample Result Set

```
{
    "sample":true
}
```

Response Arguments

Name	Type	Description
<csv export>	String	Returns the selected records in CSV format with the content headers.

Response

```
result: HTTP/1.1 200 OK
Date: Fri, 01 Mar 2013 04:05:55 GMT
Server: Apache/2.2.22 (Unix) DAV/2 PHP/5.3.17
X-Powered-By: PHP/5.3.17
Expires: Mon, 26 Jul 1997 05:00:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-
check=0
Pragma: cache
Content-Disposition: attachment; filename=Accounts.csv
```

```
Content-transfer-encoding: binary
Last-Modified: Fri, 01 Mar 2013 04:05:55 GMT
Cache-Control: post-check=0, pre-check=0
Content-Length: 1080
Connection: close
Content-Type: application/octet-stream; charset=UTF-8

"Name","ID","Website","Email Address","Office Phone","Alternate Phone"
,"Fax","Billing Street","Billing City","Billing State","Billing Postal
 Code","Billing Country","Shipping Street","Shipping City","Shipping S
tate","Shipping Postal Code","Shipping Country","Description","Type","
Industry","Annual Revenue","Employees","SIC Code","Ticker Symbol","Par
ent Account ID","Ownership","Campaign ID","Rating","Assigned User Name
","Assigned To","Team ID","Teams","Team Set ID","Date Created","Date M
odified","Modified By","Created By","Deleted","Image","last_activity_d
ate","Linkedin Company ID","Facebook Account","Twitter Account","Googl
e Plus ID"
"Arts & Crafts Inc","d43243c6-9b8e-2973-aee2-512d09bc34b4","","","(052
) 034-1853","","","777 West Filmore Ln","Santa Monica","CA","35354","U
SA","777 West Filmore Ln","Santa Monica","CA","35354","USA","","Custom
er","Transportation","","","","","","","","","sally","seed_sally_id","
West","West","West","02/26/2013 07:12 pm","02/26/2013 07:12 pm","1","1
","0","","02/26/2013 07:12 pm","","","",""
```

Change Log

Version	Change
v10	Added /<module>/export GET endpoint.

/ForecastWorksheets/filter GET

Returns a collection of ForecastWorksheet models

Summary:

This end point is used to return the Worksheets for a user at a given timeperiod. If no timeperiod is provide it will use the default timeperiod, likewise if no user is provided, it will default to the logged in user.

Also this end points use the visibility requirements set forth by the application, that if the records being requested belong to you, you will get the draft version, If the worksheet is not yours, you will only see the committed version.

Query Parameters:

Param	Description	Optional
filter	What you want to filter on	Optional
type	Which type to return, currently the ones that are supported are, opportunity and product	Optional

Possible Errors

Error	Description
412 - Invalid Parameter	When the passed in timeperiod and/or user can not be found
403 - Not Authorized	If you are not a manager, but you are tyring to view another sales rep forecast worksheet, you will receive a 403 Not Authorized Error

Url Example:

/rest/v10/ForecastWorksheets/:timeperiod_id/:user_id { "filter":[{"assigned_user_id" : "seed_jim_id"}, {"timeperiod_id":"e546185a-5889-ea75-84c8-511178d1a5ba"}], }

Output Example:

{ "next_offset": -1, "records": [{ "id": "ad3908c7-83a3-f360-c223-51117844c208", "name": "Grow-Fast Inc - 1000 units", "date_entered": "2013-02-05T21:22:00+00:00", "date_modified": "2013-02-05T21:22:00+00:00", "modified_user_id": "1", "modified_by_name": "Administrator", "created_by": "1",

"created_by_name": "Administrator", "description": "", "img": "", "deleted": "0", "assigned_user_id": "seed_jim_id", "assigned_user_name": "Jim Brennan", "team_name": [{ "id": "East", "name": "East", "name_2": "", "primary": true }], "parent_id": "50b90565-e748-ed77-d9d7-511178f5acae", "parent_type": "Opportunities", "account_name": "", "account_id": "", "likely_case": "75000", "best_case": "75000", "worst_case": "75000", "base_rate": "1", "currency_id": "-99", "currency_name": "", "currency_symbol": "", "date_closed": "2013-02-10", "date_closed_timestamp": "1360531443", "sales_stage": "Perception Analysis", "probability": "70", "commit_stage": "include", "draft": "1", "my_favorite": false, "_acl": { "fields": {} } }] }

/ForecastWorksheets/filter POST

Returns a collection of ForecastWorksheet models

Summary:

This end point is used to return the Worksheets for a user at a given timeperiod. If no timeperiod is provide it will use the default timeperiod, likewise if no user is provided, it will default to the logged in user.

Also this end points use the visibility requirements set forth by the application, that if the records being requested belong to you, you will get the draft version, If the worksheet is not yours, you will only see the committed version.

Query Parameters:

Param	Description	Optional
filter	What you want to filter on	Optional
type	Which type to return, currently the ones that are supported are, opportunity and product	Optional

Possible Errors

Error	Description
412 - Invalid Parameter	When the passed in timeperiod and/or

	user can not be found
403 - Not Authorized	If you are not a manager, but you are tyring to view another sales rep forecast worksheet, you will receive a 403 Not Authorized Error

Url Example:

/rest/v10/ForecastWorksheets/:timeperiod_id/:user_id { "filter":[{"assigned_user_id" : "seed_jim_id"}, {"timeperiod_id":"e546185a-5889-ea75-84c8-511178d1a5ba"}], }

Output Example:

{ "next_offset": -1, "records": [{ "id": "ad3908c7-83a3-f360-c223-51117844c208", "name": "Grow-Fast Inc - 1000 units", "date_entered": "2013-02-05T21:22:00+00:00", "date_modified": "2013-02-05T21:22:00+00:00", "modified_user_id": "1", "modified_by_name": "Administrator", "created_by": "1", "created_by_name": "Administrator", "description": "", "img": "", "deleted": "0", "assigned_user_id": "seed_jim_id", "assigned_user_name": "Jim Brennan", "team_name": [{ "id": "East", "name": "East", "name_2": "", "primary": true }], "parent_id": "50b90565-e748-ed77-d9d7-511178f5acae", "parent_type": "Opportunities", "account_name": "", "account_id": "", "likely_case": "75000", "best_case": "75000", "worst_case": "75000", "base_rate": "1", "currency_id": "-99", "currency_name": "", "currency_symbol": "", "date_closed": "2013-02-10", "date_closed_timestamp": "1360531443", "sales_stage": "Perception Analysis", "probability": "70", "commit_stage": "include", "draft": "1", "my_favorite": false, "_acl": { "fields": {} } }] }

/Forecasts GET

Overview

Updates a record of a specified type as a favorite for the current user.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

Name	Type	Description
<record field>	<record field type>	Returns the fields for the selected record.

Response


```
{
   "id":"bdd59d85-687b-1739-b00a-512d09f6db9e",
   "name":"Insight Marketing Inc",
   "date_entered":"2013-02-26T19:12:00+00:00",
   "date_modified":"2013-02-26T19:12:00+00:00",
   "modified_user_id":"1",
   "modified_by_name":"Administrator",
   "created_by":"1",
   "created_by_name":"Administrator",
   "description":"",
   "img":"",
   "last_activity_date":"2013-02-26T19:12:00+00:00",
   "deleted":false,
   "assigned_user_id":"seed_max_id",
   "assigned_user_name":"Max Jensen",
   "team_name":[
      {
         "id":"East",
         "name":"East",
         "name_2":"",
         "primary":false
      },
      {
         "id":1,
         "name":"Global",
         "name_2":"",
         "primary":false
      },
      {
         "id":"West",
         "name":"West",
         "name_2":"",
         "primary":true
      }
   ],
   "linkedin":"",
   "facebook":"",
   "twitter":"",
   "googleplus":"",
```

```
"account_type":"Customer",
"industry":"Electronics",
"annual_revenue":"",
"phone_fax":"",
"billing_address_street":"345 Sugar Blvd.",
"billing_address_street_2":"",
"billing_address_street_3":"",
"billing_address_street_4":"",
"billing_address_city":"San Mateo",
"billing_address_state":"CA",
"billing_address_postalcode":"56019",
"billing_address_country":"USA",
"rating":"",
"phone_office":"(927) 136-9572",
"phone_alternate":"",
"website":"www.sectionvegan.de",
"ownership":"",
"employees":"",
"ticker_symbol":"",
"shipping_address_street":"345 Sugar Blvd.",
"shipping_address_street_2":"",
"shipping_address_street_3":"",
"shipping_address_street_4":"",
"shipping_address_city":"San Mateo",
"shipping_address_state":"CA",
"shipping_address_postalcode":"56019",
"shipping_address_country":"USA",
"email1":"kid.support.vegan@example.info",
"parent_id":"",
"sic_code":"",
"parent_name":"",
"email_opt_out":false,
"invalid_email":false,
"email":[
   {
      "email_address":"kid.support.vegan@example.info",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"1"
   },
   {
      "email_address":"phone.kid@example.cn",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"0"
   }
```

```
   ],
   "campaign_id":"",
   "campaign_name":"",
   "my_favorite":true,
   "_acl":{
      "fields":{

      }
   }
}
```

Change Log

Version	Change
v10	Added /<module>/:record/favorite PUT endpoint.

/Forecasts/:timeperiod_id/:user_id/chart/:display_manager GET

Retrieve a Forecasting Information In SugarChart Format

Summary:

This endpoint is used to return the json Data for SugarCharts to use to display the needed chart.

Query Parameters:

Param	Description	Optional
timeperiod_id	Show for a specific time period	
user_id	Show for a specific user	
display_manager	Pipeline or Committed are valid values.	
dataset	Which Forecast dataset to	Optional

	show, valid values are likely, best, worst. Defaults to likely if one is not specified	
group_by	Show Which fields the y-axis shows on the chart. Can be any field in the opportunity module, defaults to Sales Stage	Optional
ranges	Pipeline or Committed are valid values.	Optional

Input Example:

{ 'user_id':'seed_max_id, 'timeperiod_id':'36f7085a-5889-ea75-84c8-50f42bd1a5ba', 'display_manager':'false', 'group_by': 'forecast', 'dataset': 'likely', 'ranges': 'include', }

Output Example:

{ "color" : ["#8c2b2b", "#468c2b", "#2b5d8c", "#cd5200", "#e6bf00", "#7f3acd", "#00a9b8", "#572323", "#004d00", "#000087", "#e48d30", "#9fba09", "#560066", "#009f92", "#b36262", "#38795c", "#3D3D99", "#99623d", "#998a3d", "#994e78", "#3d6899", "#CC0000", "#00CC00", "#0000CC", "#cc5200", "#ccaa00", "#6600cc", "#005fcc"], "label" : ["Include"], "properties" : [{ "gauge_target_list" : "Array", "goal_marker_color" : ["#000000", "#7D12B2"], "goal_marker_label" : ["Quota", "Likely Case"], "goal_marker_type" : ["group", "pareto"], "label_name" : "Include in Forecast", "labels" : "value", "legend" : "on", "print" : "on", "subtitle" : "", "thousands" : "", "title" : null, "type" : "bar chart", "value_name" : "Likely Case" }], "values" : [{ "goalmarkervalue" : ["111000.00", "36000.00"], "goalmarkervaluelabel" : ["$111,000.00", "$36,000.00"], "gvalue" : 36000, "gvaluelabel" : "$36,000.00", "label" : "January 2013", "links" : [""], "valuelabels" : ["$36,000.00"], "values" : [36000] }, { "goalmarkervalue" : ["111000.00", "61500.00"], "goalmarkervaluelabel" : ["$111,000.00", "$61,500.00"], "gvalue" : 25500, "gvaluelabel" : "$25,500.00", "label" : "February 2013", "links" : [""], "valuelabels" : ["$25,500.00"], "values" : [25500] }, { "goalmarkervalue" : ["111000.00", "61500.00"], "goalmarkervaluelabel" : ["$111,000.00", "$61,500.00"], "gvalue" : 0, "gvaluelabel" : "$0.00", "label" : "March 2013", "links" : [""], "valuelabels" : ["$0.00"], "values" : [0] }] }

/Forecasts/:timeperiod_id/progressManager/:user_id GET

Projected Manager Data

Summary:

This endpoint is used to return the json Data not already in client view for the Forecasts manager projected panel.

Query Parameters:

Param	Description	Optional
user_id	Show for a specific user, defaults to current user if not defined	Optional
timeperiod_id	Show for a specific time period, defaults to the current time period if one is not passed	Optional

Input Example:

{ user_id:seed_sally_id timeperiod_id:36f7085a-5889-ea75-84c8-50f42bd1a5ba }

Output Example:

{ closed_amount: 0 opportunities: 13 pipeline_revenue: 470000 quota_amount: 382000.000000 }

/Forecasts/:timeperiod_id/progressRep/:user_id GET

Projected Rep Data

Summary:

This endpoint is used to return the json Data not already in client view for the Forecasts rep projected panel.

Query Parameters:

Param	Description	Optional
user_id	Show for a specific user, defaults to current user if not defined	Optional
timeperiod_id	Show for a specific time period, defaults to the current time period if one is not passed	Optional

Input Example:

{ user_id:seed_sally_id timeperiod_id:36f7085a-5889-ea75-84c8-50f42bd1a5ba }

Output Example:

{ quota_amount: "80000.000000" }

/Forecasts/:timeperiod_id/quotas/direct/:user_id GET

ForecastsQuotasApi - Get

Summary:

This endpoint is used to return quota information for a specific user ID, in a specific timeperiod and by a specifc type..

Query Parameters:

Param	Description	Optional
timeperiod_id	Show for a specific time period	Required
quota_type	"rollup" or "direct" quota	Required
user_id	Show for a specific user	Required

Input Example:

Output Example:

{ "currency_id":"-99", "amount":"75.000000", "date_modified":"2013-09-17 21:23:32", "formatted_amount":"$75.00", "is_top_level_manager":true }

/Forecasts/:timeperiod_id/quotas/rollup/:user_id GET

ForecastsQuotasApi - Get

Summary:

This endpoint is used to return quota information for a specific user ID, in a specific timeperiod and by a specifc type..

Query Parameters:

Param	Description	Optional
timeperiod_id	Show for a specific time period	Required

quota_type	"rollup" or "direct" quota	Required
user_id	Show for a specific user	Required

Input Example:

Output Example:

{ "currency_id":"-99", "amount":"75.000000", "date_modified":"2013-09-17 21:23:32", "formatted_amount":"$75.00", "is_top_level_manager":true }

/Forecasts/config POST

Creates and/or updates the config settings for the Forecasts module

Summary:

This endpoint is used to create and/or update the config settings for the Forecasts module as json data. It extends the core config endpoint by adding the functionality to create the Timeperiods and updates existing Opportunities for use in the Forecasts module.

Query Parameters:

Param	Description	Optional

Input Example:

{ "is_setup": 1, "is_upgrade": 0, "has_commits": 1, "timeperiod_type": "chronological", "timeperiod_interval": "Annual", "timeperiod_leaf_interval": "Quarter", "timeperiod_start_date": "2013-01-01", "timeperiod_shown_forward": 2, "timeperiod_shown_backward": 2, "forecast_ranges": "show_binary",

"buckets_dom": "commit_stage_binary_dom", "show_binary_ranges": { "include": { "min": 70, "max": 100 }, "exclude": { "min": 0, "max": 69 } }, "show_buckets_ranges": { "include": { "min": 85, "max": 100 }, "upside": { "min": 70, "max": 84 }, "exclude": { "min": 0, "max": 69 } }, "sales_stage_won": ["Closed Won"], "sales_stage_lost": ["Closed Lost"], "show_worksheet_likely": 1, "show_worksheet_best": 1, "show_worksheet_worst": 0, "show_projected_likely": 1, "show_projected_best": 1, "show_projected_worst": 0, "show_forecasts_commit_warnings": 1 }

Output Example:

{ "is_setup": 1, "is_upgrade": 0, "has_commits": 1, "timeperiod_type": "chronological", "timeperiod_interval": "Annual", "timeperiod_leaf_interval": "Quarter", "timeperiod_start_date": "2013-01-01", "timeperiod_shown_forward": 2, "timeperiod_shown_backward": 2, "forecast_ranges": "show_binary", "buckets_dom": "commit_stage_binary_dom", "show_binary_ranges": { "include": { "min": 70, "max": 100 }, "exclude": { "min": 0, "max": 69 } }, "show_buckets_ranges": { "include": { "min": 85, "max": 100 }, "upside": { "min": 70, "max": 84 }, "exclude": { "min": 0, "max": 69 } }, "sales_stage_won": ["Closed Won"], "sales_stage_lost": ["Closed Lost"], "show_worksheet_likely": 1, "show_worksheet_best": 1, "show_worksheet_worst": 0, "show_projected_likely": 1, "show_projected_best": 1, "show_projected_worst": 0, "show_forecasts_commit_warnings": 1 }

/Forecasts/config PUT

Creates and/or updates the config settings for the Forecasts module

Summary:

This endpoint is used to create and/or update the config settings for the Forecasts module as json data. It extends the core config endpoint by adding the functionality to create the Timeperiods and updates existing Opportunities for use in the Forecasts module.

Query Parameters:

Param	Description	Optional

Input Example:

{ "is_setup": 1, "is_upgrade": 0, "has_commits": 1, "timeperiod_type": "chronological", "timeperiod_interval": "Annual", "timeperiod_leaf_interval": "Quarter", "timeperiod_start_date": "2013-01-01", "timeperiod_shown_forward": 2, "timeperiod_shown_backward": 2, "forecast_ranges": "show_binary", "buckets_dom": "commit_stage_binary_dom", "show_binary_ranges": { "include": { "min": 70, "max": 100 }, "exclude": { "min": 0, "max": 69 } }, "show_buckets_ranges": { "include": { "min": 85, "max": 100 }, "upside": { "min": 70, "max": 84 }, "exclude": { "min": 0, "max": 69 } }, "sales_stage_won": ["Closed Won"], "sales_stage_lost": ["Closed Lost"], "show_worksheet_likely": 1, "show_worksheet_best": 1, "show_worksheet_worst": 0, "show_projected_likely": 1, "show_projected_best": 1, "show_projected_worst": 0, "show_forecasts_commit_warnings": 1 }

Output Example:

{ "is_setup": 1, "is_upgrade": 0, "has_commits": 1, "timeperiod_type": "chronological", "timeperiod_interval": "Annual", "timeperiod_leaf_interval": "Quarter", "timeperiod_start_date": "2013-01-01", "timeperiod_shown_forward": 2, "timeperiod_shown_backward": 2, "forecast_ranges": "show_binary", "buckets_dom": "commit_stage_binary_dom", "show_binary_ranges": { "include": { "min": 70, "max": 100 }, "exclude": { "min": 0, "max": 69 } }, "show_buckets_ranges": { "include": { "min": 85, "max": 100 }, "upside": { "min": 70, "max": 84 }, "exclude": { "min": 0, "max": 69 } }, "sales_stage_won": ["Closed Won"], "sales_stage_lost": ["Closed Lost"], "show_worksheet_likely": 1, "show_worksheet_best": 1, "show_worksheet_worst": 0, "show_projected_likely": 1, "show_projected_best": 1, "show_projected_worst": 0, "show_forecasts_commit_warnings": 1 }

/Forecasts/enum/selectedTimePeriod GET

ForecastsApi Timeperiod filter info

Summary:

This returns the timeperiods with respect to the forward/backward options selected. For example, if we elect to use quarterly timeperiods and 2 forward and 2 backward timeperiod intervals are set, then 5 years worth of timeperiods will be returned (2 backward + current + 2 forward).

Query Parameters:

Param	Description	Optional
This endpoint does not accept any parameters.		

Input Example:

Output Example:

{ "d181230a-a7e0-3176-d10f-50f8530a51ce" : "Q1 (01/01/2012 - 03/31/2012)",
"d2bc58ef-21c4-27d5-e416-50f853db29f9" : "Q2 (04/01/2012 - 06/30/2012)",
"d3db853e-94d6-b5ac-98af-50f8530689be" : "Q3 (07/01/2012 - 09/30/2012)",
"d519f521-5303-1cfc-24f5-50f8537f5b54" : "Q4 (10/01/2012 - 12/31/2012)",
"dcc2885a-5889-ea75-84c8-50f853d1a5ba" : "Q1 (01/01/2013 - 03/31/2013)",
"ddc24224-c6c9-04aa-7869-50f853db2ea2" : "Q2 (04/01/2013 - 06/30/2013)",
"deadcfd3-41ed-5f6b-ce5b-50f853de6ef6" : "Q3 (07/01/2013 - 09/30/2013)",
"dfaa3724-6295-8ddb-6d14-50f853047fcc" : "Q4 (10/01/2013 - 12/31/2013)",
"e1ceee19-c8dd-afdd-e797-50f8538d900a" : "Q1 (01/01/2014 - 03/31/2014)",
"e307594a-c12a-6be6-74b7-50f85351c574" : "Q2 (04/01/2014 - 06/30/2014)",
"e469239c-3f71-a48e-cda1-50f853214b6a" : "Q3 (07/01/2014 - 09/30/2014)",
"e57f4889-cefd-4356-6d82-50f85350decd" : "Q4 (10/01/2014 - 12/31/2014)" }

/Forecasts/init GET

ForecastsApi - init

Summary:

This endpoint is used to return initialization data for the Forecasts module.

Query Parameters:

Param	Description	Optional
This endpoint does not accept any parameters.		

Input Example:

Output Example:

{ "initData": { "selectedUser": { "preferences": { "timezone":"GMT", "datepref":"m\/d\/Y", "timepref":"h:ia", "currency_id":"-99", "currency_name":"US Dollars", "currency_symbol":"$", "currency_iso":"USD", "currency_rate":1, "decimal_precision":"2", "decimal_separator":".", "number_grouping_separator":",", "language":"en_us", "default_teams": [{ "id":1, "display_name":"Global", "name":"Global", "name_2":"", "primary":true }] }, "module_list": ["Home", "Accounts", "Contacts", "Opportunities", "Leads","Calendar", "Reports", "Quotes", "Documents", "Emails", "Campaigns", "Calls", "Meetings", "Tasks", "Notes", "Forecasts", "Cases", "Prospects", "ProspectLists", "Bugs", "Employees"], "type":"user", "id":"seed_jim_id", "full_name":"Jim Brennan", "user_name":"jim", "picture":"46d1fbfb-c258-ce81-fa3a-50f809925709", "acl": { "Forecasts":{"fields":[] ,"admin":"no","_hash":"095777549714234c8192a7a7963c38b8"}, "ForecastWorksh eets":{"fields":[],"admin":"no","_hash":"095777549714234c8192a7a7963c38b8"}, " ForecastManagerWorksheets":{"fields":[],"admin":"no","_hash":"09577754971423 4c8192a7a7963c38b8"}, }, "my_teams": [{"id":"8833dfef- c1f4-e8b3-ca3b-50f8093616be","name":"Chris Olliver"}, {"id":"East","name":"East"},{"id":1,"name":"Global"}, {"id":"4ae6baba- f356-7374-7eb4-50f809745551","name":"Jim Brennan"}, {"id":"707a642a-710b-37f1-88f6-50f8098205c0","name":"Max Jensen"}, {"id":"63308ab9-f663-bae8-90a5-50f80951c0fe","name":"Sally Bronsen"}, {"id":"5776c705-8e6e-4bfc-44e5-50f809d12c0e","name":"Sarah Smith"}, {"id":"West","name":"West"}, {"id":"7cf7ee0b-f88e- b566-d4d3-50f809dcb717","name":"Will Westin"}], "showOpps":false, "first_name":"Jim", "last_name":"Brennan", "admin":"yes" }, "forecasts_setup":1 }, "defaultSelections": { "timeperiod_id": { "id":"a2ab9ad3-2101-36f8-7ba3-50f809b3b62c", "label":"Q1 (01\/01\/2013 - 03\/31\/2013)" }, "ranges":["include"], "group_by":"forecast", "dataset":"likely" } } }

/Forecasts/reportees/:user_id GET

ForecastsApi Reportees

Summary:

This endpoint is used to return the json Data for an array of users and their state for the forecasts users tree filter

Query Parameters:

Param	Description	Optional
user_id	Show for a specific user, defaults to current user if not defined	Optional
level	Level of child notes, defaults to 1, -1 for all levels	Optional

Input Example:

{ 'user_id':'seed_max_id' }

Output Example:

{ "attr":{ "id":"jstree_node_jim", "rel":"root" }, "children":[{ "attr":{ "id":"jstree_node_myopps_jim", "rel":"my_opportunities" }, "children":[ ], "data":"Opportunities (Jim Brennan)", "metadata":{ "first_name":"Jim", "full_name":"Jim Brennan", "id":"seed_jim_id", "last_name":"Brennan", "level":"1", "reports_to_id":"", "user_name":"jim" }, "state":"" }, { "attr":{ "id":"jstree_node_will", "rel":"manager" }, "children":[ ], "data":"Will Westin", "metadata":{ "first_name":"Will", "full_name":"Will Westin", "id":"seed_will_id", "last_name":"Westin", "level":"2", "reports_to_id":"seed_jim_id", "user_name":"will" }, "state":"closed" }, { "attr":{ "id":"jstree_node_sarah", "rel":"manager" }, "children":[ ], "data":"Sarah Smith", "metadata":{ "first_name":"Sarah", "full_name":"Sarah Smith", "id":"seed_sarah_id", "last_name":"Smith", "level":"2", "reports_to_id":"seed_jim_id", "user_name":"sarah" }, "state":"closed" }], "data":"Jim Brennan", "metadata":{ "first_name":"Jim", "full_name":"Jim Brennan",

"id":"seed_jim_id", "last_name":"Brennan", "level":"1", "reports_to_id":"", "user_name":"jim" }, "state":"open" }

/Forecasts/user/:user_id GET

ForecastsApi - user

Summary:

This endpoint is used to return a user's id, user_name, full_name, first_name, last_name, and is_manager param given a user's id.

Query Parameters:

Param	Description	Optional
user_id	Returns user data for this specific user id	

Input Example:

{ user_id:seed_sally_id }

Output Example:

{ first_name: "Sally" full_name: "Sally Bronsen" id: "seed_sally_id" is_manager: false last_name: "Bronsen" user_name: "sally" }

/Genericsearch GET

Overview

Generic search.

Summary

This endpoint searches the content of the given modules using the provider specified by the "generic_search" configuration variable. If the variable is absent, the default provider is "Elastic".

Request Arguments

Name	Type	Description	Required
q	String	The search expression.	True
module_list	String	Comma-delimited list of modules to search. If omitted, all search enabled modules will be queried. Example: KBDocuments,Bugs	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
total	Integer	The number of records found.
records	Array	An array of results containing matched records.

Response

```
  {
    "next_offset": -1,
    "total": 1,
    "records": [
        {
            "name": "Connecting to the Internet",
            "description": "To connect your device to the Internet, us
e any application that accesses the Internet. You can connect using ei
ther Wi-Fi or Bluetooth.",
            "url": "portal/index.html#KBContents/60142236-bad3-11e9-9d
32-3c15c2d57622" (for portal)
            "url": "#KBContents/60142236-bad3-11e9-9d32-3c15c2d57622"
(for base)
        }
    ]
}
```


Change Log

Version	Change
v11_9	Added /Genericsearch GET endpoint.

/Genericsearch POST

Overview

Generic search.

Summary

This endpoint searches the content of the given modules using the provider specified by the "generic_search" configuration variable. If the variable is absent, the default provider is "Elastic".

Request Arguments

Name	Type	Description	Required
q	String	The search expression.	True
module_list	String	Comma-delimited list of modules to	False

Name	Type	Description	Required
		search. If omitted, all search enabled modules will be queried. Example: KBDocuments,Bugs	
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
total	Integer	The number of records found.
records	Array	An array of results containing matched records.

Response


```
  {
    "next_offset": -1,
    "total": 1,
    "records": [
        {
            "name": "Connecting to the Internet",
            "description": "To connect your device to the Internet, us
e any application that accesses the Internet. You can connect using ei
ther Wi-Fi or Bluetooth.",
            "url": "portal/index.html#KBContents/60142236-bad3-11e9-9d
32-3c15c2d57622" (for portal)
```

```
            "url": "#KBContents/60142236-bad3-11e9-9d32-3c15c2d57622"
(for base)
        }
    ]
}
```

Change Log

Version	Change
v11_9	Added /Genericsearch POST endpoint.

/KBContents GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

Name	Type	Description	Required
filter	String	The filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query parameters in one of two ways for GET requests:	False

Name	Type	Description	Required
		1. By specifying individual filter arguments as distinct parameters. Example: filter[0][id]=1. 2. By specifying the whole filter as a single JSON-encoded string. Note that this syntax is currently not supported on certain modules. Example: filter=[{"id":"1"}].	
filter_id	String	Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND.	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
fields	String	Comma delimited	False

Name	Type	Description	Required
		list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters (applicable to link and collection fields). The fields id and date_modified will always be returned. Example: name,account_type,description,{"name":"opportunities","fields":["id","name","sales_status"],"order_by":"date_closed:desc"} For more details on additional field parameters, see Relate API and Collection API.	
view	String	Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list".	False

Name	Type	Description	Required
		Example: record	
order_by	String	How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon. Example: name:DESC,account_type:DESC,date_modified:ASC	False
q	String	A search expression, will search on this module. Cannot be used at the same time as a filter expression or id.	False
deleted	Boolean	Boolean to show deleted records in the result set.	False
nulls_last	Boolean	Boolean to return records with null values in order_by fields last in the result set.	False

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
   "filter":[
      {
         "name":"Nelson Inc"
      }
```

```
   ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
   "filter":[
      {
         "name":{
            "$starts":"Nelson"
         }
      }
   ]
}
```

Below is a list of operation types:

Operation	Description
$equals	Performs an exact match on that field.
$not_equals	Performs an exact match on that field.
$not_equals	Matches on non-matching values.
$starts	Matches on anything that starts with the value.
$ends	Matches anything that ends with the value.
$contains	Matches anything that contains the value
$in	Finds anything where field matches one of the values as

Operation		Description
		specified as an array.
	$not_in	Finds anything where field does not matches any of the values as specified as an array.
	$is_null	Checks if the field is null. This operation does not need a value specified.
	$not_null	Checks if the field is not null. This operation does not need a value specified.
	$lt	Matches when the field is less than the value.
	$lte	Matches when the field is less than or equal to the value.
	$gt	Matches when the field is greater than the value.
	$gte	Matches when the field is greater than or equal to the value.

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "$and" and "$or".

Example

```
{
   "filter":[
      {
         "$or":[
```

```
            {
               "name":"Nelson Inc"
            },
            {
               "name":"Nelson LLC"
            }
         ]
      }
   ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "$favorite" and "$owner".

Example

```
{
   "filter":[
      {
         "$favorite":"_this"
      }
   ]
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results

Name	Type	Description
		containing matched records.

Response

```
{
   "next_offset":-1,
   "records":[
      {
         "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
         "name":"Dale Spivey",
         "date_modified":"2013-02-28T05:03:00+00:00",
         "description":"",
         "opportunities": [
            {
               _module: "Opportunities",
               "id": "b0701501-1fab-8ae7-3942-540da93f5017",
               "name": "360 Vacations - 228 Units",
               "date_modified": "2014-09-08T16:05:00+03:00",
               "sales_status": "New"
            },
         ],
         "_acl": {
            "fields": {
            }
         }
      },
      {
         "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
         "name":"Florence Haddock",
         "date_modified":"2013-02-26T19:12:00+00:00",
         "description":"",
         "opportunities": [
            {
               _module: "Opportunities"
               date_modified: "2014-09-08T16:05:00+03:00"
               id: "9ce7c088-8ee4-7cd3-18f1-540da944d4c0"
               name: "360 Vacations - 312 Units"
               sales_status: "New"
            },
         ],
         "_acl": {
            "fields": {
            }
```

```
        }
      }
   ]
}
```

Change Log

Version	Change
v10	Added /<module>/filter GET endpoint.

/KBContents/:record/link POST

Overview

Creates relationships to a pre-existing record.

Request Arguments

Name	Type	Description	Required
<relationship link>	<string>	Link between targeted and related records.	True
<record ID>	<string>	The name value list of related records. Each item of the list may be either string equal to related item ID, or map containing record ID ("id" key is required in this case) and addition relationship properties.	True

Request

```
{
    link_name: "accounts"
    ids: [
        "da6a3741-2a81-ba7f-f249-512d0932e94e",
```

```
        {
            "id": "e689173e-c953-1e14-c215-512d0927e7a2",
            "role": "owner"
        }
    ]
}
```

Response Arguments

Name	Type	Description
record	Array	The record linked to related records.
related_records	Array	Records that were associated.

Response

```
    "record": {
        "id": "da6a3741-2a81-ba7f-f249-512d0932e94e",
        "name": "Slender Broadband Inc - 1000 units",
        "date_entered": "2013-02-26T19:12:00+00:00",
        "date_modified": "2013-02-26T19:12:00+00:00",
        "modified_user_id": "1",
        "modified_by_name": "Administrator",
        "created_by": "1",
        "created_by_name": "Administrator",
        "description": "",
        "img": "",
        "last_activity_date": "2013-02-28T18:21:00+00:00",
        "deleted": false,
        "assigned_user_id": "seed_max_id",
        "assigned_user_name": "Max Jensen",
        "team_name": [
            {
                "id": "East",
                "name": "East",
                "name_2": "",
                "primary": false
            },
            {
                "id": "West",
                "name": "West",
```

```
            "name_2": "",
            "primary": true
        }
    ],
    "opportunity_type": "",
    "account_name": "Slender Broadband Inc",
    "account_id": "181461c6-dc81-1115-1fe0-512d092e8f15",
    "campaign_id": "",
    "campaign_name": "",
    "lead_source": "Campaign",
    "amount": "25000",
    "base_rate": "1",
    "amount_usdollar": "25000",
    "currency_id": "-99",
    "currency_name": "",
    "currency_symbol": "",
    "date_closed": "2013-02-27",
    "date_closed_timestamp": "1361992480",
    "next_step": "",
    "sales_stage": "Needs Analysis",
    "sales_status": "New",
    "probability": "90",
    "best_case": "25000",
    "worst_case": "25000",
    "commit_stage": "include",
    "my_favorite": false,
    "_acl": {
        "fields": {

        }
    }
},
"related_records": [
    {
        "id": "e689173e-c953-1e14-c215-512d0927e7a2",
        "name": "Gus Dales",
        "date_entered": "2013-02-26T19:12:00+00:00",
        "date_modified": "2013-02-26T19:12:00+00:00",
        "modified_user_id": "1",
        "modified_by_name": "Administrator",
        "created_by": "1",
        "created_by_name": "Administrator",
        "description": "",
        "img": "",
        "deleted": false,
        "assigned_user_id": "seed_sally_id",
```

```
            "assigned_user_name": "Sally Bronsen",
            "team_name": [
                {
                    "id": "West",
                    "name": "West",
                    "name_2": "",
                    "primary": true
                }
            ],
            "salutation": "",
            "first_name": "Gus",
            "last_name": "Dales",
            "full_name": "Gus Dales",
            "title": "Director Operations",
            "linkedin": "",
            "facebook": "",
            "twitter": "",
            "googleplus": "",
            "department": "",
            "do_not_call": false,
            "phone_home": "(661) 120-2292",
            "email": [
                {
                    "email_address": "section.sugar.section@example.it
",
                    "opt_out": "1",
                    "invalid_email": "0",
                    "primary_address": "0"
                },
                {
                    "email_address": "support.qa.kid@example.co.uk",
                    "opt_out": "0",
                    "invalid_email": "0",
                    "primary_address": "1"
                }
            ],
            "phone_mobile": "(294) 447-9707",
            "phone_work": "(036) 840-3216",
            "phone_other": "",
            "phone_fax": "",
            "email1": "support.qa.kid@example.co.uk",
            "email2": "section.sugar.section@example.it",
            "invalid_email": false,
            "email_opt_out": false,
            "primary_address_street": "48920 San Carlos Ave",
            "primary_address_street_2": "",
```

```
        "primary_address_street_3": "",
        "primary_address_city": "Persistance",
        "primary_address_state": "CA",
        "primary_address_postalcode": "54556",
        "primary_address_country": "USA",
        "alt_address_street": "",
        "alt_address_street_2": "",
        "alt_address_street_3": "",
        "alt_address_city": "",
        "alt_address_state": "",
        "alt_address_postalcode": "",
        "alt_address_country": "",
        "assistant": "",
        "assistant_phone": "",
        "picture": "",
        "email_and_name1": "",
        "lead_source": "Support Portal User Registration",
        "account_name": "Arts & Crafts Inc",
        "account_id": "d43243c6-9b8e-2973-aee2-512d09bc34b4",
        "opportunity_role_fields": "",
        "opportunity_role_id": "",
        "opportunity_role": "Technical Advisor",
        "reports_to_id": "",
        "report_to_name": "",
        "portal_name": "GusDales145",
        "portal_active": true,
        "portal_password": "$1$JxYr6tmM$b.O6.KF42jP46RadSwz0N0",
        "portal_password1": "",
        "portal_app": "",
        "preferred_language": "en_us",
        "campaign_id": "",
        "campaign_name": "",
        "c_accept_status_fields": "",
        "m_accept_status_fields": "",
        "accept_status_id": "",
        "accept_status_name": "",
        "sync_contact": "",
        "my_favorite": false,
        "_acl": {
            "fields": {
            }
        }
    },
    {
        "id": "da6a3741-2a81-ba7f-f249-512d0932e94e",
        "name": "Slender Broadband Inc - 1000 units",
```

```
"date_entered": "2013-02-26T19:12:00+00:00",
"date_modified": "2013-02-26T19:12:00+00:00",
"modified_user_id": "1",
"modified_by_name": "Administrator",
"created_by": "1",
"created_by_name": "Administrator",
"description": "",
"img": "",
"last_activity_date": "2013-02-28T18:36:00+00:00",
"deleted": false,
"assigned_user_id": "seed_max_id",
"assigned_user_name": "Max Jensen",
"team_name": [
    {
        "id": "East",
        "name": "East",
        "name_2": "",
        "primary": false
    },
    {
        "id": "West",
        "name": "West",
        "name_2": "",
        "primary": true
    }
],
"opportunity_type": "",
"account_name": "Slender Broadband Inc",
"account_id": "181461c6-dc81-1115-1fe0-512d092e8f15",
"campaign_id": "",
"campaign_name": "",
"lead_source": "Campaign",
"amount": "25000",
"base_rate": "1",
"amount_usdollar": "25000",
"currency_id": "-99",
"currency_name": "",
"currency_symbol": "",
"date_closed": "2013-02-27",
"date_closed_timestamp": "1361992480",
"next_step": "",
"sales_stage": "Needs Analysis",
"sales_status": "New",
"probability": "90",
"best_case": "25000",
"worst_case": "25000",
```

```
            "commit_stage": "include",
            "my_favorite": false,
            "_acl": {
                "fields": {

                }
            }
        }
    ]
}
```

Change Log

Version	Change
v10 (7.6.0)	Added support for additional relationship values.
v10 (7.1.5)	Added /<module>/:record/link POST endpoint.

/KBContents/:record/notuseful PUT

Overview

Vote for Knowledge Base article.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

Name	Type	Description
<record field>	<record field type>	Returns the fields for the updated record.

Response

```
{
    "active_date": "2014-01-01",
    "active_rev": 1,
    "approved": false,
```

```
    "assigned_user_id": "seed_will_id",
    "assigned_user_link": {"full_name": "Will Westin", "id": "seed_wil
l_id"},
    "assigned_user_name": "Will Westin",
    "attachment_list": [],
    "cases": {"name": "", "id": ""},
    "category_id": "3e844f1c-5ce1-6d6d-496d-5714901e6666",
    "category_name": "Database",
    "created_by": "1",
    "created_by_link": {"full_name": "Administrator", "id": "1"},
    "created_by_name": "Administrator",
    "date_entered": "2016-04-18T07:43:43+00:00",
    "date_modified": "2016-04-18T07:43:43+00:00",
    "deleted": false,
    "description": "",
    "exp_date": "2014-12-31",
    "file_mime_type": "",
    "following": false,
    "id": "87758ab5-75eb-451f-2bd7-571490d29d7a",
    "is_external": false,
    "kbarticle_id": "8862943c-022e-0385-908e-5714908ac792",
    "kbarticle_name": "Resetting the device",
    "kbarticles_kbcontents": {"name": "Resetting the device", "id": "8
862943c-022e-0385-908e-5714908ac792"},
    "kbdocument_body": "
```


When things are not working as expected...

", "kbdocument_id": "87a20b6b-3172-ad19-0ca5-571490bec118",
"kbdocument_name": "Resetting the device", "kbdocuments_kbcontents": {"name":
"Resetting the device", "id": "87a20b6b-3172-ad19-0ca5-571490bec118"},
"kbsapprover_id": "", "kbsapprover_name": "", "kbsapprovers_kbcontents":
{"full_name": "", "id": ""}, "kbscase_id": "", "kbscase_name": "", "language": "en",
"modified_by_name": "Administrator", "modified_user_id": "1",
"modified_user_link": {"full_name": "Administrator", "id": "1"}, "my_favorite": false,
"name": "Resetting the device", "notuseful": 1 "related_languages": ["en"],
"revision": 1 "status": "in-review", "tag": [], "team_count": "", "team_count_link":
{"team_count": "", "id": "1"}, "team_name": [{"id": 1, "name": "Global", "name_2":
"", "primary": true}], "useful": 0, "usefulness_user_vote": "-1", "viewcount": 4 }

Change Log

Version	Change
v10	Added /KBContents/:record/useful PUT endpoint.

Version	Change
v10	Added /KBContents/:record/notuseful PUT endpoint.

/KBContents/:record/useful PUT

Overview

Vote for Knowledge Base article.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

Name	Type	Description
<record field>	<record field type>	Returns the fields for the updated record.

Response


```
{
    "active_date": "2014-01-01",
    "active_rev": 1,
    "approved": false,
    "assigned_user_id": "seed_will_id",
    "assigned_user_link": {"full_name": "Will Westin", "id": "seed_wil
l_id"},
    "assigned_user_name": "Will Westin",
    "attachment_list": [],
    "cases": {"name": "", "id": ""},
    "category_id": "3e844f1c-5ce1-6d6d-496d-5714901e6666",
    "category_name": "Database",
    "created_by": "1",
    "created_by_link": {"full_name": "Administrator", "id": "1"},
    "created_by_name": "Administrator",
    "date_entered": "2016-04-18T07:43:43+00:00",
    "date_modified": "2016-04-18T07:43:43+00:00",
    "deleted": false,
    "description": "",
    "exp_date": "2014-12-31",
    "file_mime_type": "",
    "following": false,
```

```
    "id": "87758ab5-75eb-451f-2bd7-571490d29d7a",
    "is_external": false,
    "kbarticle_id": "8862943c-022e-0385-908e-5714908ac792",
    "kbarticle_name": "Resetting the device",
    "kbarticles_kbcontents": {"name": "Resetting the device", "id": "8
862943c-022e-0385-908e-5714908ac792"},
    "kbdocument_body": "
```


When things are not working as expected...

", "kbdocument_id": "87a20b6b-3172-ad19-0ca5-571490bec118", "kbdocument_name": "Resetting the device", "kbdocuments_kbcontents": {"name": "Resetting the device", "id": "87a20b6b-3172-ad19-0ca5-571490bec118"}, "kbsapprover_id": "", "kbsapprover_name": "", "kbsapprovers_kbcontents": {"full_name": "", "id": ""}, "kbscase_id": "", "kbscase_name": "", "language": "en", "modified_by_name": "Administrator", "modified_user_id": "1", "modified_user_link": {"full_name": "Administrator", "id": "1"}, "my_favorite": false, "name": "Resetting the device", "notuseful": 1 "related_languages": ["en"], "revision": 1 "status": "in-review", "tag": [], "team_count": "", "team_count_link": {"team_count": "", "id": "1"}, "team_name": [{"id": 1, "name": "Global", "name_2": "", "primary": true}], "useful": 0, "usefulness_user_vote": "-1", "viewcount": 4 }

Change Log

Version	Change
v10	Added /KBContents/:record/useful PUT endpoint.
v10	Added /KBContents/:record/notuseful PUT endpoint.

/KBContents/config POST

Creates and/or updates the config settings for the KBContents module

Summary:

This endpoint is used to create and/or update the config settings for the KBContents module as json data. It extends the core config endpoint by adding the functionality to process deleted languages and documents related to them.

Query Parameters:

Param	Description	Optional

Input Example:

{ "languages":[{ "en":"English", "primary":true },{ "de":"German", "primary":false }], "category_root":"31696245-0438-ff7a-9144-544f8c659daf", "deleted_languages":["es","ru"] }

Output Example:

{ "languages":[{ "en":"English", "primary":true },{ "de":"German", "primary":false }], "category_root":"31696245-0438-ff7a-9144-544f8c659daf", "deleted_languages":["es","ru"] }

/KBContents/config PUT

Creates and/or updates the config settings for the KBContents module

Summary:

This endpoint is used to create and/or update the config settings for the KBContents module as json data. It extends the core config endpoint by adding the functionality to process deleted languages and documents related to them.

Query Parameters:

Param	Description	Optional

Input Example:

{ "languages":[{ "en":"English", "primary":true },{ "de":"German", "primary":false

}], "category_root":"31696245-0438-ff7a-9144-544f8c659daf",
"deleted_languages":["es","ru"] }

Output Example:

{ "languages":[{ "en":"English", "primary":true },{ "de":"German", "primary":false
}], "category_root":"31696245-0438-ff7a-9144-544f8c659daf",
"deleted_languages":["es","ru"] }

/KBContents/duplicateCheck POST

Overview

Runs a duplicate check against specified data.

Request Arguments

Name	Type	Description	Required
<record field>	<record field type>	The name value list of fields to populate.	True

Request

```
{
    "name":"Airline"
}
```

Response Arguments

Name	Type	Description
<record field>	<record field type>	Returns the potential duplicate records.

Response

```
{
   "next_offset":-1,
   "records":[
```

```
{
   "id":"c4c26496-5dbc-67dd-bf5c-512d0923889e",
   "name":"Airline Maintenance Co",
   "date_entered":"2013-02-26T19:12:00+00:00",
   "date_modified":"2013-02-26T19:12:00+00:00",
   "modified_user_id":"1",
   "modified_by_name":"Administrator",
   "created_by":"1",
   "created_by_name":"Administrator",
   "description":"",
   "img":"",
   "last_activity_date":"2013-02-26T19:12:00+00:00",
   "deleted":false,
   "assigned_user_id":"seed_sally_id",
   "assigned_user_name":"Sally Bronsen",
   "team_name":[
      {
         "id":"East",
         "name":"East",
         "name_2":"",
         "primary":false
      },
      {
         "id":1,
         "name":"Global",
         "name_2":"",
         "primary":false
      },
      {
         "id":"West",
         "name":"West",
         "name_2":"",
         "primary":true
      }
   ],
   "linkedin":"",
   "facebook":"",
   "twitter":"",
   "googleplus":"",
   "account_type":"Customer",
   "industry":"Other",
   "annual_revenue":"",
   "phone_fax":"",
   "billing_address_street":"123 Anywhere Street",
   "billing_address_street_2":"",
   "billing_address_street_3":"",
```

```
"billing_address_street_4":"",
"billing_address_city":"Sunnyvale",
"billing_address_state":"CA",
"billing_address_postalcode":"48566",
"billing_address_country":"USA",
"rating":"",
"phone_office":"(648) 452-3486",
"phone_alternate":"",
"website":"www.kidqa.it",
"ownership":"",
"employees":"",
"ticker_symbol":"",
"shipping_address_street":"123 Anywhere Street",
"shipping_address_street_2":"",
"shipping_address_street_3":"",
"shipping_address_street_4":"",
"shipping_address_city":"Sunnyvale",
"shipping_address_state":"CA",
"shipping_address_postalcode":"48566",
"shipping_address_country":"USA",
"email1":"vegan.im@example.tw",
"parent_id":"",
"sic_code":"",
"parent_name":"",
"email_opt_out":false,
"invalid_email":false,
"email":[
   {
      "email_address":"vegan.im@example.tw",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"1"
   },
   {
      "email_address":"phone85@example.tv",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"0"
   }
],
"campaign_id":"",
"campaign_name":"",
"my_favorite":false,
"_acl":{
   "fields":{
```

```
            }
         },
         "duplicate_check_rank":0
      }
   ]
}
```

Change Log

Version	Change
v10	Added /<module>/duplicateCheck POST endpoint.

/KBContents/filter GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

Name	Type	Description	Required
filter	String	The filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as	False

Name	Type	Description	Required
		query parameters in one of two ways for GET requests: 1. By specifying individual filter arguments as distinct parameters. Example: filter[0][id]=1. 2. By specifying the whole filter as a single JSON-encoded string. Note that this syntax is currently not supported on certain modules. Example: filter=[{"id":"1"}].	
filter_id	String	Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND.	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over	False

Name	Type	Description	Required
		before records are returned. Default is 0.	
fields	String	Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters (applicable to link and collection fields). The fields id and date_modified will always be returned. Example: name,account_type,description,{"name":"opportunities","fields":["id","name","sales_status"],"order_by":"date_closed:desc"} For more details on additional field parameters, see Relate API and Collection API.	False
view	String	Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with	False

Name	Type	Description	Required
		the fields argument. Common views are "record" and "list". Example: record	
order_by	String	How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon. Example: name:DESC,account_type:DESC,date_modified:ASC	False
q	String	A search expression, will search on this module. Cannot be used at the same time as a filter expression or id.	False
deleted	Boolean	Boolean to show deleted records in the result set.	False
nulls_last	Boolean	Boolean to return records with null values in order_by fields last in the result set.	False

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

{

```
   "filter":[
      {
         "name":"Nelson Inc"
      }
   ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
   "filter":[
      {
         "name":{
            "$starts":"Nelson"
         }
      }
   ]
}
```

Below is a list of operation types:

Operation	Description
$equals	Performs an exact match on that field.
$not_equals	Performs an exact match on that field.
$not_equals	Matches on non-matching values.
$starts	Matches on anything that starts with the value.
$ends	Matches anything that ends with the value.
$contains	Matches anything that contains the

Operation		Description
		value
	$in	Finds anything where field matches one of the values as specified as an array.
	$not_in	Finds anything where field does not matches any of the values as specified as an array.
	$is_null	Checks if the field is null. This operation does not need a value specified.
	$not_null	Checks if the field is not null. This operation does not need a value specified.
	$lt	Matches when the field is less than the value.
	$lte	Matches when the field is less than or equal to the value.
	$gt	Matches when the field is greater than the value.
	$gte	Matches when the field is greater than or equal to the value.

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "$and" and "$or".

Example

```
{
   "filter":[
      {
         "$or":[
            {
               "name":"Nelson Inc"
            },
            {
               "name":"Nelson LLC"
            }
         ]
      }
   ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "$favorite" and "$owner".

Example

```
{
   "filter":[
      {
         "$favorite":"_this"
      }
   ]
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional

Name	Type	Description
		results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Response

```
{
   "next_offset":-1,
   "records":[
      {
         "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
         "name":"Dale Spivey",
         "date_modified":"2013-02-28T05:03:00+00:00",
         "description":"",
         "opportunities": [
            {
               _module: "Opportunities",
               "id": "b0701501-1fab-8ae7-3942-540da93f5017",
               "name": "360 Vacations - 228 Units",
               "date_modified": "2014-09-08T16:05:00+03:00",
               "sales_status": "New"
            },
         ],
         "_acl": {
            "fields": {
            }
         }
      },
      {
         "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
         "name":"Florence Haddock",
         "date_modified":"2013-02-26T19:12:00+00:00",
         "description":"",
         "opportunities": [
            {
               _module: "Opportunities"
               date_modified: "2014-09-08T16:05:00+03:00"
               id: "9ce7c088-8ee4-7cd3-18f1-540da944d4c0"
               name: "360 Vacations - 312 Units"
               sales_status: "New"
            },
```

```
         ],
         "_acl": {
            "fields": {
            }
         }
      }
   ]
}
```

Change Log

Version	Change
v10	Added /<module>/filter GET endpoint.

/Leads POST

Create Lead with optional post-save actions

Summary:

This endpoint is used to create a Lead with the optional post-save actions for Convert Target/Prospect and Create Lead from Email operations.

Query Parameters:

Param	Description	Optional
prospect_id	Pass a prospect id if Lead is being created as part of a Convert Target/Prospect process	Optional
inbound_email_id	Pass an inbound email id if Lead is being created from an Email	Optional

/Leads/:leadId/convert POST

Convert Lead to a Contact and optionally link it to a new or existing instance of the modules specified

Summary:

This endpoint is used to convert the specified Lead to a Contact and optionally link that Contact to a new or existing instance of the modules specified. The types of modules that can be specified are limited to those that have been configured by the system administrator.

Post Parameters:

Parameter	Description
modules (required)	An object containing the Contacts module to be created as part of the conversion, along with (optionally) any modules that this new Contact record is to be related to. If relate-to modules are also specified, they must consist of a valid module type and either the id of a existing module of that type, or if new, the full record to be created for that type. Note that the allowed module types are defined by the System Administrator.

Example

```
    {        "modules": {            "Contacts": {                "del
eted": "0",                "do_not_call": false,                "porta
l_active": "0",                "preferred_language": "en_us",
       "assigned_user_id": "1",                "assigned_user_name": "
Administrator",                "salutation": "",                "first
_name": "Darius",                "last_name": "Paisley",
  "title": "Director Operations",                "department": "",
           "description": "",                "team_name": [
          {                        "id": "East",
  "name": "East",                        "name_2": "",
       "primary": true                    },                    {
                   "id": 1,                        "name": "Global",
```

```
                        "name_2": "",                           "primary":
 false                    },                   {
   "id": "West",                           "name": "West",
         "name_2": "",                           "primary": false
           }                ],                  "phone_home": "(890) 24
1-5509",                "phone_mobile": "(738) 338-2546",
   "phone_work": "(579) 448-8879",                "phone_fax": "",
           "primary_address_street": "123 Anywhere Street",
      "primary_address_city": "Salt Lake City",                "primar
y_address_state": "CA",                "primary_address_postalcode": "
81738",                "primary_address_country": "USA",
  "campaign_id": "",                "campaign_name": "",
  "email": [                   {                       "email_addres
s": "kid75@example.it",                           "invalid_email": false,
                        "opt_out": false,                           "prim
ary_address": true,                           "reply_to_address": false
                  }                ]            },            "Account
s": {                "id": "1c212ff9-c0d5-866d-d3ae-526e6cd47a31",
            "name": "Lexington Shores Corp",                "date_ente
red": "2013-10-28T09:52:46-04:00",                "date_modified": "20
13-10-28T09:52:46-04:00",                "modified_user_id": "1",
           "modified_by_name": "Administrator",                "create
d_by": "1",                "created_by_name": "Administrator",
        "description": "",                "deleted": false,
     "assigned_user_id": "seed_sally_id",                "assigned_use
r_name": "Sally Bronsen",                "team_count": "",
    "team_name": [                   {                          "id": "
West",                          "name": "West",                          "
name_2": "",                          "primary": true
 }                ],                "linkedin": "",                "fa
cebook": "",                "twitter": "",                "googleplus"
: "",                "account_type": "Customer",                "indus
try": "Electronics",                "annual_revenue": "",
   "phone_fax": "",                "billing_address_street": "111 Sili
con Valley Road",                "billing_address_street_2": "",
          "billing_address_street_3": "",                "billing_addr
ess_street_4": "",                "billing_address_city": "Santa Fe",
               "billing_address_state": "CA",                "billing_
address_postalcode": "63785",                "billing_address_country"
: "USA",                "rating": "",                "phone_office": "
(641) 347-6902",                "phone_alternate": "",
"website": "www.imphone.de",                "ownership": "",
      "employees": "",                "ticker_symbol": "",
    "shipping_address_street": "111 Silicon Valley Road",
   "shipping_address_street_2": "",                "shipping_address_s
treet_3": "",                "shipping_address_street_4": "",
```

```
      "shipping_address_city": "Santa Fe",                "shipping_a
ddress_state": "CA",                "shipping_address_postalcode": "63
785",                "shipping_address_country": "USA",
 "email": [                    {                        "email_address
": "section.qa@example.tw",                        "invalid_email": fa
lse,                        "opt_out": false,                        "
primary_address": true,                        "reply_to_address": fal
se                    },                    {                        "
email_address": "vegan29@example.de",                        "invalid_
email": false,                        "opt_out": false,
         "primary_address": false,                        "reply_to_ad
dress": false                    }                ],                "e
mail1": "section.qa@example.tw",                "parent_id": "",
          "sic_code": "",                "parent_name": "",
     "email_opt_out": false,                "invalid_email": false,
             "campaign_id": "",                "campaign_name": "",
             "my_favorite": false,                "_acl": {
         "fields": {}                },                "following": fa
lse,                "_module": "Accounts",                "duplicate_c
heck_rank": 8            },            "Opportunities": {
   "id": "5529e246-c42f-04e0-1989-526e6d3029c6",                "name"
: "Lexington Shores Corp - 311 Units",                "date_entered":
"2013-10-28T09:52:46-04:00",                "date_modified": "2013-10-
28T09:52:46-04:00",                "modified_user_id": "1",
     "modified_by_name": "Administrator",                "created_by":
 "1",                "created_by_name": "Administrator",
  "description": "",                "deleted": false,                "
assigned_user_id": "seed_sally_id",                "assigned_user_name
": "Sally Bronsen",                "team_count": "",                "t
eam_name": [                    {                        "id": "West",
                        "name": "West",                        "name_2
": "",                        "primary": true                    }
            ],                "opportunity_type": "",                "
account_name": "Lexington Shores Corp",                "account_id": "
1c212ff9-c0d5-866d-d3ae-526e6cd47a31",                "campaign_id": "
",                "campaign_name": "",                "lead_source": "
Partner",                "amount": 10413,                "base_rate":
1,                "amount_usdollar": 10413,                "currency_i
d": "-99",                "currency_name": "",                "currenc
y_symbol": "",                "date_closed": "2014-03-30",
    "date_closed_timestamp": 1396187804,                "next_step": "
",                "sales_stage": "Prospecting",                "sales_
status": "New",                "probability": 10,                "best
_case": 11795,                "worst_case": 9031,                "comm
it_stage": "exclude",                "total_revenue_line_items": 5,
            "closed_revenue_line_items": 1,                "contact_r
```

```
ole": "",                "my_favorite": false,                "_acl":
{                   "fields": {}               },               "fo
llowing": false,                "_module": "Opportunities",
     "duplicate_check_rank": 0           }        }    }
```


/Leads/:record/freebusy GET

Get a lead's FreeBusy schedule

Summary:

This endpoint returns a list of time slots for which the specified person is busy.

Request

GET /Leads/:id/freebusy

Response


```
    {
      "id": "foo"
      "module": "Users",
      "freebusy": [
                    {
                      "start": "2014-08-24T08:45:00-04:00",
                      "end": "2014-08-24T09:15:00-04:00"
                    },
                    {
                      "start": "2014-08-30T05:45:00-04:00",
                      "end": "2014-08-30T06:15:00-04:00"
                    },
                    {
                      "start": "2014-09-12T15:45:00-04:00",
                      "end": "2014-09-12T16:15:00-04:00"
                    }
                  ]
    }
```

/Leads/register POST

Overview

Creates new Leads.

Request Arguments

Name	Type	Description	Required
<record field>	<record field type>	The name value list of fields to populate.	True

Request

```
{
   "first_name":"John",
   "last_name":"Smith"
}
```

Response Arguments

Name	Type	Description
<record field>	<record field type>	Returns the fields for the newly created record.

Response

```
{
   "id":"ecba9f86-4a4a-def6-359c-505a5b33f014",
   "name":"John Smith",
   "date_entered":"2012-09-19T23:54:54+0000",
   "date_modified":"2012-09-19T23:54:54+0000",
   "modified_user_id":"1",
   "created_by":"1",
   "deleted":0,
   "team_id":"1",
   "team_set_id":"1",
   "first_name":"John",
```

```
   "last_name":"Smith",
   "full_name":"John Smith",
   "do_not_call":false,
   "converted":false,
   "lead_source":"Support Portal User Registration",
   "status":"New",
   "preferred_language":"en_us",
   "email":[

   ],
   "my_favorite":false
}
```

Change Log

Version	Change
v10	Added /Leads/register POST endpoint.

/Mail POST

Overview

Create an email and send or save as draft.

Query String Parameters

This endpoint does not accept any query string parameters.

Input Parameters

Name	Type	Description	Required
email_config	String	ID of the outbound email configuration to use when sending this email	True
to_addresses	Array	Array of name/email address pairs for the TO field, when present, only email	True

Name	Type	Description	Required
		subfield is required (not name).	
cc_addresses	Array	Array of name/email address pairs for the CC field, when present, email subfield is required.	False
bcc_addresses	Array	Array of name/email address pairs for the BCC field, when present, email subfield is required.	False
subject	String	Subject of the email	False
html_body	String	HTML body of the email	False
text_body	String	Text body of the email	False
status	String	Indicates the status of the email - 'draft' or 'ready' (ready to be sent)	True
related	Object	Contains 'parent_type' and 'parent_id' to relate this email to (for example 'Contact' and a contact's id)	False
teams	Object	Team(s) to assign this email to. 'primary' attribute is an id string for the primary team and 'other' attribute is an array of id strings for the other teams	True

Name	Type	Description	Required
		- only primary is required	
attachments	Array	Array of file attachments - each attachment consists of a 'type' (where the attachment came from: upload, document, or template), 'id' (of the file upload, note, document revision, etc), and 'name' (the file name)	False

Input Example

```
{
  "email_config":"abc181a2-5c05-b879-8e68-502279a8c401",
  "to_addresses":[
    {
      "name":"John Doe",
      "email":"john_doe@foo.com"
    },
    {
      "name":"David Madison",
      "email":"david_madison@bar.com"
    }
  ],
  "cc_addresses":[
    {
      "name":"Tom Swift",
      "email":"tswift@baz.com"
    }
  ],
  "bcc_addresses":null,
  "subject":"Minneapolis Convention",
  "html_body":"<html><body>Hello World<\/body><\/html>",
  "text_body":"Hello World"
  "status":"ready",
  "related": {
    "type":"Contacts",
```

```
    "id":"61cf0f8d-938c-c9b2-53ad-51ed7bbcf83b"
  },
  "teams": {
    "primary": "dabec868-696c-f458-e204-50227995ab50",
    "others": [
      "c3094c88-c95f-2e17-4553-50227996ad20",
      "abcde868-696c-f458-e704-58369095ab62"
    ]
  },
  "attachments": [
    {
       "type": "upload",
       "id": "cfbe4551-548d-f602-b228-45387645fc12",
       "name": "company_logo.jpg"
    },
    {
       "type": "document",
       "id": "876112a4-89c1-4ba7-a05a-7729a7a76818"
    },
    {
       "type": "template",
       "id": "002cfe6c-98e9-4342-bdb1-1660d0788872"
    }
  ],
}
```

Result

Name	Type	Description
<email record field>	<email record field type>	Returns the fields for the newly created email record.

Output Example

```
{
    "team_id": "9c61c46a-a7c5-df71-481c-51d48232f820",
    "team_set_id": "9c61c46a-a7c5-df71-481c-51d48232f820",
    "id": "d9c165d0-8863-ba61-dc85-51ed8016c476",
    "date_entered": "2013-07-22 18:57:57",
    "date_modified": "2013-07-22 18:57:57",
```

```
    "assigned_user_id": "1",
    "assigned_user_name": "",
    "modified_user_id": "1",
    "modified_by_name": "admin",
    "created_by": "1",
    "created_by_name": "",
    "deleted": 0,
    "from_addr_name": "SugarCRM",
    "reply_to_addr": "",
    "to_addrs_names": "john_doe@foo.com, david_madison@bar.com",
    "cc_addrs_names": "tswift@baz.com",
    "description_html": "<html><body>Hello World<\/body><\/html>",
    "description": "Hello World",
    "date_sent": "2013-07-22 18:57:00",
    "name": "Minneapolis Convention",
    "type": "out",
    "status": "sent",
    "parent_type": "Contacts",
    "parent_id": "61cf0f8d-938c-c9b2-53ad-51ed7bbcf83b",
}
```


Change Log

Version	Change
v11	Last version in which /Mail POST is available. Use /Emails POST instead.
v10	Added /Mail POST endpoint.

/Mail/address/validate POST

Overview

Validate one or more email addresses.

Request Arguments

Type	Description	Required
Array	Array of one or more email addresses to validate.	True

Request

```
[
    "a@b.com",
    "c@d.com",
    "invalid-email.com"
]
```

Response Arguments

Type	Description
Object	Returns an object with each email address as a key and a boolean indicating whether the email address is valid as the value.

Response

```
{
    "a@b.com": true,
    "c@d.com": true,
    "invalid-email.com": false
}
```

Change Log

Version	Change
v11	Last version in which /Mail/address/validate POST is available.
v10	Added /Mail/address/validate POST endpoint.

/Mail/archive POST

Overview

Archives an email.

Query String Parameters

This endpoint does not accept any query string parameters.

Input Parameters

Name	Type	Description	Required
date_sent	String	Date of email sent	True
from_address	String	From email address	True
to_addresses	Array	Array of name/email address pairs for the TO field, when present, only email subfield is required (not name).	True
cc_addresses	Array	Array of name/email address pairs for the CC field, when present, email subfield is required.	False
bcc_addresses	Array	Array of name/email address pairs for the BCC field, when present, email subfield is required.	False
subject	String	Subject of the email	True
html_body	String	HTML body of the email	False
text_body	String	Text body of the email	False
status	String	Indicates the status of the email - 'draft' or 'ready' (ready to be sent)	True
related	Object	Contains	False

Name	Type	Description	Required
		'parent_type' and 'parent_id' to relate this email to (for example 'Contact' and a contact's id)	
teams	Object	Team(s) to assign this email to. 'primary' attribute is an id string for the primary team and 'other' attribute is an array of id strings for the other teams - only primary is required	True
assigned_user_id	String	ID of a user this record is assigned to.	False
attachments	Array	Array of file attachments - each attachment consists of a 'type' (where the attachment came from: upload, document, or template), 'id' (of the file upload, note, document revision, etc), and 'name' (the file name)	False

Input Example

```
{
  "date_sent":"2014-02-07T00:00:00",
  "from_address":"test@foo.com"
  "to_addresses":[
    {
      "name":"John Doe",
      "email":"john_doe@foo.com",
```

```
      "module":"Leads"
    },
    {
      "name":"David Madison",
      "email":"david_madison@bar.com",
      "module":"Leads"
    }
  ],
  "cc_addresses":[
    {
      "name":"Tom Swift",
      "email":"tswift@baz.com"
    }
  ],
  "bcc_addresses":null,
  "subject":"Minneapolis Convention",
  "html_body":"<html><body>Hello World<\/body><\/html>",
  "text_body":"Hello World"
  "status":"archive",
  "related": {
    "type":"Contacts",
    "id":"61cf0f8d-938c-c9b2-53ad-51ed7bbcf83b"
  },
  "teams": {
    "primary": "dabec868-696c-f458-e204-50227995ab50",
    "others": [
      "c3094c88-c95f-2e17-4553-50227996ad20",
      "abcde868-696c-f458-e704-58369095ab62"
    ]
  },
  "assigned_user_id":"seed_jim_id",
  "attachments": [
    {
       "type": "upload",
       "id": "cfbe4551-548d-f602-b228-45387645fc12",
       "name": "company_logo.jpg"
    },
    {
       "type": "document",
       "id": "876112a4-89c1-4ba7-a05a-7729a7a76818"
    },
    {
       "type": "template",
       "id": "002cfe6c-98e9-4342-bdb1-1660d0788872"
    }
  ],
```

```
}
```

Result

Name	Type	Description
<email record field>	<email record field type>	Returns the fields for the newly created email record.

Output Example

```
{
    "team_id": "9c61c46a-a7c5-df71-481c-51d48232f820",
    "team_set_id": "9c61c46a-a7c5-df71-481c-51d48232f820",
    "id": "d9c165d0-8863-ba61-dc85-51ed8016c476",
    "date_entered": "2013-07-22 18:57:57",
    "date_modified": "2013-07-22 18:57:57",
    "assigned_user_id": "1",
    "assigned_user_name": "",
    "modified_user_id": "1",
    "modified_by_name": "admin",
    "created_by": "1",
    "created_by_name": "",
    "deleted": 0,
    "from_addr_name": "SugarCRM",
    "reply_to_addr": "",
    "to_addrs_names": "john_doe@foo.com, david_madison@bar.com",
    "cc_addrs_names": "tswift@baz.com",
    "description_html": "<html><body>Hello World<\/body><\/html>",
    "description": "Hello World",
    "date_sent": "2013-07-22 18:57:00",
    "name": "Minneapolis Convention",
    "type": "out",
    "status": "sent",
    "parent_type": "Contacts",
    "parent_id": "61cf0f8d-938c-c9b2-53ad-51ed7bbcf83b",
    "from_address":"test@foo.com"
}
```

Change Log

Version	Change
v11	Last version in which /Mail/archive POST is available. Use /Emails POST with {"state": "Archived"} instead.
v10	Added /Mail/archive POST endpoint.

/Mail/attachment POST

Overview

Upload an email attachment

Query String Parameters

This endpoint does not accept any query string parameters.

Input Parameters

Name	Type	Description	Required
email_attachment	String	The file to upload.	True

Input Example

```
{"email_attachment":"@\/path\/to\/ExampleDocument.txt"}
```

Result

Name	Type	Description
<email record field>	<email record field type>	Returns the fields for the newly created email record.

Output Example

```
{
    "guid": "61b19f41-0ac9-0f2f-d9c5-51eecaf89396",
    "name": "ExampleDocument.txt",
    "nameForDisplay": "ExampleDocument.txt"
```

```
}
```

Change Log

Version	Change
v11	Last version in which /Mail/attachment POST is available. Use /Notes/temp/file/filename POST in conjunction with /Emails/:record/link/attachments POST instead.
v10	Added /Mail/attachment POST endpoint.

/Mail/attachment/:file_guid DELETE

Overview

Delete an updated email attachment (where :file_guid is the guid value returned from the /Mail/attachment API)

Query Parameters:

This endpoint does not accept any parameters.

Input Example:

This endpoint does not accept any input.

Output Example:

If successful, the endpoint responds with a bool of true.

Change Log

Version	Change
v11	Last version in which /Mail/attachment/:file_guid DELETE is available. Use /Notes/:id/file/filename DELETE to delete a file from the filesystem. Use /Emails/:record/link/attachments/:remote_id DELETE to remove an attachment from an email. Note that removing an attachment from an email will also delete it from the filesystem.
v10	Added /Mail/attachment/:file_guid DELETE endpoint.

/Mail/attachment/cache DELETE

Overview

Clears the user's attachment cache directory

Query Parameters:

This endpoint does not accept any parameters.

Input Example:

This endpoint does not accept any input.

Output Example:

If successful, the endpoint responds with a bool of true.

Change Log

Version	Change
v11	Last version in which /Mail/attachment/cache DELETE is available. Attachments are no longer written as temporary files to the current

	user's cache directory.
v10	Added /Mail/attachment/cache DELETE endpoint.

/Mail/recipients/find GET

Overview

Finds recipients that match the search term.

Request Arguments

Name	Type	Description	Required
q	string	The search string	False
module_list	string	One of the following strings: users, accounts, contacts, leads, prospects, all Will determine which modules are searched using the given search string.	False
order_by	string	columns to sort by (one or more of $sortableColumns) with direction Example: name:asc,id:desc (will sort by last_name ASC and then id DESC)	False
offset	int	Offset of first record to return	False
max_num	int	Maximum records to return	False

Request

```
    ?q=foo&module_list=all&order_by=name:asc,id:desc&offset=1&max_num=
4
```

Response Arguments

Name	Type	Description
next_offset	int	Returns the next offset to be used if paging ahead for more records
records	Array	Array of records found based on the search string and module_list. Each record contains the ID, module, name, email address and it's ID, opt out flag of the email address, and _acl of the recipient found.

Response


```
{
    "next_offset": 3,
    "records": [
        {
            "id": "seed_sarah_id",
            "_module": "Users",
            "name": "Sarah Smith",
            "email": "sarah@example.com",
            "email_address_id": "962cefa8-2f1t-11e6-9bee-80e6577b09a3"
,
            "opt_out": false
            "_acl": {
                "fields": {
                    "assigned_user_id": {
                        "write":"no",
                        "create":"no"
                    },
                    "date_sent": {
                        "write":"no",
                        "create":"no"
                    }
                }
            }
        },
        {
            "id": "962cefa8-2f1f-11e6-9ade-80e6500b09a0",
            "_module": "Leads",
```

```
            "name": "Antonia Piermarini",
            "email": "support.sales.vegan@example.name",
            "email_address_id": "756yafa8-2f1t-11e6-9bee-80e6577b09a3"
,
            "opt_out": true
            "_acl": {
                "fields": {
                    "assigned_user_id": {
                        "write":"no",
                        "create":"no"
                    },
                    "date_sent": {
                        "write":"no",
                        "create":"no"
                    }
                }
            }
        },
        {
            "id": "962850e2-2f1f-11e6-804e-80e6500b09a0",
            "_module": "Leads",
            "name": "Santa Leffingwell",
            "email": "vegan.vegan@example.info",
            "email_address_id": "211uhhg4-2f1t-11e6-9bee-80e6577b09a3"
,
            "opt_out": false
            "_acl": {
                "fields": {
                    "assigned_user_id": {
                        "write":"no",
                        "create":"no"
                    },
                    "date_sent": {
                        "write":"no",
                        "create":"no"
                    }
                }
            }
        }
    ]
}
```

Change Log

Version	Change
v10	Added /Mail/recipients/find GET endpoint.

/Mail/recipients/lookup POST

Overview

Accepts an array of one or more recipients and tries to resolve unsupplied arguments to provide more comprehensive data for the recipient(s).

Request

```
[
  {
    "module": "Contacts",
    "id": "962cefa8-2f1f-11e6-9ade-80e6500b09a0",
    "email": "",
    "name": ""
  },
  {
    "module": "Leads",
    "id": "9c61c46a-a7c5-df71-481c-51d48232f820",
    "email": "",
    "name": "Vince Tallion"
  }
]
```

Response

```
[
  {
    "module": "Contacts",
    "id": "962cefa8-2f1f-11e6-9ade-80e6500b09a0",
    "email": "kg@example.com",
    "name": "Kelly Germaine",
    "resolved": true
  },
  {
    "module": "Leads",
    "id": "9c61c46a-a7c5-df71-481c-51d48232f820",
    "email": "vtallion@example.com",
```

```
    "name": "Vince Tallion",
    "resolved": true
  }
]
```

Change Log

Version	Change
v11	Last version in which /Mail/recipients/lookup POST is available.
v10	Added /Mail/recipients/lookup POST endpoint.

/Meetings POST

Overview

Create a single event or a series of event records.

Request Arguments

Name	Type	Description	Required
<record field>	<record field type>	The name value list of fields to populate.	True

Request

```
{
    "name":"Department Meeting",
    "description":"Weekly Department Meeting",
    "date_start":"yyyy-mm-ddThh:mm:ss-00:00",
    "date_end":"yyyy-mm-ddThh:mm:ss-00:00",
    "repeat_type":"Weekly",
    "repeat_interval":"1",  /* Every Week */
    "repeat_dow":"25",      /* Tue and Fri */
    "repeat_count":"4",     /* 4 Recurrences */
    "repeat_until":"",
}
```

Response Arguments

Name	Description	Start Date	End Date
<record field>	Returns the fields for the newly created record.		

Response

```
{

}
```

Change Log

Version	Change
v10	Added /<module> POST endpoint.

/Meetings/:record DELETE

Overview

Deletes either a single event record or a series of event records

Request Arguments

Name	Type	Description	Required
all_recurrences	Boolean	Flag to delete all events in a series.	False

Request

```
http://{site_url}/rest/v10/Meetings/2c9e5c09-6824-0d14-f5cb-5130321ac3
cf?all_recurrences=true
```

Response Arguments

Name	Type	Description
id	String	Returns the ID of the deleted record.

Response

```
{
    "id": "11cf0d0a-40af-8cb1-9da0-5057a5f511f9"
}
```

Change Log

Version	Change
v10	Added /<module>/:record DELETE endpoint.

/Meetings/:record PUT

Overview

Update a calendar event record of the specified type.

Request Arguments

Name	Type	Description	Required
all_recurrences	Boolean	Flag to update all events in a series.	False

Request

```
http://{site_url}/rest/v10/Meetings/2c9e5c09-6824-0d14-f5cb-5130321ac3
cf?all_recurrences=true
{
    "name":"Department Meeting",
    "description":"Weekly Department Meeting",
    "date_start":"yyyy-mm-ddThh:mm:ss-00:00",
    "date_end":"yyyy-mm-ddThh:mm:ss-00:00",
    "repeat_type":"Weekly",
```

```
    "repeat_interval":"1",  /* Every Week */
    "repeat_dow":"25",      /* Tue and Fri */
    "repeat_count":"4",     /* 4 Recurrences */
    "repeat_until":"",
}
```

Response Arguments

Name	Description	Start Date	End Date
<record field>	Returns the fields for the updated record.		

Response

```
{

}
```

Change Log

Version	Change
v10	Added /<module>/:record PUT endpoint.

/Meetings/:record/external GET

Retrieves info about launching an external meeting

Summary:

This endpoint is used to retrieve info about launching an external meeting. This includes whether the current user has permission to host and/or join the meeting along with the host/join URLs.

Query Parameters:

Param	Description	Optional
This endpoint does not accept any parameters.		

Input Example:

Output Example:

{ "is_host_option_allowed": false, "host_url": "", "is_join_option_allowed": true, "join_url": "//link.to/external/meeting" }

/Notifications GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

Name	Type	Description	Required
filter	String	The filter expression. Filter expressions are explained below.	False

Name	Type	Description	Required
		Note that JSON-encoded filters can be specified as query parameters in one of two ways for GET requests: 1. By specifying individual filter arguments as distinct parameters. Example: filter[0][id]=1. 2. By specifying the whole filter as a single JSON-encoded string. Note that this syntax is currently not supported on certain modules. Example: filter=[{"id":"1"}].	
filter_id	String	Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND.	False
max_num	Integer	A maximum number of records to return. Default is 20.	False

Name	Type	Description	Required
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
fields	String	Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters (applicable to link and collection fields). The fields id and date_modified will always be returned. Example: name,account_type,description,{"name":"opportunities","fields":["id","name","sales_status"],"order_by":"date_closed:desc"} For more details on additional field parameters, see Relate API and Collection API.	False
view	String	Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be	False

Name	Type	Description	Required
		used in combination with the fields argument. Common views are "record" and "list". Example: record	
order_by	String	How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon. Example: name:DESC,account_type:DESC,date_modified:ASC	False
q	String	A search expression, will search on this module. Cannot be used at the same time as a filter expression or id.	False
deleted	Boolean	Boolean to show deleted records in the result set.	False
nulls_last	Boolean	Boolean to return records with null values in order_by fields last in the result set.	False

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
   "filter":[
      {
         "name":"Nelson Inc"
      }
   ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
   "filter":[
      {
         "name":{
            "$starts":"Nelson"
         }
      }
   ]
}
```

Below is a list of operation types:

Operation	Description
$equals	Performs an exact match on that field.
$not_equals	Performs an exact match on that field.
$not_equals	Matches on non-matching values.
$starts	Matches on anything that starts with the value.
$ends	Matches anything that ends with the value.

Operation		Description
	$contains	Matches anything that contains the value
	$in	Finds anything where field matches one of the values as specified as an array.
	$not_in	Finds anything where field does not matches any of the values as specified as an array.
	$is_null	Checks if the field is null. This operation does not need a value specified.
	$not_null	Checks if the field is not null. This operation does not need a value specified.
	$lt	Matches when the field is less than the value.
	$lte	Matches when the field is less than or equal to the value.
	$gt	Matches when the field is greater than the value.
	$gte	Matches when the field is greater than or equal to the value.

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "$and" and "$or".

Example

```
{
   "filter":[
      {
         "$or":[
            {
               "name":"Nelson Inc"
            },
            {
               "name":"Nelson LLC"
            }
         ]
      }
   ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "$favorite" and "$owner".

Example

```
{
   "filter":[
      {
         "$favorite":"_this"
      }
   ]
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset

Name	Type	Description
		for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Response

```
{
   "next_offset":-1,
   "records":[
      {
         "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
         "name":"Dale Spivey",
         "date_modified":"2013-02-28T05:03:00+00:00",
         "description":"",
         "opportunities": [
            {
               _module: "Opportunities",
               "id": "b0701501-1fab-8ae7-3942-540da93f5017",
               "name": "360 Vacations - 228 Units",
               "date_modified": "2014-09-08T16:05:00+03:00",
               "sales_status": "New"
            },
         ],
         "_acl": {
            "fields": {
            }
         }
      },
      {
         "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
         "name":"Florence Haddock",
         "date_modified":"2013-02-26T19:12:00+00:00",
         "description":"",
         "opportunities": [
            {
               _module: "Opportunities"
               date_modified: "2014-09-08T16:05:00+03:00"
               id: "9ce7c088-8ee4-7cd3-18f1-540da944d4c0"
               name: "360 Vacations - 312 Units"
               sales_status: "New"
```

```
            },
         ],
         "_acl": {
            "fields": {
            }
         }
      }
   ]
}
```

Change Log

Version	Change
v10	Added /<module>/filter GET endpoint.

/Opportunities/:record PUT

Overview

Update a record of the specified type.

Request Arguments

Name	Type	Description	Required
<record field>	<record field type>	The name value list of fields to populate.	True

Request

```
{
    "name":"New Account Name",
    "phone_office":"(555) 888-5555",
    "website":"www.newsite.com",
    "meetings": {
    {
        "add": ["21e3385e-404f-b470-407e-54044e3d8024"],
        "delete": ["21e3385e-404f-b470-407e-54044e3d8023"],
        "create": [
        {
            "name": "Test Meeting"
```

```
        }
      ]
    }
}
```

Link fields

It is possible to add or delete record relations to other records and create new related records.

Action	Type	Description
add	List	List of related records ID. See RelateRecordApi for details.
delete	List	List of related records ID.
create	List	List of name to value arrays of related records.

Response Arguments

Name	Type	Description
<record field>	<record field type>	Returns the fields for the updated record.

Response

```
{
   "id":"f222265a-b755-da89-0bc7-512d09b800b6",
   "name":"New Account Name",
   "date_entered":"2013-02-26T19:12:00+00:00",
   "date_modified":"2013-02-27T22:49:00+00:00",
   "modified_user_id":"1",
   "modified_by_name":"Administrator",
   "created_by":"1",
   "created_by_name":"Administrator",
   "description":"",
   "img":"",
   "last_activity_date":"2013-02-26T19:12:00+00:00",
   "deleted":false,
   "assigned_user_id":"seed_sarah_id",
```

```
"assigned_user_name":"Sarah Smith",
"team_name":[
   {
      "id":1,
      "name":"Global",
      "name_2":"",
      "primary":false
   },
   {
      "id":"West",
      "name":"West",
      "name_2":"",
      "primary":true
   }
],
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"account_type":"Customer",
"industry":"Hospitality",
"annual_revenue":"",
"phone_fax":"",
"billing_address_street":"9 IBM Path",
"billing_address_street_2":"",
"billing_address_street_3":"",
"billing_address_street_4":"",
"billing_address_city":"San Francisco",
"billing_address_state":"CA",
"billing_address_postalcode":"43635",
"billing_address_country":"USA",
"rating":"",
"phone_office":"(555) 888-5555",
"phone_alternate":"",
"website":"www.newsite.com",
"ownership":"",
"employees":"",
"ticker_symbol":"",
"shipping_address_street":"9 IBM Path",
"shipping_address_street_2":"",
"shipping_address_street_3":"",
"shipping_address_street_4":"",
"shipping_address_city":"San Francisco",
"shipping_address_state":"CA",
"shipping_address_postalcode":"43635",
"shipping_address_country":"USA",
```

```
   "email1":"kid86@example.net",
   "parent_id":"",
   "sic_code":"",
   "parent_name":"",
   "email_opt_out":false,
   "invalid_email":false,
   "email":[
      {
         "email_address":"kid86@example.net",
         "opt_out":"0",
         "invalid_email":"0",
         "primary_address":"1"
      },
      {
         "email_address":"the.dev@example.name",
         "opt_out":"0",
         "invalid_email":"0",
         "primary_address":"0"
      }
   ],
   "campaign_id":"",
   "campaign_name":"",
   "my_favorite":false,
   "_acl":{
      "fields":{

      }
   }
}
```


Change Log

Version	Change
v10	Added /<module>/:record PUT endpoint.

/Opportunities/config POST

Overview

Opportunity Config Save

Summary

This saves the config changes to the Opportunity Module, when the **opps_view_by** attribute changes, it will perform the nessicary migrations for the Metadata and the actual Data.

Request Arguments

Name	Type	Description	Required
<config field>	<config field type>	The list of config fields to update.	False

Request

```
{
    "MyConfigOption":"MyConfigValue"
}
```

Response Arguments

Name	Type	Description
<config field>	<config field type>	The list of config fields will be returned.

Response

```
{
   "MyConfigOption":"MyConfigValue"
}
```

Change Log

Version	Change
v10	Added /Opportunities/config POST endpoint.

/Opportunities/enum/:field GET

Overview

Retrieves the enum values for a specific field.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

Name	Type	Description
<list values>	Array	Returns the enum values for a field.

Response


```
{
   "":"",
   "Analyst":"Analyst",
   "Competitor":"Competitor",
   "Customer":"Customer",
   "Integrator":"Integrator",
   "Investor":"Investor",
   "Partner":"Partner",
   "Press":"Press",
   "Prospect":"Prospect",
   "Reseller":"Reseller",
   "Other":"Other"
}
```


Change Log

Version	Change
v10	Added /<module>/enum/:field GET endpoint.

/OutboundEmail POST

Overview

Create a new OutboundEmail configuration to use to send emails over SMTP.

Summary

Each configuration that is created is validated first by attempting to connect to the server.

Request Arguments

Name	Type	Description	Required
name	String	The display name of the configuration.	True
mail_sendtype	String	Only SMTP is supported. This field is defaulted to **smtp**.	False
mail_smtptype	String	The email provider through which emails are sent. This is merely a convenience which allows for prepopulating data in the UI. Possible values are: google, exchange, outlook, other. **other** is the default.	False
mail_smtpserver	String	The address of the SMTP server.	True
mail_smtpport	Integer	The port on which to connect to the SMTP server. The default is **465**, which assumes you are using SSL.	False
mail_smtpuser	String	The username for authenticating with the SMTP server. It is only required if **mail_smtpauth_req** is **true**.	False
mail_smtppass	String	The password for authenticating with	False

Name	Type	Description	Required
		the SMTP server. It is only required if **mail_smtpauth_req** is **true**.	
mail_smtpauth_req	Boolean	Indicates whether or not the SMTP server requires authentication. The default is **false**.	False
mail_smtpssl	String	The encryption protocol used for connecting to the SMTP server. **"0"** represents no encryption. **"1"** represents SSL. **"2"** represents TLS. The default is SSL.	False

Request

```
{
  "name": "My Exchange Account",
  "mail_smtptype": "exchange",
  "mail_smtpserver": "smtp.example.com",
  "mail_smtpport": 465,
  "mail_smtpuser": "foo@example.com",
  "mail_smtppass": "P@55w0rd",
  "mail_smtpauth_req": true,
  "mail_smtpssl": "1"
}
```

Response Arguments

Name	Type	Description
name	String	The display name of the configuration.
type	String	Only **user** configuration may be created using this endpoint, so **user** will

Name	Type	Description
		always be returned. **system** and **system-override** configurations are always created by the application.
user_id	String	The current user's ID. The current user may only create configurations for himself or herself.
mail_sendtype	String	Only SMTP is supported, so **smtp** will always be returned.
mail_smtptype	String	The email provider through which emails are sent. Possible values are: google, exchange, outlook, other.
mail_smtpserver	String	The address of the SMTP server.
mail_smtpport	Integer	The port on which to connect to the SMTP server.
mail_smtpuser	String	The username for authenticating with the SMTP server.
mail_smtppass	Boolean	Indicates whether or not a password exists. **true** is returned for this field when a password does exist. This field is not included in the response when a password does not exist.
mail_smtpauth_req	Boolean	Indicates whether or not the SMTP server requires authentication.
mail_smtpssl	String	The encryption protocol used for connecting to the SMTP server.

Response

```
{
  "id": "e04ce998-960e-11e6-8628-3c15c2d582c6",
  "name": "My Exchange Account",
  "type": "user",
  "user_id": "e91b1fa7-1bd8-3c71-be96-512e643f9ca4",
  "mail_sendtype": "smtp",
  "mail_smtptype": "exchange",
  "mail_smtpserver": "smtp.example.com",
  "mail_smtpport": 465,
  "mail_smtpuser": "foo@example.com",
  "mail_smtppass": true,
  "mail_smtpauth_req": true,
  "mail_smtpssl": "1",
  "deleted": false,
  "my_favorite": false,
  "_acl": {
    "fields": {}
  },
  "_module": "OutboundEmail"
}
```


Change Log

Version	Change
v10	Added /OutboundEmail POST endpoint.

/OutboundEmail/:record PUT

Overview

Update an OutboundEmail configuration.

Summary

Each configuration that is saved is validated first by attempting to connect to the server.

Request Arguments

Name	Type	Description	Required
name	String	The display name	False

Name	Type	Description	Required
		of the configuration.	
mail_sendtype	String	Only SMTP is supported. It is recommended that you do not change this from **smtp**.	False
mail_smtptype	String	The email provider through which emails are sent. This is merely a convenience which allows for prepopulating data in the UI. Possible values are: google, exchange, outlook, other.	False
mail_smtpserver	String	The address of the SMTP server.	False
mail_smtpport	Integer	The port on which to connect to the SMTP server.	False
mail_smtpuser	String	The username for authenticating with the SMTP server. It is only required if **mail_smtpauth_req** is **true**.	False
mail_smtppass	String	The password for authenticating with the SMTP server. It is only required if **mail_smtpauth_req** is **true**.	False
mail_smtpauth_req	Boolean	Indicates whether or not the SMTP server requires authentication.	False
mail_smtpssl	String	The encryption protocol used for	False

Name	Type	Description	Required
		connecting to the SMTP server. **"0"** represents no encryption. **"1"** represents SSL. **"2"** represents TLS.	

Request

```
{
  "mail_smtpport": 587,
  "mail_smtpssl": "2"
}
```

Response Arguments

Name	Type	Description
name	String	The display name of the configuration.
type	String	Possible values are: user, system, system-override. Only the administrator may update the **system** configuration. This field may never change.
user_id	String	The current user's ID. The current user may only update configurations that he or she owns.
mail_sendtype	String	Only SMTP is supported, **smtp** will always be returned.
mail_smtptype	String	The email provider through which emails are sent. Possible values are: google, exchange, outlook, other.
mail_smtpserver	String	The address of the SMTP server.

Name	Type	Description
mail_smtpport	Integer	The port on which to connect to the SMTP server.
mail_smtpuser	String	The username for authenticating with the SMTP server.
mail_smtppass	Boolean	Indicates whether or not a password exists. **true** is returned for this field when a password does exist. This field is not included in the response when a password does not exist.
mail_smtpauth_req	Boolean	Indicates whether or not the SMTP server requires authentication.
mail_smtpssl	String	The encryption protocol used for connecting to the SMTP server.

Response

```
{
  "id": "e04ce998-960e-11e6-8628-3c15c2d582c6",
  "name": "My Exchange Account",
  "type": "user",
  "user_id": "e91b1fa7-1bd8-3c71-be96-512e643f9ca4",
  "mail_sendtype": "smtp",
  "mail_smtptype": "exchange",
  "mail_smtpserver": "smtp.example.com",
  "mail_smtpport": 587,
  "mail_smtpuser": "foo@example.com",
  "mail_smtppass": true,
  "mail_smtpauth_req": true,
  "mail_smtpssl": "2",
  "deleted": false,
  "my_favorite": false,
  "_acl": {
    "fields": {}
  },
  "_module": "OutboundEmail"
}
```

Change Log

Version	Change
v10	Added /OutboundEmail/:record PUT endpoint.

/OutboundEmailConfiguration/list GET

Overview

Lists the current user's outbound email configurations for sending email.

Summary

Used by the Emails module for selecting an outbound email configuration for sending an email.

Response

```
[
  {
     "id": "ecbf2a6c-261e-5fca-fbb6-512d093554b8",
     "name": "George Lee ",
     "type": "user",
     "default": false
  },
  {
     "id": "af5f8dae-7169-b497-1d77-512d0937ed81",
     "name": "ACME, Inc. ",
     "type": "system",
     "default": true
  }
]
```

Change Log

Version	Change

v11	Last version in which /OutboundEmailConfiguration/list GET is available. Use /Emails/enum/outbound_email_id GET instead.
v10	Added the /OutboundEmailConfiguration/list GET endpoint.

/PdfManager GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

Name	Type	Description	Required
filter	String	The filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query parameters in one of two ways for GET requests: 1. By specifying individual	False

Name	Type	Description	Required
		filter arguments as distinct parameters. Example: filter[0][id]=1. 2. By specifying the whole filter as a single JSON-encoded string. Note that this syntax is currently not supported on certain modules. Example: filter=[{"id":"1"}].	
filter_id	String	Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND.	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
fields	String	Comma delimited list of fields to return. Each field may be represented	False

Name	Type	Description	Required
		either by string, or by map containing field name and additional field parameters (applicable to link and collection fields). The fields id and date_modified will always be returned. Example: name,account_type,description,{"name":"opportunities","fields":["id","name","sales_status"],"order_by":"date_closed:desc"} For more details on additional field parameters, see Relate API and Collection API.	
view	String	Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list". Example: record	False
order_by	String	How to sort the returned records,	False

Name	Type	Description	Required
		in a comma delimited list with the direction appended to the column name after a colon. Example: name:DESC,account_type:DESC,date_modified:ASC	
q	String	A search expression, will search on this module. Cannot be used at the same time as a filter expression or id.	False
deleted	Boolean	Boolean to show deleted records in the result set.	False
nulls_last	Boolean	Boolean to return records with null values in order_by fields last in the result set.	False

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
   "filter":[
      {
         "name":"Nelson Inc"
      }
   ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
   "filter":[
      {
         "name":{
            "$starts":"Nelson"
         }
      }
   ]
}
```

Below is a list of operation types:

Operation	Description
$equals	Performs an exact match on that field.
$not_equals	Performs an exact match on that field.
$not_equals	Matches on non-matching values.
$starts	Matches on anything that starts with the value.
$ends	Matches anything that ends with the value.
$contains	Matches anything that contains the value
$in	Finds anything where field matches one of the values as specified as an array.
$not_in	Finds anything where field does not

Operation		Description
		matches any of the values as specified as an array.
	$is_null	Checks if the field is null. This operation does not need a value specified.
	$not_null	Checks if the field is not null. This operation does not need a value specified.
	$lt	Matches when the field is less than the value.
	$lte	Matches when the field is less than or equal to the value.
	$gt	Matches when the field is greater than the value.
	$gte	Matches when the field is greater than or equal to the value.

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "$and" and "$or".

Example

```
{
   "filter":[
      {
         "$or":[
            {
               "name":"Nelson Inc"
            },
            {
```

```
               "name":"Nelson LLC"
            }
         ]
      }
   ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "$favorite" and "$owner".

Example

```
{
   "filter":[
      {
         "$favorite":"_this"
      }
   ]
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Response

```
{
   "next_offset":-1,
   "records":[
      {
         "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
         "name":"Dale Spivey",
         "date_modified":"2013-02-28T05:03:00+00:00",
         "description":"",
         "opportunities": [
            {
               _module: "Opportunities",
               "id": "b0701501-1fab-8ae7-3942-540da93f5017",
               "name": "360 Vacations - 228 Units",
               "date_modified": "2014-09-08T16:05:00+03:00",
               "sales_status": "New"
            },
         ],
         "_acl": {
            "fields": {
            }
         }
      },
      {
         "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
         "name":"Florence Haddock",
         "date_modified":"2013-02-26T19:12:00+00:00",
         "description":"",
         "opportunities": [
            {
               _module: "Opportunities"
               date_modified: "2014-09-08T16:05:00+03:00"
               id: "9ce7c088-8ee4-7cd3-18f1-540da944d4c0"
               name: "360 Vacations - 312 Units"
               sales_status: "New"
            },
         ],
         "_acl": {
            "fields": {
            }
         }
      }
   ]
}
```

Change Log

Version	Change
v10	Added /<module>/filter GET endpoint.

/PdfManager/generate GET

Overview

PdfManager Generate Endpoint Help

Summary

This endpoint returns and initiates download of a PDF.

Request Arguments

Name	Type	Description	Required
module	String	The Module name of the Bean from which to create a PDF	True
record	String	The Record ID from which to create a PDF	True
pdf_template_id	String	The PDF Template ID to use to create the PDF	True
sugarpdf	String	The type of PDF to use. Defaults to 'pdfmanager'.	False

Request

```
{
   "module": "Quotes",
   "record": "abcde-fghij-12345-67890",
   "pdf_template_id": "12345-67890-abcde-fghij",
   "sugarpdf": "pdfmanager"
}
```

Response

The response is a base64 encoded string of the contents of the PDF file.

Change Log

Version	Change
v11.3	Added /PdfManager/generate GET endpoint.

/ProductTemplates/tree GET

ProductTemplate API

Summary:

This endpoint is designed to return a paged recordset of one level of product templates and product categories in a N-level deep tree representing the relationship between categories and templates. The results are loosely filtered (wildcard before and after).

Query Parameters:

Param	Description	Optional
filter	The filter used to search Product Template and Product Category names. %filter%	true
root	The ID of the root node you wish to retrieve. Defaults to null (the root level)	true
offset	Specifies the numeric value of the offset you wish to retrieve	true
max_num	Specifies the numeric value of the max number of records you wish to receive. Default is 20.	true

Response Parameters:

Param	Description
id	The id of the item (Product Category/Template)
Data	The name of the item
state	Specifies the open or closed state of the element. (used for categories)
index	Specifies the index of this element in the current level.

Response:


```
{
    "records":[
        {
            "id":"023d316a-d9d0-11e7-9f54-02425392a518",
            "type":"category",
            "data":"Christen Widgets",
            "state":"closed",
            "index":0
        },
        {
            "id":"01828a4a-d9d0-11e7-8989-02425392a518",
            "type":"category",
            "data":"Desktops",
            "state":"closed",
            "index":1
        }
    ],
    "next_offset":2
}
```

/ProductTemplates/tree POST

ProductTemplate API

Summary:

This endpoint is designed to return a paged recordset of one level of product templates and product categories in a N-level deep tree representing the relationship between categories and templates. The results are loosely filtered (wildcard before and after).

Query Parameters:

Param	Description	Optional
filter	The filter used to search Product Template and Product Category names. %filter%	true
root	The ID of the root node you wish to retrieve. Defaults to null (the root level)	true
offset	Specifies the numeric value of the offset you wish to retrieve	true
max_num	Specifies the numeric value of the max number of records you wish to receive. Default is 20.	true

Response Parameters:

Param	Description
id	The id of the item (Product Category/Template)
Data	The name of the item

state	Specifies the open or closed state of the element. (used for categories)
index	Specifies the index of this element in the current level.

Response:

```
{
    "records":[
        {
            "id":"023d316a-d9d0-11e7-9f54-02425392a518",
            "type":"category",
            "data":"Christen Widgets",
            "state":"closed",
            "index":0
        },
        {
            "id":"01828a4a-d9d0-11e7-8989-02425392a518",
            "type":"category",
            "data":"Desktops",
            "state":"closed",
            "index":1
        }
    ],
    "next_offset":2
}
```

/Quotes/:record/opportunity POST

Overview

Quote Convert to Opportunity

Summary

This endpoint converts a Quote to an Opportunity.

Request Arguments

NONE

Response Arguments

NONE

Response

Created Opportunity:


```
{
    "record": {
        "id": "31a601ee-2d3f-11e8-bfae-024289eecda3",
        "name": "Mirrors for J.K.M. Corp (HA)",
        "date_entered": "2018-03-21T19:36:47+00:00",
        "date_modified": "2018-03-21T19:36:47+00:00",
        "modified_user_id": "1",
        "modified_by_name": "admin",
        "modified_user_link": {
            "full_name": "admin",
            "id": "1",
            "_acl": {
                "fields": {
                    "pwd_last_changed": {
                        "write": "no",
                        "create": "no"
                    },
                    "last_login": {
                        "write": "no",
                        "create": "no"
                    }
                },
                "delete": "no",
                "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
            }
        },
        "created_by": "1",
        "created_by_name": "",
        "created_by_link": {
            "full_name": "",
            "id": "1",
            "_acl": {
                "fields": {
                    "pwd_last_changed": {
                        "write": "no",
                        "create": "no"
```

```
            },
            "last_login": {
                "write": "no",
                "create": "no"
            }
        },
        "delete": "no",
        "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
    }
},
"deleted": false,
"opportunity_type": "New Business",
"account_name": "",
"accounts": {
    "name": "",
    "id": "0029279c-2d3d-11e8-a121-024289eecda3",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"account_id": "0029279c-2d3d-11e8-a121-024289eecda3",
"campaign_name": "",
"campaign_opportunities": {
    "name": "",
    "id": null,
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"amount": "520.000000",
"amount_usdollar": "520.000000",
"date_closed": "2012-04-30",
"date_closed_timestamp": 1335744000,
"sales_stage": "Prospecting",
"sales_status": "In Progress",
"probability": 10,
"best_case": "520.000000",
"worst_case": "520.000000",
"commit_stage": "exclude",
"total_revenue_line_items": 1,
"closed_revenue_line_items": 0,
"included_revenue_line_items": 0,
"mkto_sync": false,
"assigned_user_id": "seed_max_id",
```

```
        "assigned_user_name": "Max Jensen",
        "assigned_user_link": {
            "full_name": "Max Jensen",
            "id": "seed_max_id",
            "_acl": {
                "fields": [],
                "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
            }
        },
        "team_id": "1",
        "team_set_id": "1",
        "acl_team_set_id": "f846e71c-2d3c-11e8-a9e4-024289eecda3",
        "team_count": "",
        "team_count_link": {
            "team_count": "",
            "id": "1",
            "_acl": {
                "fields": [],
                "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
            }
        },
        "currency_id": "-99",
        "base_rate": "1.000000",
        "currency_name": "",
        "currencies": {
            "name": "",
            "id": "-99",
            "_acl": {
                "fields": [],
                "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
            },
            "symbol": ""
        },
        "currency_symbol": "",
        "my_favorite": false,
        "_acl": {
            "fields": {}
        },
        "_module": "Opportunities"
    },
    "related_record": {
        "id": "23d19cec-2d3d-11e8-b4bf-024289eecda3",
        "name": "Mirrors for J.K.M. Corp (HA)",
        "date_entered": "2018-03-21T19:20:19+00:00",
        "date_modified": "2018-03-21T19:36:47+00:00",
        "modified_user_id": "1",
```

```
"modified_by_name": "admin",
"modified_user_link": {
    "full_name": "admin",
    "id": "1",
    "_acl": {
        "fields": {
            "pwd_last_changed": {
                "write": "no",
                "create": "no"
            },
            "last_login": {
                "write": "no",
                "create": "no"
            }
        },
        "delete": "no",
        "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
    }
},
"created_by": "1",
"created_by_name": "Administrator",
"created_by_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
        "fields": {
            "pwd_last_changed": {
                "write": "no",
                "create": "no"
            },
            "last_login": {
                "write": "no",
                "create": "no"
            }
        },
        "delete": "no",
        "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
    }
},
"description": "",
"deleted": false,
"shipper_id": "",
"shipper_name": "",
"shippers": {
    "name": "",
    "id": "",
```

```
        "_acl": {
            "fields": [],
            "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
        }
    },
    "taxrate_id": "2254f224-2d3d-11e8-925e-024289eecda3",
    "taxrate_name": "8.25 - Cupertino, CA",
    "taxrates": {
        "name": "8.25 - Cupertino, CA",
        "id": "2254f224-2d3d-11e8-925e-024289eecda3",
        "_acl": {
            "fields": [],
            "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
        }
    },
    "taxrate_value": "8.250000",
    "show_line_nums": true,
    "quote_type": "Quotes",
    "date_quote_expected_closed": "2012-04-30",
    "original_po_date": "",
    "payment_terms": "Net 15",
    "date_quote_closed": "",
    "date_order_shipped": "",
    "order_stage": "",
    "quote_stage": "Negotiation",
    "purchase_order_num": "3940021",
    "quote_num": 2,
    "subtotal": "520.000000",
    "subtotal_usdollar": "520.000000",
    "shipping": "0.000000",
    "shipping_usdollar": "0.000000",
    "discount": "",
    "deal_tot": "0.000000",
    "deal_tot_discount_percentage": 0,
    "deal_tot_usdollar": "0.000000",
    "new_sub": "520.000000",
    "new_sub_usdollar": "520.000000",
    "taxable_subtotal": "520.000000",
    "tax": "42.900000",
    "tax_usdollar": "42.900000",
    "total": "562.900000",
    "total_usdollar": "562.900000",
    "billing_address_street": "111 Silicon Valley Road",
    "billing_address_city": "Kansas City",
    "billing_address_state": "CA",
    "billing_address_postalcode": "47108",
```

```
"billing_address_country": "USA",
"shipping_address_street": "111 Silicon Valley Road",
"shipping_address_city": "Kansas City",
"shipping_address_state": "CA",
"shipping_address_postalcode": "47108",
"shipping_address_country": "USA",
"shipping_account_name": "",
"shipping_accounts": {
    "name": "",
    "id": "0029279c-2d3d-11e8-a121-024289eecda3",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"shipping_account_id": "",
"shipping_contact_name": "",
"shipping_contacts": {
    "full_name": "",
    "id": "",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"shipping_contact_id": "",
"account_name": "J.K.M. Corp (HA)",
"billing_accounts": {
    "name": "J.K.M. Corp (HA)",
    "id": "0029279c-2d3d-11e8-a121-024289eecda3",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"account_id": "0029279c-2d3d-11e8-a121-024289eecda3",
"billing_account_name": "J.K.M. Corp (HA)",
"billing_account_id": "0029279c-2d3d-11e8-a121-024289eecda3",
"billing_contact_name": "",
"billing_contacts": {
    "full_name": "",
    "id": "",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
```

```
},
"billing_contact_id": "",
"opportunity_name": "",
"opportunities": {
    "name": "",
    "id": "23d19cec-2d3d-11e8-b4bf-024289eecda3",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"opportunity_id": "",
"following": "",
"my_favorite": false,
"tag": [],
"locked_fields": [],
"assigned_user_id": "seed_max_id",
"assigned_user_name": "Max Jensen",
"assigned_user_link": {
    "full_name": "Max Jensen",
    "id": "seed_max_id",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"team_count": "",
"team_count_link": {
    "team_count": "",
    "id": "1",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"team_name": [{
    "id": "1",
    "name": "Global",
    "name_2": "",
    "primary": true,
    "selected": false
}],
"currency_id": "-99",
"base_rate": "1.000000",
"currency_name": "",
"currencies": {
```

```
            "name": "",
            "id": "-99",
            "_acl": {
                "fields": [],
                "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
            },
            "symbol": ""
        },
        "currency_symbol": "",
        "_acl": {
            "fields": {}
        },
        "_module": "Quotes"
    }
}
```


/Quotes/config GET

Overview

Quote Config GET Help

Summary

This endpoint returns dependent and related fields for all fields used in the Quoted Line Items area of the Quote record.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

Name	Type	Description
<config field>	<config field type>	The list of config fields will be returned.

Response


```
{
    "summary_columns": [
        {
            "name": "deal_tot",
```

```
            "label": "LBL_LIST_DEAL_TOT",
            "css_class": "quote-totals-row-item",
            "related_fields": [
                "deal_tot_discount_percentage"
            ],
            "type": "currency",
            "labelModule": "Quotes"
        },
        {
            "name": "new_sub",
            "css_class": "quote-totals-row-item",
            "type": "currency",
            "label": "LBL_NEW_SUB",
            "labelModule": "Quotes"
        },
        {
            "name": "tax",
            "label": "LBL_TAX_TOTAL",
            "css_class": "quote-totals-row-item",
            "type": "currency",
            "labelModule": "Quotes"
        },
        {
            "name": "shipping",
            "css_class": "quote-totals-row-item",
            "type": "currency",
            "label": "LBL_SHIPPING",
            "labelModule": "Quotes"
        },
        {
            "name": "total",
            "label": "LBL_LIST_GRAND_TOTAL",
            "css_class": "quote-totals-row-item",
            "type": "currency",
            "labelModule": "Quotes"
        }
    ],
    "worksheet_columns": [
        {
            "name": "line_num",
            "label": null,
            "widthClass": "cell-xsmall",
            "css_class": "line_num tcenter",
            "type": "line-num",
            "readonly": true
        },
```

```
{
    "name": "quantity",
    "label": "LBL_QUANTITY",
    "widthClass": "cell-small",
    "css_class": "quantity",
    "type": "float",
    "labelModule": "Products"
},
{
    "name": "product_template_name",
    "label": "LBL_ITEM_NAME",
    "widthClass": "cell-large",
    "type": "quote-data-relate",
    "required": true,
    "labelModule": "Quotes"
},
{
    "name": "mft_part_num",
    "label": "LBL_MFT_PART_NUM",
    "type": "base",
    "labelModule": "Products"
},
{
    "name": "discount_price",
    "label": "LBL_DISCOUNT_PRICE",
    "type": "currency",
    "convertToBase": true,
    "showTransactionalAmount": true,
    "related_fields": [
        "discount_price",
        "currency_id",
        "base_rate"
    ],
    "labelModule": "Products"
},
{
    "name": "discount",
    "type": "fieldset",
    "css_class": "quote-discount-percent",
    "label": "LBL_DISCOUNT_AMOUNT",
    "fields": [
        {
            "name": "discount_amount",
            "label": "LBL_DISCOUNT_AMOUNT",
            "type": "discount",
            "convertToBase": true,
```

```
                    "showTransactionalAmount": true
                },
                {
                    "type": "discount-select",
                    "name": "discount_select",
                    "no_default_action": true,
                    "buttons": [
                        {
                            "type": "rowaction",
                            "name": "select_discount_amount_button",
                            "label": "LBL_DISCOUNT_AMOUNT",
                            "event": "button:discount_select_change:cl
ick"
                        },
                        {
                            "type": "rowaction",
                            "name": "select_discount_percent_button",
                            "label": "LBL_DISCOUNT_PERCENT",
                            "event": "button:discount_select_change:cl
ick"
                        }
                    ],
                    "label": "LBL_DISCOUNT_AS_PERCENT"
                }
            ],
            "labelModule": "Products"
        },
        {
            "name": "total_amount",
            "label": "LBL_LINE_ITEM_TOTAL",
            "type": "currency",
            "widthClass": "cell-medium",
            "showTransactionalAmount": true,
            "related_fields": [
                "total_amount",
                "currency_id",
                "base_rate"
            ],
            "labelModule": "Quotes"
        }
    ],
    "footer_rows": [
        {
            "name": "new_sub",
            "type": "currency"
        },
```

```
        {
            "name": "tax",
            "type": "currency"
        },
        {
            "name": "shipping",
            "type": "quote-footer-currency",
            "css_class": "quote-footer-currency",
            "default": "0.00"
        },
        {
            "name": "total",
            "type": "currency",
            "css_class": "grand-total"
        }
    ],
    "summary_columns_related_fields": [
        "base_rate",
        "deal_tot",
        "deal_tot_usdollar",
        "shipping",
        "subtotal",
        "subtotal_usdollar",
        "tax",
        "taxable_subtotal"
    ],
    "worksheet_columns_related_fields": [
        "base_rate",
        "deal_calc",
        "discount_amount",
        "discount_price",
        "discount_select",
        "quantity",
        "subtotal",
        "tax_class",
        "total_amount",
        "description",
        "quote_id",
        "name",
        "product_template_id",
        "product_template_name"
    ],
    "footer_rows_related_fields": [
        "deal_tot",
        "deal_tot_usdollar",
        "shipping",
```

```
        "subtotal",
        "subtotal_usdollar",
        "tax",
        "taxable_subtotal"
    ],
    "dependentFields": {
        "Quotes": {
            "currency_id": {
                "related": {
                    "subtotal": {
                        "module": "Quotes",
                        "field": "subtotal",
                        "reason": "related_fields"
                    },
                    "discount": {
                        "module": "Quotes",
                        "field": "discount",
                        "reason": "related_fields"
                    },
                    "new_sub": {
                        "module": "Quotes",
                        "field": "new_sub",
                        "reason": "related_fields"
                    },
                    "tax": {
                        "module": "Quotes",
                        "field": "tax",
                        "reason": "related_fields"
                    },
                    "shipping": {
                        "module": "Quotes",
                        "field": "shipping",
                        "reason": "related_fields"
                    },
                    "subtotal_usdollar": {
                        "module": "Quotes",
                        "field": "subtotal_usdollar",
                        "reason": "related_fields"
                    }
                }
            },
            "base_rate": {
                "related": {
                    "subtotal": {
                        "module": "Quotes",
                        "field": "subtotal",
```

```
                "reason": "related_fields"
            },
            "discount": {
                "module": "Quotes",
                "field": "discount",
                "reason": "related_fields"
            },
            "new_sub": {
                "module": "Quotes",
                "field": "new_sub",
                "reason": "related_fields"
            },
            "tax": {
                "module": "Quotes",
                "field": "tax",
                "reason": "related_fields"
            },
            "shipping": {
                "module": "Quotes",
                "field": "shipping",
                "reason": "related_fields"
            },
            "subtotal_usdollar": {
                "module": "Quotes",
                "field": "subtotal_usdollar",
                "reason": "related_fields"
            }
        },
        "locked": {
            "subtotal_usdollar": {
                "module": "Quotes",
                "field": "subtotal_usdollar",
                "reason": "formula"
            },
            "deal_tot_usdollar": {
                "module": "Quotes",
                "field": "deal_tot_usdollar",
                "reason": "formula"
            }
        }
    },
    "taxable_subtotal": {
        "locked": {
            "tax": {
                "module": "Quotes",
                "field": "tax",
```

```
                "reason": "formula"
            }
        },
        "related": {
            "tax": {
                "module": "Quotes",
                "field": "tax",
                "reason": "related_fields"
            }
        }
    },
    "taxrate_value": {
        "locked": {
            "tax": {
                "module": "Quotes",
                "field": "tax",
                "reason": "formula"
            }
        },
        "related": {
            "tax": {
                "module": "Quotes",
                "field": "tax",
                "reason": "related_fields"
            }
        }
    },
    "subtotal_usdollar": {
        "locked": {
            "deal_tot_discount_percentage": {
                "module": "Quotes",
                "field": "deal_tot_discount_percentage",
                "reason": "formula"
            }
        }
    },
    "subtotal": {
        "locked": {
            "subtotal_usdollar": {
                "module": "Quotes",
                "field": "subtotal_usdollar",
                "reason": "formula"
            }
        }
    },
    "deal_tot_usdollar": {
```

```
            "locked": {
                "deal_tot_discount_percentage": {
                    "module": "Quotes",
                    "field": "deal_tot_discount_percentage",
                    "reason": "formula"
                }
            }
        },
        "deal_tot": {
            "locked": {
                "deal_tot_usdollar": {
                    "module": "Quotes",
                    "field": "deal_tot_usdollar",
                    "reason": "formula"
                }
            }
        },
        "tax": {
            "locked": {
                "total": {
                    "module": "Quotes",
                    "field": "total",
                    "reason": "formula"
                }
            }
        },
        "shipping": {
            "locked": {
                "total": {
                    "module": "Quotes",
                    "field": "total",
                    "reason": "formula"
                }
            }
        }
    },
    "ProductBundles": {
        "subtotal": {
            "locked": {
                "new_sub": {
                    "module": "ProductBundles",
                    "field": "new_sub",
                    "reason": "formula"
                }
            }
        },
```

```
        "currency_id": {
            "related": {
                "subtotal": {
                    "module": "ProductBundles",
                    "field": "subtotal",
                    "reason": "related_fields"
                },
                "deal_tot": {
                    "module": "ProductBundles",
                    "field": "deal_tot",
                    "reason": "related_fields"
                },
                "new_sub": {
                    "module": "ProductBundles",
                    "field": "new_sub",
                    "reason": "related_fields"
                }
            }
        },
        "base_rate": {
            "related": {
                "subtotal": {
                    "module": "ProductBundles",
                    "field": "subtotal",
                    "reason": "related_fields"
                },
                "deal_tot": {
                    "module": "ProductBundles",
                    "field": "deal_tot",
                    "reason": "related_fields"
                },
                "new_sub": {
                    "module": "ProductBundles",
                    "field": "new_sub",
                    "reason": "related_fields"
                }
            }
        },
        "new_sub": {
            "locked": {
                "new_sub": {
                    "module": "Quotes",
                    "field": "new_sub",
                    "reason": "rollup"
                },
                "total": {
```

```
                    "module": "Quotes",
                    "field": "total",
                    "reason": "rollup"
                }
            }
        },
        "deal_tot": {
            "locked": {
                "new_sub": {
                    "module": "ProductBundles",
                    "field": "new_sub",
                    "reason": "formula"
                }
            }
        },
        "taxable_subtotal": {
            "locked": {
                "taxable_subtotal": {
                    "module": "Quotes",
                    "field": "taxable_subtotal",
                    "reason": "rollup"
                }
            }
        }
    },
    "Products": {
        "subtotal": {
            "locked": {
                "subtotal": {
                    "module": "ProductBundles",
                    "field": "subtotal",
                    "reason": "rollup"
                }
            }
        },
        "quantity": {
            "locked": {
                "total_amount": {
                    "module": "Products",
                    "field": "total_amount",
                    "reason": "formula"
                }
            },
            "related": {
                "total_amount": {
                    "module": "Products",
```

```
                "field": "total_amount",
                "reason": "related_fields"
            }
        }
    },
    "discount_price": {
        "locked": {
            "total_amount": {
                "module": "Products",
                "field": "total_amount",
                "reason": "formula"
            }
        },
        "related": {
            "total_amount": {
                "module": "Products",
                "field": "total_amount",
                "reason": "related_fields"
            }
        }
    },
    "currency_id": {
        "related": {
            "discount_price": {
                "module": "Products",
                "field": "discount_price",
                "reason": "related_fields"
            },
            "discount_amount": {
                "module": "Products",
                "field": "discount_amount",
                "reason": "related_fields"
            },
            "total_amount": {
                "module": "Products",
                "field": "total_amount",
                "reason": "related_fields"
            }
        }
    },
    "base_rate": {
        "related": {
            "discount_price": {
                "module": "Products",
                "field": "discount_price",
                "reason": "related_fields"
```

```
            },
            "discount_amount": {
                "module": "Products",
                "field": "discount_amount",
                "reason": "related_fields"
            },
            "total_amount": {
                "module": "Products",
                "field": "total_amount",
                "reason": "related_fields"
            }
        }
    },
    "deal_calc": {
        "locked": {
            "deal_tot": {
                "module": "ProductBundles",
                "field": "deal_tot",
                "reason": "rollup"
            }
        }
    },
    "discount_select": {
        "related": {
            "discount_amount": {
                "module": "Products",
                "field": "discount_amount",
                "reason": "related_fields"
            },
            "total_amount": {
                "module": "Products",
                "field": "total_amount",
                "reason": "related_fields"
            }
        },
        "locked": {
            "total_amount": {
                "module": "Products",
                "field": "total_amount",
                "reason": "formula"
            }
        }
    },
    "discount_amount": {
        "locked": {
            "total_amount": {
```

```
                        "module": "Products",
                        "field": "total_amount",
                        "reason": "formula"
                    }
                },
                "related": {
                    "total_amount": {
                        "module": "Products",
                        "field": "total_amount",
                        "reason": "related_fields"
                    }
                }
            },
            "total_amount": {
                "locked": {
                    "taxable_subtotal": {
                        "module": "ProductBundles",
                        "field": "taxable_subtotal",
                        "reason": "rollup"
                    }
                }
            },
            "tax_class": {
                "locked": {
                    "taxable_subtotal": {
                        "module": "ProductBundles",
                        "field": "taxable_subtotal",
                        "reason": "rollup"
                    }
                }
            }
        }
    },
    "relatedFields": {
        "Products": {
            "name": {
                "locked": {
                    "product_template_name": {
                        "module": "Products",
                        "field": "name",
                        "reason": "formula"
                    }
                }
            },
            "subtotal": {
                "locked": {
```

```
            "quantity": {
                "module": "Products",
                "field": "subtotal",
                "reason": "formula"
            },
            "discount_price": {
                "module": "Products",
                "field": "subtotal",
                "reason": "formula"
            }
        },
        "related": {
            "currency_id": {
                "module": "Products",
                "field": "subtotal",
                "reason": "related_fields"
            },
            "base_rate": {
                "module": "Products",
                "field": "subtotal",
                "reason": "related_fields"
            },
            "discount_price": {
                "module": "Products",
                "field": "subtotal",
                "reason": "related_fields"
            },
            "quantity": {
                "module": "Products",
                "field": "subtotal",
                "reason": "related_fields"
            }
        }
    },
    "total_amount": {
        "locked": {
            "quantity": {
                "module": "Products",
                "field": "total_amount",
                "reason": "formula"
            },
            "discount_price": {
                "module": "Products",
                "field": "total_amount",
                "reason": "formula"
            },
```

```
            "discount_select": {
                "module": "Products",
                "field": "total_amount",
                "reason": "formula"
            },
            "discount_amount": {
                "module": "Products",
                "field": "total_amount",
                "reason": "formula"
            }
        },
        "related": {
            "currency_id": {
                "module": "Products",
                "field": "total_amount",
                "reason": "related_fields"
            },
            "base_rate": {
                "module": "Products",
                "field": "total_amount",
                "reason": "related_fields"
            },
            "quantity": {
                "module": "Products",
                "field": "total_amount",
                "reason": "related_fields"
            },
            "discount_price": {
                "module": "Products",
                "field": "total_amount",
                "reason": "related_fields"
            },
            "discount_select": {
                "module": "Products",
                "field": "total_amount",
                "reason": "related_fields"
            },
            "discount_amount": {
                "module": "Products",
                "field": "total_amount",
                "reason": "related_fields"
            }
        }
    },
    "manufacturer_name": {
        "related": {
```

```
            "manufacturer_id": {
                "module": "Products",
                "field": "manufacturer_name",
                "reason": "related_fields"
            }
        }
    },
    "cost_price": {
        "related": {
            "currency_id": {
                "module": "Products",
                "field": "cost_price",
                "reason": "related_fields"
            },
            "base_rate": {
                "module": "Products",
                "field": "cost_price",
                "reason": "related_fields"
            }
        }
    },
    "discount_price": {
        "related": {
            "currency_id": {
                "module": "Products",
                "field": "discount_price",
                "reason": "related_fields"
            },
            "base_rate": {
                "module": "Products",
                "field": "discount_price",
                "reason": "related_fields"
            }
        }
    },
    "discount_amount": {
        "related": {
            "currency_id": {
                "module": "Products",
                "field": "discount_amount",
                "reason": "related_fields"
            },
            "base_rate": {
                "module": "Products",
                "field": "discount_amount",
                "reason": "related_fields"
```

```
            },
            "discount_select": {
                "module": "Products",
                "field": "discount_amount",
                "reason": "related_fields"
            }
        }
    },
    "discount_rate_percent": {
        "locked": {
            "discount_price": {
                "module": "Products",
                "field": "discount_rate_percent",
                "reason": "formula"
            },
            "discount_amount": {
                "module": "Products",
                "field": "discount_rate_percent",
                "reason": "formula"
            }
        }
    },
    "discount_amount_usdollar": {
        "locked": {
            "discount_amount": {
                "module": "Products",
                "field": "discount_amount_usdollar",
                "reason": "formula"
            },
            "base_rate": {
                "module": "Products",
                "field": "discount_amount_usdollar",
                "reason": "formula"
            }
        }
    },
    "deal_calc": {
        "locked": {
            "discount_select": {
                "module": "Products",
                "field": "deal_calc",
                "reason": "formula"
            },
            "discount_price": {
                "module": "Products",
                "field": "deal_calc",
```

```
                "reason": "formula"
            },
            "quantity": {
                "module": "Products",
                "field": "deal_calc",
                "reason": "formula"
            },
            "discount_amount": {
                "module": "Products",
                "field": "deal_calc",
                "reason": "formula"
            }
        },
        "related": {
            "currency_id": {
                "module": "Products",
                "field": "deal_calc",
                "reason": "related_fields"
            },
            "base_rate": {
                "module": "Products",
                "field": "deal_calc",
                "reason": "related_fields"
            },
            "discount_price": {
                "module": "Products",
                "field": "deal_calc",
                "reason": "related_fields"
            },
            "quantity": {
                "module": "Products",
                "field": "deal_calc",
                "reason": "related_fields"
            },
            "discount_amount": {
                "module": "Products",
                "field": "deal_calc",
                "reason": "related_fields"
            }
        }
    },
    "deal_calc_usdollar": {
        "locked": {
            "deal_calc": {
                "module": "Products",
                "field": "deal_calc_usdollar",
```

```
                "reason": "formula"
            },
            "base_rate": {
                "module": "Products",
                "field": "deal_calc_usdollar",
                "reason": "formula"
            }
        },
        "related": {
            "currency_id": {
                "module": "Products",
                "field": "deal_calc_usdollar",
                "reason": "related_fields"
            },
            "base_rate": {
                "module": "Products",
                "field": "deal_calc_usdollar",
                "reason": "related_fields"
            }
        }
    },
    "list_price": {
        "related": {
            "currency_id": {
                "module": "Products",
                "field": "list_price",
                "reason": "related_fields"
            },
            "base_rate": {
                "module": "Products",
                "field": "list_price",
                "reason": "related_fields"
            }
        }
    },
    "cost_usdollar": {
        "locked": {
            "cost_price": {
                "module": "Products",
                "field": "cost_usdollar",
                "reason": "formula"
            },
            "base_rate": {
                "module": "Products",
                "field": "cost_usdollar",
                "reason": "formula"
```

```
            }
        },
        "related": {
            "currency_id": {
                "module": "Products",
                "field": "cost_usdollar",
                "reason": "related_fields"
            },
            "base_rate": {
                "module": "Products",
                "field": "cost_usdollar",
                "reason": "related_fields"
            }
        }
    },
    "discount_usdollar": {
        "locked": {
            "discount_price": {
                "module": "Products",
                "field": "discount_usdollar",
                "reason": "formula"
            },
            "base_rate": {
                "module": "Products",
                "field": "discount_usdollar",
                "reason": "formula"
            }
        }
    },
    "list_usdollar": {
        "locked": {
            "list_price": {
                "module": "Products",
                "field": "list_usdollar",
                "reason": "formula"
            },
            "base_rate": {
                "module": "Products",
                "field": "list_usdollar",
                "reason": "formula"
            }
        },
        "related": {
            "currency_id": {
                "module": "Products",
                "field": "list_usdollar",
```

```
                "reason": "related_fields"
            },
            "base_rate": {
                "module": "Products",
                "field": "list_usdollar",
                "reason": "related_fields"
            }
        }
    },
    "book_value": {
        "related": {
            "currency_id": {
                "module": "Products",
                "field": "book_value",
                "reason": "related_fields"
            },
            "base_rate": {
                "module": "Products",
                "field": "book_value",
                "reason": "related_fields"
            }
        }
    },
    "book_value_usdollar": {
        "locked": {
            "book_value": {
                "module": "Products",
                "field": "book_value_usdollar",
                "reason": "formula"
            },
            "base_rate": {
                "module": "Products",
                "field": "book_value_usdollar",
                "reason": "formula"
            }
        },
        "related": {
            "currency_id": {
                "module": "Products",
                "field": "book_value_usdollar",
                "reason": "related_fields"
            },
            "base_rate": {
                "module": "Products",
                "field": "book_value_usdollar",
                "reason": "related_fields"
```

```
                }
            }
        },
        "date_closed_timestamp": {
            "locked": {
                "date_closed": {
                    "module": "Products",
                    "field": "date_closed_timestamp",
                    "reason": "formula"
                }
            }
        }
    },
    "Quotes": {
        "subtotal": {
            "locked": {
                "product_bundles": {
                    "module": "Quotes",
                    "field": "subtotal",
                    "reason": "formula"
                }
            },
            "related": {
                "currency_id": {
                    "module": "Quotes",
                    "field": "subtotal",
                    "reason": "related_fields"
                },
                "base_rate": {
                    "module": "Quotes",
                    "field": "subtotal",
                    "reason": "related_fields"
                }
            }
        },
        "subtotal_usdollar": {
            "locked": {
                "subtotal": {
                    "module": "Quotes",
                    "field": "subtotal_usdollar",
                    "reason": "formula"
                },
                "base_rate": {
                    "module": "Quotes",
                    "field": "subtotal_usdollar",
                    "reason": "formula"
```

```
            }
        },
        "related": {
            "currency_id": {
                "module": "Quotes",
                "field": "subtotal_usdollar",
                "reason": "related_fields"
            },
            "base_rate": {
                "module": "Quotes",
                "field": "subtotal_usdollar",
                "reason": "related_fields"
            }
        }
    },
    "shipping": {
        "related": {
            "currency_id": {
                "module": "Quotes",
                "field": "shipping",
                "reason": "related_fields"
            },
            "base_rate": {
                "module": "Quotes",
                "field": "shipping",
                "reason": "related_fields"
            }
        }
    },
    "shipping_usdollar": {
        "locked": {
            "shipping": {
                "module": "Quotes",
                "field": "shipping_usdollar",
                "reason": "formula"
            },
            "base_rate": {
                "module": "Quotes",
                "field": "shipping_usdollar",
                "reason": "formula"
            }
        },
        "related": {
            "currency_id": {
                "module": "Quotes",
                "field": "shipping_usdollar",
```

```
                "reason": "related_fields"
            },
            "base_rate": {
                "module": "Quotes",
                "field": "shipping_usdollar",
                "reason": "related_fields"
            }
        }
    },
    "discount": {
        "related": {
            "currency_id": {
                "module": "Quotes",
                "field": "discount",
                "reason": "related_fields"
            },
            "base_rate": {
                "module": "Quotes",
                "field": "discount",
                "reason": "related_fields"
            }
        }
    },
    "deal_tot": {
        "locked": {
            "product_bundles": {
                "module": "Quotes",
                "field": "deal_tot",
                "reason": "formula"
            }
        }
    },
    "deal_tot_discount_percentage": {
        "locked": {
            "subtotal_usdollar": {
                "module": "Quotes",
                "field": "deal_tot_discount_percentage",
                "reason": "formula"
            },
            "deal_tot_usdollar": {
                "module": "Quotes",
                "field": "deal_tot_discount_percentage",
                "reason": "formula"
            }
        }
    },
```

```
"deal_tot_usdollar": {
    "locked": {
        "deal_tot": {
            "module": "Quotes",
            "field": "deal_tot_usdollar",
            "reason": "formula"
        },
        "base_rate": {
            "module": "Quotes",
            "field": "deal_tot_usdollar",
            "reason": "formula"
        }
    }
},
"new_sub": {
    "locked": {
        "product_bundles": {
            "module": "Quotes",
            "field": "new_sub",
            "reason": "formula"
        }
    },
    "related": {
        "currency_id": {
            "module": "Quotes",
            "field": "new_sub",
            "reason": "related_fields"
        },
        "base_rate": {
            "module": "Quotes",
            "field": "new_sub",
            "reason": "related_fields"
        }
    }
},
"new_sub_usdollar": {
    "locked": {
        "new_sub": {
            "module": "Quotes",
            "field": "new_sub_usdollar",
            "reason": "formula"
        },
        "base_rate": {
            "module": "Quotes",
            "field": "new_sub_usdollar",
            "reason": "formula"
```

```
            }
        },
        "related": {
            "currency_id": {
                "module": "Quotes",
                "field": "new_sub_usdollar",
                "reason": "related_fields"
            },
            "base_rate": {
                "module": "Quotes",
                "field": "new_sub_usdollar",
                "reason": "related_fields"
            }
        }
    },
    "taxable_subtotal": {
        "locked": {
            "product_bundles": {
                "module": "Quotes",
                "field": "taxable_subtotal",
                "reason": "formula"
            }
        }
    },
    "tax": {
        "locked": {
            "taxable_subtotal": {
                "module": "Quotes",
                "field": "tax",
                "reason": "formula"
            },
            "taxrate_value": {
                "module": "Quotes",
                "field": "tax",
                "reason": "formula"
            }
        },
        "related": {
            "currency_id": {
                "module": "Quotes",
                "field": "tax",
                "reason": "related_fields"
            },
            "base_rate": {
                "module": "Quotes",
                "field": "tax",
```

```
                "reason": "related_fields"
            },
            "taxrate_value": {
                "module": "Quotes",
                "field": "tax",
                "reason": "related_fields"
            },
            "taxable_subtotal": {
                "module": "Quotes",
                "field": "tax",
                "reason": "related_fields"
            }
        }
    },
    "tax_usdollar": {
        "locked": {
            "tax": {
                "module": "Quotes",
                "field": "tax_usdollar",
                "reason": "formula"
            },
            "base_rate": {
                "module": "Quotes",
                "field": "tax_usdollar",
                "reason": "formula"
            }
        },
        "related": {
            "currency_id": {
                "module": "Quotes",
                "field": "tax_usdollar",
                "reason": "related_fields"
            },
            "base_rate": {
                "module": "Quotes",
                "field": "tax_usdollar",
                "reason": "related_fields"
            }
        }
    },
    "total": {
        "locked": {
            "product_bundles": {
                "module": "Quotes",
                "field": "total",
                "reason": "formula"
```

```
                },
                "tax": {
                    "module": "Quotes",
                    "field": "total",
                    "reason": "formula"
                },
                "shipping": {
                    "module": "Quotes",
                    "field": "total",
                    "reason": "formula"
                }
            }
        },
        "total_usdollar": {
            "locked": {
                "total": {
                    "module": "Quotes",
                    "field": "total_usdollar",
                    "reason": "formula"
                },
                "base_rate": {
                    "module": "Quotes",
                    "field": "total_usdollar",
                    "reason": "formula"
                }
            },
            "related": {
                "currency_id": {
                    "module": "Quotes",
                    "field": "total_usdollar",
                    "reason": "related_fields"
                },
                "base_rate": {
                    "module": "Quotes",
                    "field": "total_usdollar",
                    "reason": "related_fields"
                }
            }
        }
    }
},
"productsFields": {
    "date_entered": {
        "type": "datetime",
        "studio": {
            "portaleditview": false
```

```
        },
        "readonly": true,
        "label": "LBL_DATE_ENTERED"
    },
    "date_modified": {
        "type": "datetime",
        "studio": {
            "portaleditview": false
        },
        "readonly": true,
        "label": "LBL_DATE_MODIFIED"
    },
    "modified_by_name": {
        "type": "relate",
        "link": "modified_user_link",
        "readonly": true,
        "label": "LBL_MODIFIED"
    },
    "created_by_name": {
        "type": "relate",
        "link": "created_by_link",
        "readonly": true,
        "label": "LBL_CREATED"
    },
    "description": {
        "type": "text",
        "label": "LBL_DESCRIPTION"
    },
    "revenuelineitem_name": {
        "type": "relate",
        "link": "revenuelineitems",
        "studio": "visible",
        "label": "LBL_REVENUELINEITEM_NAME"
    },
    "product_template_name": {
        "type": "relate",
        "link": "product_templates_link",
        "studio": {
            "editview": false,
            "detailview": false,
            "quickcreate": false
        },
        "label": "LBL_PRODUCT_TEMPLATE"
    },
    "subtotal": {
        "type": "currency",
```

```
        "default": "0",
        "related_fields": [
            "currency_id",
            "base_rate",
            "discount_price",
            "quantity"
        ],
        "label": "LBL_SUBTOTAL"
    },
    "total_amount": {
        "default": "0.00",
        "type": "currency",
        "related_fields": [
            "currency_id",
            "base_rate",
            "quantity",
            "discount_price",
            "discount_select",
            "discount_amount"
        ],
        "label": "LBL_CALCULATED_LINE_ITEM_AMOUNT"
    },
    "contact_name": {
        "type": "relate",
        "link": "contact_link",
        "label": "LBL_CONTACT_NAME"
    },
    "manufacturer_name": {
        "type": "relate",
        "link": "manufacturers",
        "related_fields": [
            "manufacturer_id"
        ],
        "label": "LBL_MANUFACTURER_NAME"
    },
    "category_name": {
        "type": "relate",
        "link": "product_categories_link",
        "studio": {
            "editview": false,
            "detailview": false,
            "quickcreate": false
        },
        "label": "LBL_CATEGORY_NAME"
    },
    "mft_part_num": {
```

```
        "type": "varchar",
        "label": "LBL_MFT_PART_NUM"
    },
    "vendor_part_num": {
        "type": "varchar",
        "label": "LBL_VENDOR_PART_NUM"
    },
    "date_purchased": {
        "type": "date",
        "label": "LBL_DATE_PURCHASED"
    },
    "discount_rate_percent": {
        "type": "decimal",
        "label": "LBL_DISCOUNT_RATE"
    },
    "discount_amount_usdollar": {
        "type": "currency",
        "studio": {
            "editview": false,
            "mobile": false
        },
        "label": "LBL_DISCOUNT_RATE_USDOLLAR"
    },
    "discount_select": {
        "type": "bool",
        "default": true,
        "label": "LBL_DISCOUNT_AS_PERCENT"
    },
    "deal_calc": {
        "type": "currency",
        "related_fields": [
            "currency_id",
            "base_rate",
            "discount_price",
            "quantity",
            "discount_amount"
        ],
        "label": "LBL_DISCOUNT_TOTAL"
    },
    "deal_calc_usdollar": {
        "type": "currency",
        "studio": {
            "editview": false,
            "mobile": false
        },
        "related_fields": [
```

```
            "currency_id",
            "base_rate"
        ],
        "label": "LBL_DISCOUNT_TOTAL_USDOLLAR"
    },
    "list_price": {
        "type": "currency",
        "related_fields": [
            "currency_id",
            "base_rate"
        ],
        "label": "LBL_LIST_PRICE"
    },
    "cost_usdollar": {
        "type": "currency",
        "studio": {
            "editview": false,
            "mobile": false
        },
        "related_fields": [
            "currency_id",
            "base_rate"
        ],
        "label": "LBL_COST_USDOLLAR"
    },
    "discount_usdollar": {
        "type": "currency",
        "studio": {
            "editview": false,
            "mobile": false
        },
        "label": "LBL_DISCOUNT_USDOLLAR"
    },
    "list_usdollar": {
        "type": "currency",
        "studio": {
            "editview": false,
            "mobile": false
        },
        "related_fields": [
            "currency_id",
            "base_rate"
        ],
        "label": "LBL_LIST_USDOLLAR"
    },
    "tax_class": {
```

```
        "type": "enum",
        "default": "Taxable",
        "label": "LBL_TAX_CLASS"
    },
    "website": {
        "type": "varchar",
        "label": "LBL_URL"
    },
    "weight": {
        "type": "decimal",
        "label": "LBL_WEIGHT"
    },
    "quantity": {
        "type": "decimal",
        "default": true,
        "label": "LBL_QUANTITY"
    },
    "support_name": {
        "type": "varchar",
        "label": "LBL_SUPPORT_NAME"
    },
    "support_description": {
        "type": "varchar",
        "label": "LBL_SUPPORT_DESCRIPTION"
    },
    "support_contact": {
        "type": "varchar",
        "label": "LBL_SUPPORT_CONTACT"
    },
    "support_term": {
        "type": "varchar",
        "label": "LBL_SUPPORT_TERM"
    },
    "date_support_expires": {
        "type": "date",
        "label": "LBL_DATE_SUPPORT_EXPIRES"
    },
    "date_support_starts": {
        "type": "date",
        "label": "LBL_DATE_SUPPORT_STARTS"
    },
    "pricing_formula": {
        "type": "varchar",
        "label": "LBL_PRICING_FORMULA"
    },
    "pricing_factor": {
```

```
        "type": "int",
        "label": "LBL_PRICING_FACTOR"
    },
    "serial_number": {
        "type": "varchar",
        "label": "LBL_SERIAL_NUMBER"
    },
    "asset_number": {
        "type": "varchar",
        "label": "LBL_ASSET_NUMBER"
    },
    "book_value": {
        "type": "currency",
        "related_fields": [
            "currency_id",
            "base_rate"
        ],
        "label": "LBL_BOOK_VALUE"
    },
    "book_value_usdollar": {
        "type": "currency",
        "studio": {
            "editview": false,
            "mobile": false
        },
        "related_fields": [
            "currency_id",
            "base_rate"
        ],
        "label": "LBL_BOOK_VALUE_USDOLLAR"
    },
    "book_value_date": {
        "type": "date",
        "label": "LBL_BOOK_VALUE_DATE"
    },
    "date_closed": {
        "type": "date",
        "label": "LBL_DATE_CLOSED"
    },
    "next_step": {
        "type": "varchar",
        "label": "LBL_NEXT_STEP"
    },
    "campaign_name": {
        "type": "relate",
        "link": "campaign_products",
```

```
        "label": "LBL_CAMPAIGN"
    },
    "opportunity_name": {
        "type": "relate",
        "link": "opportunities",
        "label": "LBL_OPPORTUNITY_NAME"
    },
    "type_name": {
        "type": "relate",
        "link": "product_types_link",
        "label": "LBL_TYPE"
    },
    "tag": {
        "type": "tag",
        "link": "tag_link",
        "studio": {
            "portal": false,
            "base": {
                "popuplist": false
            },
            "mobile": {
                "wirelesseditview": true,
                "wirelessdetailview": true
            }
        },
        "sortable": false,
        "label": "LBL_TAGS"
    },
    "team_name": {
        "type": "relate",
        "link": "team_link",
        "studio": {
            "portallistview": false,
            "portalrecordview": false
        },
        "label": "LBL_TEAMS"
    },
    "base_rate": {
        "type": "text",
        "label": "LBL_CURRENCY_RATE"
    },
    "name": {
        "type": "name",
        "link": true,
        "label": "LBL_NAME",
        "default": true
```

```
    },
    "account_name": {
        "type": "relate",
        "link": "account_link",
        "label": "LBL_ACCOUNT_NAME",
        "related_fields": [
            "account_id"
        ]
    },
    "status": {
        "type": "enum",
        "default": "",
        "label": "LBL_STATUS"
    },
    "quote_name": {
        "type": "relate",
        "link": true,
        "label": "LBL_ASSOCIATED_QUOTE",
        "related_fields": [
            "quote_id"
        ],
        "default": true
    },
    "discount_price": {
        "type": "currency",
        "default": "0",
        "related_fields": [
            "discount_price",
            "currency_id",
            "base_rate"
        ],
        "label": "LBL_DISCOUNT_PRICE"
    },
    "cost_price": {
        "type": "currency",
        "related_fields": [
            "cost_price",
            "currency_id",
            "base_rate"
        ],
        "label": "LBL_COST_PRICE"
    },
    "discount_amount": {
        "type": "discount",
        "default": "0",
        "related_fields": [
```

```
                "discount_select",
                "currency_id",
                "base_rate"
            ],
            "label": "LBL_DISCOUNT_AMOUNT"
        },
        "assigned_user_name": {
            "link": "assigned_user_link",
            "type": "relate",
            "label": "LBL_ASSIGNED_TO"
        }
    },
    "quotesFields": {
        "modified_by_name": {
            "type": "relate",
            "link": "modified_user_link",
            "readonly": true,
            "label": "LBL_MODIFIED"
        },
        "created_by_name": {
            "type": "relate",
            "link": "created_by_link",
            "readonly": true,
            "label": "LBL_CREATED"
        },
        "description": {
            "type": "text",
            "label": "LBL_DESCRIPTION"
        },
        "shipper_name": {
            "type": "relate",
            "link": "shippers",
            "label": "LBL_SHIPPING_PROVIDER"
        },
        "taxrate_name": {
            "type": "relate",
            "link": "taxrates",
            "label": "LBL_TAXRATE"
        },
        "show_line_nums": {
            "type": "bool",
            "default": 1,
            "label": "LBL_SHOW_LINE_NUMS"
        },
        "quote_type": {
            "type": "enum",
```

```
        "label": "LBL_QUOTE_TYPE"
    },
    "original_po_date": {
        "type": "date",
        "label": "LBL_ORIGINAL_PO_DATE"
    },
    "payment_terms": {
        "type": "enum",
        "label": "LBL_PAYMENT_TERMS"
    },
    "date_quote_closed": {
        "type": "date",
        "label": "LBL_DATE_QUOTE_CLOSED"
    },
    "date_order_shipped": {
        "type": "date",
        "label": "LBL_LIST_DATE_QUOTE_CLOSED"
    },
    "order_stage": {
        "type": "enum",
        "label": "LBL_ORDER_STAGE"
    },
    "purchase_order_num": {
        "type": "varchar",
        "label": "LBL_PURCHASE_ORDER_NUM"
    },
    "subtotal": {
        "type": "currency",
        "related_fields": [
            "currency_id",
            "base_rate"
        ],
        "label": "LBL_SUBTOTAL"
    },
    "subtotal_usdollar": {
        "type": "currency",
        "studio": {
            "wirelesseditview": false,
            "wirelessdetailview": false,
            "wirelesslistview": false,
            "wireless_basic_search": false,
            "wireless_advanced_search": false,
            "mobile": false
        },
        "related_fields": [
            "currency_id",
```

```
            "base_rate"
        ],
        "label": "LBL_SUBTOTAL_USDOLLAR"
    },
    "shipping": {
        "type": "currency",
        "related_fields": [
            "currency_id",
            "base_rate"
        ],
        "default": "0",
        "label": "LBL_SHIPPING"
    },
    "shipping_usdollar": {
        "type": "currency",
        "studio": {
            "wirelesseditview": false,
            "wirelessdetailview": false,
            "wirelesslistview": false,
            "wireless_basic_search": false,
            "wireless_advanced_search": false,
            "mobile": false
        },
        "related_fields": [
            "currency_id",
            "base_rate"
        ],
        "label": "LBL_SHIPPING_USDOLLAR"
    },
    "discount": {
        "type": "currency",
        "default": "0",
        "related_fields": [
            "currency_id",
            "base_rate"
        ],
        "label": "LBL_DISCOUNT_TOTAL"
    },
    "deal_tot": {
        "type": "currency",
        "label": "LBL_DEAL_TOT"
    },
    "deal_tot_discount_percentage": {
        "type": "float",
        "default": "0",
        "label": "LBL_DEAL_TOT_PERCENTAGE"
```

```
    },
    "deal_tot_usdollar": {
        "type": "currency",
        "studio": {
            "wirelesseditview": false,
            "wirelessdetailview": false,
            "wirelesslistview": false,
            "wireless_basic_search": false,
            "wireless_advanced_search": false,
            "mobile": false
        },
        "label": "LBL_DEAL_TOT_USDOLLAR"
    },
    "new_sub": {
        "type": "currency",
        "related_fields": [
            "currency_id",
            "base_rate"
        ],
        "label": "LBL_NEW_SUB"
    },
    "new_sub_usdollar": {
        "type": "currency",
        "studio": {
            "wirelesseditview": false,
            "wirelessdetailview": false,
            "wirelesslistview": false,
            "wireless_basic_search": false,
            "wireless_advanced_search": false,
            "mobile": false
        },
        "related_fields": [
            "currency_id",
            "base_rate"
        ],
        "label": "LBL_NEW_SUB_USDOLLAR"
    },
    "taxable_subtotal": {
        "type": "currency",
        "label": "LBL_TAXABLE_SUBTOTAL"
    },
    "tax": {
        "type": "currency",
        "related_fields": [
            "currency_id",
            "base_rate",
```

```
            "taxrate_value",
            "taxable_subtotal"
        ],
        "default": "0",
        "label": "LBL_TAX"
    },
    "tax_usdollar": {
        "type": "currency",
        "studio": {
            "wirelesseditview": false,
            "wirelessdetailview": false,
            "wirelesslistview": false,
            "wireless_basic_search": false,
            "wireless_advanced_search": false,
            "mobile": false
        },
        "related_fields": [
            "currency_id",
            "base_rate"
        ],
        "label": "LBL_TAX_USDOLLAR"
    },
    "billing_address_street": {
        "type": "text",
        "label": "LBL_BILLING_ADDRESS_STREET"
    },
    "billing_address_city": {
        "type": "varchar",
        "label": "LBL_BILLING_ADDRESS_CITY"
    },
    "billing_address_state": {
        "type": "varchar",
        "label": "LBL_BILLING_ADDRESS_STATE"
    },
    "billing_address_postalcode": {
        "type": "varchar",
        "label": "LBL_BILLING_ADDRESS_POSTAL_CODE"
    },
    "billing_address_country": {
        "type": "varchar",
        "label": "LBL_BILLING_ADDRESS_COUNTRY"
    },
    "shipping_address_street": {
        "type": "text",
        "label": "LBL_SHIPPING_ADDRESS_STREET"
    },
```

```
"shipping_address_city": {
    "type": "varchar",
    "label": "LBL_SHIPPING_ADDRESS_CITY"
},
"shipping_address_state": {
    "type": "varchar",
    "label": "LBL_SHIPPING_ADDRESS_STATE"
},
"shipping_address_postalcode": {
    "type": "varchar",
    "label": "LBL_SHIPPING_ADDRESS_POSTAL_CODE"
},
"shipping_address_country": {
    "type": "varchar",
    "label": "LBL_SHIPPING_ADDRESS_COUNTRY"
},
"shipping_account_name": {
    "type": "relate",
    "link": "shipping_accounts",
    "label": "LBL_SHIPPING_ACCOUNT_NAME"
},
"shipping_contact_name": {
    "type": "relate",
    "link": "shipping_contacts",
    "label": "LBL_SHIPPING_CONTACT_NAME"
},
"shipping_contact_id": {
    "type": "relate",
    "link": "shipping_contacts",
    "label": "LBL_SHIPPING_CONTACT_ID"
},
"billing_contact_name": {
    "type": "relate",
    "link": "billing_contacts",
    "label": "LBL_BILLING_CONTACT_NAME"
},
"billing_contact_id": {
    "type": "relate",
    "link": "billing_contacts",
    "label": "LBL_BILLING_CONTACT_ID"
},
"bundles": {
    "type": "collection",
    "label": "LBL_PRODUCT_BUNDLES"
},
"opportunity_name": {
```

```
        "type": "relate",
        "link": "opportunities",
        "label": "LBL_OPPORTUNITY_NAME"
    },
    "opportunity_id": {
        "type": "relate",
        "link": "opportunities",
        "label": "LBL_OPPORTUNITY_ID"
    },
    "tag": {
        "type": "tag",
        "link": "tag_link",
        "studio": {
            "portal": false,
            "base": {
                "popuplist": false
            },
            "mobile": {
                "wirelesseditview": true,
                "wirelessdetailview": true
            }
        },
        "sortable": false,
        "label": "LBL_TAGS"
    },
    "team_name": {
        "type": "relate",
        "link": "team_link",
        "studio": {
            "portallistview": false,
            "portalrecordview": false
        },
        "label": "LBL_TEAMS"
    },
    "base_rate": {
        "type": "text",
        "label": "LBL_CURRENCY_RATE"
    },
    "quote_num": {
        "type": "int",
        "readonly": true,
        "label": "LBL_LIST_QUOTE_NUM",
        "default": true
    },
    "name": {
        "type": "name",
```

```
        "label": "LBL_LIST_QUOTE_NAME",
        "default": true,
        "link": true
    },
    "billing_account_name": {
        "type": "relate",
        "link": "billing_accounts",
        "label": "LBL_LIST_ACCOUNT_NAME",
        "default": true,
        "sortable": false
    },
    "quote_stage": {
        "type": "enum",
        "label": "LBL_QUOTE_STAGE",
        "default": true
    },
    "total": {
        "type": "currency",
        "label": "LBL_TOTAL",
        "default": true,
        "related_fields": [
            "currency_id"
        ]
    },
    "total_usdollar": {
        "type": "currency",
        "studio": {
            "wirelesseditview": false,
            "wirelessdetailview": false,
            "wirelesslistview": false,
            "wireless_basic_search": false,
            "wireless_advanced_search": false,
            "mobile": false
        },
        "related_fields": [
            "currency_id",
            "base_rate"
        ],
        "label": "LBL_LIST_AMOUNT_USDOLLAR",
        "default": true
    },
    "date_quote_expected_closed": {
        "type": "date",
        "label": "LBL_LIST_DATE_QUOTE_EXPECTED_CLOSED",
        "default": true
    },
```

```
        "assigned_user_name": {
            "link": "assigned_user_link",
            "type": "relate",
            "label": "LBL_LIST_ASSIGNED_TO_NAME",
            "default": true,
            "sortable": false
        },
        "date_modified": {
            "type": "datetime",
            "studio": {
                "portaleditview": false
            },
            "readonly": true,
            "default": true,
            "label": "LBL_DATE_MODIFIED"
        },
        "date_entered": {
            "type": "datetime",
            "studio": {
                "portaleditview": false
            },
            "readonly": true,
            "default": true,
            "label": "LBL_DATE_ENTERED"
        }
    }
}
```

Change Log

Version	Change
v11.3	Added /Quotes/config GET endpoint.

/Quotes/config POST

Overview

Quote Config POST Help

Summary

This endpoint allows customizations to be made to the Quoted Line Items section

of the Quote record.

Request Arguments

Name	Type	Description	Required
worksheet_columns	array	A viewdef "fields"-style definition of the columns you want to display in the QLI section. These get written out to the custom/Products/clients/base/views/quote-data-group-list/quote-data-group-list.php file.	True
worksheet_column_related_fields	array	Array of the related field names needed for data to be returned for use by the quote-data-group-list	True
summary_columns	array	A viewdef "fields"-style definition of the columns you want to display in the Quote Summary header section. These get written out to the custom/Quotes/clients/base/views/quote-data-grand-totals-header/quote-data-grand-totals-header.php file.	True
summary_columns_related_fields	array	Array of the related field names needed for data to be returned for use by the quote-data-grand-totals-header	True

Name	Type	Description	Required
footer_rows	array	A viewdef "fields"-style definition of the columns you want to display in the Quote Grand Totals Footer section. These get written out to the custom/Quotes/clients/base/views/quote-data-grand-totals-footer/quote-data-grand-totals-footer.php file.	True
footer_rows_related_fields	array	Array of the related field names needed for data to be returned for use by the quote-data-grand-totals-footer	True

Sample Request

```
{
    "summary_columns": [
        {
            "name": "deal_tot",
            "label": "LBL_LIST_DEAL_TOT",
            "css_class": "quote-totals-row-item",
            "related_fields": [
                "deal_tot_discount_percentage"
            ],
            "type": "currency",
            "labelModule": "Quotes"
        },
        {
            "name": "new_sub",
            "css_class": "quote-totals-row-item",
            "type": "currency",
            "label": "LBL_NEW_SUB",
            "labelModule": "Quotes"
        },
        {
            "name": "tax",
```

```
            "label": "LBL_TAX_TOTAL",
            "css_class": "quote-totals-row-item",
            "type": "currency",
            "labelModule": "Quotes"
        },
        {
            "name": "shipping",
            "css_class": "quote-totals-row-item",
            "type": "currency",
            "label": "LBL_SHIPPING",
            "labelModule": "Quotes"
        },
        {
            "name": "total",
            "label": "LBL_LIST_GRAND_TOTAL",
            "css_class": "quote-totals-row-item",
            "type": "currency",
            "labelModule": "Quotes"
        }
    ],
    "summary_columns_related_fields": [
        "base_rate",
        "deal_tot",
        "deal_tot_usdollar",
        "shipping",
        "subtotal",
        "subtotal_usdollar",
        "tax",
        "taxable_subtotal"
    ],
    "worksheet_columns": [
        {
            "name": "line_num",
            "label": null,
            "widthClass": "cell-xsmall",
            "css_class": "line_num tcenter",
            "type": "line-num",
            "readonly": true
        },
        {
            "name": "quantity",
            "label": "LBL_QUANTITY",
            "widthClass": "cell-small",
            "css_class": "quantity",
            "type": "float",
            "labelModule": "Products"
```

```
    },
    {
        "name": "product_template_name",
        "label": "LBL_ITEM_NAME",
        "widthClass": "cell-large",
        "type": "quote-data-relate",
        "required": true,
        "labelModule": "Quotes"
    },
    {
        "name": "mft_part_num",
        "label": "LBL_MFT_PART_NUM",
        "type": "base",
        "labelModule": "Products"
    },
    {
        "name": "discount_price",
        "label": "LBL_DISCOUNT_PRICE",
        "type": "currency",
        "convertToBase": true,
        "showTransactionalAmount": true,
        "related_fields": [
            "discount_price",
            "currency_id",
            "base_rate"
        ],
        "labelModule": "Products"
    },
    {
        "name": "discount",
        "type": "fieldset",
        "css_class": "quote-discount-percent",
        "label": "LBL_DISCOUNT_AMOUNT",
        "fields": [
            {
                "name": "discount_amount",
                "label": "LBL_DISCOUNT_AMOUNT",
                "type": "discount",
                "convertToBase": true,
                "showTransactionalAmount": true
            },
            {
                "type": "discount-select",
                "name": "discount_select",
                "no_default_action": true,
                "buttons": [
```

```
                        {
                            "type": "rowaction",
                            "name": "select_discount_amount_button",
                            "label": "LBL_DISCOUNT_AMOUNT",
                            "event": "button:discount_select_change:cl
ick"
                        },
                        {
                            "type": "rowaction",
                            "name": "select_discount_percent_button",
                            "label": "LBL_DISCOUNT_PERCENT",
                            "event": "button:discount_select_change:cl
ick"
                        }
                    ],
                    "label": "LBL_DISCOUNT_AS_PERCENT"
                }
            ],
            "labelModule": "Products"
        },
        {
            "name": "total_amount",
            "label": "LBL_LINE_ITEM_TOTAL",
            "type": "currency",
            "widthClass": "cell-medium",
            "showTransactionalAmount": true,
            "related_fields": [
                "total_amount",
                "currency_id",
                "base_rate"
            ],
            "labelModule": "Quotes"
        }
    ],
    "worksheet_columns_related_fields": [
        "base_rate",
        "deal_calc",
        "discount_amount",
        "discount_price",
        "discount_select",
        "quantity",
        "subtotal",
        "tax_class",
        "total_amount",
        "description",
        "quote_id",
```

```
        "name",
        "product_template_id",
        "product_template_name"
    ],
    "footer_rows": [
        {
            "name": "new_sub",
            "type": "currency"
        },
        {
            "name": "tax",
            "type": "currency"
        },
        {
            "name": "shipping",
            "type": "quote-footer-currency",
            "css_class": "quote-footer-currency",
            "default": "0.00"
        },
        {
            "name": "total",
            "type": "currency",
            "css_class": "grand-total"
        }
    ],
    "footer_rows_related_fields": [
        "deal_tot",
        "deal_tot_usdollar",
        "shipping",
        "subtotal",
        "subtotal_usdollar",
        "tax",
        "taxable_subtotal"
    ]
}
```


Response Arguments

NONE

Response

Saved Config or SugarApiExceptionInvalidParameter

```
{
    "summary_columns": [
        {
            "name": "deal_tot",
            "label": "LBL_LIST_DEAL_TOT",
            "css_class": "quote-totals-row-item",
            "related_fields": [
                "deal_tot_discount_percentage"
            ],
            "type": "currency",
            "labelModule": "Quotes"
        },
        {
            "name": "new_sub",
            "css_class": "quote-totals-row-item",
            "type": "currency",
            "label": "LBL_NEW_SUB",
            "labelModule": "Quotes"
        },
        {
            "name": "tax",
            "label": "LBL_TAX_TOTAL",
            "css_class": "quote-totals-row-item",
            "type": "currency",
            "labelModule": "Quotes"
        },
        {
            "name": "shipping",
            "css_class": "quote-totals-row-item",
            "type": "currency",
            "label": "LBL_SHIPPING",
            "labelModule": "Quotes"
        },
        {
            "name": "total",
            "label": "LBL_LIST_GRAND_TOTAL",
            "css_class": "quote-totals-row-item",
            "type": "currency",
            "labelModule": "Quotes"
        }
    ],
    "worksheet_columns": [
        {
            "name": "line_num",
            "label": null,
            "widthClass": "cell-xsmall",
```

```
            "css_class": "line_num tcenter",
            "type": "line-num",
            "readonly": true
        },
        {
            "name": "quantity",
            "label": "LBL_QUANTITY",
            "widthClass": "cell-small",
            "css_class": "quantity",
            "type": "float",
            "labelModule": "Products"
        },
        {
            "name": "product_template_name",
            "label": "LBL_ITEM_NAME",
            "widthClass": "cell-large",
            "type": "quote-data-relate",
            "required": true,
            "labelModule": "Quotes"
        },
        {
            "name": "mft_part_num",
            "label": "LBL_MFT_PART_NUM",
            "type": "base",
            "labelModule": "Products"
        },
        {
            "name": "discount_price",
            "label": "LBL_DISCOUNT_PRICE",
            "type": "currency",
            "convertToBase": true,
            "showTransactionalAmount": true,
            "related_fields": [
                "discount_price",
                "currency_id",
                "base_rate"
            ],
            "labelModule": "Products"
        },
        {
            "name": "discount",
            "type": "fieldset",
            "css_class": "quote-discount-percent",
            "label": "LBL_DISCOUNT_AMOUNT",
            "fields": [
                {
```

```
                    "name": "discount_amount",
                    "label": "LBL_DISCOUNT_AMOUNT",
                    "type": "discount",
                    "convertToBase": true,
                    "showTransactionalAmount": true
                },
                {
                    "type": "discount-select",
                    "name": "discount_select",
                    "no_default_action": true,
                    "buttons": [
                        {
                            "type": "rowaction",
                            "name": "select_discount_amount_button",
                            "label": "LBL_DISCOUNT_AMOUNT",
                            "event": "button:discount_select_change:cl
ick"
                        },
                        {
                            "type": "rowaction",
                            "name": "select_discount_percent_button",
                            "label": "LBL_DISCOUNT_PERCENT",
                            "event": "button:discount_select_change:cl
ick"
                        }
                    ],
                    "label": "LBL_DISCOUNT_AS_PERCENT"
                }
            ],
            "labelModule": "Products"
        },
        {
            "name": "total_amount",
            "label": "LBL_LINE_ITEM_TOTAL",
            "type": "currency",
            "widthClass": "cell-medium",
            "showTransactionalAmount": true,
            "related_fields": [
                "total_amount",
                "currency_id",
                "base_rate"
            ],
            "labelModule": "Quotes"
        }
    ],
    "footer_rows": [
```

```
        {
            "name": "new_sub",
            "type": "currency"
        },
        {
            "name": "tax",
            "type": "currency"
        },
        {
            "name": "shipping",
            "type": "quote-footer-currency",
            "css_class": "quote-footer-currency",
            "default": "0.00"
        },
        {
            "name": "total",
            "type": "currency",
            "css_class": "grand-total"
        }
    ],
    "summary_columns_related_fields": [
        "base_rate",
        "deal_tot",
        "deal_tot_usdollar",
        "shipping",
        "subtotal",
        "subtotal_usdollar",
        "tax",
        "taxable_subtotal"
    ],
    "worksheet_columns_related_fields": [
        "base_rate",
        "deal_calc",
        "discount_amount",
        "discount_price",
        "discount_select",
        "quantity",
        "subtotal",
        "tax_class",
        "total_amount",
        "description",
        "quote_id",
        "name",
        "product_template_id",
        "product_template_name"
    ],
```

```
    "footer_rows_related_fields": [
        "deal_tot",
        "deal_tot_usdollar",
        "shipping",
        "subtotal",
        "subtotal_usdollar",
        "tax",
        "taxable_subtotal"
    ]
}
```

Change Log

Version	Change
v11.3	Added /Quotes/config POST endpoint.

/Reports/:record/:type GET

Overview

An API to run a saved report and export the result.

Summary

This endpoint will export the result of a saved report in one of the following formats: pdf, base64, csv, and json ('rows and columns' reports only).

Request Arguments

Name	Type	Description	Required
record	String	ID of a saved report	True
export_type	String	Format of exported data. It can be one of the following formats: pdf, base64, csv, and json. CSV and JSON export types are only currently available for 'Rows	True

Name	Type	Description	Required
		and Columns' reports.	

Request Example

```
GET /rest/v11/Reports/:id/json
```

Response Example

```
[
  {
      "Contact ID":"fa300a0e-0ad1-b322-9601-512d0983c19a",
      "First Name":"Dale",
      "Last Name":"Arvizu",
      "Email Address":"vegan.sugar.support@example.co.uk",
      "Account Name":"DD Furniture Inc"
  },
  {
      "Contact ID":"71c773ce-980e-11eb-8f85-f45c898a3ce7",
      "First Name":"Isis",
      "Last Name":"Arvizu",
      "Email Address":"im38@example.info",
      "Account Name":"J.K.M. Corp (HA)"
  },
]
```

Change Log

Version	Change
v11.13	Added csv and json formats.

/Reports/:record/chart GET

Overview

An API to get chart data for a saved report.

Summary

This endpoint will return chart data for a saved report.

Request Arguments

None.

Request Example

```
GET /rest/v10/Reports/:id/chart
```

Response Arguments

Name	Type	Description
reportData	Array	Report def for the chart.
chartData	Array	The chart data for a report.

Response

```
{
    "reportData":
    {
        "label":"Accounts by Industry",
        "id":"a06b4212-3509-11e7-ab6e-f45c898a3ce7",
        ...
    },
    "chartData":
    {
        "properties":[...],
        "values":[...],
        ...
    }
}
```

/Reports/:record/record_count GET

Overview

An API to get total number of filtered records from a saved report.

Summary

This endpoint will return total number of records from a saved report filtered by an expression.

Request Arguments

Name	Type	Description	Required
group_filters	String	The group filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query parameters in one of two ways for GET requests: 1. By specifying individual filter arguments as distinct parameters. Example: group_filters[0][name]=value. 2. By specifying the whole filter as a single JSON-encoded string. Example: group_filters=[{"name":"value"}]. Example: group_filt	False

Name	Type	Description	Required
		ers[0][industry]=Engineering.	
use_saved_filters	Boolean	Flag to indicate whether or not to apply saved runtime filters if exists. The default value is false. Example: use_saved_filters=true.	False

Request Example

```
GET /rest/v10/Reports/:id/record_count?group_filters[0][industry]=Engineering
```

Response Arguments

Name	Type	Description
record_count	Integer	Total number of filtered records.

Response

```
{
   "record_count": 5
}
```

/Reports/:record/records GET

Overview

An API to deliver filtered records from a saved report.

Summary

This endpoint will return a list of records from a saved report filtered by an expression.

Request Arguments

Name	Type	Description	Required
group_filters	String	The group filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query parameters in one of two ways for GET requests: 1. By specifying individual filter arguments as distinct parameters. Example: group_filters[0][name]=value. 2. By specifying the whole filter as a single JSON-encoded string. Example: group_filters=[{"name":"value"}]. Example: group_filters[0][industry]=Engineering.	False
use_saved_filters	Boolean	Flag to indicate whether or not to apply saved runtime filters if exists. The default	

Name	Type	Description	Required
		value is false. Example: use_saved_filters=true.	
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False

Request Example

```
GET /rest/v10/Reports/:id/records?group_filters[0][industry]=Engineering&use_saved_filters=true&offset=0&max_num=20
```

Response Arguments

Name	Type	Description
records	Array	An array of results containing matched records.
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.

Response

```
{
   "records":[
      {
         "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
         "name":"Dale Spivey",
         "date_modified":"2013-02-28T05:03:00+00:00",
         "description":"",
         "_acl": {
```

```
            "fields": {
            }
         }
      },
      {
         "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
         "name":"Florence Haddock",
         "date_modified":"2013-02-26T19:12:00+00:00",
         "description":"",
         "_acl": {
            "fields": {
            }
         }
      }
   ],
   "next_offset": -1
}
```


/RevenueLineItems/:record/quote POST

Convert a Revenue Line Item to a quote.

Summary:

This endpoint is used to convert a single RevenueLineItem to a quote

Query Parameters:

Param	Description	Optional
record	Which Revenue Line Item to convert	false

Valid Urls:

- /rest/v10/RevenueLineItems/:record/quote/

Output Example:

{ id:"123-456-789-0", name:"My Test Quote" }

/RevenueLineItems/multi-quote POST

Convert Multiple Revenue Line Item to a quote.

Summary:

This endpoint is used to convert multiple Revenue Line Items to a single quote

Query Parameters:

Param	Description	Optional

Valid Urls:

- POST: /rest/v10/RevenueLineItems/multi-quote

Input Example:

{ records : ['id_1', 'id_2',] opportunity_id:"123-456-789-0", account_id:"12-3-4-5" }

Output Example:

{ id:"123-456-789-0", name:"My Test Quote" }

/Tags/:record PUT

Overview

Update a record of the specified type.

Request Arguments

Name	Type	Description	Required
<record field>	<record field type>	The name value list of fields to populate.	True

Request

```
{
    "name":"New Account Name",
    "phone_office":"(555) 888-5555",
    "website":"www.newsite.com",
    "meetings": {
    {
        "add": ["21e3385e-404f-b470-407e-54044e3d8024"],
        "delete": ["21e3385e-404f-b470-407e-54044e3d8023"],
        "create": [
        {
            "name": "Test Meeting"
        }
      ]
    }
}
```

Link fields

It is possible to add or delete record relations to other records and create new related records.

Action	Type	Description
add	List	List of related records ID. See RelateRecordApi for details.
delete	List	List of related records ID.
create	List	List of name to value arrays of

Action		Type		Description
				related records.

Response Arguments

Name	Type	Description
<record field>	<record field type>	Returns the fields for the updated record.

Response


```
{
   "id":"f222265a-b755-da89-0bc7-512d09b800b6",
   "name":"New Account Name",
   "date_entered":"2013-02-26T19:12:00+00:00",
   "date_modified":"2013-02-27T22:49:00+00:00",
   "modified_user_id":"1",
   "modified_by_name":"Administrator",
   "created_by":"1",
   "created_by_name":"Administrator",
   "description":"",
   "img":"",
   "last_activity_date":"2013-02-26T19:12:00+00:00",
   "deleted":false,
   "assigned_user_id":"seed_sarah_id",
   "assigned_user_name":"Sarah Smith",
   "team_name":[
      {
         "id":1,
         "name":"Global",
         "name_2":"",
         "primary":false
      },
      {
         "id":"West",
         "name":"West",
         "name_2":"",
         "primary":true
      }
   ],
   "linkedin":"",
   "facebook":"",
   "twitter":"",
   "googleplus":"",
   "account_type":"Customer",
```

```
"industry":"Hospitality",
"annual_revenue":"",
"phone_fax":"",
"billing_address_street":"9 IBM Path",
"billing_address_street_2":"",
"billing_address_street_3":"",
"billing_address_street_4":"",
"billing_address_city":"San Francisco",
"billing_address_state":"CA",
"billing_address_postalcode":"43635",
"billing_address_country":"USA",
"rating":"",
"phone_office":"(555) 888-5555",
"phone_alternate":"",
"website":"www.newsite.com",
"ownership":"",
"employees":"",
"ticker_symbol":"",
"shipping_address_street":"9 IBM Path",
"shipping_address_street_2":"",
"shipping_address_street_3":"",
"shipping_address_street_4":"",
"shipping_address_city":"San Francisco",
"shipping_address_state":"CA",
"shipping_address_postalcode":"43635",
"shipping_address_country":"USA",
"email1":"kid86@example.net",
"parent_id":"",
"sic_code":"",
"parent_name":"",
"email_opt_out":false,
"invalid_email":false,
"email":[
   {
      "email_address":"kid86@example.net",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"1"
   },
   {
      "email_address":"the.dev@example.name",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"0"
   }
],
```

```
   "campaign_id":"",
   "campaign_name":"",
   "my_favorite":false,
   "_acl":{
      "fields":{

      }
   }
}
```

Change Log

Version	Change
v10	Added /<module>/:record PUT endpoint.

/Teams/:record/link POST

Overview

Creates relationships to a pre-existing record.

Request Arguments

Name	Type	Description	Required
<relationship link>	<string>	Link between targeted and related records.	True
<record ID>	<string>	The name value list of related records. Each item of the list may be either string equal to related item ID, or map containing record ID ("id" key is required in this case) and addition relationship properties.	True

Request

```
{
    link_name: "accounts"
    ids: [
        "da6a3741-2a81-ba7f-f249-512d0932e94e",
        {
            "id": "e689173e-c953-1e14-c215-512d0927e7a2",
            "role": "owner"
        }
    ]
}
```

Response Arguments

Name	Type	Description
record	Array	The record linked to related records.
related_records	Array	Records that were associated.

Response

```
    "record": {
        "id": "da6a3741-2a81-ba7f-f249-512d0932e94e",
        "name": "Slender Broadband Inc - 1000 units",
        "date_entered": "2013-02-26T19:12:00+00:00",
        "date_modified": "2013-02-26T19:12:00+00:00",
        "modified_user_id": "1",
        "modified_by_name": "Administrator",
        "created_by": "1",
        "created_by_name": "Administrator",
        "description": "",
        "img": "",
        "last_activity_date": "2013-02-28T18:21:00+00:00",
        "deleted": false,
        "assigned_user_id": "seed_max_id",
        "assigned_user_name": "Max Jensen",
        "team_name": [
            {
                "id": "East",
```

```
                "name": "East",
                "name_2": "",
                "primary": false
            },
            {
                "id": "West",
                "name": "West",
                "name_2": "",
                "primary": true
            }
        ],
        "opportunity_type": "",
        "account_name": "Slender Broadband Inc",
        "account_id": "181461c6-dc81-1115-1fe0-512d092e8f15",
        "campaign_id": "",
        "campaign_name": "",
        "lead_source": "Campaign",
        "amount": "25000",
        "base_rate": "1",
        "amount_usdollar": "25000",
        "currency_id": "-99",
        "currency_name": "",
        "currency_symbol": "",
        "date_closed": "2013-02-27",
        "date_closed_timestamp": "1361992480",
        "next_step": "",
        "sales_stage": "Needs Analysis",
        "sales_status": "New",
        "probability": "90",
        "best_case": "25000",
        "worst_case": "25000",
        "commit_stage": "include",
        "my_favorite": false,
        "_acl": {
            "fields": {

            }
        }
    },
    "related_records": [
        {
            "id": "e689173e-c953-1e14-c215-512d0927e7a2",
            "name": "Gus Dales",
            "date_entered": "2013-02-26T19:12:00+00:00",
            "date_modified": "2013-02-26T19:12:00+00:00",
            "modified_user_id": "1",
```

```
            "modified_by_name": "Administrator",
            "created_by": "1",
            "created_by_name": "Administrator",
            "description": "",
            "img": "",
            "deleted": false,
            "assigned_user_id": "seed_sally_id",
            "assigned_user_name": "Sally Bronsen",
            "team_name": [
                {
                    "id": "West",
                    "name": "West",
                    "name_2": "",
                    "primary": true
                }
            ],
            "salutation": "",
            "first_name": "Gus",
            "last_name": "Dales",
            "full_name": "Gus Dales",
            "title": "Director Operations",
            "linkedin": "",
            "facebook": "",
            "twitter": "",
            "googleplus": "",
            "department": "",
            "do_not_call": false,
            "phone_home": "(661) 120-2292",
            "email": [
                {
                    "email_address": "section.sugar.section@example.it
",
                    "opt_out": "1",
                    "invalid_email": "0",
                    "primary_address": "0"
                },
                {
                    "email_address": "support.qa.kid@example.co.uk",
                    "opt_out": "0",
                    "invalid_email": "0",
                    "primary_address": "1"
                }
            ],
            "phone_mobile": "(294) 447-9707",
            "phone_work": "(036) 840-3216",
            "phone_other": "",
```

```
"phone_fax": "",
"email1": "support.qa.kid@example.co.uk",
"email2": "section.sugar.section@example.it",
"invalid_email": false,
"email_opt_out": false,
"primary_address_street": "48920 San Carlos Ave",
"primary_address_street_2": "",
"primary_address_street_3": "",
"primary_address_city": "Persistance",
"primary_address_state": "CA",
"primary_address_postalcode": "54556",
"primary_address_country": "USA",
"alt_address_street": "",
"alt_address_street_2": "",
"alt_address_street_3": "",
"alt_address_city": "",
"alt_address_state": "",
"alt_address_postalcode": "",
"alt_address_country": "",
"assistant": "",
"assistant_phone": "",
"picture": "",
"email_and_name1": "",
"lead_source": "Support Portal User Registration",
"account_name": "Arts & Crafts Inc",
"account_id": "d43243c6-9b8e-2973-aee2-512d09bc34b4",
"opportunity_role_fields": "",
"opportunity_role_id": "",
"opportunity_role": "Technical Advisor",
"reports_to_id": "",
"report_to_name": "",
"portal_name": "GusDales145",
"portal_active": true,
"portal_password": "$1$JxYr6tmM$b.O6.KF42jP46RadSwz0N0",
"portal_password1": "",
"portal_app": "",
"preferred_language": "en_us",
"campaign_id": "",
"campaign_name": "",
"c_accept_status_fields": "",
"m_accept_status_fields": "",
"accept_status_id": "",
"accept_status_name": "",
"sync_contact": "",
"my_favorite": false,
"_acl": {
```

```
            "fields": {
            }
        }
    },
    {
        "id": "da6a3741-2a81-ba7f-f249-512d0932e94e",
        "name": "Slender Broadband Inc - 1000 units",
        "date_entered": "2013-02-26T19:12:00+00:00",
        "date_modified": "2013-02-26T19:12:00+00:00",
        "modified_user_id": "1",
        "modified_by_name": "Administrator",
        "created_by": "1",
        "created_by_name": "Administrator",
        "description": "",
        "img": "",
        "last_activity_date": "2013-02-28T18:36:00+00:00",
        "deleted": false,
        "assigned_user_id": "seed_max_id",
        "assigned_user_name": "Max Jensen",
        "team_name": [
            {
                "id": "East",
                "name": "East",
                "name_2": "",
                "primary": false
            },
            {
                "id": "West",
                "name": "West",
                "name_2": "",
                "primary": true
            }
        ],
        "opportunity_type": "",
        "account_name": "Slender Broadband Inc",
        "account_id": "181461c6-dc81-1115-1fe0-512d092e8f15",
        "campaign_id": "",
        "campaign_name": "",
        "lead_source": "Campaign",
        "amount": "25000",
        "base_rate": "1",
        "amount_usdollar": "25000",
        "currency_id": "-99",
        "currency_name": "",
        "currency_symbol": "",
        "date_closed": "2013-02-27",
```

```
            "date_closed_timestamp": "1361992480",
            "next_step": "",
            "sales_stage": "Needs Analysis",
            "sales_status": "New",
            "probability": "90",
            "best_case": "25000",
            "worst_case": "25000",
            "commit_stage": "include",
            "my_favorite": false,
            "_acl": {
                "fields": {

                }
            }
        }
    ]
}
```


Change Log

Version	Change
v10 (7.6.0)	Added support for additional relationship values.
v10 (7.1.5)	Added /<module>/:record/link POST endpoint.

/Teams/:record/link/:link_name/:remote_id DELETE

Overview

Deletes an existing relationship between two records.

Request Arguments

This endpoint does not accept any arguments.

Response Arguments

Name	Type	Description
record	Array	The record to disassociate from the related record.
related_record	Array	The record that was disassociated.

Response


```
{
   "record":{
      "id":"da6a3741-2a81-ba7f-f249-512d0932e94e",
      "name":"Slender Broadband Inc - 1000 units",
      "date_entered":"2013-02-26T19:12:00+00:00",
      "date_modified":"2013-02-26T19:12:00+00:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "description":"",
      "img":"",
      "last_activity_date":"2013-02-28T18:36:00+00:00",
      "deleted":false,
      "assigned_user_id":"seed_max_id",
      "assigned_user_name":"Max Jensen",
      "team_name":[
         {
            "id":"East",
            "name":"East",
            "name_2":"",
            "primary":false
         },
         {
            "id":"West",
            "name":"West",
            "name_2":"",
            "primary":true
         }
      ],
      "opportunity_type":"",
      "account_name":"Slender Broadband Inc",
      "account_id":"181461c6-dc81-1115-1fe0-512d092e8f15",
      "campaign_id":"",
      "campaign_name":"",
      "lead_source":"Campaign",
```

```
    "amount":"25000",
    "base_rate":"1",
    "amount_usdollar":"25000",
    "currency_id":"-99",
    "currency_name":"",
    "currency_symbol":"",
    "date_closed":"2013-02-27",
    "date_closed_timestamp":"1361992480",
    "next_step":"",
    "sales_stage":"Needs Analysis",
    "sales_status":"New",
    "probability":"90",
    "best_case":"25000",
    "worst_case":"25000",
    "commit_stage":"include",
    "my_favorite":false,
    "_acl":{
       "fields":{

       }
    }
 },
 "related_record":{
    "id":"e689173e-c953-1e14-c215-512d0927e7a2",
    "name":"Gus Dales",
    "date_entered":"2013-02-26T19:12:00+00:00",
    "date_modified":"2013-02-26T19:12:00+00:00",
    "modified_user_id":"1",
    "modified_by_name":"Administrator",
    "created_by":"1",
    "created_by_name":"Administrator",
    "description":"",
    "img":"",
    "deleted":false,
    "assigned_user_id":"seed_sally_id",
    "assigned_user_name":"Sally Bronsen",
    "team_name":[
       {
          "id":"West",
          "name":"West",
          "name_2":"",
          "primary":true
       }
    ],
    "salutation":"",
    "first_name":"Gus",
```

```
"last_name":"Dales",
"full_name":"Gus Dales",
"title":"Director Operations",
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(661) 120-2292",
"email":[
   {
      "email_address":"section.sugar.section@example.it",
      "opt_out":"1",
      "invalid_email":"0",
      "primary_address":"0"
   },
   {
      "email_address":"support.qa.kid@example.co.uk",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"1"
   }
],
"phone_mobile":"(294) 447-9707",
"phone_work":"(036) 840-3216",
"phone_other":"",
"phone_fax":"",
"email1":"support.qa.kid@example.co.uk",
"email2":"section.sugar.section@example.it",
"invalid_email":false,
"email_opt_out":false,
"primary_address_street":"48920 San Carlos Ave",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Persistance",
"primary_address_state":"CA",
"primary_address_postalcode":"54556",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
```

```
      "assistant":"",
      "assistant_phone":"",
      "picture":"",
      "email_and_name1":"",
      "lead_source":"Support Portal User Registration",
      "account_name":"Arts & Crafts Inc",
      "account_id":"d43243c6-9b8e-2973-aee2-512d09bc34b4",
      "opportunity_role_fields":"",
      "opportunity_role_id":"",
      "opportunity_role":"",
      "reports_to_id":"",
      "report_to_name":"",
      "portal_name":"GusDales145",
      "portal_active":true,
      "portal_password":"$1$JxYr6tmM$b.O6.KF42jP46RadSwz0N0",
      "portal_password1":"",
      "portal_app":"",
      "preferred_language":"en_us",
      "campaign_id":"",
      "campaign_name":"",
      "c_accept_status_fields":"",
      "m_accept_status_fields":"",
      "accept_status_id":"",
      "accept_status_name":"",
      "sync_contact":"",
      "my_favorite":false,
      "_acl":{
         "fields":{

         }
      }
   }
}
```


Change Log

Version	Change
v10	Added /<module>/:record/link/:link_name/:remote_id DELETE endpoint.

/Teams/:record/link/:link_name/:remote_id POST

Overview

Creates a relationship to a pre-existing record.

Query String Parameters

This endpoint does not accept any arguments.

Response Arguments

Name	Type	Description
record	Array	The record linked to the related record.
related_record	Array	The record that was associated.

Response


```
{
   "record":{
      "id":"da6a3741-2a81-ba7f-f249-512d0932e94e",
      "name":"Slender Broadband Inc - 1000 units",
      "date_entered":"2013-02-26T19:12:00+00:00",
      "date_modified":"2013-02-26T19:12:00+00:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "description":"",
      "img":"",
      "last_activity_date":"2013-02-28T18:21:00+00:00",
      "deleted":false,
      "assigned_user_id":"seed_max_id",
      "assigned_user_name":"Max Jensen",
      "team_name":[
         {
            "id":"East",
            "name":"East",
            "name_2":"",
            "primary":false
         },
         {
            "id":"West",
            "name":"West",
            "name_2":"",
```

```
            "primary":true
         }
      ],
      "opportunity_type":"",
      "account_name":"Slender Broadband Inc",
      "account_id":"181461c6-dc81-1115-1fe0-512d092e8f15",
      "campaign_id":"",
      "campaign_name":"",
      "lead_source":"Campaign",
      "amount":"25000",
      "base_rate":"1",
      "amount_usdollar":"25000",
      "currency_id":"-99",
      "currency_name":"",
      "currency_symbol":"",
      "date_closed":"2013-02-27",
      "date_closed_timestamp":"1361992480",
      "next_step":"",
      "sales_stage":"Needs Analysis",
      "sales_status":"New",
      "probability":"90",
      "best_case":"25000",
      "worst_case":"25000",
      "commit_stage":"include",
      "my_favorite":false,
      "_acl":{
         "fields":{

         }
      }
   },
   "related_record":{
      "id":"e689173e-c953-1e14-c215-512d0927e7a2",
      "name":"Gus Dales",
      "date_entered":"2013-02-26T19:12:00+00:00",
      "date_modified":"2013-02-26T19:12:00+00:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "description":"",
      "img":"",
      "deleted":false,
      "assigned_user_id":"seed_sally_id",
      "assigned_user_name":"Sally Bronsen",
      "team_name":[
```

```
      {
         "id":"West",
         "name":"West",
         "name_2":"",
         "primary":true
      }
   ],
   "salutation":"",
   "first_name":"Gus",
   "last_name":"Dales",
   "full_name":"Gus Dales",
   "title":"Director Operations",
   "linkedin":"",
   "facebook":"",
   "twitter":"",
   "googleplus":"",
   "department":"",
   "do_not_call":false,
   "phone_home":"(661) 120-2292",
   "email":[
      {
         "email_address":"section.sugar.section@example.it",
         "opt_out":"1",
         "invalid_email":"0",
         "primary_address":"0"
      },
      {
         "email_address":"support.qa.kid@example.co.uk",
         "opt_out":"0",
         "invalid_email":"0",
         "primary_address":"1"
      }
   ],
   "phone_mobile":"(294) 447-9707",
   "phone_work":"(036) 840-3216",
   "phone_other":"",
   "phone_fax":"",
   "email1":"support.qa.kid@example.co.uk",
   "email2":"section.sugar.section@example.it",
   "invalid_email":false,
   "email_opt_out":false,
   "primary_address_street":"48920 San Carlos Ave",
   "primary_address_street_2":"",
   "primary_address_street_3":"",
   "primary_address_city":"Persistance",
   "primary_address_state":"CA",
```

```
      "primary_address_postalcode":"54556",
      "primary_address_country":"USA",
      "alt_address_street":"",
      "alt_address_street_2":"",
      "alt_address_street_3":"",
      "alt_address_city":"",
      "alt_address_state":"",
      "alt_address_postalcode":"",
      "alt_address_country":"",
      "assistant":"",
      "assistant_phone":"",
      "picture":"",
      "email_and_name1":"",
      "lead_source":"Support Portal User Registration",
      "account_name":"Arts & Crafts Inc",
      "account_id":"d43243c6-9b8e-2973-aee2-512d09bc34b4",
      "opportunity_role_fields":"",
      "opportunity_role_id":"",
      "opportunity_role":"Technical Advisor",
      "reports_to_id":"",
      "report_to_name":"",
      "portal_name":"GusDales145",
      "portal_active":true,
      "portal_password":"$1$JxYr6tmM$b.O6.KF42jP46RadSwz0N0",
      "portal_password1":"",
      "portal_app":"",
      "preferred_language":"en_us",
      "campaign_id":"",
      "campaign_name":"",
      "c_accept_status_fields":"",
      "m_accept_status_fields":"",
      "accept_status_id":"",
      "accept_status_name":"",
      "sync_contact":"",
      "my_favorite":false,
      "_acl":{
         "fields":{

         }
      }
   }
}
```

Change Log

Version	Change
v10	Added /<module>/:record/link/:link_name/:remote_id POST endpoint.

/TimePeriods GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

Name	Type	Description	Required
filter	String	The filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query parameters in one of two ways for GET requests: 1. By specifying individual filter arguments as distinct	False

Name	Type	Description	Required
		parameters. Example: filter[0][id]=1. 2. By specifying the whole filter as a single JSON-encoded string. Note that this syntax is currently not supported on certain modules. Example: filter=[{"id":"1"}].	
filter_id	String	Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND.	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
fields	String	Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and	False

Name	Type	Description	Required
		additional field parameters (applicable to link and collection fields). The fields id and date_modified will always be returned. Example: name,account_type,description,{"name":"opportunities","fields":["id","name","sales_status"],"order_by":"date_closed:desc"} For more details on additional field parameters, see Relate API and Collection API.	
view	String	Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list". Example: record	False
order_by	String	How to sort the returned records, in a comma delimited list with the direction	False

Name	Type	Description	Required
		appended to the column name after a colon. Example: name:DESC,account_type:DESC,date_modified:ASC	
q	String	A search expression, will search on this module. Cannot be used at the same time as a filter expression or id.	False
deleted	Boolean	Boolean to show deleted records in the result set.	False
nulls_last	Boolean	Boolean to return records with null values in order_by fields last in the result set.	False

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
   "filter":[
      {
         "name":"Nelson Inc"
      }
   ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
   "filter":[
      {
         "name":{
            "$starts":"Nelson"
         }
      }
   ]
}
```

Below is a list of operation types:

Operation	Description
$equals	Performs an exact match on that field.
$not_equals	Performs an exact match on that field.
$not_equals	Matches on non-matching values.
$starts	Matches on anything that starts with the value.
$ends	Matches anything that ends with the value.
$contains	Matches anything that contains the value
$in	Finds anything where field matches one of the values as specified as an array.
$not_in	Finds anything where field does not matches any of the values as specified as an array.

Operation		Description
	$is_null	Checks if the field is null. This operation does not need a value specified.
	$not_null	Checks if the field is not null. This operation does not need a value specified.
	$lt	Matches when the field is less than the value.
	$lte	Matches when the field is less than or equal to the value.
	$gt	Matches when the field is greater than the value.
	$gte	Matches when the field is greater than or equal to the value.

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "$and" and "$or".

Example

```
{
   "filter":[
      {
         "$or":[
            {
               "name":"Nelson Inc"
            },
            {
               "name":"Nelson LLC"
            }
         ]
```

```
      }
   ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "$favorite" and "$owner".

Example

```
{
   "filter":[
      {
         "$favorite":"_this"
      }
   ]
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Response

```
{
   "next_offset":-1,
```

```
   "records":[
      {
         "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
         "name":"Dale Spivey",
         "date_modified":"2013-02-28T05:03:00+00:00",
         "description":"",
         "opportunities": [
            {
               _module: "Opportunities",
               "id": "b0701501-1fab-8ae7-3942-540da93f5017",
               "name": "360 Vacations - 228 Units",
               "date_modified": "2014-09-08T16:05:00+03:00",
               "sales_status": "New"
            },
         ],
         "_acl": {
            "fields": {
            }
         }
      },
      {
         "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
         "name":"Florence Haddock",
         "date_modified":"2013-02-26T19:12:00+00:00",
         "description":"",
         "opportunities": [
            {
               _module: "Opportunities"
               date_modified: "2014-09-08T16:05:00+03:00"
               id: "9ce7c088-8ee4-7cd3-18f1-540da944d4c0"
               name: "360 Vacations - 312 Units"
               sales_status: "New"
            },
         ],
         "_acl": {
            "fields": {
            }
         }
      }
   ]
}
```


Change Log

Version	Change
v10	Added /<module>/filter GET endpoint.

/TimePeriods/:date GET

Return a Timeperiod by a given date

Summary:

This endpoint is used to get a TimePeriod for a given date. If one is not found a 404 is returned.

Output Example:

{ "id":"e394485a-5889-ea75-84c8-51543bd1a5ba", "name":"Q1 (01\/01\/2013 - 03\/31\/2013)", "parent_id":"e28efe2d-c51c-be45-3d84-51543b4d2910", "start_date":"2013-01-01", "start_date_timestamp":"1356998400", "end_date":"2013-03-31", "end_date_timestamp":"1364774399", "created_by":"1", "date_entered":"2013-03-28 12:46:36", "date_modified":"2013-03-28 12:46:36", "deleted":"0", "is_fiscal":"0", "is_fiscal_year":"0", "leaf_cycle":"1", "type":"Quarter", "related_timeperiods":"" }

/TimePeriods/current GET

Get the get the current timeperiod

Summary:

This endpoint is used to get the current TimePeriod. If one is not found a 404 is returned

Output Example:

{ "id":"e394485a-5889-ea75-84c8-51543bd1a5ba", "name":"Q1 (01\/01\/2013 - 03\/31\/2013)", "parent_id":"e28efe2d-c51c-be45-3d84-51543b4d2910", "start_date":"2013-01-01", "start_date_timestamp":"1356998400",

"end_date":"2013-03-31", "end_date_timestamp":"1364774399", "created_by":"1", "date_entered":"2013-03-28 12:46:36", "date_modified":"2013-03-28 12:46:36", "deleted":"0", "is_fiscal":"0", "is_fiscal_year":"0", "leaf_cycle":"1", "type":"Quarter", "related_timeperiods":"" }

/TimePeriods/filter GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

Name	Type	Description	Required
filter	String	The filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query parameters in one of two ways for GET requests: 1. By specifying individual filter arguments as distinct parameters. Example: fil	False

Name	Type	Description	Required
		ter[0][id]=1 . 2. By specifying the whole filter as a single JSON-encoded string. Note that this syntax is currently not supported on certain modules. Example: filter=[{"id":"1"}].	
filter_id	String	Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND.	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
fields	String	Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters	False

Name	Type	Description	Required
		(applicable to link and collection fields). The fields id and date_modified will always be returned. Example: name,account_type,description,{"name":"opportunities","fields":["id","name","sales_status"],"order_by":"date_closed:desc"} For more details on additional field parameters, see Relate API and Collection API.	
view	String	Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list". Example: record	False
order_by	String	How to sort the returned records, in a comma delimited list with the direction appended to the column name after	False

Name	Type	Description	Required
		a colon. Example: name:DESC,account_type:DESC,date_modified:ASC	
q	String	A search expression, will search on this module. Cannot be used at the same time as a filter expression or id.	False
deleted	Boolean	Boolean to show deleted records in the result set.	False
nulls_last	Boolean	Boolean to return records with null values in order_by fields last in the result set.	False

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
   "filter":[
      {
         "name":"Nelson Inc"
      }
   ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
   "filter":[
      {
         "name":{
            "$starts":"Nelson"
         }
      }
   ]
}
```

Below is a list of operation types:

Operation	Description
$equals	Performs an exact match on that field.
$not_equals	Performs an exact match on that field.
$not_equals	Matches on non-matching values.
$starts	Matches on anything that starts with the value.
$ends	Matches anything that ends with the value.
$contains	Matches anything that contains the value
$in	Finds anything where field matches one of the values as specified as an array.
$not_in	Finds anything where field does not matches any of the values as specified as an array.
$is_null	Checks if the field is

Operation		Description
		null. This operation does not need a value specified.
	$not_null	Checks if the field is not null. This operation does not need a value specified.
	$lt	Matches when the field is less than the value.
	$lte	Matches when the field is less than or equal to the value.
	$gt	Matches when the field is greater than the value.
	$gte	Matches when the field is greater than or equal to the value.

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "$and" and "$or".

Example

```
{
   "filter":[
      {
         "$or":[
            {
               "name":"Nelson Inc"
            },
            {
               "name":"Nelson LLC"
            }
         ]
      }
```

```
   ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "$favorite" and "$owner".

Example

```
{
   "filter":[
      {
         "$favorite":"_this"
      }
   ]
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Response

```
{
   "next_offset":-1,
   "records":[
```

```
      {
         "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
         "name":"Dale Spivey",
         "date_modified":"2013-02-28T05:03:00+00:00",
         "description":"",
         "opportunities": [
            {
               _module: "Opportunities",
               "id": "b0701501-1fab-8ae7-3942-540da93f5017",
               "name": "360 Vacations - 228 Units",
               "date_modified": "2014-09-08T16:05:00+03:00",
               "sales_status": "New"
            },
         ],
         "_acl": {
            "fields": {
            }
         }
      },
      {
         "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
         "name":"Florence Haddock",
         "date_modified":"2013-02-26T19:12:00+00:00",
         "description":"",
         "opportunities": [
            {
               _module: "Opportunities"
               date_modified: "2014-09-08T16:05:00+03:00"
               id: "9ce7c088-8ee4-7cd3-18f1-540da944d4c0"
               name: "360 Vacations - 312 Units"
               sales_status: "New"
            },
         ],
         "_acl": {
            "fields": {
            }
         }
      }
   ]
}
```


Change Log

Version	Change
v10	Added /<module>/filter GET endpoint.

/Users GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

Name	Type	Description	Required
filter	String	The filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query parameters in one of two ways for GET requests: 1. By specifying individual filter arguments as distinct parameters. Example: filter[0][id]=1	False

Name	Type	Description	Required
		. 2. By specifying the whole filter as a single JSON-encoded string. Note that this syntax is currently not supported on certain modules. Example: filter=[{"id":"1"}].	
filter_id	String	Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND.	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
fields	String	Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters (applicable to link	False

Name	Type	Description	Required
		and collection fields). The fields id and date_modified will always be returned. Example: name,account_type,description,{"name":"opportunities","fields":["id","name","sales_status"],"order_by":"date_closed:desc"} For more details on additional field parameters, see Relate API and Collection API.	
view	String	Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list". Example: record	False
order_by	String	How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon.	False

Name	Type	Description	Required
		Example: name:DESC,account_type:DESC,date_modified:ASC	
q	String	A search expression, will search on this module. Cannot be used at the same time as a filter expression or id.	False
deleted	Boolean	Boolean to show deleted records in the result set.	False
nulls_last	Boolean	Boolean to return records with null values in order_by fields last in the result set.	False

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
   "filter":[
      {
         "name":"Nelson Inc"
      }
   ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering

on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
   "filter":[
      {
         "name":{
            "$starts":"Nelson"
         }
      }
   ]
}
```

Below is a list of operation types:

Operation	Description
$equals	Performs an exact match on that field.
$not_equals	Performs an exact match on that field.
$not_equals	Matches on non-matching values.
$starts	Matches on anything that starts with the value.
$ends	Matches anything that ends with the value.
$contains	Matches anything that contains the value
$in	Finds anything where field matches one of the values as specified as an array.
$not_in	Finds anything where field does not matches any of the values as specified as an array.
$is_null	Checks if the field is null. This operation

Operation		Description
		does not need a value specified.
	$not_null	Checks if the field is not null. This operation does not need a value specified.
	$lt	Matches when the field is less than the value.
	$lte	Matches when the field is less than or equal to the value.
	$gt	Matches when the field is greater than the value.
	$gte	Matches when the field is greater than or equal to the value.

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "$and" and "$or".

Example

```
{
   "filter":[
      {
         "$or":[
            {
               "name":"Nelson Inc"
            },
            {
               "name":"Nelson LLC"
            }
         ]
      }
   ]
```

```
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "$favorite" and "$owner".

Example

```
{
   "filter":[
      {
         "$favorite":"_this"
      }
   ]
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Response

```
{
   "next_offset":-1,
   "records":[
      {
```

```
        "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
        "name":"Dale Spivey",
        "date_modified":"2013-02-28T05:03:00+00:00",
        "description":"",
        "opportunities": [
           {
              _module: "Opportunities",
              "id": "b0701501-1fab-8ae7-3942-540da93f5017",
              "name": "360 Vacations - 228 Units",
              "date_modified": "2014-09-08T16:05:00+03:00",
              "sales_status": "New"
           },
        ],
        "_acl": {
           "fields": {
           }
        }
     },
     {
        "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
        "name":"Florence Haddock",
        "date_modified":"2013-02-26T19:12:00+00:00",
        "description":"",
        "opportunities": [
           {
              _module: "Opportunities"
              date_modified: "2014-09-08T16:05:00+03:00"
              id: "9ce7c088-8ee4-7cd3-18f1-540da944d4c0"
              name: "360 Vacations - 312 Units"
              sales_status: "New"
           },
        ],
        "_acl": {
           "fields": {
           }
        }
     }
  ]
}
```


Change Log

Version	Change

v10	Added /<module>/filter GET endpoint.

/Users/:record DELETE

Delete a User and return its ID

Summary:

This endpoint is used to delete a User. Returns the ID of the deleted User.

/Users/:record/freebusy GET

Get a user's FreeBusy schedule

Summary:

This endpoint returns a list of time slots for which the specified person is busy.

Request

GET /Users/:id/freebusy

Response

```
{
  "id": "foo"
  "module": "Users",
  "freebusy": [
                {
                  "start": "2014-08-24T08:45:00-04:00",
                  "end": "2014-08-24T09:15:00-04:00"
                },
                {
                  "start": "2014-08-30T05:45:00-04:00",
                  "end": "2014-08-30T06:15:00-04:00"
                },
                {
```

```
                    "start": "2014-09-12T15:45:00-04:00",
                    "end": "2014-09-12T16:15:00-04:00"
                  }
                ]
    }
```

/Users/:record/link POST

Overview

Creates relationships to a pre-existing record.

Request Arguments

Name	Type	Description	Required
<relationship link>	<string>	Link between targeted and related records.	True
<record ID>	<string>	The name value list of related records. Each item of the list may be either string equal to related item ID, or map containing record ID ("id" key is required in this case) and addition relationship properties.	True

Request

```
{
    link_name: "accounts"
    ids: [
        "da6a3741-2a81-ba7f-f249-512d0932e94e",
        {
            "id": "e689173e-c953-1e14-c215-512d0927e7a2",
            "role": "owner"
        }
```

```
    ]
}
```

Response Arguments

Name	Type	Description
record	Array	The record linked to related records.
related_records	Array	Records that were associated.

Response


```
    "record": {
        "id": "da6a3741-2a81-ba7f-f249-512d0932e94e",
        "name": "Slender Broadband Inc - 1000 units",
        "date_entered": "2013-02-26T19:12:00+00:00",
        "date_modified": "2013-02-26T19:12:00+00:00",
        "modified_user_id": "1",
        "modified_by_name": "Administrator",
        "created_by": "1",
        "created_by_name": "Administrator",
        "description": "",
        "img": "",
        "last_activity_date": "2013-02-28T18:21:00+00:00",
        "deleted": false,
        "assigned_user_id": "seed_max_id",
        "assigned_user_name": "Max Jensen",
        "team_name": [
            {
                "id": "East",
                "name": "East",
                "name_2": "",
                "primary": false
            },
            {
                "id": "West",
                "name": "West",
                "name_2": "",
                "primary": true
            }
        ],
```

```
    "opportunity_type": "",
    "account_name": "Slender Broadband Inc",
    "account_id": "181461c6-dc81-1115-1fe0-512d092e8f15",
    "campaign_id": "",
    "campaign_name": "",
    "lead_source": "Campaign",
    "amount": "25000",
    "base_rate": "1",
    "amount_usdollar": "25000",
    "currency_id": "-99",
    "currency_name": "",
    "currency_symbol": "",
    "date_closed": "2013-02-27",
    "date_closed_timestamp": "1361992480",
    "next_step": "",
    "sales_stage": "Needs Analysis",
    "sales_status": "New",
    "probability": "90",
    "best_case": "25000",
    "worst_case": "25000",
    "commit_stage": "include",
    "my_favorite": false,
    "_acl": {
        "fields": {

        }
    }
},
"related_records": [
    {
        "id": "e689173e-c953-1e14-c215-512d0927e7a2",
        "name": "Gus Dales",
        "date_entered": "2013-02-26T19:12:00+00:00",
        "date_modified": "2013-02-26T19:12:00+00:00",
        "modified_user_id": "1",
        "modified_by_name": "Administrator",
        "created_by": "1",
        "created_by_name": "Administrator",
        "description": "",
        "img": "",
        "deleted": false,
        "assigned_user_id": "seed_sally_id",
        "assigned_user_name": "Sally Bronsen",
        "team_name": [
            {
                "id": "West",
```

```
                "name": "West",
                "name_2": "",
                "primary": true
            }
        ],
        "salutation": "",
        "first_name": "Gus",
        "last_name": "Dales",
        "full_name": "Gus Dales",
        "title": "Director Operations",
        "linkedin": "",
        "facebook": "",
        "twitter": "",
        "googleplus": "",
        "department": "",
        "do_not_call": false,
        "phone_home": "(661) 120-2292",
        "email": [
            {
                "email_address": "section.sugar.section@example.it
",
                "opt_out": "1",
                "invalid_email": "0",
                "primary_address": "0"
            },
            {
                "email_address": "support.qa.kid@example.co.uk",
                "opt_out": "0",
                "invalid_email": "0",
                "primary_address": "1"
            }
        ],
        "phone_mobile": "(294) 447-9707",
        "phone_work": "(036) 840-3216",
        "phone_other": "",
        "phone_fax": "",
        "email1": "support.qa.kid@example.co.uk",
        "email2": "section.sugar.section@example.it",
        "invalid_email": false,
        "email_opt_out": false,
        "primary_address_street": "48920 San Carlos Ave",
        "primary_address_street_2": "",
        "primary_address_street_3": "",
        "primary_address_city": "Persistance",
        "primary_address_state": "CA",
        "primary_address_postalcode": "54556",
```

```
        "primary_address_country": "USA",
        "alt_address_street": "",
        "alt_address_street_2": "",
        "alt_address_street_3": "",
        "alt_address_city": "",
        "alt_address_state": "",
        "alt_address_postalcode": "",
        "alt_address_country": "",
        "assistant": "",
        "assistant_phone": "",
        "picture": "",
        "email_and_name1": "",
        "lead_source": "Support Portal User Registration",
        "account_name": "Arts & Crafts Inc",
        "account_id": "d43243c6-9b8e-2973-aee2-512d09bc34b4",
        "opportunity_role_fields": "",
        "opportunity_role_id": "",
        "opportunity_role": "Technical Advisor",
        "reports_to_id": "",
        "report_to_name": "",
        "portal_name": "GusDales145",
        "portal_active": true,
        "portal_password": "$1$JxYr6tmM$b.O6.KF42jP46RadSwz0N0",
        "portal_password1": "",
        "portal_app": "",
        "preferred_language": "en_us",
        "campaign_id": "",
        "campaign_name": "",
        "c_accept_status_fields": "",
        "m_accept_status_fields": "",
        "accept_status_id": "",
        "accept_status_name": "",
        "sync_contact": "",
        "my_favorite": false,
        "_acl": {
            "fields": {
            }
        }
    },
    {
        "id": "da6a3741-2a81-ba7f-f249-512d0932e94e",
        "name": "Slender Broadband Inc - 1000 units",
        "date_entered": "2013-02-26T19:12:00+00:00",
        "date_modified": "2013-02-26T19:12:00+00:00",
        "modified_user_id": "1",
        "modified_by_name": "Administrator",
```

```
"created_by": "1",
"created_by_name": "Administrator",
"description": "",
"img": "",
"last_activity_date": "2013-02-28T18:36:00+00:00",
"deleted": false,
"assigned_user_id": "seed_max_id",
"assigned_user_name": "Max Jensen",
"team_name": [
    {
        "id": "East",
        "name": "East",
        "name_2": "",
        "primary": false
    },
    {
        "id": "West",
        "name": "West",
        "name_2": "",
        "primary": true
    }
],
"opportunity_type": "",
"account_name": "Slender Broadband Inc",
"account_id": "181461c6-dc81-1115-1fe0-512d092e8f15",
"campaign_id": "",
"campaign_name": "",
"lead_source": "Campaign",
"amount": "25000",
"base_rate": "1",
"amount_usdollar": "25000",
"currency_id": "-99",
"currency_name": "",
"currency_symbol": "",
"date_closed": "2013-02-27",
"date_closed_timestamp": "1361992480",
"next_step": "",
"sales_stage": "Needs Analysis",
"sales_status": "New",
"probability": "90",
"best_case": "25000",
"worst_case": "25000",
"commit_stage": "include",
"my_favorite": false,
"_acl": {
    "fields": {
```

```
                }
            }
        }
    ]
}
```

Change Log

Version	Change
v10 (7.6.0)	Added support for additional relationship values.
v10 (7.1.5)	Added /<module>/:record/link POST endpoint.

/Users/:record/link/:link_name/:remote_id DELETE

Overview

Deletes an existing relationship between two records.

Request Arguments

This endpoint does not accept any arguments.

Response Arguments

Name	Type	Description
record	Array	The record to disassociate from the related record.
related_record	Array	The record that was disassociated.

Response

```
{
   "record":{
      "id":"da6a3741-2a81-ba7f-f249-512d0932e94e",
      "name":"Slender Broadband Inc - 1000 units",
```

```
"date_entered":"2013-02-26T19:12:00+00:00",
"date_modified":"2013-02-26T19:12:00+00:00",
"modified_user_id":"1",
"modified_by_name":"Administrator",
"created_by":"1",
"created_by_name":"Administrator",
"description":"",
"img":"",
"last_activity_date":"2013-02-28T18:36:00+00:00",
"deleted":false,
"assigned_user_id":"seed_max_id",
"assigned_user_name":"Max Jensen",
"team_name":[
   {
      "id":"East",
      "name":"East",
      "name_2":"",
      "primary":false
   },
   {
      "id":"West",
      "name":"West",
      "name_2":"",
      "primary":true
   }
],
"opportunity_type":"",
"account_name":"Slender Broadband Inc",
"account_id":"181461c6-dc81-1115-1fe0-512d092e8f15",
"campaign_id":"",
"campaign_name":"",
"lead_source":"Campaign",
"amount":"25000",
"base_rate":"1",
"amount_usdollar":"25000",
"currency_id":"-99",
"currency_name":"",
"currency_symbol":"",
"date_closed":"2013-02-27",
"date_closed_timestamp":"1361992480",
"next_step":"",
"sales_stage":"Needs Analysis",
"sales_status":"New",
"probability":"90",
"best_case":"25000",
"worst_case":"25000",
```

```
      "commit_stage":"include",
      "my_favorite":false,
      "_acl":{
         "fields":{

         }
      }
   },
   "related_record":{
      "id":"e689173e-c953-1e14-c215-512d0927e7a2",
      "name":"Gus Dales",
      "date_entered":"2013-02-26T19:12:00+00:00",
      "date_modified":"2013-02-26T19:12:00+00:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "description":"",
      "img":"",
      "deleted":false,
      "assigned_user_id":"seed_sally_id",
      "assigned_user_name":"Sally Bronsen",
      "team_name":[
         {
            "id":"West",
            "name":"West",
            "name_2":"",
            "primary":true
         }
      ],
      "salutation":"",
      "first_name":"Gus",
      "last_name":"Dales",
      "full_name":"Gus Dales",
      "title":"Director Operations",
      "linkedin":"",
      "facebook":"",
      "twitter":"",
      "googleplus":"",
      "department":"",
      "do_not_call":false,
      "phone_home":"(661) 120-2292",
      "email":[
         {
            "email_address":"section.sugar.section@example.it",
            "opt_out":"1",
```

```
            "invalid_email":"0",
            "primary_address":"0"
        },
        {
            "email_address":"support.qa.kid@example.co.uk",
            "opt_out":"0",
            "invalid_email":"0",
            "primary_address":"1"
        }
    ],
    "phone_mobile":"(294) 447-9707",
    "phone_work":"(036) 840-3216",
    "phone_other":"",
    "phone_fax":"",
    "email1":"support.qa.kid@example.co.uk",
    "email2":"section.sugar.section@example.it",
    "invalid_email":false,
    "email_opt_out":false,
    "primary_address_street":"48920 San Carlos Ave",
    "primary_address_street_2":"",
    "primary_address_street_3":"",
    "primary_address_city":"Persistance",
    "primary_address_state":"CA",
    "primary_address_postalcode":"54556",
    "primary_address_country":"USA",
    "alt_address_street":"",
    "alt_address_street_2":"",
    "alt_address_street_3":"",
    "alt_address_city":"",
    "alt_address_state":"",
    "alt_address_postalcode":"",
    "alt_address_country":"",
    "assistant":"",
    "assistant_phone":"",
    "picture":"",
    "email_and_name1":"",
    "lead_source":"Support Portal User Registration",
    "account_name":"Arts & Crafts Inc",
    "account_id":"d43243c6-9b8e-2973-aee2-512d09bc34b4",
    "opportunity_role_fields":"",
    "opportunity_role_id":"",
    "opportunity_role":"",
    "reports_to_id":"",
    "report_to_name":"",
    "portal_name":"GusDales145",
    "portal_active":true,
```

```
      "portal_password":"$1$JxYr6tmM$b.O6.KF42jP46RadSwz0N0",
      "portal_password1":"",
      "portal_app":"",
      "preferred_language":"en_us",
      "campaign_id":"",
      "campaign_name":"",
      "c_accept_status_fields":"",
      "m_accept_status_fields":"",
      "accept_status_id":"",
      "accept_status_name":"",
      "sync_contact":"",
      "my_favorite":false,
      "_acl":{
         "fields":{

         }
      }
   }
}
```


Change Log

Version	Change
v10	Added /<module>/:record/link/:link_name/:remote_id DELETE endpoint.

/Users/:record/link/:link_name/:remote_id POST

Overview

Creates a relationship to a pre-existing record.

Query String Parameters

This endpoint does not accept any arguments.

Response Arguments

Name	Type	Description
record	Array	The record linked to the related record.

Name	Type	Description
related_record	Array	The record that was associated.

Response


```
{
   "record":{
      "id":"da6a3741-2a81-ba7f-f249-512d0932e94e",
      "name":"Slender Broadband Inc - 1000 units",
      "date_entered":"2013-02-26T19:12:00+00:00",
      "date_modified":"2013-02-26T19:12:00+00:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "description":"",
      "img":"",
      "last_activity_date":"2013-02-28T18:21:00+00:00",
      "deleted":false,
      "assigned_user_id":"seed_max_id",
      "assigned_user_name":"Max Jensen",
      "team_name":[
         {
            "id":"East",
            "name":"East",
            "name_2":"",
            "primary":false
         },
         {
            "id":"West",
            "name":"West",
            "name_2":"",
            "primary":true
         }
      ],
      "opportunity_type":"",
      "account_name":"Slender Broadband Inc",
      "account_id":"181461c6-dc81-1115-1fe0-512d092e8f15",
      "campaign_id":"",
      "campaign_name":"",
      "lead_source":"Campaign",
      "amount":"25000",
      "base_rate":"1",
      "amount_usdollar":"25000",
```

```
    "currency_id":"-99",
    "currency_name":"",
    "currency_symbol":"",
    "date_closed":"2013-02-27",
    "date_closed_timestamp":"1361992480",
    "next_step":"",
    "sales_stage":"Needs Analysis",
    "sales_status":"New",
    "probability":"90",
    "best_case":"25000",
    "worst_case":"25000",
    "commit_stage":"include",
    "my_favorite":false,
    "_acl":{
       "fields":{

       }
    }
 },
 "related_record":{
    "id":"e689173e-c953-1e14-c215-512d0927e7a2",
    "name":"Gus Dales",
    "date_entered":"2013-02-26T19:12:00+00:00",
    "date_modified":"2013-02-26T19:12:00+00:00",
    "modified_user_id":"1",
    "modified_by_name":"Administrator",
    "created_by":"1",
    "created_by_name":"Administrator",
    "description":"",
    "img":"",
    "deleted":false,
    "assigned_user_id":"seed_sally_id",
    "assigned_user_name":"Sally Bronsen",
    "team_name":[
       {
          "id":"West",
          "name":"West",
          "name_2":"",
          "primary":true
       }
    ],
    "salutation":"",
    "first_name":"Gus",
    "last_name":"Dales",
    "full_name":"Gus Dales",
    "title":"Director Operations",
```

```
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(661) 120-2292",
"email":[
   {
      "email_address":"section.sugar.section@example.it",
      "opt_out":"1",
      "invalid_email":"0",
      "primary_address":"0"
   },
   {
      "email_address":"support.qa.kid@example.co.uk",
      "opt_out":"0",
      "invalid_email":"0",
      "primary_address":"1"
   }
],
"phone_mobile":"(294) 447-9707",
"phone_work":"(036) 840-3216",
"phone_other":"",
"phone_fax":"",
"email1":"support.qa.kid@example.co.uk",
"email2":"section.sugar.section@example.it",
"invalid_email":false,
"email_opt_out":false,
"primary_address_street":"48920 San Carlos Ave",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Persistance",
"primary_address_state":"CA",
"primary_address_postalcode":"54556",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
"assistant":"",
"assistant_phone":"",
"picture":"",
```

```
      "email_and_name1":"",
      "lead_source":"Support Portal User Registration",
      "account_name":"Arts & Crafts Inc",
      "account_id":"d43243c6-9b8e-2973-aee2-512d09bc34b4",
      "opportunity_role_fields":"",
      "opportunity_role_id":"",
      "opportunity_role":"Technical Advisor",
      "reports_to_id":"",
      "report_to_name":"",
      "portal_name":"GusDales145",
      "portal_active":true,
      "portal_password":"$1$JxYr6tmM$b.O6.KF42jP46RadSwz0N0",
      "portal_password1":"",
      "portal_app":"",
      "preferred_language":"en_us",
      "campaign_id":"",
      "campaign_name":"",
      "c_accept_status_fields":"",
      "m_accept_status_fields":"",
      "accept_status_id":"",
      "accept_status_name":"",
      "sync_contact":"",
      "my_favorite":false,
      "_acl":{
         "fields":{

         }
      }
   }
}
```


Change Log

Version	Change
v10	Added /<module>/:record/link/:link_name/:remote_id POST endpoint.

/VCardDownload GET

Overview

Downloads a vCard.

Request Arguments

Name	Type	Description	Required
id	String	The id of the vcard.	True
module	String	The name of the module to get the vcard from.	True

Request

```
http://{site_url}/rest/v10/VCardDownload?id=2c9e5c09-6824-0d14-f5cb-51
30321ac3cf&module=Contacts
```

Note: GET endpoint parameters are passed in the form of a query string.

Response Arguments

Name	Type	Description
<vcard information>	String;	Record in vcard format.

Response

```
BEGIN:VCARD
N;CHARSET=utf-8:Leone;Rosemarie;;
FN;CHARSET=utf-8: Rosemarie Leone
BDAY:
TEL;WORK;FAX:
TEL;HOME;CHARSET=utf-8:(692) 586-8287
TEL;CELL;CHARSET=utf-8:(117) 577-0969
TEL;WORK;CHARSET=utf-8:(844) 325-7679
EMAIL;INTERNET;CHARSET=utf-8:support.im@example.it
ADR;WORK;CHARSET=utf-8:;;777 West Filmore Ln;San Mateo;CA;74984;USA
ORG;CHARSET=utf-8:Q.R.&E. Corp;
TITLE;CHARSET=utf-8:Director Sales
END:VCARD
```

Change Log

Version	Change
v10	Added /VCardDownload GET endpoint.

/bulk POST

Overview

Run API calls in bulk.

Summary

This request will run a sequence of API requests within one query. The requests are executed sequentially and their results are returned as one response. Some requests may return failure code, that does not interrupt the execution of the batch, and the overall request will still be considered successful.

Request Arguments

Name	Type	Description	Required
requests	Array	The list of requests	True

Each of the requests can have the following fields:

Name	Type	Description	Required
url	String	The request URL, starting with version.	True
data	JSON String	The data for the POST/PUT body. Must be a JSON-encoded string.	False
headers	Array	The request headers	False
method	String	The HTTP method (default is GET)	False

Request Example

```
{"requests":
[
  {
    "url": "/v10/Accounts", "method": "POST", "data": "{\"name\": \"te
st123\"}"
```

```
  },
  {
    "url": "/v10/Accounts", "method": "GET"
  }
]
}
```

Response

The response will contain an array of response objects, each of them will correspond to the individual request. The following fields are in the response objects:

Name	Type	Description
contents	Array or String	The response contents, can be JSON object or string depending on what the individual request is supposed to return.
headers	Array	The response headers
status	Integer	HTTP status code of the response. Will be 2XX for successful requests and 4XX or 5XX for errors.

Response Example

```
[
    {
        "contents": {
            "my_favorite": false,
            "following": true,
            "id": "7d2e21a6-8a76-a74f-bb53-535620211304",
            "name": "test123",
            "date_entered": "2014-04-22T03:56:24-04:00",
            "_module": "Accounts"
        },
        "headers": [],
        "status": 200
    },
    {
        "contents": {
```

```
            "next_offset": -1,
            "records": [
                {
                    "my_favorite": false,
                    "following": true,
                    "id": "7d2e21a6-8a76-a74f-bb53-535620211304",
                    "name": "test123",
                    "date_entered": "2014-04-22T03:56:24-04:00",
                    "date_modified": "2014-04-22T03:56:24-04:00",
    }
            ]
        },
        "headers": [],
        "status": null
    }
]
```


Change Log

Version	Change
v10	Added /bulk POST endpoint.

/collection/:collection_name GET

Overview

Lists records from multiple modules at a time.

Summary

This endpoint will return a set of records fetched from multiple modules defined by :collection_name. The records will be filtered, sorted and truncated to max_num as a single collection. Collections may use a field mapping. For instance, the due_date of Tasks may be referred to as date_end. In this case the alias (date_end) should be used in filter and order_by expressions instead of real field name. Note that response data still contains the original fields for the module.

Request Arguments

Name	Type	Description	Required
fields	String	See Filter API. If the collection	False

Name	Type	Description	Required
		definition uses aliases, then aliases should be used instead of real field names.	
max_num	Integer	See Filter API.	False
filter	String	See Filter API. If collection definition uses aliases, then aliases should be used instead of real field names.	False
order_by	String	See Filter API. If collection definition uses aliases, then aliases should be used instead of real field names.	False
offset	Map	Individual offset for each module which the collection consists of. -1 (or any negative value) denotes that the module should be skipped. If module offset is not specified, it defaults to 0.	False

Response Arguments

Name	Type	Description
next_offset	Map	The next offset to retrieve records. -1 will be returned for the given module when there are no more records.
records	Array	The record result set.

Response

```
{
   "next_offset": {
      "Calls": 1,
      "Meetings": -1,
      "Tasks": -2,
   },
   "records": [
      {
         "_module": "Calls",
         "id": "8703fbf3-0ffa-c288-8d2c-512f943ecdc3",
         "name": "Discuss review process",
         "date_end": "2014-02-26T19:12:00+00:00"
      },
      {
         "_module": "Tasks",
         "id": "e1c495cb-af17-1b37-dd66-512f934fe155",
         "name": "Introduce all players",
         "due_date": "2014-02-26T19:12:00+00:00"
      },
      {
         "_module": "Tasks",
         "id": "456b7848-9959-5a64-cd34-512d0938addd",
         "name": "Follow-up on proposal",
         "due_date": "2014-02-26T19:12:00+00:00"
      }
   ]
}
```


Change Log

Version	Change
v10	Added /collection/:collection_name GET endpoint.

/connector/twitter/:twitterId GET

Overview

Responds with twitter timeline if connector is set up in administration

Request Arguments

Name	Type	Description	Required
count	Integer	The number of tweets to retrieve.	False

Response Arguments

Name	Type	Description
<response>	String	

Response


```
[
   {
      "created_at":"Tue Jun 25 23:45:35 +0000 2013",
      "id":349674766946414592,
      "id_str":"349674766946414592",
      "text":"this tweet is awesome!",
      "source":"web",
      "truncated":false,
      "in_reply_to_status_id":null,
      "in_reply_to_status_id_str":null,
      "in_reply_to_user_id":143456975,
      "in_reply_to_user_id_str":"14934565",
      "in_reply_to_screen_name":"testdesk",
      "user":{
         "id":2630841,
         "id_str":"2630841",
         "name":"SugarCRM",
         "screen_name":"sugarcrm",
         "location":"Cupertino, CA",
         "description":"Everyone Sells. Everyone Supports. Everyone Co
nnects.",
         "url":"http:\/\/t.co\/s9ejrBSlki",
         "entities":{
            "url":{
               "urls":[
                  {
                     "url":"http:\/\/t.co\/s9ejrBSlki",
                     "expanded_url":"http:\/\/www.sugarcrm.com",
                     "display_url":"sugarcrm.com",
                     "indices":[
                        0,
                        22
                     ]
                  }
```

```
            ]
         },
         "description":{
            "urls":[

            ]
         }
      },
      "protected":false,
      "followers_count":8761,
      "friends_count":6966,
      "listed_count":453,
      "created_at":"Wed Mar 28 07:16:58 +0000 2007",
      "favourites_count":27,
      "utc_offset":-28800,
      "time_zone":"Pacific Time (US \u0026 Canada)",
      "geo_enabled":false,
      "verified":false,
      "statuses_count":7488,
      "lang":"en",
      "contributors_enabled":false,
      "is_translator":false,
      "profile_background_color":"98C7EA",
      "profile_background_image_url":"http:\/\/a0.twimg.com\/profil
e_background_images\/551274192\/sugar-twitter-background.png",
      "profile_background_image_url_https":"https:\/\/si0.twimg.com
\/profile_background_images\/551274192\/sugar-twitter-background.png",
      "profile_background_tile":false,
      "profile_image_url":"http:\/\/a0.twimg.com\/profile_images\/2
027721183\/Sugar_cube_RGB_180x180_normal.png",
      "profile_image_url_https":"https:\/\/si0.twimg.com\/profile_i
mages\/2027721183\/Sugar_cube_RGB_180x180_normal.png",
      "profile_link_color":"0045AD",
      "profile_sidebar_border_color":"FFFFFF",
      "profile_sidebar_fill_color":"CCEAFF",
      "profile_text_color":"000000",
      "profile_use_background_image":true,
      "default_profile":false,
      "default_profile_image":false,
      "following":null,
      "follow_request_sent":false,
      "notifications":null
   },
   "geo":null,
   "coordinates":null,
   "place":null,
```

```
      "contributors":null,
      "retweet_count":1,
      "favorite_count":0,
      "entities":{
         "hashtags":[

         ],
         "symbols":[

         ],
         "urls":[

         ],
         "user_mentions":[
            {
               "screen_name":"testdesk",
               "name":"testdesk",
               "id":143455,
               "id_str":"ertrt75",
               "indices":[
                  0,
                  8
               ]
            }
         ],
         "media":[

         ]
      },
      "favorited":false,
      "retweeted":false,
      "possibly_sensitive":false,
      "lang":"en"
   }
]
```


Change Log

Version	Change
v10	Added /twitter/{twitterId} GET endpoint.

/connector/twitter/currentUser GET

Overview

Responds with twitter timeline if connector is set up in administration

Request Arguments

Name	Type	Description	Required
count	Integer	The number of tweets to retrieve.	False

Response Arguments

Name	Type	Description
<response>	String	

Response


```
[
   {
      "created_at":"Tue Jun 25 23:45:35 +0000 2013",
      "id":349674766946414592,
      "id_str":"349674766946414592",
      "text":"this tweet is awesome!",
      "source":"web",
      "truncated":false,
      "in_reply_to_status_id":null,
      "in_reply_to_status_id_str":null,
      "in_reply_to_user_id":143456975,
      "in_reply_to_user_id_str":"14934565",
      "in_reply_to_screen_name":"testdesk",
      "user":{
         "id":2630841,
         "id_str":"2630841",
         "name":"SugarCRM",
         "screen_name":"sugarcrm",
         "location":"Cupertino, CA",
         "description":"Everyone Sells. Everyone Supports. Everyone Co
nnects.",
         "url":"http:\/\/t.co\/s9ejrBSlki",
         "entities":{
            "url":{
               "urls":[
```

```
                  {
                     "url":"http:\/\/t.co\/s9ejrBSlki",
                     "expanded_url":"http:\/\/www.sugarcrm.com",
                     "display_url":"sugarcrm.com",
                     "indices":[
                        0,
                        22
                     ]
                  }
               ]
            },
            "description":{
               "urls":[

               ]
            }
         },
         "protected":false,
         "followers_count":8761,
         "friends_count":6966,
         "listed_count":453,
         "created_at":"Wed Mar 28 07:16:58 +0000 2007",
         "favourites_count":27,
         "utc_offset":-28800,
         "time_zone":"Pacific Time (US \u0026 Canada)",
         "geo_enabled":false,
         "verified":false,
         "statuses_count":7488,
         "lang":"en",
         "contributors_enabled":false,
         "is_translator":false,
         "profile_background_color":"98C7EA",
         "profile_background_image_url":"http:\/\/a0.twimg.com\/profil
e_background_images\/551274192\/sugar-twitter-background.png",
         "profile_background_image_url_https":"https:\/\/si0.twimg.com
\/profile_background_images\/551274192\/sugar-twitter-background.png",
         "profile_background_tile":false,
         "profile_image_url":"http:\/\/a0.twimg.com\/profile_images\/2
027721183\/Sugar_cube_RGB_180x180_normal.png",
         "profile_image_url_https":"https:\/\/si0.twimg.com\/profile_i
mages\/2027721183\/Sugar_cube_RGB_180x180_normal.png",
         "profile_link_color":"0045AD",
         "profile_sidebar_border_color":"FFFFFF",
         "profile_sidebar_fill_color":"CCEAFF",
         "profile_text_color":"000000",
         "profile_use_background_image":true,
```

```
         "default_profile":false,
         "default_profile_image":false,
         "following":null,
         "follow_request_sent":false,
         "notifications":null
      },
      "geo":null,
      "coordinates":null,
      "place":null,
      "contributors":null,
      "retweet_count":1,
      "favorite_count":0,
      "entities":{
         "hashtags":[

         ],
         "symbols":[

         ],
         "urls":[

         ],
         "user_mentions":[
            {
               "screen_name":"testdesk",
               "name":"testdesk",
               "id":143455,
               "id_str":"ertrt75",
               "indices":[
                  0,
                  8
               ]
            }
         ],
         "media":[

         ]
      },
      "favorited":false,
      "retweeted":false,
      "possibly_sensitive":false,
      "lang":"en"
   }
]
```

Change Log

Version	Change
v10	Added /twitter/{twitterId} GET endpoint.

/connectors GET

Overview

Gets general info about connectors.

Request Arguments

This endpoint does not accept any request arguments.

Result Arguments

Response

```
{
    "connectors": {
        "ext_rest_twitter": {
            "testing_enabled": true,
            "test_passed": false,
            "eapm_bean": false,
            "field_mapping": {
                "beans": {
                    "Accounts": {
                        "name": "name",
                        "id": "id"
                    },
                    "Contacts": {
                        "name": "full_name",
                        "id": "id"
                    },
                    "Leads": {
                        "name": "account_name",
                        "id": "id"
                    },
                    "Prospects": {
                        "name": "account_name",
                        "id": "id"
                    }
```

```
            }
        },
        "id": "ext_rest_twitter",
        "name": "Twitter"
    },
    "ext_eapm_google": {
        "testing_enabled": false,
        "test_passed": false,
        "eapm_bean": false,
        "field_mapping": [],
        "id": "ext_eapm_google",
        "name": "Google"
    },
    "ext_eapm_gotomeeting": {
        "testing_enabled": false,
        "test_passed": false,
        "eapm_bean": false,
        "field_mapping": [],
        "id": "ext_eapm_gotomeeting",
        "name": "GoToMeeting"
    },
    "ext_eapm_ibmsmartcloud": {
        "testing_enabled": false,
        "test_passed": false,
        "eapm_bean": false,
        "field_mapping": [],
        "id": "ext_eapm_ibmsmartcloud",
        "name": "IBM SmartCloud"
    },
    "ext_eapm_webex": {
        "testing_enabled": false,
        "test_passed": false,
        "eapm_bean": false,
        "field_mapping": [],
        "id": "ext_eapm_webex",
        "name": "WebEx"
    },
    "ext_eapm_lotuslive": {
        "testing_enabled": false,
        "test_passed": false,
        "eapm_bean": false,
        "field_mapping": [],
        "id": "ext_eapm_lotuslive",
        "name": "LotusLive"
    },
    "ext_rest_dnb": {
```

```
            "testing_enabled": false,
            "test_passed": false,
            "eapm_bean": false,
            "field_mapping": {
                "beans": {
                    "Accounts": {
                        "name": "name",
                        "id": "id"
                    }
                }
            },
            "id": "ext_rest_dnb",
            "name": "DNB"
        },
        "_hash": "\u0000d4751e1632fd5d1d2c9069a1f176e6f"
    },
    "_hash": "2cd9132603ead4f79715c69ee98e64f1"
}
```


Change Log

Version	Change
v10	Added /<connectors> GET endpoint.

/css GET

Overview

Runs LessPHP for a platform and a theme and outputs an array of css files. If not found the css files will be compiled.

Request Arguments

Name	Type	Description	Required
platform	String	The theme platform - /themes/clients/*	No. (defaults to *base*)

Name	Type	Description	Required
		PL **ATFO** **RM***** /themeName/. Accepted values are 'base' and 'portal'.	
themeName	String	The theme name - /themes/clients/platform/ ****THEMENAME******/.	No. (defaults to *default*)

Request

```
{
    platform: 'portal',
    themeName: 'default'
}
```

Response

```
{
    url: [
        'cache/themes/clients/base/default/bootstrap_97c9cf53841dbe471
c20d169e3533763.css',
        'cache/themes/clients/base/default/sugar_14c42516aad866b7f80cc
896ce5c5406.css',
    ]
}
```

Change Log

Version	Change
v10	Added /<css> GET endpoint.

/css/preview GET

Overview

Runs LessPHP for a platform and a theme and outputs the compiled CSS. It only allows to preview the theme because the file is not saved on the server.

Request Arguments

Name	Type	Description	Required
platform	String	The theme platform - /themes/clients/* **PLATFORM***/. Accepted values are 'base' and 'portal'.	No. (defaults to *base*)
themeName	String	The theme name - /themes/clients/platform/ ***THEMENAME***/.	No. (defaults to *default*)
min	Boolean	Whether or not to minify the css	No. (defaults to *false*)

Request

```
{
    platform: 'portal',
    themeName: 'default',
    min: false
}
```

Response Arguments

Response

Content-type: text/css

```
.someClass {
    any-property: any-value;
}
```

Change Log

Version	Change
v10	Added /css/preview GET endpoint.

/globalsearch GET

Overview

Global search

Summary

This endpoint exposes the global search capability using solely the Elasticsearch backend as an alternative to the /search endpoint. This endpoint can be used with a long list of request arguments. Instead of using GET request arguments, all described arguments can be used inside a JSON encoded request body using a the GET request. If your client has no support for GET requests with a body, the POST method can be used as an alternative.

Request Arguments

Name	Type	Description	Required
q	String	The search expression. Multiple terms can be specified at once. All enabled fields will be searched in. The results are ordered by relevance which is based on a multitude of settings based on	False

Name	Type	Description	Required
		token counts, hit ratio, (weighted) boost values and type of field. Currently no operators are supported in the search expression itself. By refining the search expression more relevant results will be returned as top results. If no search expression is given results are returned based on last modified date.	
module_list	String	Comma delimited list of modules to search. If omitted, all search enabled modules will be queried. Note that when consuming the endpoint /:module/globalsearch that this parameter is ignored. Example: Accounts,Contacts	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
highlights	Boolean	Wether or not to return highlighted results. Default is	False

Name	Type	Description	Required
		true.	
sort	Array	Define the sort order of the results. By default the results are returned by relevance which is the preferred approach. Using this argument any search enabled field can be used to sort on. Keep in mind the not sorting by relevance may have a negative performance impact. Example: {"date_modified":"desc","name":"asc"}	False

Request

```
{
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Response

```
{
}
```

Change Log

Version	Change
v10	Added /globalsearch GET/POST endpoint.
v10	Added /:module/globalsearch GET/POST endpoint.

/globalsearch POST

Overview

Global search

Summary

This endpoint exposes the global search capability using solely the Elasticsearch backend as an alternative to the /search endpoint. This endpoint can be used with a long list of request arguments. Instead of using GET request arguments, all described arguments can be used inside a JSON encoded request body using a the GET request. If your client has no support for GET requests with a body, the POST method can be used as an alternative.

Request Arguments

Name	Type	Description	Required
q	String	The search expression. Multiple terms can be specified at once. All enabled fields will be searched in. The results are ordered by relevance which is based on a	False

Name	Type	Description	Required
		multitude of settings based on token counts, hit ratio, (weighted) boost values and type of field. Currently no operators are supported in the search expression itself. By refining the search expression more relevant results will be returned as top results. If no search expression is given results are returned based on last modified date.	
module_list	String	Comma delimited list of modules to search. If omitted, all search enabled modules will be queried. Note that when consuming the endpoint /:module/globalsearch that this parameter is ignored. Example: Accounts,Contacts	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
highlights	Boolean	Wether or not to	False

Name	Type	Description	Required
		return highlighted results. Default is true.	
sort	Array	Define the sort order of the results. By default the results are returned by relevance which is the preferred approach. Using this argument any search enabled field can be used to sort on. Keep in mind the not sorting by relevance may have a negative performance impact. Example: {"date_modified":"desc","name":"asc"}	False

Request

```
{
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched

Name	Type	Description
		records.

Response

```
{
}
```

Change Log

Version	Change
v10	Added /globalsearch GET/POST endpoint.
v10	Added /:module/globalsearch GET/POST endpoint.

/help GET

Overview

Fetches the help documentation

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

Name	Type	Description
<html>	String	Returns the html of the help doc

Response

The HTML for this help document.

Change Log

Version	Change
v10	Added /help GET endpoint.

/help/exceptions GET

Overview

Fetches the documentation on which exceptions are thrown by the API, their HTTP codes, their default messages and a brief description of what the exception means.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

Name	Type	Description
<html>	String	Returns the html of the help doc

Response

The HTML document of the exception help page.

Change Log

Version	Change
v10	Added /help/exceptions GET endpoint.

/integrate//:lhs_sync_key_field_name/:lhs_sync_key_field_value/link/:link_name/:rhs_sync_key_field_name/:rhs_sync_key_field_value DELETE

Overview

Removes a relationship based on lhs_sync_key_field_name and rhs_sync_key_field_name. If both the LHS and RHS records can be found with the respective fields, and those records are related, then remove the relationship of the RHS record to the LHS record.

Request Arguments

Name	Type	Description	Required
module	String	The module the record belongs to	True
lhs_sync_key_field_name	String	The name of the LHS field that contains lhs_sync_key_field_value (the default field is named sync_key but can be renamed via code customization)	True
lhs_sync_key_field_value	String	A unique ID for the LHS record identifying it in an external system	True
link_name	String	A name for the link	True
rhs_sync_key_field_name	String	The name of the RHS field that contains rhs_sync_key_field_value (the default field is named sync_key but can be renamed via code customization)	True
rhs_sync_key_field_value	String	A unique ID for the RHS record identifying it in an external system	True

Request

```
  /integrate/<module>/:lhs_sync_key_field_name/:lhs_sync_key_field_val
ue/link/:link_name/:rhs_sync_key_field_name/:rhs_sync_key_field_value
```

Response Arguments

Name	Type	Description
record	String	The ID of the record.
related_record	String	The ID of the related

Name	Type	Description
		record.

Response

Status 200

```
  {
    "record": "a0328573-a252-a27c-3530-4e4297d4c9e1",
    "related_record": "a0328573-bc54-a554-3530-4e4297d4c9e1"
}
```

Status 422

```
  {
    "error": "invalid_parameter",
    "error_message": "Could not find record with :xhs_sync_key_field_n
ame::xhs_sync_key_field_value in module: <module>"
}
```

Change Log

Version	Change
v11_10	Added /integrate/<module>/:lhs_sync_key_field_name/:lhs_sync_key_field_value/link/:link_name/:rhs_sync_key_field_name/:rhs_sync_key_field_value DELETE endpoint.

/integrate//:lhs_sync_key_field_name/:lhs_sync_key_field_value/link/:link_name/:rhs_sync_key_field_name/:rhs_sync_key_field_value POST

Overview

Creates a relationship based on lhs_sync_key_field_name and rhs_sync_key_field_name. If both the LHS and RHS records can be found with the respective fields, then relate the RHS record to the LHS record.

Request Arguments

Name	Type	Description	Required
module	String	The module the record belongs to	True
lhs_sync_key_field_name	String	The name of the LHS field that contains lhs_sync_key_field_value (the default field is named sync_key but can be renamed via code customization)	True
lhs_sync_key_field_value	String	A unique ID for the LHS record identifying it in an external system	True
link_name	String	A name for the link	True
rhs_sync_key_field_name	String	The name of the RHS field that contains rhs_sync_key_field_value (the default field is named sync_key but can be renamed via code customization)	True
rhs_sync_key_field_value	String	A unique ID for the RHS record identifying it in an external system	True

Request

```
  /integrate/<module>/:lhs_sync_key_field_name/:lhs_sync_key_field_val
ue/link/:link_name/:rhs_sync_key_field_name/:rhs_sync_key_field_value
```

Response Arguments

Name	Type	Description
record	String	The ID of the record.
related_record	String	The ID of the related record.

Response

Status 200

```
  {
    "record": "a0328573-a252-a27c-3530-4e4297d4c9e1",
    "related_record": "a0328573-bc54-a554-3530-4e4297d4c9e1"
}
```

Status 422

```
  {
    "error": "invalid_parameter",
    "error_message": "Could not find record with :xhs_sync_key_field_n
ame::xhs_sync_key_field_value in module: <module>"
}
```

Change Log

Version	Change
v11_10	Added /integrate/<module>/:lhs_sync_key_field_name/:lhs_sync_key_field_value/link/:link_name/:rhs_sync_key_field_name/:rhs_sync_key_field_value POST endpoint.

/integrate//:record_id/:sync_key_field_name/:sync_key_field_value PATCH

Overview

Sets a sync key value for a record.

Request Arguments

Name	Type	Description	Required
module	String	The module the record belongs to	True
record_id	String	The ID of the record whose sync key will be set	True
sync_key_field_name	String	The name of the field that contains sync_key_field_value (the default field is named sync_key but can be renamed via code customization)	True
sync_key_field_value	String	A unique ID for the record identifying it in an external system	True

Request

```
  /integrate/<module>/:record_id/:sync_key_field_name/:sync_key_field_value
```

Response

```
{
"success": true
}
```

Change Log

Version	Change
v11_10	Added /integrate/<module>/:record_id/:sync_key_field_name/:sync_key_field_value PATCH endpoint.

/integrate//:record_id/:sync_key_field_name/:sync_

key_field_value PUT

Overview

Sets a sync key value for a record. Note that the PATCH method is recommended over the PUT method.

Request Arguments

Name	Type	Description	Required
module	String	The module the record belongs to	True
record_id	String	The ID of the record whose sync key will be set	True
sync_key_field_name	String	The name of the field that contains sync_key_field_value (the default field is named sync_key but can be renamed via code customization)	True
sync_key_field_value	String	A unique ID for the record identifying it in an external system	True

Request

```
  /integrate/<module>/:record_id/:sync_key_field_name/:sync_key_field_value
```

Response

```
{
"success": true
}
```

Change Log

Version	Change
v11_10	Added /integrate/<module>/:record_id/:sync_key_field_name/:sync_key_field_value PUT endpoint.

/integrate//:sync_key_field_name/:sync_key_field_value DELETE

Overview

Deletes the record with the given sync_key_field_name.

Request Arguments

Name	Type	Description	Required
module	String	The module the record belongs to	True
sync_key_field_name	String	The name of the field that contains sync_key_field_value (the default field is named sync_key but can be renamed via code customization)	True
sync_key_field_value	String	A unique ID for the record identifying it in an external system	True

Request

```
/integrate/<module>/:sync_key_field_name/:sync_key_field_value
```

Response Arguments

Name	Type	Description
record	String	The ID of the record that

Name	Type	Description
		was deleted.

Response

Status 200

```
{
    "id": "a0328573-a252-a27c-3530-4e4297d4c9e1"
}
```

Change Log

Version	Change
v11_10	Added /integrate/<module>/:sync_key_field_name/:sync_key_field_value DELETE endpoint.

/integrate//:sync_key_field_name/:sync_key_field_value GET

Overview

Retrieves the record with the given sync_key_field_name.

Request Arguments

Name	Type	Description	Required
module	String	The module the record belongs to	True
sync_key_field_name	String	The name of the field that contains sync_key_field_value (the default field is named sync_key but can be renamed via code	True

Name	Type	Description	Required
		customization)	
sync_key_field_value	String	A unique ID for the record identifying it in an external system	True

Request

```
/integrate/<module>/:sync_key_field_name/:sync_key_field_value
```

Response Arguments

Body: formatted JSON object.

Response

Successful Response

Status 200

Failed Response

Status 422

```
{
  "error": "invalid_parameter",
  "error_message": "Could not find record with :sync_key_field_name:
:sync_key_field_value in module: <module>"
}
```

Change Log

Version	Change
v11_10	Added /integrate/<module>/:sync_key_field_name/:sync_key_field_value GET endpoint.

/integrate//:sync_key_field_name/:sync_key_field_value PATCH

Overview

Upserts based on sync_key_field_name. If a record can be found with sync_key_field_name, then update. If the record does not exist, then create it.

Note: This endpoint cannot be used to set the sync_key_field_name for an existing Sugar record. To do so, it is recommended to use the set sync key endpoint to update the matching Sugar record ID. Once the sync_key is set for the record, then it is possible to leverage this endpoint.

Request Arguments

Name	Type	Description	Required
module	String	The module the record belongs to	True
sync_key_field_name	String	The name of the field that contains sync_key_field_value (the default field is named sync_key but can be renamed via code customization)	True
sync_key_field_value	String	A unique ID for the record identifying it in an external system	True

Request

```
/integrate/<module>/:sync_key_field_name/:sync_key_field_value
```

Response Arguments

Name	Type	Description
record	String	The ID of the record that was upserted.

Response

- Status 201: For a created record
- Status 200: For an updated record

```
{
    record: "a0328573-a252-a27c-3530-4e4297d4c9e1"
}
```

Change Log

Version	Change
v11_10	Added /integrate/<module>/:sync_key_field_name/:sync_key_field_value PATCH endpoint.

/integrate//:sync_key_field_name/:sync_key_field_value PUT

Overview

Upserts based on sync_key_field_name. If a record can be found with sync_key_field_name, then update. If the record does not exist, then create it. Note that the PATCH method is recommended over the PUT method.

Note: This endpoint cannot be used to set the sync_key_field_name for an existing Sugar record. To do so, it is recommended to use the set sync key endpoint to update the matching Sugar record ID. Once the sync_key_field_name is set for the record, then it is possible to leverage this endpoint.

Request Arguments

Name	Type	Description	Required
module	String	The module the record belongs to	True
sync_key_field_name	String	The name of the field that contains s	True

Name	Type	Description	Required
		ync_key_field_value (the default field is named sync_key but can be renamed via code customization)	
sync_key_field_value	String	A unique ID for the record identifying it in an external system	True

Request

```
/integrate/<module>/:sync_key_field_name/:sync_key_field_value
```

Response Arguments

Name	Type	Description
record	String	The ID of the record that was upserted.

Response

- Status 201: For a created record
- Status 200: For an updated record

```
{
    record: "a0328573-a252-a27c-3530-4e4297d4c9e1"
}
```

Change Log

Version	Change
v11_10	Added /integrate/<module>/ :sync_key_field_name/:sync_key_field_value PUT endpoint.

/lang/labels/dropdown PUT

Overview

Update display labels of dropdown items for dropdowns under different languages.

Summary

This endpoint is used to update display labels of dropdown items in dropdowns under different languages.

Request Arguments

Name	Type	Description	Required
name	string	The dropdown name.	True
labels	array	The new labels grouped by language keys.	True

Request

```
  [
  {
    "type": "dropdown",
    "dropdown": "ai_conv_score_classification_dropdown",
    "labels": {
      "de_DE": {
        "not_likely": "Unwahrscheinlich",
        "less_likely": "Weniger wahrscheinlich"
      },
      "ja_JP": {
        "not_likely": "???????",
        "less_likely": "??????"
      }
    }
  }
]
```

Response Example


```
  [
    {
        "name": "ai_conv_score_classification_dropdown",
        "labels": {
            "de_DE": {
                "not_likely": "Unwahrscheinlich",
                "less_likely": "Weniger wahrscheinlich"
            },
            "ja_JP": {
                "not_likely": "???????",
                "less_likely": "??????"
            }
        }
    },
    {
        "name": "account_type_dom",
        "labels": {
            "de_DE": {
                "Analyst": "Analytiker",
                "Competitor": "Wettbewerber"
            },
            "ja_JP": {
                "Analyst": "?????",
                "Competitor": "??????"
            }
        }
    }
]
```


Change Log

Version	Change
v11.13	Added /lang/labels/dropdown/ PUT endpoint.

/lang/labels/module PUT

Overview

Update display labels of fields for modules under different languages.

Summary

This endpoint is used to update display labels of fields for modules under different languages.

Request Arguments

Name	Type	Description	Required
name	string	The dropdown name.	True
labels	array	The new labels grouped by language keys.	True

Request

```
  [
  {
    "name": "Leads",
    "labels": {
      "en_us": {
          "ai_conv_score_classification_c": "Ai Conv Score Classificat
ion"
      },
      "de_DE": {
          "ai_conv_score_classification_c": "Ai Conv Score Klassifizie
rung"
      },
      "ja_JP": {
          "ai_conv_score_classification_c": "AiConv?????"
      }
    }
  }
]
```

Request example for multiple fields:

```
  [
  {
    "name": "Leads",
```

```
    "labels": {
      "en_us": {
          "ai_score_c": "Ai Score",
          "ai_conv_score_classification_c": "Ai Conv Score Classificat
ion",
          "description": "New Description"
      },
      "de_DE": {
          "ai_score_c": "Ai Punktzahl",
          "ai_conv_score_classification_c": "Ai Conv Score Klassifizie
rung",
          "description": "New German Description"
      },
      "ja_JP": {
          "ai_score_c": "????",
          "ai_conv_score_classification_c": "AiConv?????",
          "description": "New Japanese Description"
      }
    }
  },
  {
    "name": "Opportunities",
    "labels": {
      "de_DE": {
          "amount": "Menge",
          "amount_usdollar": "Betrag US-Dollar"
      },
      "ja_JP": {
          "amount": "?",
          "amount_usdollar": "?????"
      }
    }
  }
]
```

Response Example

```
  [
    {
        "name": "Leads",
        "labels": {
            "en_us": {
                "LBL_AI_SCORE": "Ai Score",
```

```
                "LBL_AI_CONV_SCORE_CLASSIFICATION": "Ai Conv Score Cla
ssification",
                "LBL_DESCRIPTION": "New Description"
            },
            "de_DE": {
                "LBL_AI_SCORE": "Ai Punktzahl",
                "LBL_AI_CONV_SCORE_CLASSIFICATION": "Ai Conv Score Kla
ssifizierung",
                "LBL_DESCRIPTION": "New German Description"
            },
            "ja_JP": {
                "LBL_AI_SCORE": "????",
                "LBL_AI_CONV_SCORE_CLASSIFICATION": "AiConv?????",
                "LBL_DESCRIPTION": "New Japanese Description"
            }
        }
    },
    {
        "type": "field",
        "module": "Opportunities",
        "labels": {
            "de_DE": {
                "LBL_LIKELY": "Menge",
                "LBL_AMOUNT_USDOLLAR": "Betrag US-Dollar"
            },
            "ja_JP": {
                "LBL_LIKELY": "?",
                "LBL_AMOUNT_USDOLLAR": "?????"
            }
        }
    }
]
```

Change Log

Version	Change
v11.13	Added /lang/labels/module/ PUT endpoint.

/logger POST

Overview

Logs a message to the Sugar Log

Request Arguments

Name	Type	Description	Required
level	String	The log level	True
message	String	The message to log	True
channel	String	Prefix for the log message, defaults to Main	false

Response Arguments

Name	Type	Description
status	bool	if the message was logged

Response

```
{
   "status":true,
}
```

Change Log

Version	Change
v10	Added /logger POST endpoint.

/login/content GET

Overview

Returns information to facilitate receiving marketing content from SugarCRM.

Request Arguments

Name	Type	Description	Required
selected_language	String	Code for the	False. Defaults to

Name	Type	Description	Required
		desired marketing content language. (Eg. "*en_us*", "*fr_FR*"). Providing a certain language code does not guarantee that content returned will be in the provided language. If an invalid language is specified, will assume "*en_us*".	"*en_us*"

Response Arguments

Name	Type	Description
content_url	String	URL suitable for embedding in an iFrame that links to new marketing material from SugarCRM. Will always be provided, but may be an empty string if there is no suitable content available.

Response

```
{
   "content_url": "https://marketing.sugarcrm.com/exciting-new-
content"
}
```

Change Log

Version	Change
v11_2	Added /login/content GET endpoint.

/login/marketingContentUrl GET

Overview

Returns the SugarCRM marketing content URL that is managed by the marketing team.

Summary

This API attempts to reach SugarCRM's marketing content URL. If the service is unreachable, a URL for the static content is returned.

Request Arguments

Name	Type	Description	Required
selected_language	String	Code for the desired marketing content language. (Eg. "*en_us*", "*fr_FR*"). Providing a certain language code does not guarantee that content returned will be in the provided language. If an invalid language is specified, will assume "*en_us*".	False. It uses the configured language if this parameter is missing. Defaults to "*en_us*".
static	Boolean	If set to true, return URL for static marketing content. Defaults to false.	False. If this parameter is missing, the API defaults to the dynamic content url.

Response Arguments

Name	Type	Description
url	String	URL suitable for embedding in an iFrame that links to marketing content from SugarCRM.

Response

```
  "https://www.sugarcrm.com/product-login-page-service/"
```

Change Log

Version	Change
v11_9	Added /login/marketingContentUrl GET endpoint.

/me GET

Overview

Returns the current user object.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

Response User

```
{
   "current_user":{
      "type":"user",
      "id":"1",
      "full_name":"Administrator",
      "timepref":"h:ia",
      "timezone":"America/Los_Angles",
      "user_name":"admin"
   }
}
```

Response Portal User

```
{
   "current_user":{
      "type":"support_portal",
      "id":"1234-567",
```

```
      "user_id":"abcd-123",
      "full_name":"Bill Williamson",
      "timepref":"h:ia",
      "timezone":"America/Denver",
      "user_name":"SupportPortalApi"
   }
}
```

Change Log

Version	Change
v10	Added /me GET endpoint.

/me PUT

Overview

Returns the current user object.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

Response User Example

```
{
   "current_user":{
      "type":"user",
      "id":"1",
      "full_name":"Administrator",
      "timepref":"h:ia",
      "timezone":"America/Los_Angles",
      "user_name":"admin"
   }
}
```

Response Portal User Example

```
{
   "current_user":{
      "type":"support_portal",
      "id":"1234-567",
      "user_id":"abcd-123",
      "full_name":"Bill Williamson",
      "timepref":"h:ia",
      "timezone":"America/Denver",
      "user_name":"SupportPortalApi"
   }
}
```


Change Log

Version	Change
v10	Added /me PUT endpoint.

/me/following GET

Overview

Returns all of the current users followed records

Request Arguments

limit and offset are available query parameters.

Response Arguments

Name	Type	Description

Response


```
{

}
```

Change Log

Version	Change
v10	Added /me/following GET endpoint.

/me/password POST

Overview

Create a new record of a specified type.

Summary

This endpoint allows for verification of the user's password. If client type is support_portal, it will update the corrending Contact. Otherwise, it will update User

Request Arguments

Name	Type	Description	Required
password_to_verify	String	The password to verify	True

Request

```
{
   "password_to_verify":"mycurrentpassword"
}
```

Response Arguments

Name	Type	Description
valid	Boolean	Returns the success of the password verification.

Response

```
{
    "valid":true
}
```

Change Log

Version	Change
v10	Added /me/password POST endpoint.

/me/password PUT

Overview

Create a new record of a specified type.

Summary

This endpoint allows for changes to the user's password. If client type is support_portal, it will update the corrending Contact. Otherwise, it will update User

Request Arguments

Name	Type	Description	Required
old_password	String	The current password.	True
new_password	String	The new password.	True

Request

```
{
   "old_password":"myoldpass",
   "new_password":"mynewpass"
}
```

Response Arguments

Name	Type	Description
valid	Boolean	Returns the success of

Name	Type	Description
		the password verification.
expiration	Date	When the password will expire.

Response

```
{
   "valid":true,
   "expiration":null
}
```

Change Log

Version	Change
v10	Added /me/password PUT endpoint.

/me/preference/:preference_name DELETE

Overview

Deletes a specific preference for the current user.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

Name	Type	Description

Response

```
{

}
```

Change Log

Version	Change
v10	Added /me/preference/:preference_name DELETE endpoint.

/me/preference/:preference_name GET

Overview

Returns a specific preference for the current user.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

Name	Type	Description

Response

```
{

}
```

Change Log

Version	Change
v10	Added /me/preference/:preference_name GET endpoint.

/me/preference/:preference_name POST

Overview

Creates a preference for the current user.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

Name	Type	Description

Response

```
{

}
```

Change Log

Version	Change
v10	Added /me/preference/:preference_name POST endpoint.

/me/preference/:preference_name PUT

Overview

Updates a specific preference for the current user.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

Name	Type	Description

Response

```
{
```

```
}
```

Change Log

Version	Change
v10	Added /me/preference/:preference_name PUT endpoint.

/me/preferences GET

Overview

Returns all the current user's stored preferences.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

Name	Type	Description

Response

```
{

}
```

Change Log

Version	Change
v10	Added /me/preferences GET endpoint.

/me/preferences PUT

Overview

Mass updates preferences for the user.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

Name	Type	Description

Response

```
{

}
```

Change Log

Version	Change
v10	Added /me/preferences PUT endpoint.

/metadata//:segment GET

Overview

[ADMIN] Gets the desired segment for a given module.

Summary

Gets the desired segment for a given module. This is used only for modules that are not visible and thus not in metatdata for the given context. This endpoint is only available to administrators.

Request Arguments

This endpoint does not accept any request arguments.

Response

```
{
    "fields": {
        Any fields associated with the module
    }
}
```

Change Log

Version	Change
v11_11	Added /metadata/<module>/:segment GET endpoint.

/mostactiveusers GET

Overview

Fetches the most active users for meetings, calls, inbound emails, and outbound emails.

Request Arguments

Name	Type	Description	Required
days	Integer	Returns most active users for the last specified quantity of days. Defaults to 30 days if not specified.	False

Request

```
http://{site_url}/rest/v10/mostactiveusers?days=30
```

Note: GET endpoint parameters are passed in the form of a query string.

Response Arguments

Response


```
{
   "meetings":{
      "user_id":"seed_sarah_id",
      "count":"20",
      "first_name":"Sarah",
      "last_name":"Smith"
   },
   "calls":{
      "user_id":"seed_will_id",
      "count":"7",
      "first_name":"Will",
      "last_name":"Westin"
   },
   "inbound_emails":{
      "user_id":"seed_sarah_id",
      "count":"20",
      "first_name":"Sarah",
      "last_name":"Smith"
   },
   "outbound_emails":{
      "user_id":"seed_max_id",
      "count":"17",
      "first_name":"Max",
      "last_name":"Jensen"
   }
}
```


Change Log

Version	Change
v10	Added /mostactiveusers GET endpoint.

/oauth2/bwc/login POST

ATTENTION: FOR INTERNAL USAGE ONLY

This endpoint is subject to change.

Overview

Retrieves a cookie from the OAuth token.

/oauth2/logout POST

Overview

Expires the token portion of the OAuth 2.0 specification.

Request Arguments

This endpoint does not accept any arguments.

Response Arguments

Name	Type	Description
success	Boolean	The success of the logout.

Response

```
{
    "success":true
}
```

Change Log

Version	Change
v10	Added /oauth2/logout POST endpoint.

/oauth2/sudo/:user_name POST

Overview

Get an access token as another user. The current user must be an admin in order to access this endpoint. This method is useful for integrations in order to be able to access the system with the same permission restrictions as a specified user. The calling user does not lose their existing token, this one is granted in addition.

Request Arguments

Name	Type	Description	Required
platform	String	Which platform on the session, defaults to "base"	False
client_id	String	The client id for the session, defaults to "sugar"	False

Request

```
{
   "client_id":"sugar",
   "platform":"base"
}
```

Response Arguments

Name	Type	Description
access_token	String	The access token needed to authenticate for other methods.
expires_in	Integer	The length of time until the access_token expires.
token_type	String	Should always be bearer.
scope	String	There is no scope implementation in the current release of Sugar.

Response

```
{
   "access_token":"c19fff9b-b767-233e-ebb4-512e369d3e39",
   "expires_in":3600,
   "token_type":"bearer",
   "scope":null
}
```

Change Log

Version	Change
v10	Added /oauth2/token/sudo/:user POST endpoint.

/oauth2/token POST

Overview

Retrieves the token portion of the OAuth 2.0 specification.

Request Arguments

Name	Type	Description	Required
grant_type	String	Type of request. Available grant types are "password" and "refresh_token".	True
client_id	String	Used to identify the client. A client_id of "sugar" will automatically create an OAuth Key in the system that is used for "password" authentication. A client_id of "support_portal" will create an OAuth Key that will allow for portal authentication. Additional client_id's can be created by the administrator in Admin > OAuth Keys to allow for additional grant types. If the client	True

Name	Type	Description	Required
		secret is populated, it will be validated against the client id.	
client_secret;	String	The clients secret key.	True
username	String	The username of the user authenticating to the system.	True
password	String	The plaintext password the user authenticating to the system.	True
platform	String	The platform type. Available types are "base", "mobile", and "portal".	True

Request for Password Grant Types

```
{
   "grant_type":"password",
   "client_id":"sugar",
   "client_secret":"",
   "username":"admin",
   "password":"password",
   "platform":"base"
}
```

Request for Refresh Token Grant Types

```
{
    "grant_type":"refresh_token",
    "refresh_token":"c1be5132-655b-1ca3-fb44-512e36709871",
    "client_id":"sugar",
    "client_secret":"",
    "platform":"base"
}
```

Response Arguments

Name	Type	Description
access_token	String	The access token needed to authenticate for other methods.
expires_in	Integer	The length of time until access_token expires in seconds.
token_type	String	The token type. Currently only "bearer" is supported.
null|		The Oauth scope. Normally returned as null.
refresh_token	String	The token needed to extend the access_token expiration timeout.
refresh_expires_in	Integer	The length of time until refresh_token expires in seconds.
download_token	String	The token used to download images and files.

Response


```
{
   "access_token":"802b64c0-5eac-8431-a541-5342d38ac527",
   "expires_in":3600,
   "token_type":"bearer",
   "scope":null,
   "refresh_token":"85053198-24b1-4521-b1a1-5342d382e0b7",
   "refresh_expires_in":1209600,
   "download_token":"8c9b5461-0d95-8d87-6084-5342d357b39e"
}
```


Change Log

Version	Change

v10	Added /oauth2/token POST endpoint.

/password/request GET

Overview

Sends an email request to reset a users password.

Request Arguments

Name	Type	Description	Required
email	String	The email of the user.	True
username	String	The username of the user.	True

Request

```
http://{site_url}/rest/v10/password/request?email=admin%40sugar.crm&username=admin
```

Note: GET endpoint parameters are passed in the form of a query string.

Response Arguments

Name	Type	Description
Success	boolean	Returns the result of sending the email.

Response

```
true
```

Change Log

Version	Change
v10	Added /password/request POST endpoint.

/ping GET

Overview

Responds with "pong" if the access_token is valid.

Request Arguments

This endpoint does not accept any arguments.

Response Arguments

Name	Type	Description
<response>	String	Returns pong is authenticated.

Response

pong

Change Log

Version	Change
v10	Added /ping GET endpoint.

/ping/whattimeisit GET

Overview

Responds with the current time in server format.

Request Arguments

This endpoint does not accept any arguments.

Response Arguments

Name	Type	Description
<server time>	String	Returns the servers time.

Response

```
<time>
```

Change Log

Version	Change
v10	Added /ping/whattimeisit GET endpoint.

/pmse_Business_Rules GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

Name	Type	Description	Required
filter	String	The filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query parameters in one of two ways for GET requests:	False

Name	Type	Description	Required
		1. By specifying individual filter arguments as distinct parameters. Example: filter[0][id]=1. 2. By specifying the whole filter as a single JSON-encoded string. Note that this syntax is currently not supported on certain modules. Example: filter=[{"id":"1"}].	
filter_id	String	Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND.	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
fields	String	Comma delimited	False

Name	Type	Description	Required
		list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters (applicable to link and collection fields). The fields id and date_modified will always be returned. Example: name,account_type,description,{"name":"opportunities","fields":["id","name","sales_status"],"order_by":"date_closed:desc"} For more details on additional field parameters, see Relate API and Collection API.	
view	String	Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list".	False

Name	Type	Description	Required
		Example: record	
order_by	String	How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon. Example: name:DESC,account_type:DESC,date_modified:ASC	False
q	String	A search expression, will search on this module. Cannot be used at the same time as a filter expression or id.	False
deleted	Boolean	Boolean to show deleted records in the result set.	False
nulls_last	Boolean	Boolean to return records with null values in order_by fields last in the result set.	False

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
   "filter":[
      {
         "name":"Nelson Inc"
      }
```

```
   ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
   "filter":[
      {
         "name":{
            "$starts":"Nelson"
         }
      }
   ]
}
```

Below is a list of operation types:

Operation	Description
$equals	Performs an exact match on that field.
$not_equals	Performs an exact match on that field.
$not_equals	Matches on non-matching values.
$starts	Matches on anything that starts with the value.
$ends	Matches anything that ends with the value.
$contains	Matches anything that contains the value
$in	Finds anything where field matches one of the values as

Operation		Description
		specified as an array.
	$not_in	Finds anything where field does not matches any of the values as specified as an array.
	$is_null	Checks if the field is null. This operation does not need a value specified.
	$not_null	Checks if the field is not null. This operation does not need a value specified.
	$lt	Matches when the field is less than the value.
	$lte	Matches when the field is less than or equal to the value.
	$gt	Matches when the field is greater than the value.
	$gte	Matches when the field is greater than or equal to the value.

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "$and" and "$or".

Example

```
{
   "filter":[
      {
         "$or":[
```

```
            {
               "name":"Nelson Inc"
            },
            {
               "name":"Nelson LLC"
            }
         ]
      }
   ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "$favorite" and "$owner".

Example

```
{
   "filter":[
      {
         "$favorite":"_this"
      }
   ]
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results

Name	Type	Description
		containing matched records.

Response

```
{
   "next_offset":-1,
   "records":[
      {
         "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
         "name":"Dale Spivey",
         "date_modified":"2013-02-28T05:03:00+00:00",
         "description":"",
         "opportunities": [
            {
               _module: "Opportunities",
               "id": "b0701501-1fab-8ae7-3942-540da93f5017",
               "name": "360 Vacations - 228 Units",
               "date_modified": "2014-09-08T16:05:00+03:00",
               "sales_status": "New"
            },
         ],
         "_acl": {
            "fields": {
            }
         }
      },
      {
         "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
         "name":"Florence Haddock",
         "date_modified":"2013-02-26T19:12:00+00:00",
         "description":"",
         "opportunities": [
            {
               _module: "Opportunities"
               date_modified: "2014-09-08T16:05:00+03:00"
               id: "9ce7c088-8ee4-7cd3-18f1-540da944d4c0"
               name: "360 Vacations - 312 Units"
               sales_status: "New"
            },
         ],
         "_acl": {
            "fields": {
            }
```

```
            }
        }
    ]
}
```

Change Log

Version	Change
v10	Added /<module>/filter GET endpoint.

/pmse_Business_Rules/:record/brules GET

Overview

Exports a .pbr file with a Process Business Rules definition

Summary

This endpoint will retrieve a .pbr file containing JSON encoded data of a Process Business Rules definition identified by the record input parameter.

Request Arguments

Name	Type	Description	Required
record	String	The Process Business Rule record ID	True

Response Arguments

Name	Type	Description
N/A		

Response

Note: The response is a MIME type application/pbr file attachment without any response parameters.

/pmse_Business_Rules/file/businessrules_import POST

Overview

Imports a Process Business Rules definition from a .pbr file

Summary

This endpoint will import a Process Business Rules definition from the uploaded .pbr file containing JSON encoded data.

Request Arguments

Name	Type	Description	Required
N/A			

Request

Note: A JSON encoded .pbr file attachment should be included with the request as part of the form-data.

Response Arguments

Name	Type	Description
businessrules_import	Object	Result of the import operation

Response

```
{
    "businessrules_import":
    {
         "success":true,
         "id":"e4396c91-81cd-c097-8149-5733bab68316"
    }
}
```

/pmse_Emails_Templates GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

Name	Type	Description	Required
filter	String	The filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query parameters in one of two ways for GET requests: 1. By specifying individual filter arguments as distinct parameters. Example: filter[0][id]=1. 2. By specifying the whole filter as a single JSON-encoded string. Note that this syntax is currently not	False

Name	Type	Description	Required
		supported on certain modules. Example: filter=[{"id":"1"}].	
filter_id	String	Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND.	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
fields	String	Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters (applicable to link and collection fields). The fields id and date_modified will always be returned. Example: name,account_type,description,{"name":"opportunities","fields":["id","name","sales_status"],"order_by":"date_closed:desc"}	False

Name	Type	Description	Required
		For more details on additional field parameters, see Relate API and Collection API.	
view	String	Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list". Example: record	False
order_by	String	How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon. Example: name:DESC,account_type:DESC,date_modified:ASC	False
q	String	A search expression, will search on this module. Cannot be used at the same time as a filter expression or id.	False

Name	Type	Description	Required
deleted	Boolean	Boolean to show deleted records in the result set.	False
nulls_last	Boolean	Boolean to return records with null values in order_by fields last in the result set.	False

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
   "filter":[
      {
         "name":"Nelson Inc"
      }
   ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
   "filter":[
      {
         "name":{
            "$starts":"Nelson"
         }
      }
```

```
    ]
}
```

Below is a list of operation types:

Operation	Description
$equals	Performs an exact match on that field.
$not_equals	Performs an exact match on that field.
$not_equals	Matches on non-matching values.
$starts	Matches on anything that starts with the value.
$ends	Matches anything that ends with the value.
$contains	Matches anything that contains the value
$in	Finds anything where field matches one of the values as specified as an array.
$not_in	Finds anything where field does not matches any of the values as specified as an array.
$is_null	Checks if the field is null. This operation does not need a value specified.
$not_null	Checks if the field is not null. This operation does not need a value specified.
$lt	Matches when the field is less than the value.

Operation		Description
	$lte	Matches when the field is less than or equal to the value.
	$gt	Matches when the field is greater than the value.
	$gte	Matches when the field is greater than or equal to the value.

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "$and" and "$or".

Example

```
{
   "filter":[
      {
         "$or":[
            {
               "name":"Nelson Inc"
            },
            {
               "name":"Nelson LLC"
            }
         ]
      }
   ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently

accepted module expressions are "$favorite" and "$owner".

Example

```
{
   "filter":[
      {
         "$favorite":"_this"
      }
   ]
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Response

```
{
   "next_offset":-1,
   "records":[
      {
         "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
         "name":"Dale Spivey",
         "date_modified":"2013-02-28T05:03:00+00:00",
         "description":"",
         "opportunities": [
            {
               _module: "Opportunities",
               "id": "b0701501-1fab-8ae7-3942-540da93f5017",
               "name": "360 Vacations - 228 Units",
               "date_modified": "2014-09-08T16:05:00+03:00",
               "sales_status": "New"
            },
```

```
         ],
         "_acl": {
            "fields": {
            }
         }
      },
      {
         "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
         "name":"Florence Haddock",
         "date_modified":"2013-02-26T19:12:00+00:00",
         "description":"",
         "opportunities": [
            {
               _module: "Opportunities"
               date_modified: "2014-09-08T16:05:00+03:00"
               id: "9ce7c088-8ee4-7cd3-18f1-540da944d4c0"
               name: "360 Vacations - 312 Units"
               sales_status: "New"
            },
         ],
         "_acl": {
            "fields": {
            }
         }
      }
   ]
}
```

Change Log

Version	Change
v10	Added /<module>/filter GET endpoint.

/pmse_Emails_Templates/:/find_modules GET

Overview

Get the related module list for a module

Summary

This endpoint will retrieve a list of related modules that can be accessed in the

Process Email Templates definition from the specified module. Input parameters are passed in as the URL query string. Pagination is supported.

Request Arguments

Name	Type	Description	Required
module_list	String	The name of the module the retrieved module list is related to.	True

Request

```
rest/v11/pmse_Emails_Templates/Quotes/find_modules?module_list=Quotes
```

Response Arguments

Name	Type	Description
search	String	The value of the input module_list parameter.
success	Boolean	Whether the related module list has been successfully retrieved.
result	Array	An array of related module objects.

Response


```
{
    "search":"Quotes",
    "success":true,
    "result":[
        {
            "value":"Quotes",
            "text":"\u003CQuotes\u003E",
            "module":"Quotes",
            "module_label":"Quotes",
            "module_name":"Quotes",
            "relationship":"Quotes"
        },
        {
```

```
        "value":"billing_accounts",
        "text":"Accounts (Bill to Account: billing_accounts)",
        "module":"Accounts",
        "module_label":"Accounts",
        "module_name":"Accounts",
        "relationship":"quotes_billto_accounts"
    },
    {
        "value":"shipping_accounts",
        "text":"Accounts (Ship to Account: shipping_accounts)",
        "module":"Accounts",
        "module_label":"Accounts",
        "module_name":"Accounts",
        "relationship":"quotes_shipto_accounts"
    },
    {
        "value":"billing_contacts",
        "text":"Contacts (Bill to Contact: billing_contacts)",
        "module":"Contacts",
        "module_label":"Contacts",
        "module_name":"Contacts",
        "relationship":"quotes_contacts_billto"
    },
    {
        "value":"shipping_contacts",
        "text":"Contacts (Ship to Contact: shipping_contacts)",
        "module":"Contacts",
        "module_label":"Contacts",
        "module_name":"Contacts",
        "relationship":"quotes_contacts_shipto"
    },
    {
        "value":"contracts",
        "text":"Contracts (Contracts: contracts)",
        "module":"Contracts",
        "module_label":"Contracts",
        "module_name":"Contracts",
        "relationship":"contracts_quotes"
    },
    {
        "value":"opportunities",
        "text":"Opportunities (Opportunity: opportunities)",
        "module":"Opportunities",
        "module_label":"Opportunities",
        "module_name":"Opportunities",
        "relationship":"quotes_opportunities"
```

```
        },
        {
            "value":"assigned_user_link",
            "text":"Users (Assigned to User: assigned_user_link)",
            "module":"Users",
            "module_label":"Users",
            "module_name":"Users",
            "relationship":"quotes_assigned_user"
        },
        {
            "value":"created_by_link",
            "text":"Users (Created by User: created_by_link)",
            "module":"Users",
            "module_label":"Users",
            "module_name":"Users",
            "relationship":"quotes_created_by"
        },
        {
            "value":"modified_user_link",
            "text":"Users (Modified by User: modified_user_link)",
            "module":"Users",
            "module_label":"Users",
            "module_name":"Users",
            "relationship":"quotes_modified_user"
        }
    ]
}
```


/pmse_Emails_Templates/:record/etemplate GET

Overview

Exports a .pet file with a Process Email Templates definition

Summary

This endpoint will retrieve a .pet file containing JSON encoded data of a Process Email Templates definition identified by the record input parameter.

Request Arguments

Name	Type	Description	Required
record	String	The Process Email	True

Name	Type	Description	Required
		Template record ID	

Response Arguments

Name	Type	Description
N/A		

Response

Note: The response is a MIME type application/pet file attachment without any response parameters.

/pmse_Emails_Templates/file/emailtemplates_import POST

Overview

Imports a Process Email Templates definition from a .pet file

Summary

This endpoint will import a Process Email Templates definition from the uploaded .pet file containing JSON encoded data.

Request Arguments

Name	Type	Description	Required
N/A			

Request

Note: A JSON encoded .pet file attachment should be included with the request as part of the form-data.

Response Arguments

Name	Type	Description
emailtemplates_import	Object	Result of the import operation

Response

```
{
    "emailtemplates_import":
    {
         "success":true,
         "id":"aa5f7fd9-73a9-8338-fd1a-573635ce5c51"
    }
}
```

/pmse_Emails_Templates/variables/find GET

Overview

Get the variable list for a module

Summary

This endpoint will retrieve a list of variables that can be used in the Process Email Templates definition from the specified module. Input parameters are passed in as the URL query string. Pagination is supported.

Request Arguments

Name	Type	Description	Required
base_module	String	The name of the target module of the Process Email Templates definition.	True
module_list	String	The name of the module the variable list is retrieved from. Should be related to the base_module.	True
q	String	Only variables with name containing this search string will be returned.	False
max_num	Integer	Maximum number of variables to return for this request.	False

Name	Type	Description	Required
offset	Integer	The index of the first variable in the whole list to return for this request. Use "end" to indicate no variable should be returned.	False
order_by	String	Property to sort the variable list by. For example, "name:asc,id:desc" will sort the returned variable list by name ASC first and then id DESC.	False

Request

```
rest/v11/pmse_Emails_Templates/variables/find?max_num=20&order_by=date
_modified%3Adesc&module_list=Quotes&base_module=Quotes
```

Response Arguments

Name	Type	Description
next_offset	Integer	The offset value that should be used to retrieve the next batch of the list.
records	Array	A sorted array of variable objects.

Response

```
{
    "next_offset":20,
    "records":[
        {
            "id":"date_quote_expected_closed",
            "_module":"Quotes",
            "name":"Valid Until",
```

```
        "rhs_module":"Quotes"
    },
    {
        "id":"total_usdollar",
        "_module":"Quotes",
        "name":"Total (US Dollar)",
        "rhs_module":"Quotes"
    },
    {
        "id":"total",
        "_module":"Quotes",
        "name":"Total",
        "rhs_module":"Quotes"
    },
    {
        "id":"tax_usdollar",
        "_module":"Quotes",
        "name":"Tax (US Dollar)",
        "rhs_module":"Quotes"
    },
    {
        "id":"tax",
        "_module":"Quotes",
        "name":"Tax",
        "rhs_module":"Quotes"
    },
    {
        "id":"subtotal_usdollar",
        "_module":"Quotes",
        "name":"Subtotal (US Dollar)",
        "rhs_module":"Quotes"
    },
    {
        "id":"subtotal",
        "_module":"Quotes",
        "name":"Subtotal",
        "rhs_module":"Quotes"
    },
    {
        "id":"shipping_address_state",
        "_module":"Quotes",
        "name":"Shipping State",
        "rhs_module":"Quotes"
    },
    {
        "id":"shipping_address_postalcode",
```

```
        "_module":"Quotes",
        "name":"Shipping Postal Code",
        "rhs_module":"Quotes"
    },
    {
        "id":"shipping_address_country",
        "_module":"Quotes",
        "name":"Shipping Country",
        "rhs_module":"Quotes"
    },
    {
        "id":"shipping_address_city",
        "_module":"Quotes",
        "name":"Shipping City",
        "rhs_module":"Quotes"
    },
    {
        "id":"shipping_address_street",
        "_module":"Quotes",
        "name":"Shipping Address",
        "rhs_module":"Quotes"
    },
    {
        "id":"shipping_usdollar",
        "_module":"Quotes",
        "name":"Shipping (US Dollar)",
        "rhs_module":"Quotes"
    },
    {
        "id":"shipping",
        "_module":"Quotes",
        "name":"Shipping",
        "rhs_module":"Quotes"
    },
    {
        "id":"quote_type",
        "_module":"Quotes",
        "name":"Quote Type",
        "rhs_module":"Quotes"
    },
    {
        "id":"name",
        "_module":"Quotes",
        "name":"Quote Subject",
        "rhs_module":"Quotes"
    },
```

```
        {
            "id":"quote_stage",
            "_module":"Quotes",
            "name":"Quote Stage",
            "rhs_module":"Quotes"
        },
        {
            "id":"quote_num",
            "_module":"Quotes",
            "name":"Quote Number",
            "rhs_module":"Quotes"
        },
        {
            "id":"purchase_order_num",
            "_module":"Quotes",
            "name":"Purchase Order Num",
            "rhs_module":"Quotes"
        },
        {
            "id":"payment_terms",
            "_module":"Quotes",
            "name":"Payment Terms",
            "rhs_module":"Quotes"
        }
    ]
}
```


/pmse_Inbox GET

Overview

Returns a list of Processes by user using filters

Summary

This endpoint gets a list of processes by user using filters

Request Arguments

Name	Type	Description	Required
order_by	string	How to sort the returned records, in a comma	false

Name	Type	Description	Required
		delimited list with the direction appended to the column name after a colon.	
q	string	A search expression, will search on this module. Cannot be used at the same time as a filter expression or id.	false
filter	string	The filter expression	false
filter_id	string	Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND	false
max_num	integer	A maximum number of records to return. Default is 20.	false
offset	integer	The number of records to skip over before records are returned. Default is 0.	false
fields	string	Comma delimited list of fields to return	false
deleted	boolean	Boolean to show deleted records in the result set.	false

Response Arguments

Name	Type	Description
records	<field>:<value>	Process records
next_offset	integer	Next offset for results

Response


```
{
    "next_offset":-1,
    "records":
    [
        {
            "id":"839ccd22-2925-11e6-83f3-6c4008960436",
            "date_modified":"2016-06-02T17:52:55-07:00",
            "created_by":"1",
            "cas_finish_date":"",
            "assigned_user_id":"1",
            "assigned_user_name":"Administrator",
            "assigned_user_link":
            {
                "full_name":"Administrator",
                "id":"1",
                "_acl":
                {
                    "fields":
                    {
                        "pwd_last_changed":
                        {
                            "write":"no",
                            "create":"no"
                        },
                        "last_login":
                        {
                            "write":"no",
                            "create":"no"
                        }
                    },
                    "delete":"no",
                    "_hash":"08b99a97c2e8d792f7a44d8882b5af6d"
                }
            },
            "_acl":
            {
                "fields":{}
            },
            "_module":"pmse_Project\/pmse_BpmFlow",
            "_locked_fields":[],
            "cas_id":"4",
            "act_assignment_method":"static",
            "cas_title":"PQR1",
```

```
            "pro_title":"PD1",
            "date_entered":"2016-06-02T17:52:55-07:00",
            "name":"PQR1",
            "cas_create_date":"2016-06-02T17:52:55-07:00",
            "flow_id":"839ccd22-2925-11e6-83f3-6c4008960436",
            "id2":"8380e512-2925-11e6-9600-6c4008960436",
            "task_name":"Activity # 1",
            "cas_assignment_method":"static",
            "cas_sugar_module":"Accounts",
            "cas_sugar_object_id":"82d5c470-2925-11e6-a269-6c400896043
6",
            "prj_id":"6e025038-2842-11e6-b4f7-6c4008960436",
            "in_time":true,
            "cas_user_id":"1",
            "prj_created_by":"1",
            "cas_user_id_full_name":"Administrator",
            "prj_user_id_full_name":"Administrator"
        },
    ]
}
```


/pmse_Inbox/:record/file/:field GET

Overview

Returns the process status image file

Summary

This endpoint returns the process status image file.

Request Arguments

Name	Type	Description	Required
record	string	The id of the record or image to get	true
field	string	The image type field to get the image from	true
_project	string	If set, will get an image for a project instead of a process	false

Response

There is no Response to this endpoint

/pmse_Inbox/AdhocReassign PUT

Overview

Deprecated endpoint.

Summary

This endpoint is deprecated and will be removed in a future release.

/pmse_Inbox/AdhocReassign/:data/:flowId GET

Overview

Deprecated endpoint.

Summary

This endpoint is deprecated and will be removed in a future release.

/pmse_Inbox/ReassignForm PUT

Overview

Deprecated endpoint.

Summary

This endpoint is deprecated and will be removed in a future release.

/pmse_Inbox/ReassignForm/:data/:flowId GET

Overview

Deprecated endpoint.

Summary

This endpoint is deprecated and will be removed in a future release.

/pmse_Inbox/cancelCases PUT

Overview

Call methods to cancel a process

Summary

This endpoint cancels one or more processes.

Request Arguments

Name	Type	Description	Required
cas_id	array	Collection of process trigger sequence ids	true

Response Arguments

Name	Type	Description
success	boolean	Result of the cancel operation

Response

```
{
    "success":true
}
```

/pmse_Inbox/case/:id/:idflow GET

Overview

Retrieve information of the process record

Summary

This endpoint gets information for a process record

Request Arguments

Name	Type	Description	Required
id	string	The id of the process requiring action	true
idflow	string	The id of the process trigger sequence	true

Response Arguments

Name	Type	Description
case	object	Process record

Response


```
{
    "case":
    {
        "flow":
        {
            "id":"b2d418ba-28ef-11e6-8ae6-6c4008960436",
            "name":"",
            "date_entered":"2016-06-02 18:27:42",
            "date_modified":"2016-06-02 18:27:42",
            "modified_user_id":"1",
            "created_by":"1",
            "description":null,
            "deleted":"0",
            "cas_id":"3",
            "cas_index":"3",
            "pro_id":"6e172e2c-2842-11e6-a4bb-6c4008960436",
            "cas_previous":"2",
            "cas_reassign_level":"0",
            "bpmn_id":"7c62acc2-2842-11e6-9b9c-6c4008960436",
            "bpmn_type":"bpmnActivity",
            "cas_assignment_method":"static",
            "cas_user_id":"1",
            "cas_thread":"1",
            "cas_flow_status":"FORM",
            "cas_sugar_module":"Accounts",
            "cas_sugar_object_id":"b2103fd0-28ef-11e6-8916-6c400896043
6",
            "cas_sugar_action":"None",
            "cas_adhoc_type":"",
```

```
            "cas_adhoc_parent_id":"",
            "cas_adhoc_actions":"[\u0022link_cancel\u0022,\u0022approv
e\u0022,\u0022reject\u0022,\u0022edit\u0022]",
            "cas_task_start_date":null,
            "cas_delegate_date":"2016-06-02 18:27:42",
            "cas_start_date":null,
            "cas_finish_date":null,
            "cas_due_date":null,
            "cas_queue_duration":"0",
            "cas_duration":"0",
            "cas_delay_duration":"0",
            "cas_started":"0",
            "cas_finished":"0",
            "cas_delayed":"0",
            "assigned_user_id":"1",
            "au_first_name":null,
            "au_last_name":"Administrator",
            "cbu_first_name":null,
            "cbu_last_name":"Administrator",
            "mbu_first_name":null,
            "mbu_last_name":"Administrator",
            "my_favorite":null
        },
        "reclaim":false,
        "buttons":
        [
            {
                "type":"button",
                "name":"cancel_button",
                "label":"Cancel",
                "css_class":"btn-invisible btn-link",
                "showOn":"edit",
                "events":
                {
                    "click":"button:cancel_button:click"
                }
            },
            {
                "type":"rowaction",
                "event":"case:approve",
                "name":"approve_button",
                "label":"Approve",
                "css_class":"btn btn-success"
            },
            {
                "type":"rowaction",
```

```
            "event":"case:reject",
            "name":"reject_button",
            "label":"Reject",
            "css_class":"btn btn-danger"
        },
        {
            "type":"actiondropdown",
            "name":"main_dropdown",
            "primary":true,
            "showOn":"view",
            "buttons":
            [
                {
                    "type":"rowaction",
                    "event":"button:edit_button:click",
                    "name":"edit_button",
                    "label":"Edit",
                    "acl_action":"edit"
                },
                {
                    "type":"rowaction",
                    "name":"history",
                    "label":"History",
                    "event":"case:history"
                },
                {
                    "type":"rowaction",
                    "name":"status",
                    "label":"Status",
                    "event":"case:status"
                },
                {
                    "type":"rowaction",
                    "name":"add-notes",
                    "label":"Show Notes",
                    "event":"case:add:notes"
                }
            ]
        }
    ],
    "readonly":[],
    "required":[],
    "title":
    {
        "time":
        {
```

```
                "expected_time_warning":false,
                "expected_time_message":false,
                "expected_time_view":false,
                "expected_time":""
            },
            "process":"PD1",
            "rec_name":"ABC3",
            "activity":"Activity # 1"
        },
        "inboxId":"b2b6e20e-28ef-11e6-bba8-6c4008960436",
        "flowId":"b2d418ba-28ef-11e6-8ae6-6c4008960436"
    }
}
```


/pmse_Inbox/casesList GET

Overview

Returns a list with the processes for Process Management

Summary

This endpoint gets a list of Processes.

Request Arguments

Name	Type	Description	Required
module_list	string	Identifier used to determine which status to filter against	false
order_by	string	How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon.	false
q	string	A search expression, will search on this	false

Name	Type	Description	Required
		module. Cannot be used at the same time as a filter expression or id.	
filter	string	The filter expression	false
filter_id	string	Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND.	false
max_num	integer	A maximum number of records to return. Default is 20.	false
offset	integer	The number of records to skip over before records are returned. Default is 0.	false
fields	string	Comma delimited list of fields to return	false
deleted	boolean	Boolean to show deleted records in the result set.	false

Response Arguments

Name	Type	Description
next_offset	Integer	Next offset
records	<field>/<value>	List of process attributes

Response

```
{
    {
        "next_offset":-1,
        "records":
        [
```

```
            {
                "id":"3922976c-284f-11e6-9f40-6c4008960436",
                "name":"ABC2",
                "date_entered":"2016-06-01T16:18:59-07:00",
                "date_modified":"2016-06-01T16:18:59-07:00",
                "modified_user_id":"1",
                "created_by":"1",
                "deleted":"0",
                "cas_id":"2",
                "cas_parent":"0",
                "cas_status":"IN PROGRESS",
  "pro_id":"6e172e2c-2842-11e6-a4bb-6c4008960436",
  "cas_title":"ABC2",
  "cas_custom_status":"",
  "cas_init_user":"1",
  "cas_create_date":"2016-06-01T16:18:59-07:00",
  "cas_update_date":"2016-06-01 23:18:59",
  "cas_finish_date":null,
  "cas_assigned_status":"UNASSIGNED",
  "cas_module":"Accounts",
  "team_id":"1",
  "team_set_id":"1",
  "assigned_user_id":"1",
  "assigned_user_name":"Administrator",
  "prj_id":"6e025038-2842-11e6-b4f7-6c4008960436",
  "prj_created_by":"1",
  "prj_module":"Accounts",
  "cas_sugar_module":"Accounts",
  "cas_sugar_object_id":"38beb03a-284f-11e6-b059-6c4008960436",
  "pro_title":"PD1",
  "prj_deleted":"0",
  "prj_user_id_full_name":"Administrator",
  "cas_user_id_full_name":"Sarah Smith"
            },
    }
}
```


/pmse_Inbox/changeCaseUser/:cas_id GET

Overview

Deprecated endpoint.

Summary

This endpoint is deprecated and will be removed in a future release.

/pmse_Inbox/clearLog/:typelog PUT

Overview

Clear the PMSE.log file log

Summary

This endpoint clears the contents of the PMSE.log file

Request Arguments

There are no request arguments for this endpoint

Response Arguments

Name	Type	Description
none	boolean	Result of the clear operation

Response

```
true
```

/pmse_Inbox/delete_notes/:id DELETE

Overview

Deletes a process note

Summary

TBD

Request Arguments

Name	Type	Description	Required
id	string	The note id to	true

Name	Type	Description	Required
		delete	

Response Arguments

Name	Type	Description
id	string	Id of the deleted note

Response

```
{
    "id":"da13553a-298a-11e6-b125-6c4008960436"
}
```

/pmse_Inbox/engine_claim PUT

Overview

Claims the processes to the current user

Summary

This endpoint claims the processes to the current user.

Request Arguments

Name	Type	Description	Required
cas_id	string	The id of the process trigger sequence	true
cas_index	string	The step in the process	true

Response

There is no Response to this endpoint

/pmse_Inbox/engine_route PUT

Overview

Evaluates the response of the user form Show Process [Approve, Reject, Route]

Summary

This endpoint handles Approve, Reject and Route actions for a process record.

Request Arguments

Name	Type	Description	Required
frm_action	string	The action to take when it comes to routing the record.	true

Response Arguments

Name	Type	Description
success	boolean	Result of the routing operation

Response

```
{
    "success":true,
}
```

/pmse_Inbox/filter GET

Overview

Returns a list of Processes by user

Summary

This endpoint gets a list of processes by user

Request Arguments

Name	Type	Description	Required
order_by	string	How to sort the returned records, in a comma delimited list with	false

Name	Type	Description	Required
		the direction appended to the column name after a colon.	
q	string	A search expression, will search on this module. Cannot be used at the same time as a filter expression or id.	false
filter	string	The filter expression	false
filter_id	string	Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND	false
max_num	integer	A maximum number of records to return. Default is 20.	false
offset	integer	The number of records to skip over before records are returned. Default is 0.	false
fields	string	Comma delimited list of fields to return	false
deleted	boolean	Boolean to show deleted records in the result set.	false

Response Arguments

Name	Type	Description
records	<field>:<value>	Process records
next_offset	integer	Next offset for results

Response

```
{
    "next_offset":-1,
    "records":
    [
        {
            "id":"839ccd22-2925-11e6-83f3-6c4008960436",
            "date_modified":"2016-06-02T17:52:55-07:00",
            "created_by":"1",
            "cas_finish_date":"",
            "assigned_user_id":"1",
            "assigned_user_name":"Administrator",
            "assigned_user_link":
            {
                "full_name":"Administrator",
                "id":"1",
                "_acl":
                {
                    "fields":
                    {
                        "pwd_last_changed":
                        {
                            "write":"no",
                            "create":"no"
                        },
                        "last_login":
                        {
                            "write":"no",
                            "create":"no"
                        }
                    },
                    "delete":"no",
                    "_hash":"08b99a97c2e8d792f7a44d8882b5af6d"
                }
            },
            "_acl":
            {
                "fields":{}
            },
            "_module":"pmse_Project\/pmse_BpmFlow",
            "_locked_fields":[],
            "cas_id":"4",
            "act_assignment_method":"static",
            "cas_title":"PQR1",
            "pro_title":"PD1",
            "date_entered":"2016-06-02T17:52:55-07:00",
```

```
            "name":"PQR1",
            "cas_create_date":"2016-06-02T17:52:55-07:00",
            "flow_id":"839ccd22-2925-11e6-83f3-6c4008960436",
            "id2":"8380e512-2925-11e6-9600-6c4008960436",
            "task_name":"Activity # 1",
            "cas_assignment_method":"static",
            "cas_sugar_module":"Accounts",
            "cas_sugar_object_id":"82d5c470-2925-11e6-a269-6c400896043
6",
            "prj_id":"6e025038-2842-11e6-b4f7-6c4008960436",
            "in_time":true,
            "cas_user_id":"1",
            "prj_created_by":"1",
            "cas_user_id_full_name":"Administrator",
            "prj_user_id_full_name":"Administrator"
        },
    ]
}
```


/pmse_Inbox/getLog GET

Overview

Return the text of the PMSE.log file

Summary

This endpoint gets the contents of the PMSE.log file

Request Arguments

There are no request arguments for this endpoint

Response Arguments

Name	Type	Description
none	string	Log Entries from PMSE.log

Response


```
"Test Log Entry1\nTest Log Entry2\n"
```

/pmse_Inbox/historyLog/:filter GET

Overview

Gets the history log for a process

Summary

This endpoint gets the history log for a process

Request Arguments

Name	Type	Description	Required
filter	string	The id of the process trigger sequence (cas_id)	true

Response Arguments

Name	Type	Description
success	boolean	Result of the operation
result	<field>:<value>	History records

Response


```
{
    "success":true,
    "result":
    [
        {
            "image":"index.php?entryPoint=download\u0026id=0b2dc2f0-23
77-11e6-9d2a-6c4008960436\u0026type=SugarFieldImage\u0026isTempFile=1"
,
            "user":"Administrator",
            "current_user":"Administrator",
            "due_date":"2016-06-03T16:18:59-07:00",
            "end_date":"2016-06-01T16:18:59-07:00",
            "current_date":"2016-06-02T10:53:14-07:00",
            "delegate_date":"2016-06-01T16:18:59-07:00",
            "start_date":"2016-06-01T16:18:59-07:00",
            "completed":true,
            "cas_user_id":"1",
            "data_info":"has created Process #2  on the Account record
```

```
  with the event \u0027Start Event # 1\u0027."
        },
    ]
}
```

/pmse_Inbox/note_list/:cas_id GET

Overview

Returns the notes list for a process

Summary

This endpoint gets the notes for a process

Request Arguments

Name	Type	Description	Required
cas_id	string	The id of the process trigger sequence	true

Response Arguments

Name	Type	Description
rowList	object	Note records
totalRows	integer	Total notes
currentOffset	integer	Current offset
currentDate	integer	Current date

Response

```
{
    "rowList":
    [
        {
            "pmse_bpm_notes__date_entered":"2016-06-02 16:52:21",
            "id":"602d4cb0-28e2-11e6-88c9-6c4008960436",
            "date_entered":"2016-06-02T09:52:21-07:00",
            "date_modified":"2016-06-02T09:52:21-07:00",
            "cas_id":"2",
```

```
            "cas_index":"1",
            "not_user_id":"1",
            "not_user_recipient_id":"",
            "not_type":"GENERAL",
            "not_date":null,
            "not_status":"ACTIVE",
            "not_availability":"",
            "not_content":"Test Note1",
            "not_recipients":"",
            "first_name":null,
            "last_name":"Administrator",
            "picture":"0b2dc2f0-2377-11e6-9d2a-6c4008960436"
        }
    ],
    "totalRows":1,
    "currentOffset":0,
    "currentDate":"2016-06-02 16:52:39"
}
```


/pmse_Inbox/reactivateFlows PUT

Overview

Deprecated endpoint.

Summary

This endpoint is deprecated and will be removed in a future release.

/pmse_Inbox/reassignFlows PUT

Overview

Call methods to reassign processes

Summary

This endpoint reassigns a collection of processes

Request Arguments

Name	Type	Description	Required
flow_data	array	Collection of flow data arrays, each containing a user_id property	true

Response Arguments

Name	Type	Description
success	boolean	Result of the reassign operation

Response

```
{
    "success":true,
}
```

/pmse_Inbox/reassignFlows/:record GET

Overview

Retrieve information to reassign processes

Summary

This endpoint gets information necessary to reassign a process

Request Arguments

Name	Type	Description	Required
record	string	The id of the process trigger sequence	true

Response Arguments

Name	Type	Description
success	boolean	Result of the routing operation
next_offset	integer	Next offset for more

Name	Type	Description
		results
records	<field>:<value>	List of process attributes

Response

```
{
    "success":true,
    "next_offset":-1,
    "records":
    [
        {
            "cas_id":"2",
            "cas_index":"3",
            "cas_task_start_date":null,
            "cas_delegate_date":"2016-06-01T16:18:59-07:00",
            "cas_flow_status":"FORM",
            "cas_user_id":"seed_sarah_id",
            "cas_due_date":"",
            "cas_sugar_module":"Accounts",
            "bpmn_id":"7c62acc2-2842-11e6-9b9c-6c4008960436",
            "act_name":"Activity # 1",
            "act_assignment_method":"static",
            "act_expected_time":"eyJ0aW1lIjpudWxsLCJ1bml0IjoiaG91ciJ9"
,
            "cas_expected_time":"",
            "assigned_user":"Sarah Smith"
        }
    ]
}
```

/pmse_Inbox/save_notes POST

Overview

Creates a new note for a process

Summary

This endpoint creates a new note for a process

Request Arguments

Name	Type	Description	Required
data	array	Collection of data for a note containing the following:	true
data.cas_id	string	The id of the process trigger sequence	true
data.cas_index	string	The step in the process	true
data.not_content	string	The content of the note	true
data.not_type	string	The type of note, defaults to 'GENERAL'	false

Response Arguments

Name	Type	Description
success	boolean	Result of the update
id	string	Id of new note
date_entered	date	Date of creation

Response

```
{
    "success":true,
    "id":"06c075d0-28e7-11e6-8bb1-6c4008960436",
    "date_entered":"2016-06-02T10:25:39-07:00"
}
```

/pmse_Inbox/settings GET

Overview

Retrieve settings for the PA engine

Summary

This endpoint gets settings for the SugarBPM™ engine

Request Arguments

Name	Type	Description	Required
fields	string	Comma separated list of SugarBPM™ config keys to return. If not sent, all config settings will be returned.	false

Response Arguments

Name	Type	Description
logger_level	string	Level of logging
error_number_of_cycles	integer	Error number of cycles
error_timeout	integer	Timeout for errors

Response

```
{
    "logger_level":"critical",
    "error_number_of_cycles":"10",
    "error_timeout":"40"
}
```

/pmse_Inbox/settings PUT

Overview

Update settings for the SugarBPM™ engine

Summary

This endpoint updates settings for the SugarBPM™ engine

Request Arguments

Name	Type	Description	Required
data	array	Key/value map of config keys and their associated	false

Name	Type	Description	Required
		values for the SugarBPM™ engine to update	

Response Arguments

Name	Type	Description
success	boolean	Result of update
data	object	Key/Value map of config keys and their new values

Response

```
{
    "success":true,
    "data":
    {
        "logger_level":"critical",
        "error_number_of_cycles":"10",
        "error_timeout":"40"
    }
}
```

/pmse_Inbox/unattendedCases GET

Overview

Retrieves the processes to show on Unattended Process view

Summary

This endpoint gets the list of unattended processes

Request Arguments

Name	Type	Description	Required
module_list	string	Identifier used to determine which field to filter against	true

Name	Type	Description	Required
q	string	Filter applied to a field to determine which processes to return	false
order_by	string	Sort definition to apply to the result	false

Response Arguments

Name	Type	Description
next_offset	integer	Next offset for more results
records	<field>:<value>	List of unattended process attributes

Response


```
{
    "next_offset":"-1",
    "records":
    [
        {
            "id":"3922976c-284f-11e6-9f40-6c4008960436",
            "assigned_user_id":"1",
            "date_modified":"2016-06-01 23:18:59",
            "date_entered":"2016-06-01T16:18:59-07:00",
            "name":"ABC2",
            "cas_id":"2",
            "cas_title":"ABC2",
            "cas_status":"IN PROGRESS",
            "pro_title":"PD1",
            "pro_id":"6e172e2c-2842-11e6-a4bb-6c4008960436",
            "cas_init_user":"1",
            "cas_user_full_name":"Sarah Smith",
            "prj_id":"6e025038-2842-11e6-b4f7-6c4008960436",
            "prj_user_id_full_name":"Administrator",
            "cas_sugar_object_id":"38beb03a-284f-11e6-b059-6c400896043
6",
            "cas_sugar_module":"Accounts",
            "assigned_user_name":"Administrator"
        }
    ]
}
```

/pmse_Inbox/userListByTeam/:id GET

Overview

Deprecated endpoint.

Summary

This endpoint is deprecated and will be removed in a future release.

/pmse_Project GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

Name	Type	Description	Required
filter	String	The filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query parameters in one of two ways for GET requests:	False

Name	Type	Description	Required
		1. By specifying individual filter arguments as distinct parameters. Example: filter[0][id]=1. 2. By specifying the whole filter as a single JSON-encoded string. Note that this syntax is currently not supported on certain modules. Example: filter=[{"id":"1"}].	
filter_id	String	Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND.	False
max_num	Integer	A maximum number of records to return. Default is 20.	False
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False
fields	String	Comma delimited	False

Name	Type	Description	Required
		list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters (applicable to link and collection fields). The fields id and date_modified will always be returned. Example: name,account_type,description,{"name":"opportunities","fields":["id","name","sales_status"],"order_by":"date_closed:desc"} For more details on additional field parameters, see Relate API and Collection API.	
view	String	Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list".	False

Name	Type	Description	Required
		Example: record	
order_by	String	How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon. Example: name:DESC,account_type:DESC,date_modified:ASC	False
q	String	A search expression, will search on this module. Cannot be used at the same time as a filter expression or id.	False
deleted	Boolean	Boolean to show deleted records in the result set.	False
nulls_last	Boolean	Boolean to return records with null values in order_by fields last in the result set.	False

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
   "filter":[
      {
         "name":"Nelson Inc"
      }
```

```
   ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
   "filter":[
      {
         "name":{
            "$starts":"Nelson"
         }
      }
   ]
}
```

Below is a list of operation types:

Operation	Description
$equals	Performs an exact match on that field.
$not_equals	Performs an exact match on that field.
$not_equals	Matches on non-matching values.
$starts	Matches on anything that starts with the value.
$ends	Matches anything that ends with the value.
$contains	Matches anything that contains the value
$in	Finds anything where field matches one of the values as

Operation		Description
		specified as an array.
	$not_in	Finds anything where field does not matches any of the values as specified as an array.
	$is_null	Checks if the field is null. This operation does not need a value specified.
	$not_null	Checks if the field is not null. This operation does not need a value specified.
	$lt	Matches when the field is less than the value.
	$lte	Matches when the field is less than or equal to the value.
	$gt	Matches when the field is greater than the value.
	$gte	Matches when the field is greater than or equal to the value.

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "$and" and "$or".

Example

```
{
   "filter":[
      {
         "$or":[
```

```
            {
               "name":"Nelson Inc"
            },
            {
               "name":"Nelson LLC"
            }
         ]
      }
   ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "$favorite" and "$owner".

Example

```
{
   "filter":[
      {
         "$favorite":"_this"
      }
   ]
}
```

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records.
records	Array	An array of results

Name	Type	Description
		containing matched records.

Response

```
{
   "next_offset":-1,
   "records":[
      {
         "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
         "name":"Dale Spivey",
         "date_modified":"2013-02-28T05:03:00+00:00",
         "description":"",
         "opportunities": [
            {
               _module: "Opportunities",
               "id": "b0701501-1fab-8ae7-3942-540da93f5017",
               "name": "360 Vacations - 228 Units",
               "date_modified": "2014-09-08T16:05:00+03:00",
               "sales_status": "New"
            },
         ],
         "_acl": {
            "fields": {
            }
         }
      },
      {
         "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
         "name":"Florence Haddock",
         "date_modified":"2013-02-26T19:12:00+00:00",
         "description":"",
         "opportunities": [
            {
               _module: "Opportunities"
               date_modified: "2014-09-08T16:05:00+03:00"
               id: "9ce7c088-8ee4-7cd3-18f1-540da944d4c0"
               name: "360 Vacations - 312 Units"
               sales_status: "New"
            },
         ],
         "_acl": {
            "fields": {
            }
```

```
            }
        }
    ]
}
```

Change Log

Version	Change
v10	Added /<module>/filter GET endpoint.

/pmse_Project/:record/dproject GET

Overview

Exports a .bpm file with a Process Definition

Summary

This endpoint will retrieve a .bpm file containing JSON encoded data of a Process Definition identified by the record input parameter.

Request Arguments

Name	Type	Description	Required
record	String	The Process Definition record ID	True

Response Arguments

Name	Type	Description
N/A		

Response

Note: The response is a MIME type application/bpm file attachment without any response parameters.

/pmse_Project/:record/verify GET

Overview

Verifies whether the Process Definition has any pending processes

Summary

This endpoint will verify whether the Process Definition identified by the record input parameter has any currently pending processes based on it.

Request Arguments

Name	Type	Description	Required
record	String	The Process Definition record ID	True

Response Arguments

Name	Type	Description
N/A		

Response

Note: The response is a string of either "true" or "false" indicating whether any pending processes exist.

/pmse_Project/ActivityDefinition/:record GET

Overview

Retrieves the definition data for an activity

Summary

This endpoint will retrieve the JSON encoded definition data for the Process Definition activity identified by the record input parameter.

Request Arguments

Name	Type	Description	Required
record	String	The value for the act_uid field in the pmse_bpmn_activity record	True

Response Arguments

Name	Type	Description
success	Boolean	The status of the response
<field>	<value>	Field value in the pmse_bpmn_activity record

Response


```
{
    "success":true,
    "id":"566f0156-0ce3-75b4-da56-573cb73984aa",
    "name":"Action # 1",
    "date_entered":"2016-05-18 18:41:29",
    "date_modified":"2016-05-18 18:42:00",
    "modified_user_id":"1",
    "created_by":"1",
    "description":null,
    "deleted":"0",
    "pro_id":"9f8c4fcd-8d4c-949d-50b8-573caddf8c0c",
    "act_type":"TASK",
    "act_duration":"0",
    "act_duration_unit":"DAYS",
    "act_send_notification":"0",
    "act_assignment_method":"balanced",
    "act_assign_team":"",
    "act_assign_user":"",
    "act_value_based_assignment":"",
    "act_reassign":"0",
    "act_reassign_team":"",
    "act_adhoc":"0",
    "act_adhoc_behavior":"",
    "act_adhoc_team":"",
    "act_response_buttons":"",
    "act_last_user_assigned":"",
    "act_field_module":"Accounts",
    "act_fields":"[{\u0022name\u0022:\u0022Assigned to\u0022,\u0022fie
ld\u0022:\u0022assigned_user_id\u0022,\u0022value\u0022:\u0022currentu
ser\u0022,\u0022type\u0022:\u0022user\u0022,\u0022label\u0022:\u0022Cu
rrent user\u0022}]",
    "act_readonly_fields":
    [
        {
```

```
        "name":"phone_alternate",
        "label":"Alternate Phone",
        "readonly":false
    },
    {
        "name":"annual_revenue",
        "label":"Annual Revenue",
        "readonly":false
    },
    {
        "name":"billing_address_city",
        "label":"Billing City",
        "readonly":false
    },
    {
        "name":"billing_address_country",
        "label":"Billing Country",
        "readonly":false
    },
    {
        "name":"billing_address_postalcode",
        "label":"Billing Postal Code",
        "readonly":false
    },
    {
        "name":"billing_address_state",
        "label":"Billing State",
        "readonly":false
    },
    {
        "name":"billing_address_street",
        "label":"Billing Street",
        "readonly":false
    },
    {
        "name":"created_by",
        "label":"Created By",
        "readonly":false
    },
    {
        "name":"description",
        "label":"Description",
        "readonly":false
    },
    {
        "name":"duns_num",
```

```
        "label":"DUNS",
        "readonly":false
    },
    {
        "name":"employees",
        "label":"Employees",
        "readonly":false
    },
    {
        "name":"facebook",
        "label":"Facebook Account",
        "readonly":false
    },
    {
        "name":"phone_fax",
        "label":"Fax",
        "readonly":false
    },
    {
        "name":"googleplus",
        "label":"Google Plus ID",
        "readonly":false
    },
    {
        "name":"industry",
        "label":"Industry",
        "readonly":false
    },
    {
        "name":"modified_user_id",
        "label":"Modified By",
        "readonly":false
    },
    {
        "name":"name",
        "label":"Name",
        "readonly":false
    },
    {
        "name":"phone_office",
        "label":"Office Phone",
        "readonly":false
    },
    {
        "name":"ownership",
        "label":"Ownership",
```

```
        "readonly":false
    },
    {
        "name":"rating",
        "label":"Rating",
        "readonly":false
    },
    {
        "name":"shipping_address_city",
        "label":"Shipping City",
        "readonly":false
    },
    {
        "name":"shipping_address_country",
        "label":"Shipping Country",
        "readonly":false
    },
    {
        "name":"shipping_address_postalcode",
        "label":"Shipping Postal Code",
        "readonly":false
    },
    {
        "name":"shipping_address_state",
        "label":"Shipping State",
        "readonly":false
    },
    {
        "name":"shipping_address_street",
        "label":"Shipping Street",
        "readonly":false
    },
    {
        "name":"sic_code",
        "label":"SIC Code",
        "readonly":false
    },
    {
        "name":"ticker_symbol",
        "label":"Ticker Symbol",
        "readonly":false
    },
    {
        "name":"twitter",
        "label":"Twitter Account",
        "readonly":false
```

```
    },
    {
        "name":"account_type",
        "label":"Type",
        "readonly":false
    },
    {
        "name":"website",
        "label":"Website",
        "readonly":false
    }
],
"act_expected_time":
{
    "time":"",
    "unit":"hour"
},
"act_required_fields":
[
    {
        "name":"phone_alternate",
        "label":"Alternate Phone",
        "required":false
    },
    {
        "name":"annual_revenue",
        "label":"Annual Revenue",
        "required":false
    },
    {
        "name":"billing_address_city",
        "label":"Billing City",
        "required":false
    },
    {
        "name":"billing_address_country",
        "label":"Billing Country",
        "required":false
    },
    {
        "name":"billing_address_postalcode",
        "label":"Billing Postal Code",
        "required":false
    },
    {
        "name":"billing_address_state",
```

```
        "label":"Billing State",
        "required":false
    },
    {
        "name":"billing_address_street",
        "label":"Billing Street",
        "required":false
    },
    {
        "name":"created_by",
        "label":"Created By",
        "required":false
    },
    {
        "name":"description",
        "label":"Description",
        "required":false
    },
    {
        "name":"duns_num",
        "label":"DUNS",
        "required":false
    },
    {
        "name":"employees",
        "label":"Employees",
        "required":false
    },
    {
        "name":"facebook",
        "label":"Facebook Account",
        "required":false
    },
    {
        "name":"phone_fax",
        "label":"Fax",
        "required":false
    },
    {
        "name":"googleplus",
        "label":"Google Plus ID",
        "required":false
    },
    {
        "name":"industry",
        "label":"Industry",
```

```
        "required":false
    },
    {
        "name":"modified_user_id",
        "label":"Modified By",
        "required":false
    },
    {
        "name":"phone_office",
        "label":"Office Phone",
        "required":false
    },
    {
        "name":"ownership",
        "label":"Ownership",
        "required":false
    },
    {
        "name":"rating",
        "label":"Rating",
        "required":false
    },
    {
        "name":"shipping_address_city",
        "label":"Shipping City",
        "required":false
    },
    {
        "name":"shipping_address_country",
        "label":"Shipping Country",
        "required":false
    },
    {
        "name":"shipping_address_postalcode",
        "label":"Shipping Postal Code",
        "required":false
    },
    {
        "name":"shipping_address_state",
        "label":"Shipping State",
        "required":false
    },
    {
        "name":"shipping_address_street",
        "label":"Shipping Street",
        "required":false
```

```
        },
        {
            "name":"sic_code",
            "label":"SIC Code",
            "required":false
        },
        {
            "name":"ticker_symbol",
            "label":"Ticker Symbol",
            "required":false
        },
        {
            "name":"twitter",
            "label":"Twitter Account",
            "required":false
        },
        {
            "name":"account_type",
            "label":"Type",
            "required":false
        },
        {
            "name":"website",
            "label":"Website",
            "required":false
        }
    ],
    "act_related_modules":[],
    "act_service_url":null,
    "act_service_params":null,
    "act_service_method":null,
    "act_update_record_owner":null,
    "execution_mode":"DEFAULT",
    "assigned_user_id":"",
    "au_first_name":null,
    "au_last_name":null,
    "cbu_first_name":null,
    "cbu_last_name":"Administrator",
    "mbu_first_name":null,
    "mbu_last_name":"Administrator",
    "my_favorite":null
}
```

/pmse_Project/ActivityDefinition/:record PUT

Overview

Updates the definition data for an activity

Summary

This endpoint will update the Process Definition activity identified by the record input parameter with the data provided in the request payload.

Request Arguments

Name	Type	Description	Required
record	String	The value for the act_uid field in the pmse_bpmn_activity record	True

Request Payload

```
{
    "data":
    {
        "act_field_module":"Accounts",
        "act_fields":"[{\"name\":\"Assigned to\",\"field\":\"assigned_user_id\",\"value\":\"owner\",\"type\":\"user\",\"label\":\"Record owner\"}]"
    }
}
```

Response Arguments

Name	Type	Description
success	Boolean	The status of the update operation

Response

```
{
    "success":true
```

```
}
```

/pmse_Project/CrmData/:data/:filter GET

Overview

Retrieves information about Fields, Modules, Users, Roles, etc.

Summary

This endpoint will retrieve various data related to the Process Definition.

Request Arguments

Name	Type	Description	Required
data	String	The type of data to be retrieved	True
filter	String	Filtering criteria to be applied to data retrieved	False

Request

Note: Additional input parameters can be provided in the form of a URL query string.

```
/rest/v11/pmse_Project/CrmData/related/Accounts?cardinality=one
```

Response Arguments

Name	Type	Description
success	Boolean	The status of the response
search	String	The value of the filter input parameter
result	Array	The data requested

Response

```
{
    "search":"Accounts",
    "success":true,
    "result":
    [
        {
            "value":"Accounts",
            "text":"\u003CAccounts\u003E",
            "module":"Accounts",
            "module_label":"Accounts",
            "module_name":"Accounts",
            "relationship":"Accounts"
        },
        {
            "value":"member_of",
            "text":"Accounts (Member of: member_of)",
            "module":"Accounts",
            "module_label":"Accounts",
            "module_name":"Accounts",
            "relationship":"member_accounts"
        },
        {
            "value":"campaign_accounts",
            "text":"Campaigns (Campaigns: campaign_accounts)",
            "module":"Campaigns",
            "module_label":"Campaigns",
            "module_name":"Campaigns",
            "relationship":"campaign_accounts"
        },
        {
            "value":"assigned_user_link",
            "text":"Users (Assigned to User: assigned_user_link)",
            "module":"Users",
            "module_label":"Users",
            "module_name":"Users",
            "relationship":"accounts_assigned_user"
        },
        {
            "value":"created_by_link",
            "text":"Users (Created by User: created_by_link)",
            "module":"Users",
            "module_label":"Users",
            "module_name":"Users",
            "relationship":"accounts_created_by"
        },
        {
```

```
            "value":"modified_user_link",
            "text":"Users (Modified by User: modified_user_link)",
            "module":"Users",
            "module_label":"Users",
            "module_name":"Users",
            "relationship":"accounts_modified_user"
        }
    ]
}
```


/pmse_Project/CrmData/:record/:filter PUT

Overview

Updates information about Fields, Modules, Users, Roles, etc.

Summary

This endpoint will update various data related to the Process Definition with the data provided in the request payload.

Request Arguments

Name	Type	Description	Required
record	String	The type of data to be updated	True
filter	String	Filtering criteria to be applied to data updated	False

Request

```
/rest/v11/pmse_Project/CrmData/clearEventCriteria/383414365573cf64b8ea
a65039872849
```

Request Payload

```
{
    "data":null
}
```

Response Arguments

Name	Type	Description
success	Boolean	The status of the update operation

Response

```
{
    "success":true
}
```

/pmse_Project/EventDefinition/:record GET

Overview

Retrieves the definition data for an event

Summary

This endpoint will retrieve the JSON encoded definition data for the Process Definition event identified by the record input parameter.

Request Arguments

Name	Type	Description	Required
record	String	The value for the evn_uid field in the pmse_bpmn_event record	True

Request

Note: Additional input parameters can be provided in the form of a URL query string.

```
    /rest/v11/pmse_Project/EventDefinition/632263607573cb74a582507093735454?related=modules
```

Response Arguments

Name	Type	Description
<field>	<value>	Field value in the pmse_bpmn_activity record

Response


```
{
    "id":"80a37c82-8be8-c668-a547-573cb791b905",
    "name":"",
    "date_entered":"2016-05-18 18:41:29",
    "date_modified":"2016-05-18 18:41:42",
    "created_by":"1",
    "description":null,
    "deleted":"0",
    "evn_status":"ACTIVE",
    "evn_type":"START",
    "evn_module":"Accounts",
    "evn_criteria":"",
    "evn_params":"new",
    "evn_script":"",
    "execution_mode":"DEFAULT",
    "au_first_name":null,
    "au_last_name":null,
    "cbu_first_name":null,
    "cbu_last_name":"Administrator",
    "mbu_first_name":null,
    "mbu_last_name":"Administrator",
    "my_favorite":null,
    "evn_uid":"632263607573cb74a582507093735454",
    "related":
    {
        "modules":
        [
            {
                "value":"Accounts",
                "text":"Accounts"
            },
            {
                "value":"Bugs",
                "text":"Bugs"
```

```
},
{
    "value":"Calls",
    "text":"Calls"
},
{
    "value":"Campaigns",
    "text":"Campaigns"
},
{
    "value":"Cases",
    "text":"Cases"
},
{
    "value":"Contacts",
    "text":"Contacts"
},
{
    "value":"Contracts",
    "text":"Contracts"
},
{
    "value":"Documents",
    "text":"Documents"
},
{
    "value":"KBContents",
    "text":"Knowledge Base"
},
{
    "value":"Leads",
    "text":"Leads"
},
{
    "value":"Meetings",
    "text":"Meetings"
},
{
    "value":"Notes",
    "text":"Notes"
},
{
    "value":"Opportunities",
    "text":"Opportunities"
},
{
```

```
                "value":"ProductTemplates",
                "text":"Product Catalog"
            },
            {
                "value":"ProjectTask",
                "text":"Project Tasks"
            },
            {
                "value":"Project",
                "text":"Projects"
            },
            {
                "value":"Products",
                "text":"Quoted Line Items"
            },
            {
                "value":"Quotes",
                "text":"Quotes"
            },
            {
                "value":"RevenueLineItems",
                "text":"Revenue Line Items"
            },
            {
                "value":"Prospects",
                "text":"Targets"
            },
            {
                "value":"Tasks",
                "text":"Tasks"
            }
        ]
    }
}
```


/pmse_Project/EventDefinition/:record PUT

Overview

Updates the definition data for an event

Summary

This endpoint will update the Process Definition event identified by the record input parameter with the data provided in the request payload.

Request Arguments

Name	Type	Description	Required
record	String	The value for the evn_uid field in the pmse_bpmn_event record	True

Request Payload

There are 5 event types in SugarBPM™.

Start Event

```
{
    "data":
    {
        "evn_module":"Accounts",
        "evn_params":"updated",
        "evn_criteria":null
    }
}
```

Timer Event

```
{
    "data":
    {
        "evn_timer_type":"duration",
        "evn_duration_criteria":"10",
        "evn_duration_params":"minute",
        "evn_criteria":null
    }
}
```

Receive Message Event

```
{
```

```
    "data":
    {
        "evn_criteria":[
            {
              "expType":"MODULE",
              "expSubtype":"DropDown",
              "expLabel":"Accounts (Industry is equal to Banking)",
              "expValue":"Banking",
              "expOperator":"equals",
              "expModule":"Accounts",
              "expField":"industry"
            }
        ]
    }
}
```

Send Message Event

```
{
    "data":
    {
        "evn_module":"Accounts",
        "evn_criteria":"8444baae-c73f-11e8-afa5-6c400895ea84",
        "evn_params":
        {
            "to":[
               {
                  "type":"recipient",
                  "module":"calls",
                  "moduleLabel":"Calls",
                  "value":"email1",
                  "label":"%MODULE% : Email Address",
                  "filter":{
                     "expType":"MODULE",
                     "expSubtype":"DropDown",
                     "expLabel":"Calls (Status is equal to Scheduled)"
,
                     "expValue":"Planned",
                     "expOperator":"equals",
                     "expModule":"calls",
                     "expField":"status"
                  },
                  "chainedRelationship":{
```

```
                    "module":"contact_parent",
                    "moduleLabel":"Contacts",
                    "filter":{
                       "expType":"MODULE",
                       "expSubtype":"DropDown",
                       "expLabel":"Contacts (Lead Source is equal to
Cold Call)",
                       "expValue":"Cold Call",
                       "expOperator":"equals",
                       "expModule":"contact_parent",
                       "expField":"lead_source"
                    }
                 }
              }
           ],
           "cc":[],
           "bcc":[]
        }
     }
}
```

End Event

```
{
   "data":{
      "activities":{
      },
      "events":{
         "4849630305bb7e68ae078d5012588603":{
            "evn_uid":"4849630305bb7e68ae078d5012588603",
            "evn_behavior":"THROW",
            "evn_marker":"EMPTY",
            "evn_message":"",
            "action":"UPDATE"
         }
      },
      "gateways":{
      },
      "flows":{
      },
      "artifacts":{
      },
      "prj_uid":"389d3d5a-c8ee-11e8-bc27-6c400895ea84"
```

```
   },
   "id":"389d3d5a-c8ee-11e8-bc27-6c400895ea84",
   "operation":"update",
   "wrapper":"Project"
}
```

Response Arguments

Name	Type	Description
N/A		

Response

```
null
```

/pmse_Project/GatewayDefinition/:record GET

Overview

Retrieves the definition data for a gateway

Summary

This endpoint will retrieve the JSON encoded definition data for the Process Definition gateway identified by the record input parameter.

Request Arguments

Name	Type	Description	Required
record	String	The value for the gat_uid field in the pmse_bpmn_gateway record	True

Response Arguments

Name	Type	Description
success	Boolean	The status of the response

Name	Type	Description
data	Array	The definition data for the gateway

Response


```
{
    "success":true,
    "data":
    [
        {
            "flo_uid":"985528634573cabc8901d04029355451",
            "flo_condition":"[{\u0022expType\u0022:\u0022BUSINESS_RULE
S\u0022,\u0022expLabel\u0022:\u0022Action # 1 is equal to \\\u0022TRUE
\\\u0022\u0022,\u0022expValue\u0022:\u0022TRUE\u0022,\u0022expOperator
\u0022:\u0022equals\u0022,\u0022expField\u0022:\u0022b6d2500c-
bf29-2ad7-67a2-569d5debc7d9\u0022}]"
        }
    ]
}
```

/pmse_Project/GatewayDefinition/:record PUT

Overview

Updates the definition data for a gateway

Summary

This endpoint will update the Process Definition gateway identified by the record input parameter with the data provided by the request payload.

Request Arguments

Name	Type	Description	Required
record	String	The value for the gat_uid field in the pmse_bpmn_gateway record	True

Request Payload

```
{
    "data":
    [
        {
            "flo_uid":"985528634573cabc8901d04029355451",
            "flo_condition":""
        }
    ]
}
```

Response Arguments

Name	Type	Description
success	Boolean	The status of the update operation

Response

```
{
    "success":true
}
```

/pmse_Project/file/project_import POST

Overview

Imports a Process Definition from a .bpm file

Summary

This endpoint will import a Process Definition from the uploaded .bpm file containing JSON encoded data.

Request Arguments

Name	Type	Description	Required
selectedIds	array	A list of IDs for dependent elements to import.	False

Request

Note: A JSON encoded .bpm file attachment should be included with the request as part of the form-data. An optional array of IDs can be included to indicate which dependent Business Rules and Email Templates to import and link with the Process Definition. The file must contain the metadata for the dependent elements as well for the selectedIds to be used. **Important:** The request format must be multipart/form-data.

Response Arguments

Name	Type	Description
project_import	Object	Result of the import operation

Response

```
{
    "project_import":
    {
        "success":true,
        "id":"459f05de-3c3a-c0e7-ce1e-573cabe79e7c",
        "br_warning":true,
        "et_warning":true
    }
}
```

Compatibility

SugarBPM™ files exported from Sugar version 7.6.1.0 or earlier cannot be imported to Sugar instances running 7.6.2.0 or later.

Change Log

Version	Change
v11_2	Added support for selectedIds to specify which dependent elements to import.
v10	Added /pmse_Project/file/project_import POST endpoint.

/pmse_Project/project/:record GET

Overview

Retrieves the schema data to be used by the Process Definition designer

Summary

This endpoint will retrieve the JSON encoded schema data for the Process Definition identified by the record input parameter.

Request Arguments

Name	Type	Description	Required
record	String	The Process Definition record ID	True

Response Arguments

Name	Type	Description
success	Boolean	The status of the response
project	Object	The schema data for the Process Definition

Response


```
{
    "success":true,
    "project":
    {
        "id":"2119da13-f748-4741-3858-573caddda034",
        "name":"PD_AC",
        "date_entered":"2016-05-18 17:58:07",
        "date_modified":"2016-05-18 18:41:45",
        "modified_user_id":"1",
        "created_by":"1",
        "description":null,
        "deleted":"0",
        "prj_uid":"2119da13-f748-4741-3858-573caddda034",
        "prj_target_namespace":null,
        "prj_expression_language":null,
        "prj_type_language":null,
```

```
"prj_exporter":null,
"prj_exporter_version":null,
"prj_author":null,
"prj_author_version":null,
"prj_original_source":null,
"prj_status":"INACTIVE",
"prj_module":"Accounts",
"team_id":"1",
"team_set_id":"1",
"team_set_selected_id":"",
"assigned_user_id":"1",
"au_first_name":null,
"au_last_name":"Administrator",
"cbu_first_name":null,
"cbu_last_name":"Administrator",
"mbu_first_name":null,
"mbu_last_name":"Administrator",
"tn_name":"Global",
"tn_name_2":"",
"my_favorite":null,
"following":"0",
"pro_id":"9f8c4fcd-8d4c-949d-50b8-573caddf8c0c",
"diagram":
[
    {
        "id":"9ce21d87-f81a-5be2-86e8-573cad955d06",
        "name":"PD_AC",
        "date_entered":"2016-05-18 17:58:07",
        "date_modified":"2016-05-18 17:58:07",
        "modified_user_id":"1",
        "created_by":"1",
        "description":null,
        "deleted":"0",
        "dia_uid":"343198086573cad309cc1d6078585899",
        "prj_id":"2119da13-f748-4741-3858-573caddda034",
        "dia_is_closable":"0",
        "assigned_user_id":"1",
        "activities":
        [
            {
                "id":"9ce21d87-f81a-5be2-86e8-573cad955d06",
                "name":"PD_AC",
                "date_entered":"2016-05-18 17:58:07",
                "date_modified":"2016-05-18 17:58:07",
                "modified_user_id":"1",
                "created_by":"1",
```

```
                        "description":null,
                        "deleted":"0",
                        "dia_uid":"343198086573cad309cc1d6078585899",
                        "prj_id":"2119da13-f748-4741-3858-573caddda034
",
                        "dia_is_closable":"0",
                        "assigned_user_id":"1",
                        "activities":
                        [
                            {
                                "id":"566f0156-0ce3-75b4-da56-573cb739
84aa",
                                "name":"Action # 1",
                                "date_entered":"2016-05-18 18:41:29",
                                "date_modified":"2016-05-18 18:41:45",
                                "created_by":"1",
                                "description":"",
                                "deleted":"0",
                                "act_uid":"552527013573cb7565825f20504
63919",
                                "act_type":"TASK",
                                "act_is_for_compensation":"0",
                                "act_start_quantity":"1",
                                "act_completion_quantity":"0",
                                "act_task_type":"SCRIPTTASK",
                                "act_implementation":"",
                                "act_instantiate":"0",
                                "act_script_type":"CHANGE_FIELD",
                                "act_script":"",
                                "act_loop_type":"NONE",
                                "act_test_before":"0",
                                "act_loop_maximum":"0",
                                "act_loop_condition":"",
                                "act_loop_cardinality":"0",
                                "act_loop_behavior":"",
                                "act_is_adhoc":"0",
                                "act_is_collapsed":"0",
                                "act_completion_condition":"",
                                "act_ordering":"PARALLEL",
                                "act_cancel_remaining_instances":"1",
                                "act_protocol":"",
                                "act_method":"",
                                "act_is_global":"0",
                                "act_referer":"",
                                "act_default_flow":"0",
                                "act_master_diagram":"",
```

```
            "bou_x":"245",
            "bou_y":"106",
            "bou_width":"35",
            "bou_height":"35",
            "bou_container":"bpmnDiagram",
            "act_name":"Action # 1"
        }
    ],
    "id":"566f0156-0ce3-75b4-da56-573cb73984aa",
    "name":"Action # 1",
    "date_entered":"2016-05-18 18:41:29",
    "date_modified":"2016-05-18 18:41:45",
    "created_by":"1",
    "description":"",
    "deleted":"0",
    "act_uid":"552527013573cb7565825f2050463919",
    "act_type":"TASK",
    "act_is_for_compensation":"0",
    "act_start_quantity":"1",
    "act_completion_quantity":"0",
    "act_task_type":"SCRIPTTASK",
    "act_implementation":"",
    "act_instantiate":"0",
    "act_script_type":"CHANGE_FIELD",
    "act_script":"",
    "act_loop_type":"NONE",
    "act_test_before":"0",
    "act_loop_maximum":"0",
    "act_loop_condition":"",
    "act_loop_cardinality":"0",
    "act_loop_behavior":"",
    "act_is_adhoc":"0",
    "act_is_collapsed":"0",
    "act_completion_condition":"",
    "act_ordering":"PARALLEL",
    "act_cancel_remaining_instances":"1",
    "act_protocol":"",
    "act_method":"",
    "act_is_global":"0",
    "act_referer":"",
    "act_default_flow":"0",
    "act_master_diagram":"",
    "bou_x":"245",
    "bou_y":"106",
    "bou_width":"35",
    "bou_height":"35",
```

```
            "bou_container":"bpmnDiagram",
            "act_name":"Action # 1"
        }
    ],
    "events":
    [
        {
            "id":"80a37c82-8be8-c668-a547-573cb791b905",
            "name":"Start Event # 1",
            "date_entered":"2016-05-18 18:41:29",
            "date_modified":"2016-05-18 18:41:29",
            "created_by":"1",
            "description":"",
            "deleted":"0",
            "evn_uid":"632263607573cb74a582507093735454",
            "evn_type":"START",
            "evn_marker":"MESSAGE",
            "evn_is_interrupting":"1",
            "evn_attached_to":"",
            "evn_cancel_activity":"0",
            "evn_activity_ref":"",
            "evn_wait_for_completion":"1",
            "evn_error_name":"",
            "evn_error_code":"",
            "evn_escalation_name":"",
            "evn_escalation_code":"",
            "evn_condition":"",
            "evn_message":"",
            "evn_operation_name":"",
            "evn_operation_implementation":"",
            "evn_time_date":"",
            "evn_time_cycle":"",
            "evn_time_duration":"",
            "evn_behavior":"CATCH",
            "bou_x":"132",
            "bou_y":"108",
            "bou_width":"33",
            "bou_height":"33",
            "bou_container":"bpmnDiagram",
            "evn_name":"Start Event # 1"
        },
        {
            "id":"82894d13-1af8-f039-2cd3-573cb755559e",
            "name":"End Event # 1",
            "date_entered":"2016-05-18 18:41:29",
            "date_modified":"2016-05-18 18:41:29",
```

```
            "created_by":"1",
            "description":"",
            "deleted":"0",
            "evn_uid":"282428179573cb7585826d6045140545",
            "evn_type":"END",
            "evn_marker":"EMPTY",
            "evn_is_interrupting":"1",
            "evn_attached_to":"",
            "evn_cancel_activity":"0",
            "evn_activity_ref":"",
            "evn_wait_for_completion":"1",
            "evn_error_name":"",
            "evn_error_code":"",
            "evn_escalation_name":"",
            "evn_escalation_code":"",
            "evn_condition":"",
            "evn_message":"",
            "evn_operation_name":"",
            "evn_operation_implementation":"",
            "evn_time_date":"",
            "evn_time_cycle":"",
            "evn_time_duration":"",
            "evn_behavior":"THROW",
            "bou_x":"349",
            "bou_y":"107",
            "bou_width":"33",
            "bou_height":"33",
            "bou_container":"bpmnDiagram",
            "evn_name":"End Event # 1"
        }
    ],
    "gateways":[],
    "artifacts":[],
    "flows":
    [
        {
            "id":"974e5216-9d76-5e30-7241-573cb73b7761",
            "name":"",
            "date_entered":"2016-05-18 18:41:36",
            "date_modified":"2016-05-18 18:41:36",
            "created_by":"1",
            "description":"",
            "deleted":"0",
            "flo_uid":"287649205573cb75b582a78086279138",
            "flo_type":"SEQUENCE",
            "flo_element_origin":"632263607573cb74a5825070
```

```
93735454",
                        "flo_element_origin_type":"bpmnEvent",
                        "flo_element_origin_port":"0",
                        "flo_element_dest":"552527013573cb7565825f2050
463919",
                        "flo_element_dest_type":"bpmnActivity",
                        "flo_element_dest_port":"0",
                        "flo_is_inmediate":"",
                        "flo_condition":"",
                        "flo_eval_priority":"0",
                        "flo_x1":"165",
                        "flo_y1":"125",
                        "flo_x2":"243",
                        "flo_y2":"122",
                        "flo_state":
                        [
                            {
                                "x":165,
                                "y":125
                            },
                            {
                                "x":204,
                                "y":125
                            },
                            {
                                "x":204,
                                "y":122
                            },
                            {
                                "x":243,
                                "y":122
                            }
                        ],
                        "prj_id":"2119da13-f748-4741-3858-573caddda034
"
                    },
                    {
                        "id":"9fc25911-3e2d-89f1-8d37-573cb73639bd",
                        "name":"",
                        "date_entered":"2016-05-18 18:41:36",
                        "date_modified":"2016-05-18 18:41:36",
                        "created_by":"1",
                        "description":"",
                        "deleted":"0",
                        "flo_uid":"121706749573cb75d582c86016080199",
                        "flo_type":"SEQUENCE",
```

```
                        "flo_element_origin":"552527013573cb7565825f20
50463919",
                        "flo_element_origin_type":"bpmnActivity",
                        "flo_element_origin_port":"0",
                        "flo_element_dest":"282428179573cb7585826d6045
140545",
                        "flo_element_dest_type":"bpmnEvent",
                        "flo_element_dest_port":"0",
                        "flo_is_inmediate":"",
                        "flo_condition":"",
                        "flo_eval_priority":"0",
                        "flo_x1":"282",
                        "flo_y1":"122",
                        "flo_x2":"349",
                        "flo_y2":"124",
                        "flo_state":
                        [
                            {
                                "x":282,
                                "y":122
                            },
                            {
                                "x":315,
                                "y":122
                            },
                            {
                                "x":315,
                                "y":124
                            },
                            {
                                "x":349,
                                "y":124
                            }
                        ],
                        "prj_id":"2119da13-f748-4741-3858-573caddda034
"
                    }
                ]
            }
        ],
        "process_definition":
        {
            "id":"9f8c4fcd-8d4c-949d-50b8-573caddf8c0c",
            "name":"",
            "date_entered":"2016-05-18 17:58:07",
            "date_modified":"2016-05-18 17:58:07",
```

```
    "created_by":"1",
    "description":null,
    "deleted":"0",
    "pro_module":"Accounts",
    "pro_status":"INACTIVE",
    "pro_locked_variables":[],
    "pro_terminate_variables":"",
    "execution_mode":"SYNC"
},
"prj_name":"PD_AC",
"prj_description":null,
"script_tasks":
{
    "add_related_record":
    {
       "1":
        {
            "value":"members",
            "text":"Accounts (Members: members)",
            "module":"Accounts",
            "module_label":"Accounts",
            "module_name":"Accounts",
            "relationship":"member_accounts"
        },
        "2":
        {
            "value":"bugs",
            "text":"Bugs (Bugs: bugs)",
            "module":"Bugs",
            "module_label":"Bugs",
            "module_name":"Bugs",
            "relationship":"accounts_bugs"
        },
        "3":
        {
            "value":"calls",
            "text":"Calls (Calls: calls)",
            "module":"Calls",
            "module_label":"Calls",
            "module_name":"Calls",
            "relationship":"account_calls"
        },
        "4":
        {
            "value":"cases",
            "text":"Cases (Cases: cases)",
```

```
        "module":"Cases",
        "module_label":"Cases",
        "module_name":"Cases",
        "relationship":"account_cases"
    },
    "5":
    {
        "value":"contacts",
        "text":"Contacts (Contacts: contacts)",
        "module":"Contacts",
        "module_label":"Contacts",
        "module_name":"Contacts",
        "relationship":"accounts_contacts"
    },
    "6":
    {
        "value":"contracts",
        "text":"Contracts (Contracts: contracts)",
        "module":"Contracts",
        "module_label":"Contracts",
        "module_name":"Contracts",
        "relationship":"account_contracts"
    },
    "7":
    {
        "value":"documents",
        "text":"Documents (Documents: documents)",
        "module":"Documents",
        "module_label":"Documents",
        "module_name":"Documents",
        "relationship":"documents_accounts"
    },
    "8":
    {
        "value":"leads",
        "text":"Leads (Leads: leads)",
        "module":"Leads",
        "module_label":"Leads",
        "module_name":"Leads",
        "relationship":"account_leads"
    },
    "9":
    {
        "value":"meetings",
        "text":"Meetings (Meetings: meetings)",
        "module":"Meetings",
```

```
                    "module_label":"Meetings",
                    "module_name":"Meetings",
                    "relationship":"account_meetings"
                },
                "10":
                {
                    "value":"notes",
                    "text":"Notes (Notes: notes)",
                    "module":"Notes",
                    "module_label":"Notes",
                    "module_name":"Notes",
                    "relationship":"account_notes"
                },
                "11":
                {
                    "value":"opportunities",
                    "text":"Opportunities (Opportunity: opportunities)
",
                    "module":"Opportunities",
                    "module_label":"Opportunities",
                    "module_name":"Opportunities",
                    "relationship":"accounts_opportunities"
                },
                "12":
                {
                    "value":"project",
                    "text":"Projects (Projects: project)",
                    "module":"Projects",
                    "module_label":"Projects",
                    "module_name":"Project",
                    "relationship":"projects_accounts"
                },
                "13":
                {
                    "value":"products",
                    "text":"Quoted Line Items (Products: products)",
                    "module":"Quoted Line Items",
                    "module_label":"Quoted Line Items",
                    "module_name":"Products",
                    "relationship":"products_accounts"
                },
                "14":
                {
                    "value":"quotes_shipto",
                    "text":"Quotes (Quotes Ship to: quotes_shipto)",
                    "module":"Quotes",
```

```
                    "module_label":"Quotes",
                    "module_name":"Quotes",
                    "relationship":"quotes_shipto_accounts"
                },
                "15":
                {
                    "value":"quotes",
                    "text":"Quotes (Quotes: quotes)",
                    "module":"Quotes",
                    "module_label":"Quotes",
                    "module_name":"Quotes",
                    "relationship":"quotes_billto_accounts"
                },
                "16":
                {
                    "value":"tasks",
                    "text":"Tasks (Tasks: tasks)",
                    "module":"Tasks",
                    "module_label":"Tasks",
                    "module_name":"Tasks",
                    "relationship":"account_tasks"
                }
            }
        }
    }
}
```


/pmse_Project/project/:record PUT

Overview

Updates the schema data from the Process Definition designer

Summary

This endpoint will update the Process Definition identified by the record input parameter with data provided in the request payload.

Request Arguments

Name	Type	Description	Required
record	String	The Process Definition record	True

Name	Type	Description	Required
		ID	

Request Payload

```
{
    "data":
    {
        "activities":{},
        "gateways":
        {
            "407410601573ce3125db315056173234":
            {
                "action":"REMOVE",
                "gat_uid":"407410601573ce3125db315056173234"
            }
        },
        "events":{},
        "artifacts":{},
        "flows":{},
        "prj_uid":"2119da13-f748-4741-3858-573caddda034"
    },
    "id":"2119da13-f748-4741-3858-573caddda034",
    "operation":"update",
    "wrapper":"Project"
}
```

Response Arguments

Name	Type	Description
success	Boolean	The status of the update operation

Response

```
{
    "success":true
}
```

/pmse_Project/validateCrmData/:data/:filter GET

Overview

Validates information about Fields, Modules, Users, Roles, etc.

Summary

This endpoint will validate the existence of various data related to the Process Definition.

Request Arguments

Name	Type	Description	Required
data	String	The type of data to be retrieved	True
filter	String	Filtering criteria to be applied to data retrieved	False
key	String	Key of the entry to search for within the given data and filter	True

Request

Note: "key" should be provided in the form of a URL query string.

Example Request:

```
  /rest/v11_10/pmse_Project/validateCrmData/field/Accounts?key=data_en
tered
```

Response Arguments

Name	Type	Description
result	Boolean	The result of the validation. True if the key exists in the specified data set; false otherwise

Response

```
  {
    result: true
}
```

Change Log

Version	Change
v11_10	Added /pmse_Project/validateCrmData/:data/:filter GET endpoint.

/recent GET

Overview

Returns all of the current users recently viewed records.

Request Arguments

Name	Type	Description	Required
module_list	String	Comma delimited list of modules to return recently viewed records. Example: Accounts,Contacts	True
fields	String	Comma delimited list of fields to return. The field date_modified will always be returned. Example: id,name	False
date	String	The date expression. Filter recently viewed records from this date.	False
limit	Integer	A maximum number of records to return. Default is 20.	False

Name	Type	Description	Required
offset	Integer	The number of records to skip over before records are returned. Default is 0.	False

Request

```
http://{site_url}/rest/v10/recent?module_list=Accounts%2CContacts
```

Note: GET endpoint parameters are passed in the form of a query string.

Response Arguments

Name	Type	Description
next_offset	String	The next offset to start fetching additional records.
records	Array	An array of recently viewed records.

Response


```
{
   "next_offset":-1,
   "records":[
      {
         "my_favorite":false,
         "following":false,
         "id":"e4213959-35cd-119b-5cd6-5342e8be16f6",
         "name":"Leila Purifoy",
         "date_entered":"2014-04-07T13:58:50-04:00",
         "date_modified":"2014-04-07T13:58:50-04:00",
         "modified_user_id":"1",
         "modified_by_name":"Administrator",
         "created_by":"1",
         "created_by_name":"Administrator",
         "doc_owner":"",
         "description":"",
         "deleted":false,
         "assigned_user_id":"seed_will_id",
         "assigned_user_name":"Will Westin",
```

```
"team_count":"",
"team_name":[
   {
      "id":"East",
      "name":"East",
      "name_2":"",
      "primary":true
   },
   {
      "id":"West",
      "name":"West",
      "name_2":"",
      "primary":false
   }
],
"email":[
   {
      "email_address":"support62@example.biz",
      "invalid_email":false,
      "opt_out":true,
      "primary_address":false,
      "reply_to_address":false
   },
   {
      "email_address":"sales39@example.com",
      "invalid_email":false,
      "opt_out":false,
      "primary_address":true,
      "reply_to_address":true
   }
],
"email1":"sales39@example.com",
"email2":"support62@example.biz",
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"",
"salutation":"",
"first_name":"Leila",
"last_name":"Purifoy",
"full_name":"Leila Purifoy",
"title":"President",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
```

```
"phone_home":"(841) 469-8223",
"phone_mobile":"(286) 010-9553",
"phone_work":"(369) 075-2809",
"phone_other":"",
"phone_fax":"",
"primary_address_street":"345 Sugar Blvd.",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Santa Monica",
"primary_address_state":"NY",
"primary_address_postalcode":"52255",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
"assistant":"",
"assistant_phone":"",
"picture":"",
"email_and_name1":"",
"lead_source":"Trade Show",
"account_name":"Sea Region Inc",
"account_id":"b1cd3a55-7e84-e53b-954e-5342e85b63f1",
"dnb_principal_id":"",
"opportunity_role_fields":"",
"opportunity_role_id":"",
"opportunity_role":"",
"reports_to_id":"",
"report_to_name":"",
"birthdate":"",
"portal_name":"LeilaPurifoy5",
"portal_active":true,
"portal_password":true,
"portal_password1":null,
"portal_app":"",
"preferred_language":"",
"campaign_id":"",
"campaign_name":"",
"c_accept_status_fields":"",
"m_accept_status_fields":"",
"accept_status_id":"",
"accept_status_name":"",
"accept_status_calls":"",
```

```
      "accept_status_meetings":"",
      "sync_contact":false,
      "mkto_sync":false,
      "mkto_id":null,
      "mkto_lead_score":null,
      "_acl":{
         "fields":{

         }
      },
      "_module":"Contacts",
      "_last_viewed_date":"2014-04-07T14:43:46-04:00"
   },
   {
      "my_favorite":false,
      "following":false,
      "id":"e8f7eb6d-2647-7e57-df30-5342e83c622d",
      "name":"Draft Diversified Energy Inc",
      "date_entered":"2014-04-07T13:58:50-04:00",
      "date_modified":"2014-04-07T13:58:50-04:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "doc_owner":"",
      "description":"",
      "deleted":false,
      "assigned_user_id":"seed_max_id",
      "assigned_user_name":"Max Jensen",
      "team_count":"",
      "team_name":[
         {
            "id":"West",
            "name":"West",
            "name_2":"",
            "primary":true
         }
      ],
      "email":[
         {
            "email_address":"dev.kid.kid@example.de",
            "invalid_email":false,
            "opt_out":false,
            "primary_address":true,
            "reply_to_address":false
         },
```

```
      {
         "email_address":"info.sales@example.co.uk",
         "invalid_email":false,
         "opt_out":false,
         "primary_address":false,
         "reply_to_address":false
      }
   ],
   "email1":"dev.kid.kid@example.de",
   "email2":"info.sales@example.co.uk",
   "invalid_email":false,
   "email_opt_out":false,
   "email_addresses_non_primary":"",
   "facebook":"",
   "twitter":"",
   "googleplus":"",
   "account_type":"Customer",
   "industry":"Education",
   "annual_revenue":"",
   "phone_fax":"",
   "billing_address_street":"111 Silicon Valley Road",
   "billing_address_street_2":"",
   "billing_address_street_3":"",
   "billing_address_street_4":"",
   "billing_address_city":"Los Angeles",
   "billing_address_state":"CA",
   "billing_address_postalcode":"26022",
   "billing_address_country":"USA",
   "rating":"",
   "phone_office":"(790) 406-0049",
   "phone_alternate":"",
   "website":"www.phonesugar.de",
   "ownership":"",
   "employees":"",
   "ticker_symbol":"",
   "shipping_address_street":"111 Silicon Valley Road",
   "shipping_address_street_2":"",
   "shipping_address_street_3":"",
   "shipping_address_street_4":"",
   "shipping_address_city":"Los Angeles",
   "shipping_address_state":"CA",
   "shipping_address_postalcode":"26022",
   "shipping_address_country":"USA",
   "parent_id":"",
   "sic_code":"",
   "duns_num":"",
```

```
         "parent_name":"",
         "campaign_id":"",
         "campaign_name":"",
         "_acl":{
            "fields":{

            }
         },
         "_module":"Accounts",
         "_last_viewed_date":"2014-04-07T14:43:36-04:00"
      }
   ]
}
```


Change Log

Version	API	Change
7.2.0	v10	Added /recent GET endpoint.

/rssfeed GET

Overview

Consumes an RSS feed as a proxy and returns the feed up to a certain number of entries.

Request Arguments

Name	Type	Description	Required
feed_url	String	A fully qualified URL to an RSS feed.	True
limit	Integer	Maximum number of entries to fetch. If not provided, the API will return up to $sugar_config['rss_feed_max_entries'] or 20 if no config option is set.	False

Request

```
http://{site_url}/rest/v10/rssfeed?feed_url=http%3A%2F%2Ffqd.rssfeed.u
rl%2F&limit=10
```

Note: GET endpoint parameters are passed in the form of a query string.

Response Arguments

Name	Type	Description
title	String	The title of the RSS Feed. This can be blank.
link	String	The URL to the source of the RSS Feed. This can be blank.
description	String	A description of the feed. This can be blank.
publication_date	Timestamp	A timestamp of when the feed was published.
entries	Array	An array of entries, each of which will contain the following values: • title • description • link • publication_date • source • author

Response

```
{
    "feed":{
        "title":"Sample Feed Title",
        "link":"http:\/\/www.samplefeed.com\/feed-link-url.htm",
        "description":"A sample of a feed description",
        "publication_date":"Tue, 10 Aug 2014 13:38:55 -0800",
        "entries":[
            {
                "title":"First entry title",
                "description":"A blurb about the first entry. This wil
```

```
l be HTML encoded on return.",
                "link":"http:\/\/www.samplefeed.com\/feed-
entry-1.htm",
                "publication_date":"Tue, 10 Aug 2014 13:38:55 -0800",
                "source":"Reuters",
                "author":"John Doe"
            },
            {
                "title":"Second entry title",
                "description":"A blurb about the Second entry. This wi
ll be HTML encoded on return.",
                "link":"http:\/\/www.samplefeed.com\/feed-
entry-2.htm",
                "publication_date":"Tue, 10 Aug 2014 13:38:55 -0800",
                "source":"BBC",
                "author":""
            }
        ]
    }
}
```


Change Log

Version	Change
v10	Added /<rssfeed> GET endpoint.

/search GET

Overview

List records in a module. Searching, filtering and ordering can be applied to only fetch the records you are interested in. Additionally the set of returned fields can be restricted to speed up processing and reduce download times.

Request Arguments

Name	Type	Description	Required
q	String	The search text to match records on. This will search through any fields	False

Name	Type	Description	Required
		on the module that has unified_search set to true.	
max_num	Integer	A maximum number of records to return.	False
offset	Integer	The number of records to skip over before records are returned.	False
fields	String	Comma delimited list of fields to return. The field date_modified will always be returned. Example: name,account_type,description	False
order_by	String	How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon. Example: name:DESC,account_type:DESC,date_modified:ASC	False
favorites	Boolean	Only fetch the current users favorited records.	False
my_items	Boolean	Only fetch items assigned to the current user.	False

Response Arguments

Name	Type	Description
next_offset	Integer	Displays the next offset for retrieval of additional

Name	Type	Description
		results. -1 will be returned when there are no more records.
records	Array	An array of results containing matched records.

Response


```
{
   "next_offset":2,
   "records":[
      {
         "id":"ecbf2a6c-261e-5fca-fbb6-512d093554b8",
         "name":"Avery Software Co",
         "date_modified":"2013-02-26T19:12:56+00:00",
         "description":"",
         "my_favorite":false,
         "_acl":{
            "fields":{

            }
         },
         "_module":"Accounts",
         "_search":{
            "score":1
         }
      },
      {
         "id":"af5f8dae-7169-b497-1d77-512d0937ed81",
         "name":"Avery Software Co",
         "date_modified":"2013-02-26T19:12:56+00:00",
         "description":"",
         "my_favorite":false,
         "_acl":{
            "fields":{

            }
         },
         "_module":"Accounts",
         "_search":{
            "score":1
         }
      }
```

```
   ]
}
```

Change Log

Version	Change
v10	Added /<module> GET endpoint.

/theme GET

Overview

Fetches the customizable variables of a theme.

Request Arguments

Name	Type	Description	Required
platform	String	The theme platform - /themes/clients/***PLATFORM***/themeName/. Accepted values are 'base' and 'portal'.	False
themeName	String	The theme name - /themes/clients/platform/***THEME NAME***/.	False

Request

```
http://{site_url}/rest/v10/theme?platform=base&themeName=default
```

Note: GET endpoint parameters are passed in the form of a query string.

Response Arguments

Name	Type	Description
mixins	Array	An array of name value

Name	Type	Description
		pairs detailing mixin colors
hex	Array	A array of name value pairs detailing css colors
rgba	Array	An array of name value pairs detailing colors
rel	Array	An array of name value pairs detailing relationships
bg	Array	An array of name value pairs detailing backgroup colors

Response

```
{
   "colors":[
      {
         "name":"BorderColor",
         "value":"#E61718"
      },
      {
         "name":"NavigationBar",
         "value":"#000000"
      },
      {
         "name":"PrimaryButton",
         "value":"#177EE5"
      }
   ]
}
```

Change Log

Version	Change
v10	Added /<theme> GET endpoint.

/theme POST

Overview

Updates the variables.less (less file containing customizable vars in the theme folder) with the set of variables passed as arguments.

Request Arguments

Name	Type	Description	Required
platform	String	The theme platform - /themes/clients/***PLATFORM***/themeName/. Accepted values are 'base' and 'portal'.	True
themeName	String	The theme name - /themes/clients/platform/***THEMENAME***/.	True
<theme variable>	String	The variables of the theme	False

Request

```
{
    platform:'portal',
    themeName:'default',
    primary:'default',
    secondary:'#aaaaaa',
    primaryBtn:'#bbbbbb',
}
```

Response Arguments

Name	Type	Description
<css path>	String	Returns the path of the new bootstrap.css file.

Response

```
"http:\/\/sugarcrm\/cache\/themes\/clients\/base\/default\/6a031485bf5
```

```
239b9462e4aaba72a4646.css"
```

Change Log

Version	Change
v10	Added /<theme> POST endpoint.

Extending Endpoints

Overview

How to add your own custom endpoints to the REST API.

Custom Endpoints

With the REST architecture, you can easily define your own endpoint. All custom global endpoints will be located in ./custom/clients/<client>/api/. All custom endpoints specific to a module will be located in ./custom/modules/<module>/clients/<client>/api/.

Note: The Sugar application client type is "base". More information on the various client types can be found in the Clients section.

Defining New Endpoints

The endpoint class will be created in ./custom/clients/<client>/api/ and will extend SugarApi. Within this class, you will need to implement a registerApiRest() function that will define your endpoint paths. When creating a new endpoint, please note that the file name must match the class name for the endpoint definitions. The example below demonstrates creating a GET endpoint.

./custom/clients/base/api/MyEndpointsApi.php

```
  <?php

if(!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry Point
');

class MyEndpointsApi extends SugarApi
{
    public function registerApiRest()
```

```
    {
        return array(
            //GET
            'MyGetEndpoint' => array(
                //request type
                'reqType' => 'GET',

                //set authentication
                'noLoginRequired' => false,

                //endpoint path
                'path' => array('MyEndpoint', 'GetExample', '?'),

                //endpoint variables
                'pathVars' => array('', '', 'data'),

                //method to call
                'method' => 'MyGetMethod',

                //short help string to be displayed in the help docume
ntation
                'shortHelp' => 'An example of a GET endpoint',

                //long help to be displayed in the help documentation
                'longHelp' => 'custom/clients/base/api/help/MyEndPoint
_MyGetEndPoint_help.html',
            ),
        );
    }

    /**
     * Method to be used for my MyEndpoint/GetExample endpoint
     */
    public function MyGetMethod($api, $args)
    {
        //custom logic
        return $args;
    }

}

?>
```

Note: The file name must match the class name for the endpoint definitions.

Given the example above, a class name of MyEndpointsApi must be created as MyEndpointsApi.php.

registerApiRest() Method

Your extended class will contain two methods. The first method is registerApiRest() which will return an array that defines your REST endpoints. Each endpoint will need the following properties:

Name	Type	Description
reqType	Array | String	An array of one or many request types of the endpoint. Possible values are: GET, PUT, POST and DELETE.
noLoginRequired	Boolean	Determines whether the endpoint is authenticated. Setting this value to true will remove the authentication requirement.
path	Array	The endpoint path. An endpoint path of: `'path' => array('MyEndpoint', 'GetExample'),` Will equate to the URL path of http://{site url}/rest/{REST VERSION}/MyEndpoint/GetExample. To pass in a variable string to your endpoint, you will add a path of '?'. This will allow you to pass URL data to your selected method given that it is specified in the pathVars.
pathVars	Array	The path variables. For each path on your

		endpoint, you can opt to pass its value in as a parameter to your selected method. An empty path variable will ignore the path. Example: `'path' => array('MyEndpoint', 'GetExample', '?'),'pathVars' => array('', '', 'data'),` The above will pass information to your selected method as the $args parameter when calling http://{site url}/rest/{REST VERSION}/MyEndpoint/GetExample/MyData
method	String	The method to pass the pathVars to. This can be named anything you choose and will be located in your SugarApi extended class.
shortHelp	String	A short help string for developers when looking at the help documentation.
longHelp	String	Path to a long help file. This file will be loaded when a user clicks an endpoint from the help documentation.
minVersion	Integer	The minimum API Version this endpoint can be used with. See Scoring Section below.
maxVersion	Integer	The maximum API Version this endpoint can be used

		with. See Scoring Section below.
extraScore	Integer	Add an extra score value to the Scoring of this endpoint, to place priority on its usage. See Scoring Section below.

Endpoint Method

The second method can be named anything you choose but it is important to note that it will need to match the name of the method you specified for your endpoint. This method will require the parameters $api and $args. The MyGetMethod() in the example above will be used when the endpoint MyEndpoint/GetExample is called. Any path variables, query string parameters or posted data will be available in the $args parameter for you to work with.

```
  public function MyEndpointMethod($api, $args)
{
    //logic
    return $returnData;
}
```

Help Documentation

The final step once you have your endpoint working is to document it. This file can reside anywhere in the system and will be referenced in your endpoints longHelp property. An example of the help documentation can be found below:

custom/clients/base/api/help/MyEndPoint_MyGetEndPoint_help.html

```
  <h2>Overview</h2>
<span class="lead">
    A custom GET endpoint. Returns the $args parameter that is passed
to the endpoint method.
</span>

<h2>Path Variables</h2>
<table class="table table-hover">
    <thead>
    <tr>
        <th>Name</th>
        <th>Description</th>
    </tr>
```

```
    </thead>
    <tbody>
    <tr>
        <td>
            :data
        </td>
        <td>
            Data string to pass to the endpoint.
        </td>
    </tr>
    </tbody>
</table>

<h2>Input Parameters</h2>
<span class="lead">
    This endpoint does not accept any input parameters.
</span>

<h2>Result</h2>
<table class="table table-hover">
    <thead>
    <tr>
        <th>Name</th>
        <th>Type</th>
        <th>Description</th>
    </tr>
    </thead>
    <tbody>
    <tr>
        <td>
            __sugar_url
        </td>
        <td>
            String
        </td>
        <td>
            The endpoint path.
        </td>
    </tr>
    <tr>
        <td>
            data
        </td>
        <td>
            String
        </td>
```

```
        <td>
            The data from path variable.
        </td>
    </tr>
    </tbody>
</table>

<h3>Output Example</h3>
<pre class="pre-scrollable">
{
    "__sugar_url":"v10\/MyEndpoint\/GetExample\/MyData",
    "data":"MyData"
}
</pre>

<h2>Change Log</h2>
<table class="table table-hover">
    <thead>
    <tr>
        <th>Version</th>
        <th>Change</th>
    </tr>
    </thead>
    <tbody>
    <tr>
        <td>
            v10
        </td>
        <td>
            Added <code>/MyEndpoint/GetExample/:data</code> GET endpoi
nt.
        </td>
    </tr>
    </tbody>
</table>
```

Quick Repair and Rebuild

Once all of the files are in place, you will need to navigate to Admin > Repair > Quick Repair and Rebuild. This will rebuild the ./cache/file_map.php and ./cache/include/api/ServiceDictionary.rest.php files to make your endpoint available.

Example

./custom/clients/base/api/MyEndpointsApi.php

```
  <?php

if(!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry Point
');

class MyEndpointsApi extends SugarApi
{
    public function registerApiRest()
    {
        return array(
            //GET & POST
            'MyGetEndpoint' => array(
                //request type
                'reqType' => array('GET','POST'),

                //set authentication
                'noLoginRequired' => false,

                //endpoint path
                'path' => array('MyEndpoint', 'GetExample', '?'),

                //endpoint variables
                'pathVars' => array('', '', 'data'),

                //method to call
                'method' => 'MyGetMethod',

                //short help string to be displayed in the help docume
ntation
                'shortHelp' => 'An example of a GET endpoint',

                //long help to be displayed in the help documentation
                'longHelp' => 'custom/clients/base/api/help/MyEndPoint
_MyGetEndPoint_help.html',
            ),
        );
    }

    /**
     * Method to be used for my MyEndpoint/GetExample endpoint
     */
    public function MyGetMethod($api, $args)
    {
```

```
        //custom logic
        return $args;
    }

}

?>
```

Redefining Existing Endpoints

With endpoints, you may have a need to extend an existing endpoint to meet your needs. When doing this it is important that you do not remove anything from the return array of the endpoint. Doing so could result in unexpected behavior due to other functionality using the same endpoint.

For this example, we will extend the ping endpoint to return "Pong <timestamp>". To do this, we will need to extend the existing PingApi class and define our method overrides. When creating a new endpoint, please note that the file name must match the class name for the endpoint definitions. For our example, we will prefix the original class name with "custom" to create our overriding endpoint.

./custom/clients/base/api/CustomPingApi.php

```
  <?php

if(!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry Point
');

require_once("clients/base/api/PingApi.php");

class CustomPingApi extends PingApi
{
    public function registerApiRest()
    {
        //in case we want to add additional endpoints
        return parent::registerApiRest();
    }

    //override to modify the ping function of the ping endpoint
    public function ping($api, $args)
    {
        $result = parent::ping($api, $args);

        //append the current timestamp
```

```
        return $result . ' ' . time();
    }
}
```

Note: The file name must match the class name for the endpoint definitions. Given the example above, a class name of CustomPingApi must be created as CustomPingApi.php.

As you can see, we extended registerApiRest to fetch existing endpoint definitions in case we want to add our own. We then added an override for the ping method to return our new value. Once the file is in place you will need to navigate to Admin > Repair > Quick Repair and Rebuild. Once completed, any call made to <url>/rest/{REST VERSION}/ping will result in a response of "ping <timestamp>".

Endpoint Scoring

When generating custom endpoints or overriding existing endpoints, the system can determine which endpoint to use based on the score placed on the registered endpoints. The scoring calculation works in the following manner for registered endpoints.

Route Path	**Score**
?	0.75
<module>	1.0
Exact Match	1.75
Custom	0.5
Endpoint 'extraScore' property	Defined Extra Score

The defined Path and extraScore properties in the registerApiRest() method, help determine the score for any given Endpoint. The higher the score allows the API to determine which Endpoint is being used for a given request. For example, if you had extended the /Accounts/:record GET API Endpoint as follows:

/custom/modules/Accounts/clients/base/api/CustomAccountsApi.php

```
  <?php

if(!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry Point
```

```
');

require_once("clients/base/api/ModuleApi.php");

class CustomAccountsApi extends ModuleApi
{
    public function registerApiRest()
    {
        return array(
            //GET & POST
            'getAccount' => array(
                //request type
                'reqType' => array('GET'),

                //set authentication
                'noLoginRequired' => false,

                //endpoint path
                'path' => array('Accounts', '?'),

                //endpoint variables
                'pathVars' => array('module', 'record'),

                //method to call
                'method' => 'retrieveRecord',

                //short help string to be displayed in the help docume
ntation
                'shortHelp' => 'An example of a GET endpoint',

                //long help to be displayed in the help documentation
                'longHelp' => 'custom/clients/base/api/help/MyEndPoint
_MyGetEndPoint_help.html',
            ),
        );
    }

    /**
     * Method to be used for my Accounts/:record endpoint
     */
    public function retrieveRecord($api, $args)
    {
        //custom logic
        return parent::retrieveRecord($api,$args);
    }
```

```
}

?>
```

This Endpoint will be used when accessing /Accounts/<record_id> since the Exact Match of Accounts in the path property gives a greater score than the <module> match that comes with the standard /<module>/:record Endpoint defined in ModuleApi.

Endpoint Versioning

Part of the scoring calculation allows for versioning of the API Endpoints to accommodate the same endpoint path in the Rest API, for other versions of the API. If you have a custom Endpoint at /rest/v10/CustomEndpoint, and would like to alter the logic slightly for another use without deprecating or removing the old Endpoint, you can use the versioning techniques to have the new logic reside at /rest/v11/CustomEndpoint. The following table describes how the Versioning properties on the Endpoint definition affect the score of the Endpoint to allow the API to determine which Endpoint should be used for a given request.

Property	**Score**
minVersion	0.02 + minVersion/1000
maxVersion	0.02
minVersion === maxVersion (i.e. Both minVersion and maxVersion properties are defined on Endpoint to the same value)	0.06 + minVersion/1000

To allow for additional Versions in the API, you will first need to customize the 'maxVersion' property in the $apiSettings array. To do this, create ./custom/include/api/metadata.php. Inside the file you can set the $apiSettings['maxVersion'] property as follows:

```
  <?php

$apiSettings['maxVersion'] = 11;
```

Once you have configured the Max Version property, you can then set the minVersion and maxVersion properties on the Endpoint definition in the custom registerApiRest() method. For example, the following Endpoint definitions:

```
  <?php
...
    return array(
        'foo1' => array(
            'reqType' => 'GET',
            'path' => array('Accounts','config',
            'pathVars' => array('module', 'record'),
            'method' => 'foo1',
        ),
        'foo2' => array(
            'reqType' => 'GET',
            'minVersion' => 11,
            'path' => array('Accounts','config',
            'pathVars' => array('module', 'record'),
            'method' => 'foo2',
        ),
        'foo3' => array(
            'reqType' => 'GET',
            'minVersion' => 11,
            'maxVersion' => 11,
            'path' => array('Accounts','config',
            'pathVars' => array('module', 'record'),
            'method' => 'foo3',
        ),
        'foo4' => array(
            'reqType' => 'GET',
            'minVersion' => 11,
            'maxVersion' => 12,
            'path' => array('Accounts','config',
            'pathVars' => array('module', 'record'),
            'method' => 'foo3',
        ),
        'foo5' => array(
            'reqType' => 'GET',
            'minVersion' => 14,
            'path' => array('Accounts','config',
            'pathVars' => array('module', 'record'),
            'method' => 'foo5',
        ),
    );
...
```

With the above definitions, the following table provides the Expected Request for each Endpoint.

Request URI	Best Route
/rest/v10/Accounts/config	foo1
/rest/v11/Accounts/config	foo3
/rest/v12/Accounts/config	foo4
/rest/v13/Accounts/config	foo2
/rest/v14/Accounts/config	foo5

API Exceptions

Overview

Sugar comes with some predefined API Exceptions, located in ./include/api/, that can be called from API endpoints. These exceptions return a specific HTTP code and a message.

Stock Exceptions

Exception Class	HTTP Code	Error Label	Language Label Key	Language Label Value
SugarApiExceptionError	500	fatal_error	EXCEPTION_FATAL_ERROR	Your request failed to complete. A fatal error occurred. Check logs for more details.
SugarApiExceptionIncorrectVersion	301	incorrect_version	EXCEPTION_INCORRECT_VERSION	The version of the API you are using is not correct for the current request.
SugarApiExceptionNeedLogin	401	need_login	EXCEPTION_NEED_LOGIN	You need to be logged in to perform this action.
SugarApiExceptionInvalidGran	401	invalid_grant	EXCEPTION_INVALID_TOKE	Your authentication

t			N	token is invalid.
SugarApiExceptionNotAuthorized	403	not_authorized	EXCEPTION_NOT_AUTHORIZED	You are not authorized to perform this action. Contact your administrator if you need access.
SugarApiExceptionPortalUserInactive	403	inactive_portal_user	EXCEPTION_INACTIVE_PORTAL_USER	You cannot access Portal because your portal account is inactive. Please contact customer support if you need access.
SugarApiExceptionPortalNotConfigured	403	portal_not_configured	EXCEPTION_PORTAL_NOT_CONFIGURED	Portal is not configured properly. Contact your Portal Administrator for assistance.
SugarApiExceptionNoMethod	404	no_method	EXCEPTION_NO_METHOD	Your request was not supported. Could not find the HTTP method of your request for this path.
SugarApiExceptionNotFound	404	not_found	EXCEPTION_NOT_FOUND	Your requested resource was not found. Could not find a handler for the path specified in the request.
SugarApiExceptionEditConflict	409	edit_conflict	EXCEPTION_EDIT_CONFLICT	Edit conflict, please reload the record

				data.
SugarApiExceptionInvalidHash	412	metadata_out_of_date	EXCEPTION_METADATA_OUT_OF_DATE	Your metadata or user hash did not match the server. Please resync your metadata.
SugarApiExceptionRequestTooLarge	413	request_too_large	EXCEPTION_REQUEST_TOO_LARGE	Your request is too large to process.
SugarApiExceptionMissingParameter	422	missing_parameter	EXCEPTION_MISSING_PARAMTER	A required parameter in your request was missing.
SugarApiExceptionInvalidParameter	422	invalid_parameter	EXCEPTION_INVALID_PARAMETER	A parameter in your request was invalid.
SugarApiExceptionRequestMethodFailure	424	request_failure	EXCEPTION_REQUEST_FAILURE	Your request failed to complete.
SugarApiExceptionClientOutdated	433	client_outdated	EXCEPTION_CLIENT_OUTDATED	Your software is out of date, please update your client before attempting to connect again.
SugarApiExceptionConnectorResponse	502	bad_gateway	EXCEPTION_CONNECTOR_RESPONSE	A connector or an integration request resulted in a failed response.
SugarApiExceptionMaintenance	503	maintenance	EXCEPTION_MAINTENANCE	SugarCRM is in maintenance mode. Only admins can login. Please contact your administrator for details.
SugarApiExcep	503	service_unavail	EXCEPTION_S	The server

tionServiceUnavailable		able	ERVICE_UNAVAILABLE	cannot process your request because it is busy or unavailable at this time.
SugarApiExceptionSearchUnavailable	400	search_unavailable	EXCEPTION_SEARCH_UNAVAILABLE	Search engine is temporarily unavailable.
SugarApiExceptionSearchRuntime	400	search_runtime	EXCEPTION_SEARCH_RUNTIME	A search engine runtime error occurred. Please contact your System Administrator.

Using Exceptions

When extending endpoints in Sugar, the stock API exceptions can be used to return errors and messages. The exception classes accept a single parameter on creation for the message value to return. This parameter can accept language label keys or plain text. If no parameter is passed exception, the error will default to the exception's default language label key. To call a stock API exception from custom code add the following snippet :

```
  //throwing an api exception using the stock message
throw new SugarApiExceptionError();

//throwing an api exception using a custom language label key
throw new SugarApiExceptionError('EXCEPTION_CSTM_LABEL_KEY');

//throwing an api exception with text
throw new SugarApiExceptionError('There has been an error.');
```

This will return an http response code of 500 with the following response data:

```
  {
    "error": "fatal_error",
    "error_message": "There has been an error."
}
```

Custom Exceptions

Sugar gives you the ability to create custom API exceptions by creating a new class in ./custom/include/api/ that extends the stock SugarApiException class. When extending the class you can provide the following properties in your custom code:

$httpCode	The http response code to send back in the header. Defaults to 400 if not specified.
$errorLabel	The string value returned in the 'error' key response.
$messageLabel	The label to return as a default response if a message is not provided.

Example

The following example will demonstrate how to create a custom exception.

./custom/include/api/<name>.php

```
  <?php

require_once 'include/api/SugarApiException.php';

/**
 * Custom error.
 */
class cstmSugarApiExceptionError extends SugarApiException
{
    public $httpCode = 404;
    public $errorLabel = 'my_error';
    public $messageLabel = 'EXCEPTION_CSTM_LABEL_KEY';
}
```

Create a custom language label using the extension framework:

./custom/Extension/application/Ext/Language/<name>.php

```
  <?php

$app_strings['EXCEPTION_CSTM_LABEL_KEY'] = 'My Exception Message.';
```

You can call your new exception by using the following code :

```
  require_once 'custom/include/api/<name>.php';

//throwing custom api exception using the default message
throw new cstmSugarApiExceptionError();

//throwing custom api exception using a custom language label key
throw new cstmSugarApiExceptionError('EXCEPTION_CSTM_LABEL_KEY');

//throwing custom api exception with text
throw new cstmSugarApiExceptionError('There has been an error.');
```

You will then need to navigate to Admin > Repair > Quick Repair and Rebuild before using your exception.

The exception error will return a response of:

```
  {
    "error": "my_error",
    "error_message": "There has been an error."
}
```

Legacy API

Overview

v1 - v4.1 API documentation.

SOAP VS REST

There are significant differences between how the legacy REST and SOAP protocols function on an implementation level (e.g. Performance, response size, etc). Deciding which protocol to use is up to the individual developer and is beyond the scope of this guide. Starting in SugarCRM version 6.2.0, there are some deviations between the protocols with the v4 API. There are additional core calls that are only made available through the REST protocol. They are listed below:

- get_module_layout
- get_module_layout_md5
- get_quotes_pdf method
- get_report_pdf method

- snip_import_emails
- snip_update_contacts
- job_queue_cycle
- job_queue_next
- job_queue_run
- oauth_access method
- oauth_access_token
- oauth_request_token

REST

REST stands for 'Representational State Transfer'. This protocol is used by Sugar to exchange information both internally and externally.

How do I access the REST service?

The legacy REST services in SugarCRM and be found by navigating to:

```
http://{site url)/service/{version}/rest.php
```

Where 'site url' is the URL of your Sugar instance and 'version' is the latest version of the API specific to your release of Sugar. You can find out more about versioning in the Web Services documentation.

Input / Output Datatypes

The default input / output datatype for REST is JSON / PHP serialize.

These datatype files, SugarRestJSON.php and SugarRestSerialize.php, are in:

```
./service/core/REST/
```

Defining your own Datatypes

You can also define you own datatype. To do this, you need to create a new file such as:

```
./service/core/REST/SugarRest<CustomDataType>.php
```

Next, you will need to override generateResponse() and serve() functions. The

Serve function decodes or unserializes the appropriate datatype based on the input type; the generateResponse function encodes or serializes it based on the output type.

See service/test.html for more examples on usage. In this file, the getRequestData function, which generates URL with json, is both the input_type and the response_type. That is, the request data from the JavaScript to the server is JSON and response data from server is also JSON. You can mix and match any datatype as input and output. For example, you can have JSON as the input_type and serialize as the response_type based on your application's requirements.

REST Failure Response

If a call failure should occur, the result will be as shown below:

Name	Type	Description
name	String	Error message.
number	Integer	Error number.
description	String	Description of error.

SOAP

SOAP stands for 'Simple Object Access Protocol'. SOAP is a simple XML-based protocol that is used to allow applications to exchange information.

How do I access the SOAP service?

The legacy SOAP service in SugarCRM and be found by navigating to:

```
http://{sugar_url)/service/{version}/soap.php
```

Where 'sugar_url' is the url of your Sugar instance and 'version' is the latest version of the API specific to your release of Sugar. You can find out more about versioning in the section titled 'API: Versioning'. The default WSDL is formatted as rpc/encoded.

WS-I 1.0 Compliancy

Sugar supports generating a URL that is WS-I compliant. When accessing the soap entry point, you can access the WSDL at:

```
http://{sugar_url)/service/{version}/soap.php?wsdl
```

By default, the WSDL is formatted as rpc/encoded, however, this can be changed by specifying a 'style' and 'use' url-paramater. An example of this is:

```
  http://{sugar_url)/service/{version}/soap.php?wsdl&style=rpc&use=lit
eral
```

URL Parameters

style

- rpc

use

- encoded
- literal

Validation

This WSDL (rpc/literal) was successfully verified against Apache CXF 2.2.

SOAP Failure Response

If a call failure should occur, the result will be as shown below:

Name	Type	Description
faultcode	Integer	Fault ID.
faultactor	String	Provides information about what caused the fault to happen.
faultstring	String	Fault Message.
detail	String	Description of fault.

What is NuSOAP?

Overview

NuSOAP is a SOAP Toolkit for PHP that doesn't require PHP extensions.

Where Can I Get It?

NuSOAP can be downloaded from http://nusoap.sourceforge.net

How Do I Use It?

After you have downloaded NuSOAP, you will need to extract the zip file contents to a storage directory. Once extracted, you will reference "lib/nusoap.php" in your PHP SOAP application.

Example

```
<?php

//require NuSOAP
require_once("./lib/nusoap.php");

//retrieve WSDL
$client = new nusoap_client("http://{site_url}/service/v4/soap.php?wsdl", 'wsdl');
```

Methods

Web Service Method Calls.

get_available_modules

Overview

Retrieves a list of available modules in the system.

Available APIs

- SOAP
- REST

Definition

get_available_modules(session, filter)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
filter	String	String to filter the modules with. Possible values are 'default', 'mobile', 'all'.

Result

Name	Type	Description
result	new_module_fields | Array	The call result.
result.modules	Array	The list of avaialble modules.
result.modules[].module_key	String	The modules key.
result.modules[].module_label	String	The display label for the module.
result.modules[].favorite_enabled	String	Whether favorites are enabled for the module.
result.modules[].acls	Array	The ACL list for the module

Change Log

Version	Change
v3	Added ***filter*** parameter. Accepts the values 'default', 'mobile', 'all'.

Examples

PHP

```
$get_available_modules_parameters = array(
    //Session id
    "session" => $session_id,
```

```
    //Module filter. Possible values are 'default', 'mobile', 'all'.
    "filter" => 'all',
);
```

get_document_revision

Overview

Retrieves a specific document revision.

Available APIs

- SOAP
- REST

Definition

get_document_revision(session, i)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
i	String	The ID of the document revision.

Result

Name	Type	Description
result	new_return_document_revision| Array	The call result.
result.document_revision	Array	The details of the document revision.
result.document_revision.id	String	The document ID.
result.document_revision.document_name	String	The document name.
result.document_revision.revision	String	The document revision number

Name	Type	Description
result.document_revision.filename	String	The filename of the file.
result.document_revision.file	String	The binary contents of the file.

Change Log

Version	Change
v2	Return type was changed from return_document_revision to ***new_return_document_revision***.

Examples

PHP

```
$get_document_revision_parameters = array(
    //Session id
    "session" => $session_id,

    //The attachment details
    "i" => '723b7dcb-27b3-e53d-b348-50bd283f8e48',
);
```

---

get_entries

Overview

Retrieves a list of beans based on specified record IDs.

Available APIs

- SOAP
- REST

Definition

get_entries(session, module_name, ids, select_fields, link_name_to_fields_array, track_view)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
module_name	String	The name of the module from which to retrieve records. Note: This is the modules key which may not be the same as the modules display name.
ids	String	The list of record IDs to retrieve.
select_fields	select_fields | Array	The list of fields to be returned in the results. Specifying an empty array will return all fields.
link_name_to_fields_array	link_names_to_fields_array | Array	A list of link names and the fields to be returned for each link.
track_view	Boolean	Flag the record as a recently viewed item.

Result

Name	Type	Description
result	get_entry_result_version2 | Array	The call result.
result.entry_list	Array	The record's name-value pair for the simple datatypes excluding the link field data. If you do not have access to the object, entry_list[].name_value_list will notify you.
result.relationship_list	Array	The records link field data.

Change Log

Version	Change

v3_1	Added ***track_view*** parameter.
v2	Added ***link_name_to_fields_array*** parameter.
v2	Return type was changed from get_entry_result to ***get_entry_result_version2.***

Examples

PHP

```
$get_entries_parameters = array(
    //session id
    'session' => $session_id,

    //The name of the module from which to retrieve records
    'module_name' => 'Accounts',

    //An array of record IDs
    'ids' => array(
        '14b0c0ca-3ea2-0ee8-f3be-50aa57c11ee7',
    ),

    //The list of fields to be returned in the results
    'select_fields' => array(
        'name',
        'billing_address_state',
        'billing_address_country'
    ),

    //A list of link names and the fields to be returned for each link
 name
    'link_name_to_fields_array' => array(
        array(
            'name' => 'email_addresses',
            'value' => array(
                'email_address',
                'opt_out',
                'primary_address'
            ),
        ),
    ),

    //Flag the record as a recently viewed item
    'track_view' => true,
);
```

get_entries_count

Overview

Retrieves a list of beans based on query specifications.

Available APIs

- SOAP
- REST

Definition

get_entries_count(session, module_name, query, deleted)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
module_name	String	The name of the module from which to retrieve records. Note: This is the modules key which may not be the same as the modules display name.
query	String	The SQL WHERE clause without the word "where".
deleted	Integer	If deleted records should be included in the results.

Result

Name	Type	Description
result	get_entries_count_result | Array	The call result.
result.result_count	Integer	The count of total records.

Change Log

Version	Change

Examples

PHP

```
$get_entries_count_parameters = array(
    //Session id
    'session' => $session_id,

    //The name of the module from which to retrieve records
    'module_name' => 'Accounts',

    //The SQL WHERE clause without the word "where".
    'query' => " accounts.name like '%example text%' ",

    //If deleted records should be included in results.
    'deleted' => false
);
```

---

get_entry

Overview

Retrieves a single bean based on record ID.

Available APIs

- SOAP
- REST

Definition

get_entry(session, module_name, id, select_fields, link_name_to_fields_array, track_view)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
module_name	String	The name of the module from which to retrieve

Name	Type	Description
		records. Note: This is the modules key which may not be the same as the modules display name.
id	String	The ID of the record to retrieve.
select_fields	select_fields | Array	The list of fields to be returned in the results. Specifying an empty array will return all fields.
link_name_to_fields_array	link_names_to_fields_array | Array	A list of link names and the fields to be returned for each link.
track_view	Boolean	Flag the record as a recently viewed item.

Result

Name	Type	Description
result	get_entry_result_version2 | Array	The call result.
result.entry_list	Array	The record's name-value pair for the simple datatypes excluding the link field data. If you do not have access to the object, entry_list[].name_value_list will notify you.
result.relationship_list	Array	The records link field data.

Change Log

Version	Change
v3_1	Added ***track_view*** parameter.
v2	Added ***link_name_to_fields_array*** parameter.
v2	Return type was changed from get_entry_result to ***get_entry_result_version2***.

Examples

PHP

```
$get_entry_parameters = array(
    //session id
    'session' => $session_id,

    //The name of the module from which to retrieve records
    'module_name' => "Contacts",

    //The ID of the record to retrieve.
    'id' => "18df70e4-1422-8bff-6f5f-50aa571fe4e5",

    //The list of fields to be returned in the results
    'select_fields' => array(
        'id',
        'first_name',
        'last_name',
    ),

    //A list of link names and the fields to be returned for each link
 name
    'link_name_to_fields_array' => array(
        array(
            'name' => 'email_addresses',
            'value' => array(
                'email_address',
                'opt_out',
                'primary_address'
            ),
        ),
    ),

    //Flag the record as a recently viewed item
    'track_view' => true,
);
```

get_entry_list

Overview

Retrieves a list of beans based on query specifications.

Available APIs

- SOAP
- REST

Definition

get_entry_list(session, module_name, query, order_by, offset, select_fields, link_name_to_fields_array, max_results, deleted, favorites)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
module_name	String	The name of the module from which to retrieve records. Note: This is the modules key which may not be the same as the modules display name.
query	String	The SQL WHERE clause without the word "where". You should remember to specify the table name for the fields to avoid any ambiguous column errors.
order_by	String	The SQL ORDER BY clause without the phrase "order by".
offset	Integer	The record offset from which to start.
select_fields	select_fields | Array	The list of fields to be returned in the results. Specifying an empty array will return all fields.
link_name_to_fields_array	link_names_to_fields_array | Array	A list of link names and the fields to be returned for each link.
max_results	Integer	The maximum number of results to return.
deleted	Integer	If deleted records should

Name	Type	Description
		be included in the results.
favorites	Boolean	If only records marked as favorites should be returned.

Result

Name	Type	Description
result	get_entry_result_version2 | Array	The call result.
result.result_count	Integer	The total number of records returned in the call.
result.total_count	Integer	The total number of records.
resultnext_offset	Integer	The next offset to retrieve records.
result.entry_list	Array	The record's name-value pair for the simple datatypes excluding the link field data. If you do not have access to the object, entry_list[].name_value_list will notify you.
result.relationship_list	Array	The records link field data.

Change Log

Version	Change
v3_1	Added ***favorites*** parameter.
v2	Added ***link_name_to_fields_array*** parameter.
v2	Return type was changed from get_entry_list_result to ***get_entry_list_result_version2***.

Examples

PHP

```
$get_entry_list_parameters = array(
    //session id
    'session' => $session_id,

    //The name of the module from which to retrieve records
    'module_name' => 'Leads',

    //The SQL WHERE clause without the word "where".
    'query' => "",

    //The SQL ORDER BY clause without the phrase "order by".
    'order_by' => "",

    //The record offset from which to start.
    'offset' => 0,

    //A list of fields to include in the results.
    'select_fields' => array(
        'id',
        'name',
        'title',
    ),

    //A list of link names and the fields to be returned for each link
 name.
    'link_name_to_fields_array' => array(
        array(
            'name' => 'email_addresses',
            'value' => array(
                'email_address',
                'opt_out',
                'primary_address'
            ),
        ),
    ),

    //The maximum number of results to return.
    'max_results' => 2,

    //If deleted records should be included in results.
    'deleted' => 0,

    //If only records marked as favorites should be returned.
    'favorites' => false,
);
```

get_language_definition

Overview

Retrieves the language label strings for the specified modules.

Available APIs

- REST

Definition

get_available_modules(session, modules, md5)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
modules	Array	The list of modules to retrieve language definitions for.
md5	Boolean	Setting this to true will return an MD5 hash of the modules labels strings instead of the label strings themselves.

Result

Name	Type	Description
result	Array	The call result. Contains a list of modules.
result[]	Array or String	The list of language definitions or MD5 hashes. This is dependent on the 'md5' parameter.

Change Log

Version	Change

v3_1	Added ***get_language_definition*** method.

Examples

PHP

```
$get_language_definition_parameters = array(
    //Session id
    'session' => $session_id,

    //The list of modules
    'modules' => array(
        'Accounts'
    ),

    //Whether to return the results as an MD5 hash
    'md5' => false,
);
```

---

get_last_viewed

Overview

Retrieves a list of recently viewed records by module for the current user.

Available APIs

- SOAP
- REST

Definition

get_last_viewed(session, module_names)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
module_names	module_names | Array	The list of modules to retrieve last viewed records for.

Result

Name	Type	Description
result	last_viewed_list | Array	The call result. Contains a list of reciently viewed records.
result[].id	Integer	The result ID.
result[].item_id	String	The reciently viewed record ID.
result[].item_summary	String	The name of the reciently viewed record.
result[].module_name	String	The name of the module.
result[].monitor_id	String	The monitor ID from the tracker table.
result[].date_modified	String	The date the record was viewed.

Change Log

Version	Change

Examples

PHP

```
$get_last_viewed_parameters = array(
    //Session id
    "session" => $session_id,

    //The name of the modules to retrieve last viewed for
    'module_names' => array(
        'Contacts',
        'Accounts'
    ),
);
```

get_modified_relationships

Overview

Retrieves a list of modified relationships between a specific date range. Helps facilitate sync operations for users.

Available APIs

- SOAP
- REST

Definition

get_modified_relationships(session, module_name, related_module, from_date, to_date, offset, max_results, deleted, module_user_id, select_fields, relationship_name, deletion_date)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
module_name	String	The module key to retrieve relationships against.
related_module	String	The related module key to retrieve records off of. This parameter should always be 'Users'.
from_date	String	Start date in YYYY-MM-DD HH:MM:SS format for the date range.
to_date	String	End date in YYYY-MM-DD HH:MM:SS format for the date range.
offset	Integer	The offset to begin returning records from.
max_results	Integer	The max_results to return.
deleted	Integer	Whether or not to include deleted records. Set to 1 to find deleted records.
module_user_id	String	*This parameter is no longer used and is only

Name	Type	Description
		present for backward compatibility purposes.
select_fields	select_fields	List of fields to select and return as name/value pairs.
relationship_name	String	The name of the relationship name to search on.
deletion_date	String	Date value in YYYY-MM-DD HH:MM:SS format for filtering on deleted records whose date_modified falls within range.

Result

Name	Type	Description
result	modified_relationship_result | Array	The call result.
result.result_count	Integer	The result count.
result.next_offset	Integer	The next offset to retrieve from.
result.entry_list	Array	List of found records.
result.error	Array	Error Message.
result.error.number	Integer	The error number.
result.error.name	String	The name of the error.
result.error.description	String	The description of the error.

Change Log

Version	Change
v4_1	Added ***get_modified_relationships*** method.

Considerations

-

The **module_name** parameter should **always** be 'Users'.

Examples

PHP

```
$get_modified_relationships_parameters = array(
    //Session id
    'session' => $session_id,

    //The module key to retrieve relationships against.
    //This parameter should always be 'Users'.
    'module_name' => 'Users',

    //The related module key to retrieve records off of.
    'related_module' => 'Meetings',

    //Start date in YYYY-MM-DD HH:MM:SS format for the date range.
    'from_date' => '2000-01-01 01:01:01',

    //End date in YYYY-MM-DD HH:MM:SS format for the date range
    'to_date' => '2013-01-01 01:01:01',

    //The offset to begin returning records from.
    'offset' => 0,

    //The max_results to return.
    'max_results' => 5,

    //Whether or not to include deleted records. Set to 1 to find dele
ted records.
    'deleted' => 0,

    //This parameter is not used.
    'module_user_id' => '',

    //List of fields to select and return as name/value pairs.
    'select_fields' => array(),

    //The name of the relationship name to search on.
    'relationhip_name' => 'meetings_users',

    //Date value in YYYY-MM-
DD HH:MM:SS format for filtering on deleted records.
```

```
    'deletion_date' => '2012-01-01 01:01:01'
);
```

get_module_fields

Overview

Retrieves the list of field vardefs for a specific module.

Available APIs

- SOAP
- REST

Definition

get_module_fields(session, module_name, fields)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
module_name	String	The name of the module from which to retrieve records. Note: This is the modules key which may not be the same as the modules display name.
fields	select_fields | Array	The list of fields to retrieve. An empty parameter will return all.

Result

Name	Type	Description
result	new_module_fields | Array	The call result.
result.module_name	String	The name of the module.
result.table_name	String	The name of the modules primary table.

Name	Type	Description
result.module_fields	Array	The vardefs for each individual field.

Change Log

Version	Change
v2	Added ***fields*** parameter.
v2	Return type was changed from module_fields to ***new_module_fields***.

Examples

PHP

```
$get_module_fields_parameters = array(
    //Session id
    "session" => $session_id,

    //The name of the module from which to retrieve fields
    'module_name' => "Contacts",

    //List of specific fields
    'fields' => array(),
);
```

---

get_module_fields_md5

Overview

Retrieves the MD5 hash of the vardefs for the specified modules.

Available APIs

- SOAP
- REST

Definition

get_module_fields_md5(session, module_names)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
module_names	select_fields | Array	The list of modules to retrieve MD5 hashes for.

Result

Name	Type	Description
result	md5_results | Array	The call result. Contains a list of module field hashes. Order is based on the module_names parameter.

Change Log

Version	Change

Examples

PHP

```
$get_module_fields_md5_parameters = array(
    //Session id
    "session" => $session_id,

    //The name of the modules to retrieve field hashes for
    'module_names' => array(
        'Contacts',
        'Accounts'
    ),
);
```

get_module_layout

Overview

Retrieves the layout metadata for a given module given a specific type and view.

Available APIs

- REST

Definition

get_module_layout(session, modules, types, views, acl_check, md5)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
modules	Array	The list of modules to retrieve layouts for.
types	Array	The types of views requested. Current supported types are 'default' (for application) and 'wireless'.
views	Array	The views requested. Current supported types are 'edit', 'detail', 'list', and 'subpanel'.
acl_check	Boolean	Whether or not to check for ACL access.
md5	Boolean	Setting this to true will return an MD5 hash of the modules layouts instead of the layouts themselves.

Result

Name	Type	Description
result	Array	The call result. Contains a list of modules.
result[][$type]	Array	The list of types requested.
result[][$view]	Array or String	The list of layout views requested or MD5 hashes. This is dependent on the

Name	Type	Description
		'md5' parameter.

Change Log

Version	Change
v3_1	Added ***acl_check*** parameter.
v3	Added ***get_module_layout*** method.

Examples

PHP

```
$get_module_layout_parameters = array(
    //Session id
    'session' => $session_id,

    //The list of modules
    'modules' => array(
        'Accounts'
    ),

    //The types of views requested
    'types' => array(
        'default',
    ),

    //The views requested
    'views' => array(
        'edit'
    ),

    //Whether or not to check for ACL access
    'acl_check' => false,

    //Whether to return the results as an MD5 hash
    'md5' => true,
);
```

get_module_layout_md5

Overview

Retrieves the MD5 hash value for a layout given a specific module, type and view.

Available APIs

- REST

Definition

get_module_layout_md5(session, modules, types, views, acl_check)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
modules	Array	The list of modules to retrieve layouts for.
types	Array	The types of views requested. Current supported types are 'default' (for application) and 'wireless'.
views	Array	The views requested. Current supported types are 'edit', 'detail', 'list', and 'subpanel'.
acl_check	Boolean	Whether or not to check for ACL access.

Result

Name	Type	Description
result	Array	The call result. Contains a list of modules.
result[][$type]	Array	The list of types requested.
result[][$view]	String	The list of MD5 layout view hashes requested.

Change Log

Version	Change
v3_1	Added ***acl_check*** parameter.
v3	Added ***get_module_layout_md5*** method.

Examples

PHP

```
  $get_module_layout_md5_parameters = array(
    //Session id
    'session' => $session_id,

    //The list of modules
    'modules' => array(
        'Accounts'
    ),

    //The types of views requested
    'types' => array(
        'default',
    ),

    //The views requested
    'views' => array(
        'edit'
    ),

    //Whether or not to check for ACL access
    'acl_check' => false,
);
```

get_note_attachment

Overview

Retrieves an attachment associated with a specific note record.

Available APIs

- SOAP
- REST

Definition

get_note_attachment(session, id)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
id	String	The ID of the note record to associate the attachment to.

Result

Name	Type	Description
result	new_return_note_attachment | Array	The call result.
result.note_attachment	Array	The details of the file attachment.
result.note_attachment.id	String	The ID of the note record / attachment.
result.note_attachment.filename	String	The filename of the attachment.
result.note_attachment.file	String	The binary contents of the file.
result.note_attachment.related_module_id	String	The related parent ID.
result.note_attachment.related_module_name	String	The related parent module.

Change Log

Version	Change
v2	Return type was changed from return_note_attachment to ***new_return_note_attachment***.

Examples

PHP

```
$get_note_attachment_parameters = array(
    //Session id
    "session" => $session_id,

    //The ID of the note containing the attachment.
    'id' => "9057784d-de17-4f28-c5f9-50bd0f260a43",
);
```

get_quotes_pdf

Overview

Generates a quote PDF for a specific quote.

Available APIs

- REST

Definition

get_quotes_pdf(session, quote_id, pdf_format)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
quote_id	String	The ID of the quote record to generate the PDF for.
pdf_format	String	The pdf type requested. 'Standard' is an example.

Result

Name	Type	Description
result	Array	The call result.
result.file_contents	String	The binary contents of the PDF file.

Change Log

Version	Change
v3_1	Added ***get_quotes_pdf*** method.

Examples

PHP

```
$get_quotes_pdf_parameters = array(
    //Session id
    'session' => $session_id,

    //The quote to generate the pdf for
    'quote_id' => '490cc844-f83a-9b74-9888-50aa575b517c',

    //The pdf type
    'pdf_format' => 'Standard',
);
```

---

get_relationships

Overview

Retrieves a specific relationship link for a specified record.

Available APIs

- SOAP
- REST

Definition

get_relationships(session, module_name, module_id, link_field_name, related_module_query, related_fields, related_module_link_name_to_fields_array, deleted, order_by, offset, limit)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
module_name	String	The name of the module from which to retrieve

Name	Type	Description
		records. Note: This is the modules key which may not be the same as the modules display name.
module_id	String	The ID of the specified module record.
link_field_name	String	The name of the link field for the related module.
related_module_query	String	The list of related record IDs you are relating
related_fields	select_fields | Array	An array specifying relationship fields to populate. An example of this is contact_role between Opportunities and Contacts.
related_module_link_name_to_fields_array	link_names_to_fields_array | Array	For every related record returned, specify link field names to field information.
deleted	Integer	
order_by	String	The SQL ORDER BY clause without the phrase "order by".
offset	Integer	The record offset from which to start.
limit	Integer	The maximum number of results to return.

Result

Name	Type	Description
result	get_entry_result_version2 | Array	The call result.
result.entry_list	Array	The record's name-value pair for the simple datatypes excluding the link field data. If you do not have access to the object,

Name	Type	Description
		entry_list[].name_value_list will notify you.
result.relationship_list	Array	The records link field data.

Change Log

Version	Change
v3	Added ***order_by*** parameter.
v2	Removed ***related_module*** parameter.
v2	Added ***link_field_name*** parameter.
v2	Added ***related_fields*** parameter.
v2	Added ***related_module_link_name_to_fields_array*** parameter.
v2	Return type was changed from get_relationships_result to ***get_entry_result_version2***.

Examples

PHP

```
$get_relationships_parameters = array(
    //session id
    'session' => $session_id,

    //The name of the module from which to retrieve records.
    'module_name' => 'Accounts',

    //The ID of the specified module bean.
    'module_id' => '9f0c0ceb-c512-7103-9456-50aa5787c3f6',

    //The relationship name of the linked field from which to return r
ecords.
    'link_field_name' => 'opportunities',

    //The portion of the WHERE clause from the SQL statement used to f
ind the related items.
    'related_module_query' => " opportunities.name IS NOT NULL ",

    //The related fields to be returned.
```

```
    'related_fields' => array(
        'id',
        'name'
    ),

    //For every related bean returned,
    //specify link field names to field information.
    'related_module_link_name_to_fields_array' => array(
        array(
            'name' => 'contacts',
            'value' => array(
                'id',
                'first_name',
                'last_name',
            ),
        ),
    ),

    //To exclude deleted records
    'deleted'=> 0,

    //order by
    'order_by' => ' opportunities.name ',

    //offset
    'offset' => 0,

    //limit
    'limit' => 200,
);
```

get_report_entries

Overview

Retrieves a list of report entries based on specified record IDs.

Available APIs

- SOAP
- REST

Definition

get_report_entries(session, ids, select_fields)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
ids	select_fields | Array	An array of record IDs to retrieve.
select_fields	select_fields | Array	The list of fields to be included in the results.

Result

Name	Type	Description
result	get_entry_result_for_reports | Array	The call result.
result.field_list	Array	The list of selected fields.
result.field_list[].name	String	The name of the report.
result.field_list[].type	String	The type of the report.
result.field_list[].label	String	The label of the report.
result.field_list[].required	Boolean	
result.field_list[].options	Array	
result.entry_list	Array	The list of report results
result.entry_list[].id	String	The result ID. This is not the record ID.
result.entry_list[].module_name	String	The name of the module. Normally contains the value 'Reports'.
result.entry_list[].name_value_list	Array	The name value list of the report results.

Change Log

Version	Change
v2	Method ***get_report_entries*** was added.

Considerations

- This method is not available in CE.

Examples

PHP

```
$get_report_entries_parameters = array(
    //Session id
    'session' => $session_id,

    //An array of record IDs to retrieve.
    'ids' => array(
        '63f1b905-d206-14cb-cb95-50aa5734815f'
    ),

    //The list of fields to be included in the results.
    'select_fields' => array()
);
```

---

get_report_pdf

Overview

Generates a PDF for a specific report.

Available APIs

- REST

Definition

get_report_pdf(session, report_id)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
report_id	String	The ID of the report record to generate the PDF for.

Result

Name	Type	Description
result	Array	The call result.
result.file_contents	String	The binary contents of the PDF file.

Change Log

Version	Change
v3_1	Added ***get_report_pdf*** method.

Examples

PHP

```
$get_report_pdf_parameters = array(
    //Session id
    'session' => $session_id,

    //The report to generate the pdf for.
    'report_id' => '68ca4de9-7486-72e1-1a56-50aa5757aaab',
);
```

---

get_server_info

Overview

Retrieves info about the SugarCRM instance.

Available APIs

- SOAP
- REST

Definition

get_server_info()

Parameters

Name	Type	Description
null	null	No parameters available.

Result

Name	Type	Description
result	get_server_info_result | Array	The call result.
result.flavor	String	The flavor of the instance.
result.version	String	The version of the instance.
result.gmt_time	String	The GMT time of the server.

Change Log

Version	Change
v2	Method ***get_server_info*** was added to replace ***get_server_time***, ***get_server_version*** and ***get_sugar_flavor***.
v2	Method ***get_server_time*** was removed.
v2	Method ***get_server_version*** was removed.
v2	Method ***get_sugar_flavor*** was removed.

Examples

PHP

```
//this method does not have any parameters
$get_server_info_parameters = array();
```

---

get_upcoming_activities

Overview

Retrieves a list of upcoming activities for the current user.

Available APIs

- SOAP
- REST

Definition

get_upcoming_activities(session)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.

Result

Name	Type	Description
result	upcoming_activities_list | Array	The call result. Contains a list of upcoming activity records.
result[].id	Integer	The record ID.
result[].module	String	The activity module.
result[].date_due	String	The date the activity is due.
result[].summary	String	The summary of the activity.

Change Log

Version	Change

Examples

PHP

```
$get_upcoming_activities_parameters = array(
    //Session id
    "session" => $session_id,
);
```

get_user_id

Overview

Retrieves the id of the user currently logged in.

Available APIs

- SOAP
- REST

Definition

get_user_id(session)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.

Result

Name	Type	Description
id	String	The users ID.

Change Log

Version	Change

Examples

PHP

```
$get_user_id_parameters = array(
    //Session id
    "session" => $session_id,
);
```

get_user_team_id

Overview

Retrieves the ID of the default team of the user who is logged into the current session.

Available APIs

- SOAP
- REST

Definition

get_user_team_id(session)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.

Result

Name	Type	Description
default_team_id	Integer	The ID of the current users default ream

Change Log

Version	Change
v2	Added **get_user_team_id** method.

Examples

PHP

```
$get_user_team_id_parameters = array(
    //Session id
    "session" => $session_id,
);
```

---

job_queue_cycle

Overview

Runs through the scheduler cleanup process and cycles the scheduler jobs.

Available APIs

- REST

Definition

job_queue_cycle(session, clientid)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
clientid	String	The client id calling the application. This parameter is of your choosing for the calling application.

Result

Name	Type	Description
result	Array	The call result.
result.results	String	The cycle result. Returns 'ok' on success.

Change Log

Version	Change
v4	Added ***job_queue_cycle*** method.

Examples

PHP

```
$job_queue_cycle_parameters = array(
    //Session id
    'session' => $session_id,
```

```
    //The ID of the calling application.
    'clientid' => 'MyAppID',
);
```

job_queue_next

Overview

Retrieves the next job from the job queue and marks it as 'In Progress'.

Available APIs

- REST

Definition

job_queue_next(session, clientid)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
clientid	String	The client id calling the application. This parameter is of your choosing for the calling application.

Result

Name	Type	Description
result	Array	The call result.
result.results	String	The next job ID.

Change Log

Version	Change
v4	Added ***job_queue_next*** method.

Examples

PHP

```
$job_queue_next_parameters = array(
    //Session id
    'session' => $session_id,

    //The ID of the calling application.
    'clientid' => 'MyAppID',
);
```

---

job_queue_run

Overview

Runs the specified job.

Available APIs

- REST

Definition

job_queue_run(session, jobid, clientid)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
jobid	String	The ID of the job to run.
clientid	String	The client id calling the application. This parameter is of your choosing for the calling application.

Result

Name	Type	Description
result	Array	The call result.
result.results	Boolean	The result of the job run.
result.message	String	This is only returned if a failure occurs.

Change Log

Version	Change
v4	Added ***job_queue_run*** method.

Examples

PHP

```
$job_queue_run_parameters = array(
    //Session id
    'session' => $session_id,

    //The ID of the job to run.
    'jobid' => 'd141efd3-d2c7-8a9c-9c02-50c11b491f16',

    //The ID of the calling application.
    'clientid' => 'MyAppID',
);
```

---

login

Overview

Logs a user into the SugarCRM application.

Available APIs

- SOAP
- REST

Definition

login(user_auth, application_name, name_value_list)

Parameters

Name	Type	Description
user_auth	user_auth | Array	Contains the parameters to authenticate a user.
user_auth.user_name	String	The user name of your user
user_auth.password	String	The MD5 hash of the users password.
application name	String	The name of the application logging in.
name_value_list	name_value_list | Array	Sets the name_value pair. The parameter is used to set values for the 'language' and 'notifyonsave' user settings.
name_value_list.language	String	The language for the user.
name_value_list.notifyonsave	Boolean	Alerts users on new record creations when set to true.

Result

Name	Type	Description
result	entry_value | Array	The call result
result.id	String	This is the session id required to make other method calls.
result.module_name	String	Returns the 'Users' module.
result.name_value_list	Array	Authenticated user properties.
result.name_value_list.user_id	String	ID of the authenitcated user.
result.name_value_list.user_name	String	Username of the authenticated user.
result.name_value_list.user_language	String	Default language of the authenticated user.
result.name_value_list.use	String	Default currency ID of the

Name	Type	Description
r_currency_id		authenticated user.
result.name_value_list.user_is_admin	String	Admin status of the authenticated user.
result.name_value_list.user_default_team_id	String	Default team of the authenticated user.
result.name_value_list.user_default_dateformat	String	Default date format for the authenticated user.
result.name_value_list.user_default_timeformat	String	Default time format for the authenticated user.
result.name_value_list.user_number_seperator	String	Number seperator for the authenticated user.
result.name_value_list.user_decimal_seperator	String	Decimal sperator for the authenticated user.
result.name_value_list.mobile_max_list_entries	String	Max list entires for the authenticated user.
result.name_value_list.mobile_max_subpanel_entries	String	Max subpanel entries for the authenticated user.
result.name_value_list.user_currency_name	String	Default currency name for the authenticated user.

Change Log

Version	Change
v3_1	Added additional return values to ***name_value_list***. The list now also includes ***user_number_seperator***, ***user_decimal_seperator***, ***mobile_max_list_entries***, ***mobile_max_subpanel_entries***.
v3	Added additional return values to ***name_value_list***. The list now also includes ***user_is_admin***, ***user_default_team_id***, ***user_default_dateformat***, ***user_default_timeformat***.
v2	Added ***name_value_list*** to response. Returns ***user_id***, ***user_name***, ***user_language***, ***user_currency_id, user_currency_name***.
v2	Added ***module_name*** to response.
v2	Removed ***error*** from response.
v2	Added ***name_value_list*** parameter
v2	Return type was changed from set_entry_result to

	entry_value.

Examples

PHP

```
  $login_parameters = array(
    //user authentication
    "user_auth" => array(
        "user_name" => $username,
        "password" => md5($password),
    ),

    //application name
    "application_name" => "My Application",

    //name value list for 'language' and 'notifyonsave'
    "name_value_list" => array(
        array(
            'name' => 'language',
            'value' => 'en_us',
        ),
        array(
            'name' => 'notifyonsave',
            'value' => true
        ),
    ),
);
```

logout

Overview

Logs a user out of the SugarCRM application.

Available APIs

- SOAP
- REST

Definition

logout(session)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous call to login.

Result

Name	Type	Description
null	void	No return value.

Change Log

Version	Change
v2	Return type was changed from error_value to ***void***.

Example

PHP

```
$logout_parameters = array(
    //session id to expire
    "session" => $session_id,
);
```

---

oauth_access

Overview

Retrieves the OAuth access token.

Available APIs

- REST

Definition

oauth_access()

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.

Result

Name	Type	Description
result	Array	The call result.
result.id	String	The OAuth access token.

Change Log

Version	Change
v4	Added ***oauth_access*** method.

Examples

PHP

```
$oauth_access_parameters = array(
    //Session id
    'session' => $session_id,
);
```

---

seamless_login

Overview

Verifies that a session is authenticated.

Available APIs

- SOAP
- REST

Definition

seamless_login(session)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.

Result

Name	Type	Description
result	Integer	Returns 1 if the session is authenticated. Otherwise 0 will be returned.

Change Log

Version	Change

Considerations

If you are attempting to log a user into SugarCRM seamlessly, you can do this by passing the validated session_id in the url.

An example is shown below:

```
http://{site_url}/index.php?module=Home&action=index&MSID={session_id}
```

Examples

PHP

```
$seamless_login_parameters = array(
    //Session id
    "session" => $session_id,
);
```

---

search_by_module

Overview

Searches modules for a string and returns matched records.

Available APIs

- SOAP
- REST

Definition

search_by_module(session, search_string, modules, offset, max_results, assigned_user_id, select_fields, unified_search_only, favorites)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
search_string	String	The string to search for.
modules	Integer	The list of modules to query.
offset	Integer	The record offset from which to start.
max_results	Integer	The maximum number of records to return.
assigned_user_id	String	Filters records by the assigned user ID. Leave this empty if no filter should be applied.
select_fields	select_fields | Array	An array of fields to return. If empty the default return fields will be from the active listviewdefs.
unified_search_only	Boolean	If the search is only against modules participating in the unified search.
favorites	Boolean	If only records marked as favorites should be returned.

Result

Name	Type	Description
result	return_search_result | Array	Call result.
result.entry_list	Array	The count of records in paged result.
result.entry_list[].name	String	The .name of the module
result.entry_list[].records	Array	A list of name_value lists for each record matched.

Change Log

Version	Change
v3_1	Added ***unified_search_only*** parameter.

Examples

PHP

```
$search_by_module_parameters = array(
    //Session id
    "session" => $session_id,

    //The string to search for.
    'search_string' => 'example text',

    //The list of modules to query.
    'modules' => array(
        'Accounts',
    ),

    //The record offset from which to start.
    'offset' => 0,

    //The maximum number of records to return.
    'max_results' => 100,

    //Filters records by the assigned user ID.
    //Leave this empty if no filter should be applied.
    'assigned_user_id' => '',

    //An array of fields to return.
    //If empty the default return fields will be from the active listv
iewdefs.
```

```
    'select_fields' => array(
        'id',
        'name',
    ),

    //If the search is only search modules participating in the unifie
d search.
    'unified_search_only' => false,

    //If only records marked as favorites should be returned.
    'favorites' => false
);
```

set_campaign_merge

Overview

Handles campaign log entry creation for mail-merge activity given a specified campaign.

Available APIs

- SOAP
- REST

Definition

set_campaign_merge(session, targets, campaign_id)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous call to login.
targets	select_fields | Array	A string array of IDs identifying the targets used in the merge. The IDs used in this parameter come from the column 'pr ospect_lists_prospects.id'.
campaign_id	String	The campaign ID used for the mail merge.

Result

Name	Type	Description
null	void	No return value.

Change Log

Version	Change
v2	Return type was changed from error_value to ***void***.

Example

PHP

```
$set_campaign_merge_parameters = array(
    //Session id
    "session" => $session_id,

    //A string array of IDs identifying the targets used in the merge.
    //The IDs used in this parameter come from the column 'prospect_li
sts_prospects.id'.
    "targets" => array(
        '403787cc-ab19-bec8-3ef4-50bd4896c9b3',
        'c5341c8d-4b0a-2b56-7108-50bd48b91213'
    ),

    //The campaign ID used for the mail merge.
    "campaign_id" => '781d4471-fb48-8dd2-ae62-50bd475950b2'
);
```

set_document_revision

Overview

Creates a new document revision for a specific document record.

Available APIs

- SOAP
- REST

Definition

set_document_revision(session, note)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
note	document_revision | Array	The file attachment details
note.id	String	The ID of the note record to associate the attachment to.
note.file	String	The binary contents of the file.
note.filename	String	The file name of the file attachment.
note.revision	String	The revision number

Result

Name	Type	Description
result	new_set_entry_result | array	The results from the call.
result.id	String	The ID of the document revision.

Change Log

Version	Change
v2	Return type was changed from set_entry_result to ***new_set_entry_result***.

Examples

PHP

```
$file_contents = file_get_contents("/path/to/example_document.txt");
$set_document_revision_parameters = array(
    //Session id
    "session" => $session_id,
```

```
    //The attachment details
    "note" => array(
        //The ID of the parent document.
        'id' => $document_id,

        //The binary contents of the file.
        'file' => base64_encode($file_contents),

        //The name of the file
        'filename' => 'example_document.txt',

        //The revision number
        'revision' => '1',
    ),
);
```

set_entries

Overview

Create or update a list of records.

Available APIs

- SOAP
- REST

Definition

set_entries(session, module_name, name_value_lists)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
module_name	String	The name of the module from which to retrieve records. Note: This is the modules key which may not be the same as the

Name	Type	Description
		modules display name.
name_value_lists	name_value_lists | Array	The an array of name/value lists containing the record attributes.

Result

Name	Type	Description
result	new_set_entries_result | Array	The call result.
result.ids	Array	The list of record IDs that were created or updated

Change Log

Version	Change
v2	Return type was changed from set_entry_result to ***new_set_entries_result***.

Considerations

- To update an existing record, you will need to specify 'id' for the name_value_list item in the name_value_lists parameter.
- To create a new record with a specific ID, you will need to set 'new_with_id' in the name_value_list item in the name_value_lists parameter.

Examples

PHP

```
$set_entries_parameters = array(
    //Session id
    "session" => $session_id,

    //The name of the module from which to retrieve records.
    "module_name" => "Accounts",

    //Record attributes
    "name_value_lists" => array(
        array(
            //to update a record
```

```
            /*
            array(
                "name" => "id",
                "value" => "da0b107d-cfbc-cb08-4f90-50b7b9cb9ad7"
            ),
            */

            //to create a new record with a specific ID
            /*
            array(
                "name" => "new_with_id",
                "value" => 1
            ),
            */
            array(
                "name" => "name",
                "value" => "Example Account 1"
            ),
        ),
        array(
            //to update a record
            /*
            array(
                "name" => "id",
                "value" => "da0b107d-cfbc-cb08-4f90-50b7b9cb9ad7"
            ),
            */

            //to create a new record with a specific ID
            /*
            array(
                "name" => "new_with_id",
                "value" => 1
            ),
            */

            array(
                "name" => "name",
                "value" => "Example Account 2"
            ),
        ),
    )
,);
```

set_entry

Overview

Creates or updates a specific record.

Available APIs

- SOAP
- REST

Definition

set_entry(session, module_name, name_value_list)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
module_name	String	The name of the module from which to retrieve records. Note: This is the modules key which may not be the same as the modules display name.
name_value_list	name_value_list | Array	The name/value list of the record attributes.

Result

Name	Type	Description
result	new_set_entry_result | Array	The call result.
result.id	String	The ID of the record that was created/updated.

Change Log

Version	Change
v2	Return type was changed from set_entry_result to

	new_set_entry_result.

Considerations

- To update an existing record, you will need to specify 'id' in the name_value_list parameter.
- To create a new record with a specific ID, you will need to set 'new_with_id' in the name_value_list parameter.

Examples

PHP

```
$set_entry_parameters = array(
    //session id
    "session" => $session_id,

    //The name of the module from which to retrieve records.
    "module_name" => "Accounts",

    //Record attributes
    "name_value_list" => array(
        //to update a record
        /*
        array(
            "name" => "id",
            "value" => "da0b107d-cfbc-cb08-4f90-50b7b9cb9ad7"
        ),
        */

        //to create a new record with a specific ID
        /*
        array(
            "name" => "new_with_id",
            "value" => true
        ),
        */

        array(
            "name" => "name",
            "value" => "Example Account"
        ),
    ),
);
```

set_note_attachment

Overview

Creates an attachment and associated it to a specific note record.

Available APIs

- SOAP
- REST

Definition

set_note_attachment(session, note)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
note	new_note_attachment | Array	The file attachment details
note.id	String	The ID of the note record to associate the attachment to.
note.filename	String	The file name of the file attachment.
note.file	String	The binary contents of the file.

Result

Name	Type	Description
result	new_set_entry_result | Array	The call result.
result.id	String	The ID of the note record / attachment

Change Log

Version	Change

v2	Return type was changed from set_entry_result to ***new_set_entry_result***.
v2	Parameter note type change from note_attachment to ***new_note_attachment***.

Examples

PHP

```
$file_contents = file_get_contents("/path/to/example_file.php");
$set_note_attachment_parameters = array(
    //Session id
    "session" => $session_id,

    //The attachment details
    "note" => array(
        //The ID of the note containing the attachment.
        'id' => $note_id,

        //The file name of the attachment.
        'filename' => 'example_file.php',

        //The binary contents of the file.
        'file' => base64_encode($file_contents),
    ),
);
```

set_relationship

Overview

Sets relationships between two records. You can relate multiple records to a single record using this.

Available APIs

- SOAP
- REST

Definition

set_relationship(session, module_name, module_id, link_field_name, related_ids, name_value_list, delete)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
module_name	String	The name of the module from which to retrieve records. Note: This is the modules key which may not be the same as the modules display name.
module_id	String	The ID of the specified module record.
link_field_name	String	The name of the link field for the related module.
related_ids	select_fields | Array	The list of related record IDs you are relating
name_value_list	name_value_list | Array	An array specifying relationship fields to populate. An example of this is contact_role between Opportunities and Contacts.
delete	Integer	Determines whether the relationship is being created or deleted. 0:create, 1:delete

Result

Name	Type	Description
result	new_set_relationship_list_result | Array	The call result
result.created	Integer	The number of relationships created.
result.failed	Integer	Determines whether or not the relationship failed. This is normally thrown when the parameters module_name or link_field_name are incorrect.

Name	Type	Description
result.deleted	Integer	The number of relationships deleted.

Change Log

Version	Change
v2	Removed ***set_relationship_value*** parameter.
v2	Added ***module_name*** parameter.
v2	Added ***module_id*** parameter.
v2	Added ***link_field_name*** parameter.
v2	Added ***related_ids*** parameter.
v2	Added ***name_value_list*** parameter.
v2	Added ***delete*** parameter.
v2	Return type was changed from error_value to ***new_set_relationship_list_result***.

Examples

PHP

```
$set_relationship_parameters = array(
    //session id
    'session' => $session_id,

    //The name of the module.
    'module_name' => 'Opportunities',

    //The ID of the specified module bean.
    'module_id' => '15e79b92-5025-827f-0784-50aa578270d8',

    //The relationship name of the linked field from which to relate r
ecords.
    'link_field_name' => 'contacts',

    //The list of record ids to relate
    'related_ids' => array(
        '19b8799e-64ae-9502-588c-50aa575454c9',
    ),

    //Sets the value for relationship based fields
    'name_value_list' => array(
```

```
        array(
            'name' => 'contact_role',
            'value' => 'Other'
        )
    ),

    //Whether or not to delete the relationship. 0:create, 1:delete
    'delete'=> 0,
);
```

set_relationships

Overview

Sets multiple relationships between mulitiple record sets.

Available APIs

- SOAP
- REST

Definition

set_relationships(session, module_names, module_ids, link_field_names, related_ids, name_value_lists, delete_array)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
module_names	select_fields | Array	The list of modules from which to retrieve records. Note: This is the modules key which may not be the same as the modules display name.
module_ids	select_fields | Array	The list of IDs for the specified module records.
link_field_names	select_fields | Array	The list of link names for the related modules.

Name	Type	Description
related_ids	new_set_relationhip_ids | Array	The list of related record IDs you are relating.
name_value_lists	name_value_lists | Array	An array of arrays specifying relationship fields to populate. An example of this is contact_role between Opportunities and Contacts.
delete_array	deleted_array | Array	An array determining whether the relationships are being created or deleted. 0:create, 1:delete

Result

Name	Type	Description
result	new_set_relationship_list_result | Array	The call result.
result.created	Integer	The number of relationships created.
result.failed	Integer	Determines whether or not the relationship failed. This is normally thrown when the parameters module_name or link_field_name are incorrect.
result.deleted	Integer	The number of relationships deleted.

Change Log

Version	Change
v2	Removed ***set_relationship_value*** parameter.
v2	Added ***module_names*** parameter.
v2	Added ***module_ids*** parameter.
v2	Added ***link_field_names*** parameter.
v2	Added ***related_ids*** parameter.

v2	Added ***name_value_list***s parameter.
v2	Added ***delete_array*** parameter.
v2	Return type was changed from set_relationship_list_result to ***new_set_relationship_list_result***.

Examples

PHP

```
$set_relationships_parameters = array(
    //session id
    'session' => $session_id,

    //The name of the modules from which to relate records.
    'module_names' => array(
        'Opportunities',
        'Accounts',
    ),

    //The IDs of the specified module beans.
    'module_ids' => array(
        '15e79b92-5025-827f-0784-50aa578270d8', //Opportunity ID
        '27035f04-f6ec-492d-b89e-50aa57f5247f'  //Account ID
    ),

    //The relationship names of the linked fields from which to relate
 records.
    'link_field_names' => array(
        'contacts', //Contacts link field to Opportunities
        'leads' //Leads link field to Accounts
    ),

    //The lists of record ids to relate
    'related_ids' => array(
        //Contact IDs
        array(
            '19b8799e-64ae-9502-588c-50aa575454c9'
        ),

        //Lead IDs
        array(
            '16d8d519-5f56-0984-2092-50aa576a7333',
            '15ae07eb-63f0-dbac-6e4c-50aa57c5a609'
        ),
```

```
    ),

    //Sets the value for relationship based fields
    'name_value_lists' => array(
        //Opportunity-Contact relationship fields
        array(
            array(
                'name' => 'contact_role',
                'value' => 'Other'
            ),
        ),

        //Account-Lead relationship fields
        array(),
    ),

    //Whether or not to delete the relationships. 0:create, 1:delete
    'delete_array'=> array(
        0, //Opportunity-Contact
        0 //Account-Lead
    ),
);
```

snip_import_emails

Overview

Used to imports an email record from the SNIP archiving service.

Available APIs

- REST

Definition

snip_import_emails(session, email)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.

Name	Type	Description
email	Array	The contents of the email being imported.
email.message	Array	Contains the email attributes.
email.message.message_id	String	The ID of the email message.
email.message.subject	String	Email subject.
email.message.attachments	Array	The list of attachments to be imported
email.message.from_name	String	From sender name.
email.message.description	String	Plain text content body.
email.message.description_html	String	HTML email content body.
email.message.to_addrs	String	Email addresses the email was sent to.
email.message.cc_addrs	String	Email addresses the email was CCed to.
email.message.bcc_addrs	String	Email addresses the email was BCCed to.
email.message.date_sent	String	Date the email was sent.

Result

Name	Type	Description
result	Array	The call result.
result.results	Boolean	The success of the import.
result.count	Integer	The count of records imported.
result.message	String	The return message.

Change Log

Version	Change
v4	Added ***snip_import_emails*** method.

snip_update_contacts

Overview

Retrieves new contact emails since a timestamp for the current user.

Available APIs

- REST

Definition

snip_update_contacts(session, report_id)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
report_id	String	The ID of the report record to generate the PDF for.

Result

Name	Type	Description
result	Array	The call result.
result.results	Boolean	The new emails.
result.count	Integer	The count of results.
result.message	String	The return message.

Change Log

Version	Change
v4	Added ***snip_update_contacts*** method.

REST Release Notes

Overview

Lists changes between the different versions of the REST API.

Release Notes

v4_1

- get_modified_relationships method was added.
- get_relationships had the parameter $limit added.
- get_relationships had the parameter $offset added.

v4

- get_entries had the parameter $track_view removed.
- job_queue_cycle method was added.
- job_queue_next method was added.
- job_queue_run method was added.
- oauth_access method was added.
- oauth_access_token method was added.
- oauth_request_token method was added.
- search_by_module had the parameter $favorites added.
- snip_import_emails method was added.
- snip_update_contacts method was added.

v3_1

- get_entries had the parameter $track_view added.
- get_entry had the parameter $track_view added.
- get_entry_list had the parameter $favorites added.
- get_language_definition method was added.
- get_module_layout had the parameter $acl_check added.
- get_module_layout_md5 had the parameter $acl_check added.
- get_quotes_pdf method was added.
- get_report_pdf method was added.
- search_by_module had the parameter $unified_search_only added.
- set_entry had the parameter $track_view added.

v3

- get_available_modules had the parameter $filter added.
- get_last_viewed method was added.
- get_module_fields_md5 method was added.
- get_module_layout method was added.
- get_module_layout_md5 method was added.
- get_relationships had the parameter $order_by added.
- get_upcoming_activities method was added.
- search_by_module had the parameter $assigned_user_id added.
- search_by_module had the parameter $select_fields added.

v2_1

- get_entry_list method logic was modified.
- get_report_entries method logic was modified.
- md5 method was removed.

v2 (REST API was introduced into SugarCRM)

- get_available_modules method was added.
- get_document_revision method was added.
- get_entries method was added.
- get_entries_count method was added.
- get_entry method was added.
- get_entry_list method was added.
- get_module_fields method was added.
- get_note_attachment method was added.
- get_relationships method was added.
- get_report_entries method was added.
- get_server_info method was added.
- get_user_id method was added.
- get_user_team_id method was added.
- login method was added.
- logout method was added.
- md5 method was added.
- seamless_login method was added.
- search_by_module method was added.
- set_campaign_merge method was added.
- set_document_revision method was added.
- set_entries method was added.
- set_entry method was added.
- set_note_attachment method was added.
- set_relationship method was added.
- set_relationships method was added.SOAP Release Notes

SOAP Release Notes

Overview

Lists changes between the different versions of the SOAP API.

Release Notes

v4_1

- get_modified_relationships method was added.
- get_relationships had the parameter $limit added.
- get_relationships had the parameter $offset added.

v4

- search_by_module had the parameter $favorites added.

v3_1

- get_entries had the parameter $track_view added.
- get_entry had the parameter $track_view added.
- get_entry_list had the parameter $favorites added.
- search_by_module had the parameter $unified_search_only added.

v3

- get_available_modules had the parameter $filter added.
- get_last_viewed method was added.
- get_module_fields_md5 method was added.
- get_relationships had the parameter $order_by added.
- get_upcoming_activities method was added.
- search_by_module had the parameter $assigned_user_id added.
- search_by_module had the parameter $select_fields added.

v2_1

- get_entry_list method logic was modified.
- get_report_entries method logic was modified.

v2

- contact_by_email method was removed.
- create_account method was removed.
- create_case method was removed.
- create_contact method was removed.
- create_lead method was removed.
- create_opportunity method was removed.
- create_session method was removed.
- end_session method was removed.
- get_attendee_list method was removed.
- get_contact_relationships method was removed.
- get_disc_client_file_list method was removed.
- get_document_revision return type was changed from return_document_revision to new_return_document_revision.

- get_encoded_file method was removed.
- get_encoded_portal_zip_file method was removed.
- get_encoded_zip_file method was removed.
- get_entries had the parameter $link_name_to_fields_array added.
- get_entries return type was changed from get_entry_result to get_entry_result_version2.
- get_entry had the parameter $link_name_to_fields_array added.
- get_entry return type was changed from get_entry_result to get_entry_result_version2.
- get_entry_list had the parameter $link_name_to_fields_array added.
- get_entry_list return type was changed from get_entry_list_result to get_entry_list_result_version2.
- get_gmt_time method was removed.
- get_mailmerge_document method was removed.
- get_mailmerge_document2 method was removed.
- get_modified_entries method was removed.
- get_module_fields had the parameter $fields added.
- get_module_fields return type was changed from module_fields to new_module_fields.
- get_note_attachment return type was changed from return_note_attachment to new_return_note_attachment.
- get_quick_sync_data method was removed.
- get_related_notes method was removed.
- get_relationships had the parameter $link_field_name added.
- get_relationships had the parameter $related_fields added.
- get_relationships had the parameter $related_module removed.
- get_relationships had the parameter $related_module_link_name_to_fields_array added.
- get_relationships return type was changed from get_relationships_result to get_entry_result_version2.
- get_report_entries method was added.
- get_required_upgrades method was removed.
- get_server_info method was added.
- get_server_time method was removed.
- get_server_version method was removed.
- get_sugar_flavor method was removed.
- get_system_status method was removed.
- get_unique_system_id method was removed.
- is_loopback method was removed.
- is_user_admin method was removed.
- login had the parameter $name_value_list added.
- login return type was changed from set_entry_result to entry_value.
- logout return type was changed from error_value to void.
- offline_client_available method was removed.
- relate_note_to_module method was removed.
- search method was removed.

- search_by_module had the parameter $password removed.
- search_by_module had the parameter $session added.
- search_by_module had the parameter $user_name removed.
- search_by_module return type was changed from get_entry_list_result to return_search_result.
- set_campaign_merge return type was changed from error_value to void.
- set_document_revision return type was changed from set_entry_result to new_set_entry_result.
- set_entries return type was changed from set_entries_result to new_set_entries_result.
- set_entries_details method was removed.
- set_entry return type was changed from set_entry_result to new_set_entry_result.
- set_note_attachment had the parameter $note type change from note_attachment to new_note_attachment.
- set_note_attachment return type was changed from set_entry_result to new_set_entry_result.
- set_relationship had the parameter $delete added.
- set_relationship had the parameter $link_field_name added.
- set_relationship had the parameter $module_id added.
- set_relationship had the parameter $module_name added.
- set_relationship had the parameter $name_value_list added.
- set_relationship had the parameter $related_ids added.
- set_relationship had the parameter $set_relationship_value removed.
- set_relationship return type was changed from error_value to new_set_relationship_list_result.
- set_relationships had the parameter $delete_array added.
- set_relationships had the parameter $link_field_names added.
- set_relationships had the parameter $module_ids added.
- set_relationships had the parameter $module_names added.
- set_relationships had the parameter $name_value_lists added.
- set_relationships had the parameter $related_ids added.
- set_relationships had the parameter $set_relationship_list removed.
- set_relationships return type was changed from set_relationship_list_result to new_set_relationship_list_result.
- sudo_user method was removed.
- sync_get_entries method was removed.
- sync_get_modified_relationships method was removed.
- sync_get_relationships method was removed.
- sync_set_entries method was removed.
- sync_set_relationships method was removed.
- track_email method was removed.
- update_portal_user method was removed.
- user_list method was removed.

SugarHttpClient

Overview

The SugarHttpClient class is used to make REST calls.

The SugarHttpClient Class

The SugarHttpClient class is located in 'include/SugarHttpClient.php'. It contains a callRest() method that will allow you to post a request to a REST service via cURL without having to worry about the overhead or the restrictions on the file_get_contents() method when doing outbound webservice calls.

Making a Request

```
<?php

  // specify the REST web service to interact with
  $url = 'http://{sugar_url}/service/v4_1/rest.php';

  // Open a SugarHttpClient session for making the call
  require_once('include/SugarHttpClient.php');
  $client = new SugarHttpClient;

  // Set the POST arguments to pass to the Sugar server
  $parameters = array(
      'user_auth' => array(
          'user_name' => 'username',
          'password' => md5('password'),
      ),
  );

  $json = json_encode($parameters);

  $postArgs = array(
      'method' => 'login',
      'input_type' => 'JSON',
      'response_type' => 'JSON',
      'rest_data' => $json,
  );

  $postArgs = http_build_query($postArgs);

  // Make the REST call, returning the result
  $response = $client->callRest($url, $postArgs);
```

```
    if ( $response === false )
    {
        die("Request failed.\n");
    }

    // Convert the result from JSON format to a PHP array
    $result = json_decode($response);

    if ( !is_object($result) )
    {
        die("Error handling result.\n");
    }

    if ( !isset($result->id) )
    {
        die("Error: {$result->name} - {$result->description}\n.");
    }

    // Get the session id
    $sessionId = $result->id;

?>
```

ExternalResourceClient

Overview

In Sugar 12.1, ExternalResourceClient was introduced to replace cURL. This has also been backported to all supported versions of Sugar. Therefore any apps or integrations for Sugar that use a Module Loadable Package (MLP) that includes any of the PHP curl_*, socket_*, or stream_* functions will need to change to use the ExternalResourceClient lib.

Self-Signed Certificates

ExternalResourceClient is designed to be as secure as possible. We will not support disabling the check for authenticity of the peer's certificate as you could in cURL with using the CURLOPT_SSL_VERIFYPEER option. This is not a security best practice and is therefore not allowed in the new client.

If your target URL has an invalid or self-signed certificate, you can benefit from using Let's Encrypt. Let's Encrypt is a free, automated, and open Certificate Authority (CA) that makes it possible to set up an HTTPS server and have it

automatically obtain a browser-trusted certificate without any human intervention. You can get started here.

IP vs Domain/Hostnames via DNS

ExternalResourceClient does not support using an IP address in your URL. Editing /etc/hosts on your server is a very easy and fast way of accomplishing this, but if that is not an option and you have a proper DNS server configured, you can use one of the following articles to help you understand what DNS is and how you can configure them in different OSs and infrastructures.

- Build your own DNS server on Linux
- DNS Configuration: Everything You Need to Know
- How to Setup DNS Server on Windows Server 2012

Note: If your domain is behind a LoadBalancer, ExternalResourceClient will reverse DNS it for an IP and that resolution may bring different IPs every time. LoadBalancer will balance the load to the IP at that particular moment and return it to ExternalResourceClient. If you are in a restricted DMZ behind a firewall and need to open connections to those IPs, you must get all IPs served by the LoadBalancer and add them all to your firewall.

Local Endpoints

If you do not know what server-side request forgery (SSRF) is, please refer to this detailed explanation and examples of how an attacker can reach your local network using malicious HTTP requests.

ExternalResourceClient prevents this type of attack by taking the resolved IP from your URL and checking against a Sugar Config $sugar_config['security']['private_ips'] array list.

This list contains a wide range of internal IPs blocked by default ['10.0.0.0|10.255.255.255', '172.16.0.0|172.31.255.255', '192.168.0.0|192.168.255.255', '169.254.0.0|169.254.255.255', '127.0.0.0|127.255.255.255'] .

For example, if your URL is http://an-internal-url/api/get and "an-internal-url" resolves to an IP 10.0.0.87, it will throw an exception because it is within the range of your config.

Add your critical IP ranges to this config so no attacker will get to sensitive IPs through ExternalResourceClient's HTTP Request.

Sugar Proxy

To get proxy settings, ExternalResourceClient relies on Administration::getSettings('proxy'). This utility queries Sugar's infrastructure (cache or DB) to retrieve those settings, therefore it needs to be bootstrapped into SugarCRM.

In the case of MLPs that use standard modules via the web, you don't need to manually bootstrap SugarCRM, so you may use ExternalResourceClient in your methods without requiring entryPoint.php

If you are writing a CLI script, you must include SugarCRM entryPoint.php:

```
  <?php
if (!defined('sugarEntry')) {
   define('sugarEntry', true);
}

define('ENTRY_POINT_TYPE', 'api');
require_once 'include/entryPoint.php';

?>
```

Test your customization

You can easily test your package by enabling PackageScanner checks in the supported versions for now ($sugar_config['moduleInstaller']['enableEnhancedModuleChecks'] = true), remember, it comes disabled by default in 12.1.

If your code doesn't go through PackageScanner, it will not be enforced, that's the premise we're working with.

Examples of Replacing cURL

The below examples can be used as a guideline for replacing your current cURL code with ExternalResourceClient.

You must import this client and its exceptions (if needed) from:

```
  <?php

use Sugarcrm\Sugarcrm\Security\HttpClient\ExternalResourceClient;
use Sugarcrm\Sugarcrm\Security\HttpClient\RequestException;
```

HTTP GET

The following code snippet provides you with a code replacement from cURL GET to ExternalResourceClient.

Before:

```
  // cURL GET
$ch = curl_init();

$url = 'https://httpbin.org/get';
curl_setopt_array($ch, array(
    CURLOPT_URL => $url,
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_ENCODING => "",
    CURLOPT_MAXREDIRS => 10,
    CURLOPT_TIMEOUT => 60,
    CURLOPT_FOLLOWLOCATION => true,
    CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_2TLS,
));

$response = curl_exec($ch);

$httpCode = curl_getinfo($ch, CURLINFO_HTTP_CODE);
$errorNo = curl_errno($ch);
$error = curl_error($ch);

if ($errorNo !== 0) {
    $GLOBALS['log']->log("fatal", "curl error ($errorNo): $error");
    throw new \SugarApiExceptionError("curl error ($errorNo): $error")
;
}

curl_close($ch);

if ($httpCode === 0 || $httpCode >= 400) {
    $GLOBALS['log']->log("fatal", "Error message: $error");
    throw new \SugarApiException($error, null, null, $httpCode ? $http
Code : 500);
}

echo $response;
```

After:

```
  // ExternalResourceClient GET
try {
    // Set timeout to 60 seconds and 10 max redirects
    $response = (new ExternalResourceClient(60, 10))->get($url);
} catch (RequestException $e) {
    $GLOBALS['log']->log('fatal', 'Error: ' . $e->getMessage());
    throw new \SugarApiExceptionError($e->getMessage());
}
$httpCode = $response->getStatusCode();
if ($httpCode >= 400) {
    $GLOBALS['log']->log("fatal", "Request failed with status: " . $ht
tpCode);
    throw new \SugarApiException("Request failed with status: " . $htt
pCode, null, null, $httpCode);
}
echo $response->getBody()->getContents();
```

Note that timeout cannot be 0 or negative.

HTTP POST

The following code snippet provides you with a code replacement from cURL POST to ExternalResourceClient.

Before:

```
  // cURL POST JSON
$url = 'https://httpbin.org/post';

$ch = curl_init($url);
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 2);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, 1);
curl_setopt($ch, CURLOPT_HTTPHEADER, array ("Content-
Type: application/json"));
curl_setopt($ch, CURLOPT_POSTFIELDS, json_encode(['foo' => 'bar']));

$response = curl_exec($ch);
curl_close($ch);
$parsed = !empty($response) ? json_decode($response, true) : null;
var_dump($parsed);
```

After:

```
  // ExternalResourceClient POST JSON
try {
    $response = (new ExternalResourceClient())->post($url, json_encode
(['foo' => 'bar']), ['Content-Type' => "application/json"]);
} catch (RequestException $e) {
    throw new \SugarApiExceptionError($e->getMessage());
}

$parsed = !empty($response) ? json_decode($response->getBody()->getCon
tents(), true) : null;
var_dump($parsed);
```

Authenticated Endpoints (CURLOPT_USERPWD)

The following code snippet provides you with a code replacement from cURL to ExternalResourceClient passing authentication params.

Before:

```
  $username = 'foo';
$password = 'bar';
$url = 'https://httpbin.org/basic-auth/{$username}/{$password}';

$ch = curl_init($url);
curl_setopt($ch, CURLOPT_USERPWD, $username . $password);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 2);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, json_encode(['foo' => 'bar']));

$response = curl_exec($ch);
curl_close($ch);
$parsed = !empty($response) ? json_decode($response, true) : null;
var_dump($parsed);
```

After:

```
  $username = 'foo';
$password = 'bar';
$auth     = base64_encode( "{$username}:{$password}" );
```

```
 //Passing custom 'Authorization' header
// try + catch is omitted for brevity
$response = (new ExternalResourceClient())->get("https://httpbin.org/b
asic-auth/{$username}/{$password}", [
    'Authorization' => 'Basic ' . $auth,
]);
echo $response->getBody()->getContents();

// OR the same by using user:password@hostname.tld URL format
 // try + catch is omitted for brevity
$response = (new ExternalResourceClient())->get("https://{$username}:{
$password}@httpbin.org/basic-auth/{$username}/{$password}");
echo $response->getBody()->getContents();
```

Stream

The following code snippet provides you with a code replacement from stream to ExternalResourceClient.

Before:

```
  <?php
// Sending GET
if ($stream = fopen('https://httpbin.org/get', 'r')) {
    echo stream_get_contents($stream);
}

// Sending POST
$options = [
    'http' => [
        'method' => 'POST',
        'header' => ["Content-Type: application/x-www-form-
urlencoded"],
        'content' => http_build_query(['fieldName' => 'value', 'fieldN
ame2' => 'another value']),
        'ignore_errors' => true,
    ],
];
$context = stream_context_create($options);
if ($stream = fopen('https://httpbin.org/post', 'r', false, $context))
 {
    echo stream_get_contents($stream);
}
```

After:

```
  // Using ExtenalResourceClient
// GET
echo (new ExternalResourceClient())
->get('https://httpbin.org/get')
->getBody()
->getContents();
// POST
echo (new ExternalResourceClient())
->post(
'https://httpbin.org/post',
['fieldName' => 'value', 'fieldName2' => 'another value']
)
->getBody()
->getContents();
```

Socket

The following code snippet provides you with a code replacement from socket to ExternalResourceClient.

Before:

```
  <?php

// Sending GET
$fp = fsockopen("httpbin.org", 80, $errno, $errstr, 30);
if (!$fp) {
    echo "$errstr ($errno)\n";
} else {
    $out = "GET /get HTTP/1.1\r\n";
    $out .= "Host: httpbin.org\r\n";
    $out .= "Connection: Close\r\n\r\n";
    fwrite($fp, $out);
    while (!feof($fp)) {
        echo fgets($fp, 128);
    }
    fclose($fp);
}
// Sending POST

$fp = fsockopen("httpbin.org", 80, $errno, $errstr, 30);
if (!$fp) {
```

```
    echo "$errstr ($errno)\n";
} else {
    $postData = http_build_query(['fieldName' => 'value', 'fieldName2'
 => 'another value']);
    $out = "POST /post HTTP/1.1\r\n";
    $out .= "Host: httpbin.org\r\n";
    $out .= "Content-type: application/x-www-form-urlencoded\r\n";
    $out .= "Content-length: " . strlen($postData) . "\r\n";
    $out .= "Connection: Close\r\n\r\n";
    $out .= "{$postData}\r\n\r\n";
    fwrite($fp, $out);
    while (!feof($fp)) {
        echo fgets($fp, 128);
    }
    fclose($fp);
}
```

After:

```
  // Using ExtenalResourceClient
// GET
echo (new ExternalResourceClient())
->get('http://httpbin.org/get')
->getBody()
->getContents();
// POST
echo (new ExternalResourceClient())
->post(
'http://httpbin.org/post',
['fieldName' => 'value', 'fieldName2' => 'another value']
)
->getBody()
->getContents();
```

Upload a file

The following code snippet provides you with a code example using ExternalResourceClient to upload files from Sugar.

```
  // Uploading files from `upload` directory
// File data will be accessible on the target server as $_FILES['file_
field_name']
```

```
echo 'File upload', PHP_EOL, PHP_EOL;
$file = new UploadFile('file_field_name');
// relative to `upload` directory
$filename = 'f68/2cad6f68-e016-11ec-9c9c-0242ac140007';
$file->temp_file_location = $file->get_upload_path($filename);
// try + catch is omitted for brevity
echo (new ExternalResourceClient())->upload(
    $file,
    'https://httpbin.org/post',
    'POST',
    ['foo' => 'bar'], // Optional, additional form data
    ['custom-header-name' => 'custom-header-
value'] // Optional, additional headers
)->getBody()->getContents();
```

Migration

Migrating Sugar instances and external data.

Migrating From On-Site to SugarCloud

Overview

The following article describes the process of migrating an on-site deployment to the SugarCloud environment.

Prerequisites

A few requirements must be met before an instance can be migrated to the SugarCloud environment:

- The on-site deployment must be running MySQL. If you are using SQL Server or Oracle, you will need to speak with your Sugar Customer Advocate about data migration options.
- The instance must be updated to a supported version of Sugar. Refer to the Managing Your Sugar Subscription article for information on determining your Sugar version. Once identified, you can check your version against our Supported Versions.

Migration

Once the above requirements have been met you are ready to migrate.

Case Portal

Inform Sugar Support of your intention to migrate by opening a support case. They will provide you with an FTP site and a set of credentials so you can transfer your instance. Sugar Support will expect you to provide two files. One file will be an archive (zip, tar, etc.) containing all the files and folders of your Sugar instance. The second file will be an export of your SQL database; it is a good idea to archive the resulting SQL export as well.

Files and Folders

You should be aware of the location of your Sugar instance on your on-site server. If you do not, you can locate the path to your instance by navigating to Admin > Schedulers.

Once there, you should see something similar to this:

To Set Up Crontab

Note: In order to run Sugar Schedulers, add the following line to the crontab file:
* * * * * **cd <*path to sugar instance*>; php -f cron.php > /dev/null 2>&1**

Now that you have located the path to your Sugar instance, archive the entire contents of the Sugar root directory using the archive utility of your choice (zip, tar, WinZip, WinRar, 7zip, etc.). The Sugar root directory is the directory that contains the files config.php, cron.php, sugarcrm.log, and the folders custom, cache, and modules among others.

Database

The following describes how to export a MySQL database using the command line utility "mysqldump". If you prefer, you may choose to use a tool such as phpMyAdmin to export your database. The command to export a MySQL database is:

```
  mysqldump -h localhost -u [MySQL user, e.g. root] -p[database passwo
rd] [name of the database] > backup.sql
```

If you do not know the host, username, password, or database name you may refer to the "config.php" file of your Sugar instance. The "dbconfig" array in the "config.php" file contains all the required information. The example above showed the following "dbconfig" array:

```
  'dbconfig' =>  array (
    'db_host_name' => 'localhost',
    'db_host_instance' => 'SQLEXPRESS',
    'db_user_name' => 'sugarcrm',
    'db_password' => 'MyP@ssword',
    'db_name' => 'sugarcrm',
    'db_type' => 'mysql',
),
```

Using this information we can rewrite the command:

```
  mysqldump -h localhost -u sugarcrm -pMyP@ssword sugarcrm > backup.sq
l
```

Upload

Finally, upload the two files to the FTP site provided by the Sugar Support team. The instance will be deployed to the SugarCloud environment and a URL to the instance will be provided to you.

Migrating From SugarCloud to On-Site

Overview

Occasionally, system administrators will have the need to deploy versions of their instance hosted on Sugar's cloud service to an on-site system. Reasons for this type of deployment are:

- Testing locally-developed modules
- Migrating from SugarCloud to on-site

Prerequisites

- Before migrating Sugar to an on-site environment, you will need to request a backup of your SugarCloud system by filing a case with the Sugar Support team. Once the backup request is received, SugarCRM will provide an FTP account where the following files can be downloaded:
    - instance_name-YYYYMMDDHHmm.sql.gz (backup of database)
    - instance_name-YYYYMMDDHHmm-triggers.sql.gz (backup of

database triggers)
    - instance_name-YYYYMMDDHHmm.files.tgz (backup of file system)
- The local system must be running MySQL. Converting the database to another system, such as SQL Server or Oracle, requires special handling. For more information regarding this type of conversion, please contact your Account Manager.
- On-site system administrators must be familiar with their stack and the tools (gunzip, tar, mysqladmin, mysql, etc.) referenced in this guide.

Note: It is the system administrator's responsibility to diagnose and troubleshoot issues specific to the stack (permissions, environment variables, etc.).

Steps to Complete

Deploy the backup files to a local system using the following steps:

1. Extract and import the SQL data as follows:
    - Extract the SQL file via an archive utility (e.g. 7-Zip) or via command line such as:

      ```
      gunzip instance_name-YYYYMMDDHHmm.sql.gz
      ```

    - Create a database on your MySQL server via command line:

      ```
      mysqladmin -u mysql_username -p create instance_name
      ```

      Or, if already logged into MySQL, with a command such as:

      ```
      CREATE DATABASE instance_name;
      ```

    - Import the SQL data to your MySQL server via the command line:

      ```
      mysql -u mysql_username database_name -p < instance_name-
      YYYYMMDDHHmm.sql
      ```

    - Extract the Triggers file via an archive utility (e.g. 7-Zip) or via command line such as:

      ```
      gunzip instance_name-YYYYMMDDHHmm.triggers.sql.gz
      ```

- Import the Triggers data to your MySQL server via the command line:

```
mysql -u mysql_username database_name -p < instance_name-YYYYMMDDHHmm.triggers.sql
```

2. Extract the tar file to your web server's web root directory (e.g. /var/www/html) with the following command:

```
tar -C /var/www/html -xzf instance_name-YYYYMMDDHHmm.files.tgz
```

   This will create a directory named "instance_name-YYYYMMDDHHmm" in your web root directory.
3. Rename the newly created directory:

```
mv /var/www/html/instance_name-YYYYMMDDHHmm /var/www/html/instance_name
```

4. For Linux-based servers, perform the following actions:
   - Change the ownership of the directory to be accessible by the Apache user and group. Please note that the user and group (e.g. apache, www-data, etc.) values can vary depending on your web server configuration.

```
chown -R apache:apache /var/www/html/instance_name
```

   - Change the permissions of the directory to ensure files can be accessed by the application. The actual permission values may differ depending on server security settings.

```
chmod -R 770 /var/www/html/instance_name
```

5. Edit the ./config.php file to point to your database.
   - Open the ./config.php file:

```
vi /var/www/html/instance_name/config.php
```

   - Locate and update the dbconfig array with the information appropriate for your MySQL server as follows:

```
'dbconfig' => array (
 'db_host_name' => 'localhost',
 'db_host_instance' => 'SQLEXPRESS',
 'db_user_name' => 'mysql_username',
 'db_password' => 'mysql_password',
 'db_name' => 'instance_name',
 'db_type' => 'mysql',
 'db_port' => '',
 'db_manager' => 'MysqliManager',
),
```

6. Edit the config.php file and modify the following parameters:
    - site_url should be the URL of the instance on your server (e.g. https://www.mysugarinstance.com)
    - host_name should be the URL of the instance without the https protocol (e.g. www.mysugarinstance.com)
7. Modify the .htaccess file in the root directory of the instance.
    - Remove the following lines if they exist in the file:

      ```
      #Added by operations to force SSL
      RewriteEngine On
      RewriteCond %{QUERY_STRING} !^entryPoint=WebToLeadCapture
      RewriteCond %{HTTP:X-SSL-CLUSTER} !^od2$
      RewriteRule ^(.*)$ https://%{HTTP_HOST}%{REQUEST_URI} [R=3
      01,L]
      ```

    - If your on-site deployment does not reside in the root of your domain (e.g. https://www.example.com/mysugar/), change the following line from:

      ```
      RewriteBase /
      ```

      to:

      ```
      RewriteBase /mysugar/
      ```

8. If you are migrating permanently to an on-site deployment, there are additional modifications that should be made to ensure full functionality for your instance of Sugar.

    - To restore the ability to perform upgrades, open the ./config.php file and remove the following line:

      ```
      'disable_uw_upload' => true,
      ```

- To display the full-text search configuration options under Admin > Search, open the ./config.php file and remove the following line:

  ```
  'hide_full_text_engine_config' => true,
  ```

- To display the Sugar log settings under Admin > System Settings, open the ./config.php file and remove the following line:

  ```
  'logger_visible' => false,
  ```

- To bypass security checks when installing packages via Admin > Module Loader, open the ./config.php file and change the following line from:

  ```
  'packageScan' => true,
  ```

  to:

  ```
  'packageScan' => false,
  ```

- To ensure full-text search functions properly, open the ./config.php file and modify the following lines to point to your ElasticSearch configuration:

  ```
  'full_text_engine' => array (
    'Elastic' => array (
     'host' => '<your_elastic_search_host>',
     'port' => '<your_elastic_search_port>',
    ),
   ),
  ```

9. Disable the SugarCloud Insights service as this is only available for SugarCloud instances. Open the ./config.php file and change 'enabled' => true, to 'enabled' => false, in the following array:

   ```
   'cloud_insight' =>
     array (
       'enabled' => true,
       'url' => 'https://sugarcloud-insights.service.sugarcrm.com',
       'key' => '<insights key>',
     ),
   ```

10. Finally, please navigate to Admin > Repair and perform a "Quick Repair and Rebuild" to clear all cached elements from the SugarCloud instance.

You should now have a working version of your SugarCloud instance accessible from your local server.

Migrating From SugarCloud to On-Site

Overview

Occasionally, system administrators will have the need to deploy versions of their SugarCloud instance to an on-site system. Reasons for this type of deployment include:

- Testing locally developed modules
- Migrating from SugarCloud to on-site

Prerequisites

- Before migrating Sugar to an on-site environment, you will need to request a backup of your SugarCloud system by filing a case with the Sugar Support team. Once the backup request is received, SugarCRM will provide an FTP account where the following files can be downloaded:
    - instance_name-YYYYMMDDHHmm.sql.gz (backup of the database)
    - instance_name-YYYYMMDDHHmm-triggers.sql.gz (backup of database triggers)
    - instance_name-YYYYMMDDHHmm.files.tgz (backup of the file system)
- The local system must be running MySQL. Converting the database to another system, such as SQL Server or Oracle, requires special handling. For more information regarding this type of conversion, please contact your Account Manager.
- On-site system administrators must be familiar with their stack and the tools (gunzip, tar, mysqladmin, mysql, etc.) referenced in this guide.

Note: It is the system administrator's responsibility to diagnose and troubleshoot issues specific to the stack (permissions, environment variables, etc.).

Steps to Complete

Deploy the backup files to a local system using the following steps:

1. Extract and import the SQL data as follows:
    - Extract the SQL file via an archive utility (e.g. 7-Zip) or via command line such as:

      ```
      gunzip instance_name-YYYYMMDDHHmm.sql.gz
      ```

    - Create a database on your MySQL server via command line:

      ```
      mysqladmin -u mysql_username -p create instance_name
      ```

      Or, if already logged into MySQL, with a command such as:

      ```
      CREATE DATABASE instance_name;
      ```

    - Import the SQL data to your MySQL server via the command line:

      ```
      mysql -u mysql_username database_name -p < instance_name-
      YYYYMMDDHHmm.sql
      ```

    - Extract the Triggers file via an archive utility (e.g. 7-Zip) or via command line such as:

      ```
      gunzip instance_name-YYYYMMDDHHmm.triggers.sql.gz
      ```

    - Import the Triggers data to your MySQL server via the command line:

      ```
      mysql -u mysql_username database_name -p < instance_name-
      YYYYMMDDHHmm.triggers.sql
      ```

2. Extract the tar file to your web server's web root directory (e.g. /var/www/html) with the following command:

   ```
   tar -C /var/www/html -xzf instance_name-YYYYMMDDHHmm.files.tgz
   ```

   This will create a directory named "instance_name-YYYYMMDDHHmm" in your web root directory.
3. Rename the newly created directory:

```
mv /var/www/html/instance_name-
YYYYMMDDHHmm /var/www/html/instance_name
```

4. For Linux-based servers, perform the following actions:
    - Change the ownership of the directory to be accessible by the Apache user and group. Please note that the user and group (e.g. apache, www-data, etc.) values can vary depending on your web server configuration.

      ```
      chown -R apache:apache /var/www/html/instance_name
      ```

    - Change the permissions of the directory to ensure files can be accessed by the application. The actual permission values may differ depending on server security settings.

      ```
      chmod -R 770 /var/www/html/instance_name
      ```

5. Edit the config.php file to point to your database.
    - Open the config.php file:

      ```
      vi /var/www/html/instance_name/config.php
      ```

    - Locate and update the dbconfig array with the information appropriate for your MySQL server as follows:

      ```
      'dbconfig' =>
       array (
         'db_host_name' => 'localhost',
         'db_host_instance' => 'SQLEXPRESS',
         'db_user_name' => 'mysql_username',
         'db_password' => 'mysql_password',
         'db_name' => 'instance_name',
         'db_type' => 'mysql',
         'db_port' => '',
         'db_manager' => 'MysqliManager',
       ),
      ```

6. Edit the config.php file and modify the following parameters:
    - site_url should be the URL of the instance on your server (e.g. https://www.mysugarinstance.com)
    - host_name should be the URL of the instance without the https protocol (e.g. www.mysugarinstance.com)

7. Modify the .htaccess file in the root directory of the instance.
    - Remove the following lines if they exist in the file:

```
#Added by operations to force SSL
RewriteEngine On
RewriteCond %{QUERY_STRING} !^entryPoint=WebToLeadCapture
RewriteCond %{HTTP:X-SSL-CLUSTER} !^od2$
RewriteRule ^(.*)$ https://%{HTTP_HOST}%{REQUEST_URI} [R=3
01,L]
```

    - If your on-site deployment does not reside in the root of your domain (e.g. https://www.example.com/mysugar/), change the following line from:

```
RewriteBase /
```

      to:

```
RewriteBase /mysugar/
```

8. If you are migrating permanently to an on-site deployment, there are additional modifications that should be made to ensure full functionality for your instance of Sugar.
    - To restore the ability to perform ad-hoc file backups, open the ./config.php file and remove the following line:

```
 'hide_admin_backup' => true,
```

    - To restore the ability to perform upgrades, open the ./config.php file and remove the following line:

```
  'disable_uw_upload' => true,
```

    - To display the full-text search configuration options under Admin > Search, open the ./config.php file and remove the following line:

```
  'hide_full_text_engine_config' => true,
```

    - To display the Sugar log settings under Admin > System Settings, open the ./config.php file and remove the following line:

```
'logger_visible' => false,
```

- To bypass security checks when installing packages via Admin > Module Loader, open the ./config.php file and change the following line from:

```
'packageScan' => true,
```

to:

```
'packageScan' => false,
```

- To ensure full-text search functions properly, open the ./config.php file and modify the following lines to point to your ElasticSearch configuration:

```
'full_text_engine' =>
 array (
 'Elastic' =>
 array (
 'host' => '<your_elastic_search_host>',
 'port' => '<your_elastic_search_port>',
 ),
 ),
```

9. Finally, please navigate to Admin > Repair and perform a "Quick Repair and Rebuild" to clear all cached elements from the cloud instance.

You should now have a working version of your SugarCloud instance accessible from your local server.

Migrating From a Broken Instance to a Clean Install

Overview

The following article describes the process of migrating a potentially broken instance to a clean install.

This is beneficial in the use case of corrupted core files. This will not correct issues caused by broken customizations.

Requirements

To migrate your instance to a clean install you will need to do the following:

- Have full permissions to modify the system files.
- Identify your Sugar version.
  Note: The version of your instance can be found by clicking the Help link in Sugar's footer or navigating to the About page from your user menu.
- Once you have identified your Sugar version, you will need to download the full installer package of your Sugar version from your account's Downloads page.
- If you are on 14.1.0 Pro, you must download the SugarPro-14.1.0.zip package or the migration will not work.

Migrating to a New Instance

1. Make a complete backup of your broken instance. This includes both the filesystem and database.
2. Extract the contents of your downloaded Sugar package (Sugar<Edition>-<Version>.zip) to your web directory. This will be your clean Sugar directory that we will migrate your existing instance to.
3. You will then need to copy the following directories and files from your broken instance to the clean instance:
    - ./config.php
    - ./config_override.php (if it exists)
    - ./custom/
    - ./upload/
    - ./cache/images/ - This is optional and contains the embedded emails displayed in the UI.
    - **Note**: If you are on a version prior to 6.4.0, you will also need to copy over the entire ./cache/ directory.
4. Next, you will need to identify if you have any custom modules. These folders will reside in your ./modules/ directory and have a naming format of <key>_<module name>. You will need to copy these files to their corresponding directory in the clean instance you extracted in step 2.
5. Once the files have been moved to the clean instance, you can remove your old instances files and move the clean instance in its place. If you choose to move the instance to a new location on the server, you will need to update the $sugar_config['site_url'] parameter in your ./config.php and/or ./config_override.php files.
6. Reset your filesystem permissions.
7. Log into your instance as an administrator and navigate to Admin > Repair

> Quick Repair and Rebuild.

Importing Records

Covers the Sugar structure when migrating data into the application.

Importing Email Addresses

Overview

Recommended approaches when Importing email addresses.

Importing via API

Sugar comes out of the box with an API that can be called from custom applications utilizing the REST interface. The API can be used to mass create and update records in Sugar with external data. For more information on the REST API in Sugar, please refer to the Web Services documentation.

Importing New Records

When importing new records into Sugar through the API, modules with relationships to email addresses can utilize the email1 field or the email link field to specify email addresses for a record.

Email1 Field

When using the email1 field, the default functionality is to import the email address specified as the primary address. Assuming the email address does not already exist in the database, the email address is then flagged as being valid and is not opted out. Using the **/<module> POST** endpoint, you can send the following JSON payload to create a contact record with a primary email address using the email1 field:
POST URL: http://<site url>/rest/v10/Contacts

```
  {
    "first_name":"Rob",
    "last_name":"Robertson",
    "email1":"rob.robertson@sugar.crm"
}
```

Note: For importing multiple email addresses, you will need to use the email link field described below.

Email Link Field

When using the email link field, you can specify multiple email addresses to assign to the record. You may specify the following additional information regarding each email address being added:

- **invalid_email** : Specify this email address as being invalid
- **opt_out** : Specify this email address as being opted out
- **primary_address** : Specify this email address as the primary

Using the **/<module> POST** endpoint, you can send the following JSON payload to create a contact record with multiple email addresses using the email link field:
POST URL: http://<site url>/rest/v10/Contacts

```
  {
   "first_name":"Rob",
   "last_name":"Robertson",
   "email":[
      {
         "email_address":"rob.robertson@sugar.crm",
         "primary_address":"1",
         "invalid_email":"0",
         "opt_out":"0"
      },
      {
         "email_address":"rob@sugar.crm",
         "primary_address":"0",
         "invalid_email":"0",
         "opt_out":"1"
      }
   ]
}
```

For more information on the **/<module>/:record POST** endpoint, you can refer to your instance's help documentation found at:

```
http://<site url>/rest/v10/help
```

Or you can reference the <module> POST PHP example.

Updating Existing Records

When updating existing records in Sugar through the API, modules with relationships to email addresses can also utilize the email1 field or the email link field to specify email addresses for a record.

Email1 Field

When using the email1 field, the default functionality is to replace the existing email primary address. Assuming the email does not already exist in the database, the new email address is flagged as being valid and is not opted out. Using the **/<module>/:record PUT** endpoint, you can send the following JSON payload to update a contact records primary email address:

PUT URL: http://<site url>/rest/v10/Contacts/<record id>

```
  {
    "email1":"rob.robertson@sugar.crm"
}
```

Note: This will replace the current email address on the record with the new data. The old email address will no longer be associated with the record.

Email Link Field

When using the email link field, you can specify multiple email addresses to update the record with. You may specify the following additional information regarding each email address being added:

- **invalid_email** : Specify this email address as being invalid
- **opt_out** : Specify this email address as being opted out
- **primary_address** : Specify this email address as the primary

Using the **/<module>/:record PUT** endpoint, you can send the following JSON payload to update a contact record with multiple email addresses:
PUT URL: http://<site url>/rest/v10/Contacts/<record id>

```
  {
   "email":[
      {
         "email_address":"rob.robertson@sugar.crm",
         "primary_address":"1",
         "invalid_email":"0",
         "opt_out":"0"
```

```
      },
      {
         "email_address":"rob@sugar.crm",
         "primary_address":"0",
         "invalid_email":"0",
         "opt_out":"1"
      }
   ]
}
```

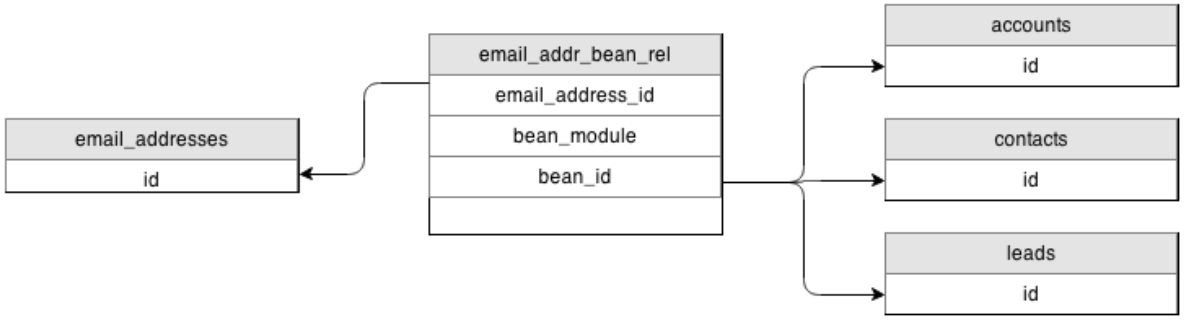
For more information on the **/<module>/:record PUT** endpoint, you can refer to your instance's help documentation found at:

```
http://<site url>/rest/v10/help
```

Or you can reference the <module>/:record PUT PHP example.

Importing via Database

When importing records into Sugar directly via the database, it is important that you understand the data structure involved before loading data. Email addresses are not stored directly on the table for the module being imported in, but are related via the email_addr_bean_rel table.



The table structure for email addresses can be seen from the database via the following SQL statement:

```
  SELECT
 email_addr_bean_rel.bean_id,
 email_addr_bean_rel.bean_module,
 email_addresses.email_address
```

```
FROM email_addr_bean_rel
INNER JOIN email_addresses
 ON email_addresses.id = email_addr_bean_rel.email_address_id
 AND email_addr_bean_rel.deleted = 0
WHERE email_addresses.deleted = 0;
```

Checking for Duplicates

Email addresses can become duplicated in Sugar from a variety of sources including API calls, imports, and from data entry. There are a few ways to have the system check for duplicate contact records, but not many of those methods work for checking email addresses for duplicates. The following section will demonstrate how to find and clean up duplicate email addresses using SQL.

The following SQL query can be used to locate if any email addresses are being used against more than one bean record within Sugar:

```
  SELECT
 email_addresses.email_address,
 COUNT(*) AS email_address_count
FROM email_addr_bean_rel
INNER JOIN email_addresses
 ON email_addresses.id = email_addr_bean_rel.email_address_id
 AND email_addr_bean_rel.deleted = 0
WHERE email_addresses.deleted = 0
GROUP BY email_addresses.email_address
HAVING COUNT(*) > 1;
```

Note: If you convert a Lead record to a Contact record, both the Lead and the Contact will be related to the same Email Address and will return as having duplicates in this query. You can add the following line to the WHERE clause to filter the duplicate check down to only one bean type:

```
  AND email_addr_bean_rel.bean_module = 'Contacts'
```

Email addresses can not only be duplicated in the system but can occasionally be missing critical data. Each bean record with an email address assigned to it should have an email address designated the primary. The following query will locate any bean records that have at least one email address, where there is not an email address designated as the primary:

```
  SELECT
 email_addr_bean_rel.bean_module,
 email_addr_bean_rel.bean_id,
 COUNT(*) AS email_count,
 COUNT(primary_email_addr_bean_rel.id) AS primary_email_count
FROM email_addr_bean_rel
LEFT JOIN email_addr_bean_rel primary_email_addr_bean_rel
 ON primary_email_addr_bean_rel.bean_module = email_addr_bean_rel.bean
_module
 AND primary_email_addr_bean_rel.bean_id = email_addr_bean_rel.bean_id
 AND primary_email_addr_bean_rel.primary_address = '1'
 AND primary_email_addr_bean_rel.deleted = '0'
WHERE email_addr_bean_rel.deleted = '0'
GROUP BY email_addr_bean_rel.bean_module,
 email_addr_bean_rel.bean_id
HAVING primary_email_count < 1;
```

Removing Duplicates

If it is determined you have duplicate email addresses being used in your system, you can use the following query to clean up the records:

```
  START TRANSACTION;
CREATE
 TABLE email_addr_bean_rel_tmp
SELECT
 *
FROM email_addr_bean_rel
WHERE deleted = '0'
GROUP BY email_address_id,
 bean_module,
 bean_id
ORDER BY primary_address DESC;
TRUNCATE TABLE email_addr_bean_rel;
INSERT INTO email_addr_bean_rel
 SELECT
 *
 FROM email_addr_bean_rel_tmp;
SELECT
 COUNT(*) AS repetitions,
 date_modified,
 bean_id,
 bean_module
FROM email_addr_bean_rel
```

```
WHERE deleted = '0'
GROUP BY bean_id,
 bean_module,
 email_address_id
HAVING repetitions > 1;
COMMIT;
```

Security

Security documents for securing Sugars infastructure.

Endpoints and Entry Points

Overview

This document describes how to disable out of the box REST API endpoints and legacy MVC entry points.

Advisory ID: sugarcrm-sr-2015-001
Revision: 1.1
Last updated: 2015-10-01

Description

SugarCRM has both legacy entry points and REST API endpoints which are shipped out of the box. Not every customer uses all capabilities of the SugarCRM product. To harden the configuration both entry points can be disabled based on the customer's requirements.

Legacy Entry Points

All stock entry points are defined in include/MVC/Controller/entry_point_registry.php. Using the SugarCRM Extension framework the configuration directives can be overridden in an upgrade safe manner. As an example consider the entrypoint "removeme". To disable this entrypoint use the following procedure.

Create a new php file ./custom/Extension/application/Ext/EntryPointRegistry/disable_removeme.php and add the following content:

```
  <?php
```

```
if (isset($entry_point_registry['removeme'])) {
    unset($entry_point_registry['removeme']);
}
```

Execute a quick repair and rebuild as SugarCRM administrator. The entry point is now fully disabled and no longer accessible to respond to any calls. Note that when trying to hit an non-existing (or disabled) entry point, the application will redirect you to the homepage (or login screen if the user has no session).

REST API Endpoints

To disable the HelpAPI which is located at clients/base/api/HelpApi.php use the following procedure.

Create a new php file custom/clients/base/api/CustomHelpApi.php and add the following content:

```
  <?php

require_once 'clients/base/api/HelpApi.php';

class CustomHelpApi extends HelpApi
{
    public function getHelp($api, $args)
    {
        throw new SugarApiExceptionNotFound();
    }
}
```

Execute a quick repair and rebuild as SugarCRM administrator. The entry point is now fully disabled. When making a REST API call to /rest/v10/help the followingHTTP 404 Not Found error will be returned:

```
  {
    "error": "not_found",
    "error_message": "Your requested resource was not found."
}
```

Publication History

2015-10-01	Adding REST API endpoint example
2015-06-16	Initial publication

A stand-alone copy of this document that omits the distribution URL is an uncontrolled copy, and may lack important information or contain factual errors. SugarCRM reserves the right to change or update this document at any time.

Web Server Configuration

Overview

This document serves as a guideline to harden the web server configuration with regard to running SugarCRM. Note that this is a guideline and certain settings will depend on your specific environment and setup. This guideline focuses on Apache web server as this is SugarCRM's primary supported web server. However, the recommendations in this document apply to all web servers in general.

Advisory ID: sugarcrm-sr-2015-003
Revision: 1.1
Last updated: 2015-10-01

SugarCRM .htaccess file

During installation, SugarCRM deploys an .htaccess file in SugarCRM's root folder. The content of this file may change during upgrades as well. The primary configuration directives managed by SugarCRM are focused around disabling direct access to certain directories and files. Secondly, the configuration contains specific URL rewrite rules which are required for SugarCRM.

Although this file can be used to ship most of the settings which are explained in this document to harden the web server, SugarCRM has chosen not to do so as most of them depend on the customer's setup which cannot be fully controlled by the SugarCRM application itself.

All directives which are managed by SugarCRM are placed between the following markers inside the .htaccess file:

```
    # BEGIN SUGARCRM RESTRICTIONS
RedirectMatch 403 .*\.log$
...
# END SUGARCRM RESTRICTIONS
```

Customers can add additional directives if need. Make sure to put those directives outside of the above-mentioned markers. SugarCRM can rebuild during upgrades all directives enclosed in the above markers.

Securing .htaccess

Apache allows admins to drop in .htaccess files in any given directory. When a request comes in, Apache will scan the directory recursively for any .htaccess file and apply its configuration directives. This ability is orchestrated by the AllowOverride directive which is enabled by default on most Apache configurations.

To disable this dynamic Apache configuration behavior, disable AllowOverride and make sure to copy paste the .htaccess file content as well.

```
    <Directory "/var/www/html">
    AllowOverride None
    # BEGIN SUGARCRM RESTRICTIONS
    RedirectMatch 403 .*\.log$
    ...
    # END SUGARCRM RESTRICTIONS
</Directory>
```

When choosing for this security measure, make sure to verify if any changes have happened in the .htaccess after an upgrade. Those changes need to be applied manually back in the Apache configuration to ensure SugarCRM functions properly. Disabling the .htaccess file support also has a positive impact on the performance as Apache will no longer need to scan for the presence of .htaccess files.

Although Apache does not allow to download .htaccess files, to tighten the security around .htaccess, even more, the following directives can be added:

```
    <Files ".ht*">
    Order deny,allow
    Deny from all
</Files>
```

Prevent version exposure

By default Apache web server exposes its version and OS details in most *nix distributions in both the HTTP response headers as well as on the default error pages. To prevent this behavior use the following configuration directives:

```
    ServerSignature Off
ServerTokens Prod
```

The exposure of the PHP version can be disabled in your php.ini file. When exposed an X-Powered-By response header will be added containing the PHP version information.

```
    expose_php = Off
```

Another option is to unset the header explicitly in the Apache configuration:

```
    <IfModule mod_headers.c>
    Header unset X-Powered-By
</IfModule>
```

Note that unsetting the Server header is not possible using this way. The web server will always respond with a Server header containing the string "Apache". If it is desired to remove this header as well, additional modules need to be used for this (i.e. mod_security can do this).

Directory listing and index page

Make sure to configure a default index page which should only be index.php for SugarCRM. This is mostly configured but worth checking. Additional SugarCRM does not require the directory browsing support and this is recommended to be disabled. This can be accomplished by removing Options Indexes and/or completely disable the mod_autoindex module.

Optionally additional Allow/Deny directives can be used to apply access lists to certain directories. It is not recommended to add additional authentication through Apache as SugarCRM already fully manages user authentication. The Allow/Deny directives can still be used to apply any IP access list if required by the customer.

The following example secures the root directory (default configuration), disable directory browsing and an additional IP access list is applied.

```
    DocumentRoot "/var/www/html"

<Directory />
    AllowOverride none
    Order allow,deny
    Deny from all
</Directory>

<Directory "/var/www/html">
    DirectoryIndex index.php
    Options -Indexes
    Order deny,allow
    Allow from 10.0.0.0/8
    Deny from all
</Directory>
```

Additional Options directives

SugarCRM does not rely on both CGI and SSI and it is recommended to disable this functionality explicitly. Optionally the FollowSymLinks options can also be disabled depending on your directory setup.

```
    <Directory "/var/www/html">
    Options -ExecCGI -Includes -FollowSymLinks
</Directory>
```

Web server permissions

Ensure your web server is configured to run as a dedicated non-privileged user. Certain *nix distributions use out of the box the user/group nobody. It is recommended to use a dedicated user/group instead.

```
    User wwwrun
Group www
```

Ensure the configured user/group have the proper file and directory permissions for SugarCRM. See the installation/upgrade guide for more detailed instructions regarding the required permissions.

Disable unnecessary loadable modules

Apache is pluggable by means of loadable modules. To minimize the chances of becoming a victim of an attack it is recommended to disable all loadable modules which are not needed. SugarCRM recommends the following base Apache modules:

- mod_authz_host
- mod_dir
- mod_expires
- mod_rewrite
- mod_headers
- mod_mime
- mod_alias

From an Apache perspective to disable a loadable module, it is sufficient to comment out the LoadModule directives. However certain *nix distribution has additional configuration scripts to manage the loaded Apache modules. Please refer to your *nix distribution for this.

Based on your setup additional Apache modules can be used. When using MPM prefork you will also need mod_php5 and when terminating secure HTTP connection on the web server directly also mod_ssl (see below).

HTTP methods

HTTP uses different methods (verbs) in requests. As per RFC-7231 for HTTP/1.1 different request methods are defined which serve the primary source of the request semantics. The following methods are used by SugarCRM:

For new REST API:

- GET
- POST
- PUT

- DELETE
- HEAD (*)
- OPTIONS (*)

Bootstrap, BWC access, old SOAP/REST API:

- GET
- POST

(*) Currently not in use, but may be implemented in the future.

The new REST API is accessed using the URL "/rest/vxx/..." where xxx represents the API version number. To harden the usage of the additional HTTP methods the following directives can be used for the new REST API endpoint while only allowing GET and POST on the legacy access.

```
    <Directory "/var/www/html">
    <LimitExcept GET POST>
        Order deny,allow
        Deny from all
    </LimitExcept>
</Directory>

<Location "/rest">
    <LimitExcept GET POST PUT DELETE>
        Order deny,allow
        Deny from all
    </LimitExcept>
</Location>

<Location "/api/rest.php">
    <LimitExcept GET POST PUT DELETE>
        Order deny,allow
        Deny from all
    </LimitExcept>
</Location>
```

Instead of using the Limit directive, an alternative is using mod_allowmethods which is available since Apache 2.4. However this module is flagged as experimental.

Secure HTTP traffic

SugarCRM requires the usage of HTTPS transport without any exception. Depending on your setup you can terminate the secure connection on a separate endpoint (load balancer, dedicated HTTPS termination point) or directly on the web server(s). You will also need a valid private key and X.509 certificate. In most cases, the customer needs to acquire an X.509 certificate through a public certification authority of their choice. For internal deployments, this may not be necessary when a local PKI environment is available.

When terminating the secure connection directly on the Apache web server, mod_ssl is required. A dedicated VirtualHost needs to be configured on port 443 (standard HTTPS port). The following is a basic SSL/TLS configuration as a reference. More details are covered in the next paragraphs.

```
    <VirtualHost 1.2.3.4:443>
    SSLEngine on
    SSLCertificateFile /etc/pki/tls/certs/example.com.crt
    SSLCertificateKeyFile /etc/pki/tls/certs/example.com.key
    SSLCertificateChainFile /etc/pki/tls/certs/example_bundle.crt
    ServerName crm.example.com
</VirtualHost>
```

Make sure to have your server name properly configured as it should match the X.509 certificate's CommonName (or one of the SNA's, subject alternative names). More information regarding the certificate requests including the certificate chain can be obtained from your certification authority.

Ensure to redirect all non-secure HTTP (port 80) traffic to the secure virtual host.

```
    <VirtualHost *:80>
    Redirect "/" "https://crm.example.com/"
</VirtualHost >
```

Cipher suite hardening

As per the publish date of this document, the usage of any version of SSL protocol, as well as TLSv1.0 and TLSv1.1 are recommended to be disabled. The current safe secure transport protocols are TLSv1.2 and TLSv1.3 with properly configured cipher suites.

For proper deployment of Diffie-Hellman (DH) key exchange algorithms the

following guidelines apply:

- No longer use out-dated export cipher suites
- Ensure the usage of Elliptic Curve Diffie-Hellman (ECDHE) based algorithms
- Use of non-common Diffie-Hellman groups

More detail around the DH configuration and a detailed list of recommended cipher suites including web server configuration examples can be found at https://weakdh.org/sysadmin.html. Additionally, an online SSL/TLS checker can be used to generate a full report of your current configuration in case your web server is publically reachable. A recommended tool is https://www.ssllabs.com/ssltest/analyze.html. The output of this SSL/TLS test suite will guide you into further (optional) details which can be tweaked regarding your configuration (i.e. OCSP stapling, certificate pinning, …).

Please note that using the above instruction "perfect forward secrecy" will be enabled. This basically means that captured TLS traffic can only be decoded using the session key between a specific client and server. Nonperfect forwarding secrecy connections can always be decrypted by having access to only the server's private key. If the customer's setup has intermediate IDS/WAF systems to inspect traffic looking inside the secure traffic, having perfect forwarding secrecy configured may not be desirable.

Public trusted root CA's

It is recommended that the *nix system where Apache is running always has the latest public root CA (certification authority) files available to authenticate server certificates during the SSL/TLS handshake. Most *nix distributions take care of this through their online update process. If this is not the case, the system administrator is responsible to ensure the latest trusted root CA's are up-to-date. Out of the box, the SugarCRM application contacts the SugarCRM heartbeat and licensing server using a secure connection.

HSTS header

As all traffic requires HTTPS for SugarCRM, it is recommended to configure a global HSTS (HTTP Strict Transport Security) header. This is an additional opt-in security enhancement supported by most modern browsers. The HSTS will instruct the browser to always use HTTPS in the future for the given FQDN. More details regarding HSTS can be found here: https://www.owasp.org/index.php/HTTP_Strict_Transport_Security.

Before enabling the HSTS header, ensure that no other non-HTTPS website is running for the given FQDN and any subdomains.

```
    Header always set Strict-Transport-Security \
"max-age=10886400; includeSubdomains; preload"
```

Optionally when your SugarCRM application is publically reachable, you can submit your FQDN on the preload which is maintained by Chrome and included by Firefox, Safari, Internet Explorer 11 and Edge browser. Understand the consequences and requirements to end up on the preload list. Once registered, this cannot be undone easily. Submissions for the HSTS preload list can be found at https://hstspreload.appspot.com.

CSP header

A Content Security Policy header can be added to offer additional protection against XSS (cross-site scripting). Most modern browsers support CSP and will apply the limitation as directed by the CSP header. The CSP header defines the allowed sources through different directives which are allowed regarding JScript, CSS, images, fonts, form actions, plugins, … More information can be found at https://www.owasp.org/index.php/Content_Security_Policy.

The management of CSP headers can become complex and may depend on different types of customizations which can be deployed on the SugarCRM platform.

For now, CSP headers can be configured directly on the web server. A basic configuration looks like this:

```
    Header always set Content-Security-Policy \
"default-src 'self'; script-src 'self'"
```

Note: This is a syntax example of a CSP header and not a recommended value.

Older browsers use the header X-Content-Security-Policy or X-WebKit-CSP instead of the standardized one. The CSP header can also safely be added first in "report only" mode before deploying it in production using Content-Security-Policy-Report-Only.

Additional security headers

To prevent clickjacking an X-Frame-Options header can be added. This header can

also be used to iframe the SugarCRM application inside another web site if required. This can be better controlled with the above mentioned CSP header. Both can coexist together but it is encouraged to used the CSP approach.

```
    Header always set X-Frame-Options "SAMEORIGIN"
Header always set X-Frame-Options "ALLOW-
FROM https://someothersite.com"
```

To protect against "drive-by download" attacks, it is encouraged to disable Content-Type sniffing. This is particularly targeted at Chrome and IE.

```
    Header always set X-Content-Type-Options "nosniff"
```

Ensure XSS browser protection is enabled by adding the following header. By default, this feature should be enabled in every browser. In case it has been disabled, sending this header will re-enable it.

```
    Header always set X-XSS-Protection "1; mode=block"
```

Secure cookie settings

Ensure the following settings are applied in php.ini to ensure legacy cookies are properly protected against XSS attacks and are only transmitted over a secure HTTPS connection.

```
    session.cookie_httponly = 1
session.cookie_secure = 1
```

Additionally, the session.cookie_path can be tweaked if other applications are running on the same web server.

Security modules

Additional Apache loadable security modules exist which can optionally be installed to better protect your system from malicious requests and/or DDOS attacks. The following modules are available for Apache 2.4:

- mod_security - This module adds WAF (web application firewall) capabilities to your Apache setup. In combination with the OWASP CRS (core rule set) gives you a solid starting point for a WAF implementation.
- mod_evasive - This module helps to prevent (D)DOS attacks

Covering both modules in details is out of the scope of this Security Response. Refer to the module documentation for additional details. In the future, SugarCRM may publish specific settings and fine-tuning for both modules.

Additional resources

Generic Apache security tips
http://httpd.apache.org/docs/2.4/misc/security_tips.html

Online SSL/TLS tester
https://www.ssllabs.com/ssltest/analyze.html

Useful HTTP headers
https://www.owasp.org/index.php/List_of_useful_HTTP_headers

Publication History

2015-10-01	Adding HTTP method restrictions
2015-09-08	Initial publication

A stand-alone copy of this document that omits the distribution URL is an uncontrolled copy and may lack important information or contain factual errors. SugarCRM reserves the right to change or update this document at any time.

XSS Prevention

Overview

This document describes how to prevent cross-site scripting (XSS) attacks in Sugar customizations. XSS attacks occur when malicious entities are able to inject client-side scripts into web pages.

Best Practices

When creating custom code for Sugar, be sure to respect the following best practices. These rules serve to protect the elements of your customizations that are most susceptible to XSS vulnerabilities:

- Create safe variables when injecting HTML via Handlebars templates or JavaScript methods such as: $(selector).html(variable), $(selector).before(variable), or $(selector).after(variable).
- Avoid using triple-bracket (e.g. {{{variable}}}) syntax to allow unescaped HTML content in Handlebars templates.
- Never use input from an unknown source as it may contain unsafe data.
- Protect dynamic content, which may also contain unsafe data.

Please refer to the following sections for more information and examples that demonstrate these best practices.

Creating Safe Variables

You must always encode a variable before you inject or display it on a webpage. The piece that is not encoded should be isolated and marked as a safe string very carefully. The the following sections outline protecting your system when using JavaScript and Handlebars templates.

JavaScript

To ensure displayed JavaScript variables are safe, respect these two rules:

- If you are displaying plain text without any HTML, use .text() instead of .html() with your selectors.
- If your text contains HTML formatting, you can use .html() as long as it does not contain any dynamic data coming from a variable.

Vulnerable Setup

This is an example of bad code because it utilizes the .html() method to display a dynamic-value variable, leaving the JavaScript vulnerable to injection:

```
var inputData = $('#myInput').val(); // "<script>alert(0)</script>"
$(selector).html('This is the value: ' + inputData); // "This is the v
alue: <script>alert(0)</script>"
```

Protected Setup

This is an example of good code, which uses the safe .text() method to protect the JavaScript from potential injection:

```
var status = 'safe';
$(selector).text('This is very ' + status); // "This is very safe"
```

This is also an acceptable approach, which uses
the Handlebars.Utils.escapeExpression to safely escape the content:

```
var inputData = $('#myinput').val(); // "<script>alert(0)</script>"
var safeData = Handlebars.Utils.escapeExpression(inputData); // "&lt;s
cript&gt;alert(0)&lt;/script&gt;"
$(selector).html('This is the value: ' + safeData); // "This is the va
lue: &lt;script&gt;alert(0)&lt;/script&gt;"
```

Handlebars Templates

Handlebars, by default, handles the encoding of data passed to the template in double brackets (e.g. {{variable}}), however, it also allows you to bypass this encoding by using triple brackets (e.g. {{{variable}}}). Text passed via triple brackets will not be encoded. As a rule, do not use triple brackets. If you don't want Handlebars to encode a piece of a string, use double brackets and execute Handlebars.SafeString() in your JavaScript controller.

As an example, we will apply best practices to a complex use case with dynamic values that need to be encoded. For this example, we want to display a message similar to:

```
Congrats! You just created the record <a href="/#Accounts/abcd">ABCD</
a>
```

The values ABCD and /#Accounts/abcd are dynamic values and, therefore, must be encoded. But the message must also be displayed as HTML, so the overall element cannot be entirely encoded. To address this use case, we must properly script the JavaScript controller and the Handlebars template.

The following JavaScript encodes the message's dynamic values, marks the displayed hyperlink as a safe string, and then outputs the safe message:

```
var record = {
    id: 'abcd',
    name: 'ABCD'
```

```
};

// escape `ABCD`
var safeName = Handlebars.Utils.escapeExpression(record.name);

// escape `abcd`
var safeId = Handlebars.Utils.escapeExpression(record.id);

// Mark the link as a SafeString now that the unsafe pieces are encode
d and the content has safe HTML.
var link = new Handlebars.SafeString('<a href="/#Accounts/' + safeId +
 '">' + safeName . '</a>');

// This can be displayed with double brackets in the template because
html parts will not be encoded.
this.safeMessage = new Handlebars.SafeString('Congrats! You just creat
ed the record ' + link + '.');
```

To display the safe message in the application, use the following syntax in your Handlebars template:

```
<div class="success">{{safeMessage}}</div>
```

Please refer to the HTML Escaping documentation on the Handlebars website for more information.

Cookbook

Welcome to the SugarCRM Developer Cookbook! This library is filled with real-world code examples for common Sugar customizations and best practices. Sugar's DevClub is always cooking up new ideas, so be sure to check in often and see what's new. Get started by browsing the categories below.

New to Sugar? Sugar's On-Boarding Developer Framework guides you through the process to start developing with Sugar

---

Adding Buttons to the Application Footer

Overview

This example explains how to add additional buttons to the application footer. We will create a custom view and then append the view component to the footer-buttons layout metadata. The additional button will merely show an alert, but can be expanded on to do much more.

Steps To Complete

This tutorial requires the following steps, which are explained in the sections below:

1. Creating the Custom View
2. Appending the View to Layout Metadata

Creating the Custom View

To add a button to the application footer, you will first need to create the custom view that will contain your button and logic. To create the custom view create a folder in ./custom/clients/base/views with the name of your view, such as ./custom/clients/base/views/footer-greet-button/. Once your folder is in place, you can create the three primary files that make up your view:

1. footer-greet-button.hbs - Contains the handlebar template for your view
2. footer-greet-button.js - Contains the JavaScript controller logic
3. footer-greet-button.php - Contains the metadata your view might use. For this example, we simply use the metadata to define whether the greeting will auto close or not.

./custom/clients/base/views/footer-greet-button/footer-greet-button.hbs

```
  {{!
    Define HTML for our new button.  We will mimic the style of other
buttons
    in the footer so we remain consistent.
    Also we will only show the button if a User is logged into the sys
tem
}}

{{#if isAuthenticated}}
<button class="btn btn-invisible" type="button" data-action="greet">
    <i class="fa fa-bell"></i>
    <span class="action-label"> {{str "Greet!"}}</span>
</button>
{{/if}}
```

./custom/clients/base/views/footer-greet-button/footer-greet-button.js

```
  ({
    // tagName attribute is inherited from Backbone.js.
    // We set it to "span" instead of default "div" so that our "butto
n" element is displayed inline.
    tagName: "span",
    events: {
        //On click of our "button" element
        'click [data-action=greet]': 'greet'
    },
    isAuthenticated: false,

    _renderHtml: function() {
        this.isAuthenticated = app.api.isAuthenticated();
        this._super('_renderHtml');
    },
    greet: function(){
        // Use Sugar 7 API to pop one of our standard alert message bo
xes.
        app.alert.show('greeting-alert', {
            level: 'info',
            messages: 'Hello '+app.user.get('full_name')+'!',
            autoClose: this.meta.autoClose || false
        });
    }

})
```

./custom/clients/base/views/footer-greet-button/footer-greet-button.php

```
  <?php

$viewdefs['base']['view']['footer-greet-button'] = array(
    'autoClose' => true
);
```

Appending the View to Layout Metadata

Once the view is created, you simply need to add the view to the Footer-Buttons Layout metadata. The Footer-Buttons Layout is located in ./clients/base/layotus/footer-buttons/, so appending the view to this layout can be

done via the Extension Framework. Create a file in ./custom/Extension/application/Ext/clients/base/layouts/footer-buttons/ and the Extension Framework will append the metadata to the Footer-Buttons Layout.

./custom/Extension/application/Ext/clients/base/layouts/footer-buttons/greetingButton.php

```
  <?php

$viewdefs['base']['layout']['footer-
buttons']['components'][] = array (
    'view' => 'footer-greet-button',
);
```

Once all the files are in place, you can run a Quick Repair and Rebuild and the new button will show in the footer when logged into your system.

Adding Buttons to the Record View

Overview

This example explains how to create additional buttons on the record view and add events. We will extend and override the stock Accounts record view to add a custom button. The custom button will be called "Validate Postal Code" and ping the Zippopotamus REST service to validate the records billing state and postal code.

Steps To Complete

This tutorial requires the following steps, which are explained in the sections below:

1. Defining the Metadata
2. Adding Custom Buttons
3. Defining the Button Label
4. Extending and Overriding the Controller

Defining the Metadata

To add a button to the record view, you will first need to create the custom metadata for the view if it doesn't exist. You can easily do this by opening and saving your modules record layout in studio. Depending on your module, the path

will then be ./custom/modules/<module>/clients/base/views/record/record.php. Once your file is in place, you will need to copy the button array from ./clients/base/views/record/record.php and add it to the $viewdefs['<module>']['base']['view']['record'] portion of your metadata array. An example of the button array is shown below:

```
  'buttons' => array(
        array(
            'type' => 'button',
            'name' => 'cancel_button',
            'label' => 'LBL_CANCEL_BUTTON_LABEL',
            'css_class' => 'btn-invisible btn-link',
            'showOn' => 'edit',
        ),
        array(
            'type' => 'rowaction',
            'event' => 'button:save_button:click',
            'name' => 'save_button',
            'label' => 'LBL_SAVE_BUTTON_LABEL',
            'css_class' => 'btn btn-primary',
            'showOn' => 'edit',
            'acl_action' => 'edit',
        ),
        array(
            'type' => 'actiondropdown',
            'name' => 'main_dropdown',
            'primary' => true,
            'showOn' => 'view',
            'buttons' => array(
                array(
                    'type' => 'rowaction',
                    'event' => 'button:edit_button:click',
                    'name' => 'edit_button',
                    'label' => 'LBL_EDIT_BUTTON_LABEL',
                    'acl_action' => 'edit',
                ),
                array(
                    'type' => 'shareaction',
                    'name' => 'share',
                    'label' => 'LBL_RECORD_SHARE_BUTTON',
                    'acl_action' => 'view',
                ),
                array(
                    'type' => 'pdfaction',
                    'name' => 'download-pdf',
                    'label' => 'LBL_PDF_VIEW',
```

```
                'action' => 'download',
                'acl_action' => 'view',
            ),
            array(
                'type' => 'pdfaction',
                'name' => 'email-pdf',
                'label' => 'LBL_PDF_EMAIL',
                'action' => 'email',
                'acl_action' => 'view',
            ),
            array(
                'type' => 'divider',
            ),
            array(
                'type' => 'rowaction',
                'event' => 'button:find_duplicates_button:click',
                'name' => 'find_duplicates_button',
                'label' => 'LBL_DUP_MERGE',
                'acl_action' => 'edit',
            ),
            array(
                'type' => 'rowaction',
                'event' => 'button:duplicate_button:click',
                'name' => 'duplicate_button',
                'label' => 'LBL_DUPLICATE_BUTTON_LABEL',
                'acl_module' => $module,
            ),
            array(
                'type' => 'rowaction',
                'event' => 'button:audit_button:click',
                'name' => 'audit_button',
                'label' => 'LNK_VIEW_CHANGE_LOG',
                'acl_action' => 'view',
            ),
            array(
                'type' => 'divider',
            ),
            array(
                'type' => 'rowaction',
                'event' => 'button:delete_button:click',
                'name' => 'delete_button',
                'label' => 'LBL_DELETE_BUTTON_LABEL',
                'acl_action' => 'delete',
            ),
        ),
    ),
```

```
        array(
            'name' => 'sidebar_toggle',
            'type' => 'sidebartoggle',
        ),
    ),
```

Note: When copying this array into your metadata, you will need to replace $module with the text string of your module's name.

For standard button types, the button definitions will contain the following properties:

Property	Potential Values	Description
type	button, rowaction, shareaction, actiondropdown	The widget type
name		The name of the button
label		The label string key for the display text of the button
css_class		The CSS class to append to the button
showOn	edit, view	The ACL action of the button

For this example, we will add the custom button to the main dropdown. For actiondropdown types, there is an additional buttons array for you to specify the dropdown list of buttons. The button definitions in this array will contain the following properties:

Property	Potential Values	Description
type	button, rowaction, shareaction, actiondropdown	The widget type; Most custom buttons are 'rowaction'
event	button:button_name:click	The event name of the button
name		The name of the button
label		The label string key for the display text of the button

acl_action	edit, view	The ACL action of the button

Adding Custom Buttons

For this example, modify the accounts' metadata to add the button definition to main_dropdown:

```
  array(
    'type' => 'rowaction',
    'event' => 'button:validate_postal_code:click',
    'name' => 'validate_postal_code',
    'label' => 'LBL_VALIDATE_POSTAL_CODE',
    'acl_action' => 'view',
),
```

A full example is shown below:

./custom/modules/Accounts/clients/base/views/record/record.php

```
  <?php

$viewdefs['Accounts'] =
array (
  'base' =>
  array (
    'view' =>
    array (
      'record' =>
      array (
        'buttons' =>
        array (
          0 =>
          array (
            'type' => 'button',
            'name' => 'cancel_button',
            'label' => 'LBL_CANCEL_BUTTON_LABEL',
            'css_class' => 'btn-invisible btn-link',
            'showOn' => 'edit',
          ),
          1 =>
          array (
            'type' => 'rowaction',
            'event' => 'button:save_button:click',
```

```
    'name' => 'save_button',
    'label' => 'LBL_SAVE_BUTTON_LABEL',
    'css_class' => 'btn btn-primary',
    'showOn' => 'edit',
    'acl_action' => 'edit',
  ),
  2 =>
  array (
    'type' => 'actiondropdown',
    'name' => 'main_dropdown',
    'primary' => true,
    'showOn' => 'view',
    'buttons' =>
    array (
      0 =>
      array (
        'type' => 'rowaction',
        'event' => 'button:edit_button:click',
        'name' => 'edit_button',
        'label' => 'LBL_EDIT_BUTTON_LABEL',
        'acl_action' => 'edit',
      ),
      1 =>
      array (
        'type' => 'shareaction',
        'name' => 'share',
        'label' => 'LBL_RECORD_SHARE_BUTTON',
        'acl_action' => 'view',
      ),
      2 =>
      array (
        'type' => 'rowaction',
        'event' => 'button:validate_postal_code:click',
        'name' => 'validate_postal_code',
        'label' => 'LBL_VALIDATE_POSTAL_CODE',
        'acl_action' => 'view',
      ),
      3 =>
      array (
        'type' => 'divider',
      ),
      4 =>
      array (
        'type' => 'rowaction',
        'event' => 'button:duplicate_button:click',
        'name' => 'duplicate_button',
```

```
          'label' => 'LBL_DUPLICATE_BUTTON_LABEL',
          'acl_module' => 'Accounts',
        ),
        5 =>
        array (
          'type' => 'rowaction',
          'event' => 'button:audit_button:click',
          'name' => 'audit_button',
          'label' => 'LNK_VIEW_CHANGE_LOG',
          'acl_action' => 'view',
        ),
        6 =>
        array (
          'type' => 'divider',
        ),
        7 =>
        array (
          'type' => 'rowaction',
          'event' => 'button:delete_button:click',
          'name' => 'delete_button',
          'label' => 'LBL_DELETE_BUTTON_LABEL',
          'acl_action' => 'delete',
        ),
      ),
    ),
    3 =>
    array (
      'name' => 'sidebar_toggle',
      'type' => 'sidebartoggle',
    ),
  ),
  'panels' =>
  array (
    0 =>
    array (
      'name' => 'panel_header',
      'header' => true,
      'fields' =>
      array (
        0 =>
        array (
          'name' => 'picture',
          'type' => 'avatar',
          'width' => 42,
          'height' => 42,
          'dismiss_label' => true,
```

```
        'readonly' => true,
      ),
      1 => 'name',
      2 =>
      array (
        'name' => 'favorite',
        'label' => 'LBL_FAVORITE',
        'type' => 'favorite',
        'dismiss_label' => true,
      ),
      3 =>
      array (
        'name' => 'follow',
        'label' => 'LBL_FOLLOW',
        'type' => 'follow',
        'readonly' => true,
        'dismiss_label' => true,
      ),
    ),
  ),
  1 =>
  array (
    'name' => 'panel_body',
    'columns' => 2,
    'labelsOnTop' => true,
    'placeholders' => true,
    'fields' =>
    array (
      0 => 'website',
      1 => 'industry',
      2 => 'parent_name',
      3 => 'account_type',
      4 => 'assigned_user_name',
      5 => 'phone_office',
    ),
  ),
  2 =>
  array (
    'name' => 'panel_hidden',
    'hide' => true,
    'columns' => 2,
    'labelsOnTop' => true,
    'placeholders' => true,
    'fields' =>
    array (
      0 =>
```

```
array (
  'name' => 'fieldset_address',
  'type' => 'fieldset',
  'css_class' => 'address',
  'label' => 'LBL_BILLING_ADDRESS',
  'fields' =>
  array (
    0 =>
    array (
      'name' => 'billing_address_street',
      'css_class' => 'address_street',
      'placeholder' => 'LBL_BILLING_ADDRESS_STREET',
    ),
    1 =>
    array (
      'name' => 'billing_address_city',
      'css_class' => 'address_city',
      'placeholder' => 'LBL_BILLING_ADDRESS_CITY',
    ),
    2 =>
    array (
      'name' => 'billing_address_state',
      'css_class' => 'address_state',
      'placeholder' => 'LBL_BILLING_ADDRESS_STATE',
    ),
    3 =>
    array (
      'name' => 'billing_address_postalcode',
      'css_class' => 'address_zip',
      'placeholder' => 'LBL_BILLING_ADDRESS_POSTALCODE',
    ),
    4 =>
    array (
      'name' => 'billing_address_country',
      'css_class' => 'address_country',
      'placeholder' => 'LBL_BILLING_ADDRESS_COUNTRY',
    ),
  ),
),
1 =>
array (
  'name' => 'fieldset_shipping_address',
  'type' => 'fieldset',
  'css_class' => 'address',
  'label' => 'LBL_SHIPPING_ADDRESS',
  'fields' =>
```

```
                array (
                  0 =>
                  array (
                    'name' => 'shipping_address_street',
                    'css_class' => 'address_street',
                    'placeholder' => 'LBL_SHIPPING_ADDRESS_STREET',
                  ),
                  1 =>
                  array (
                    'name' => 'shipping_address_city',
                    'css_class' => 'address_city',
                    'placeholder' => 'LBL_SHIPPING_ADDRESS_CITY',
                  ),
                  2 =>
                  array (
                    'name' => 'shipping_address_state',
                    'css_class' => 'address_state',
                    'placeholder' => 'LBL_SHIPPING_ADDRESS_STATE',
                  ),
                  3 =>
                  array (
                    'name' => 'shipping_address_postalcode',
                    'css_class' => 'address_zip',
                    'placeholder' => 'LBL_SHIPPING_ADDRESS_POSTALCODE'
,
                  ),
                  4 =>
                  array (
                    'name' => 'shipping_address_country',
                    'css_class' => 'address_country',
                    'placeholder' => 'LBL_SHIPPING_ADDRESS_COUNTRY',
                  ),
                  5 =>
                  array (
                    'name' => 'copy',
                    'label' => 'NTC_COPY_BILLING_ADDRESS',
                    'type' => 'copy',
                    'mapping' =>
                    array (
                      'billing_address_street' => 'shipping_address_st
reet',
                      'billing_address_city' => 'shipping_address_city
',
                      'billing_address_state' => 'shipping_address_sta
te',
                      'billing_address_postalcode' => 'shipping_addres
```

```
s_postalcode',
                        'billing_address_country' => 'shipping_address_c
ountry',
                      ),
                    ),
                  ),
                ),
                2 =>
                array (
                  'name' => 'phone_alternate',
                  'label' => 'LBL_OTHER_PHONE',
                ),
                3 => 'email',
                4 => 'phone_fax',
                5 => 'campaign_name',
                6 =>
                array (
                  'name' => 'description',
                  'span' => 12,
                ),
                7 => 'sic_code',
                8 => 'ticker_symbol',
                9 => 'annual_revenue',
                10 => 'employees',
                11 => 'ownership',
                12 => 'rating',
                13 =>
                array (
                  'name' => 'date_entered_by',
                  'readonly' => true,
                  'type' => 'fieldset',
                  'label' => 'LBL_DATE_ENTERED',
                  'fields' =>
                  array (
                    0 =>
                    array (
                      'name' => 'date_entered',
                    ),
                    1 =>
                    array (
                      'type' => 'label',
                      'default_value' => 'LBL_BY',
                    ),
                    2 =>
                    array (
                      'name' => 'created_by_name',
```

```
            ),
          ),
        ),
        14 => 'team_name',
        15 =>
        array (
          'name' => 'date_modified_by',
          'readonly' => true,
          'type' => 'fieldset',
          'label' => 'LBL_DATE_MODIFIED',
          'fields' =>
          array (
            0 =>
            array (
              'name' => 'date_modified',
            ),
            1 =>
            array (
              'type' => 'label',
              'default_value' => 'LBL_BY',
            ),
            2 =>
            array (
              'name' => 'modified_by_name',
            ),
          ),
          'span' => 12,
        ),
      ),
    ),
  ),
  'templateMeta' =>
  array (
    'useTabs' => false,
    'tabDefs' =>
    array (
      'LBL_RECORD_BODY' =>
      array (
        'newTab' => false,
        'panelDefault' => 'expanded',
      ),
      'LBL_RECORD_SHOWMORE' =>
      array (
        'newTab' => false,
        'panelDefault' => 'expanded',
      ),
```

```
          ),
        ),
      ),
    ),
  ),
);
```

Defining the Button Label

Next, define the label for the button:

./custom/Extension/modules/Accounts/Ext/Language/en_us.validatePostalCode.php

```
  <?php

$mod_strings['LBL_VALIDATE_POSTAL_CODE'] = 'Validate Postal Code';
```

Extending and Overriding the Controller

Once the button has been added to the metadata, extend and override the record view controller:

./custom/modules/Accounts/clients/base/views/record/record.js

```
  ({

    extendsFrom: 'RecordView',

    zipJSON: {},

    initialize: function (options) {
        this._super('initialize', [options]);

        //add listener for custom button
        this.context.on('button:validate_postal_code:click', this.vali
date_postal_code, this);
    },

    validate_postal_code: function() {
        //example of getting field data from current record
        var AcctID = this.model.get('id');
        var currentCity = this.model.get('billing_address_city');
```

```
        var currentZip = this.model.get('billing_address_postalcode');

        //jQuery AJAX call to Zippopotamus REST API
        $.ajax({
            url: 'http://api.zippopotam.us/us/' + currentZip,
            success: function(data) {
                    this.zipJSON = data;
                    var city = this.zipJSON.places[0]['place name'];

                    if (city === currentCity)
                    {
                        app.alert.show('address-ok', {
                            level: 'success',
                            messages: 'City and Zipcode match.',
                            autoClose: true
                        });
                    }
                    else
                    {
                        app.alert.show('address-ok', {
                            level: 'error',
                            messages: 'City and Zipcode do not match.'
,
                            autoClose: false
                        });
                    }
                }
            });
    }
})
```

Once the files are in place, navigate to Admin > Repair > Quick Repair and Rebuild.

Adding Buttons without Overriding View Controllers

Sometimes your customization might need to add the same buttons to multiple views, but you don't want to overwrite other possible customizations already on the view. The following will walk through creating a custom button like the above example, add adding the buttons to views without overriding the module view controllers.

Create a Custom Button

Buttons typically come in two forms, a standard button and a 'rowaction' in an Action Menu. We can create two buttons to handle both scenarios.

./clients/base/fields/zipcode-check-button/zipcode-check-button.js

```
  /**
 * Zipcode Check button will check if zipcode matches city field
 *
 * @class View.Fields.Base.ZipcodeCheckButtonField
 * @alias SUGAR.App.view.fields.BaseZipcodeCheckButtonField
 * @extends View.Fields.Base.ButtonField
 */
({
    extendsFrom: 'ButtonField',

    defaultZipCodeField: 'billing_address_postalcode',

    defaultCityField: 'billing_address_city',

    /**
     * @inheritdoc
     */
    initialize: function(options) {
        options.def.events = _.extend({}, options.def.events, {
            'click .zip-check-btn': 'validatePostalCode'
        });

        this._super('initialize', [options]);

        this.def.zip_code_field = _.isEmpty(this.def.zip_code_field)?t
his.defaultZipCodeField:this.def.zip_code_field;
        this.def.city_field = _.isEmpty(this.def.city_field)?this.defa
ultCityField:this.def.city_field;
    },

    validatePostalCode: function(evt) {
        var currentCity = this.model.get(this.def.city_field);
        var currentZip = this.model.get(this.def.zip_code_field);

        //jQuery AJAX call to Zippopotamus REST API
        $.ajax({
            url: 'http://api.zippopotam.us/us/' + currentZip,
            success: function(data) {
                this.zipJSON = data;
                var city = this.zipJSON.places[0]['place name'];
```

```
                if (city === currentCity)
                {
                    app.alert.show('address-ok', {
                        level: 'success',
                        messages: 'City and Zipcode match.',
                        autoClose: true
                    });
                }
                else
                {
                    app.alert.show('address-ok', {
                        level: 'error',
                        messages: 'City and Zipcode do not match.',
                        autoClose: false
                    });
                }
            }
        });
    }
})
```

./clients/base/fields/zipcode-check-rowaction/zipcode-check-rowaction.js

```
  /**
 * Zipcode Check button will check if zipcode matches city field
 *
 * @class View.Fields.Base.ZipcodeCheckRowactionField
 * @alias SUGAR.App.view.fields.BaseZipcodeCheckRowactionField
 * @extends View.Fields.Base.RowactionField
 */
({
    extendsFrom: 'RowactionField',

    defaultZipCodeField: 'billing_address_postalcode',

    defaultCityField: 'billing_address_city',

    /**
     * @inheritdoc
     */
    initialize: function(options) {
        this._super('initialize', [options]);

        this.def.zip_code_field = _.isEmpty(this.def.zip_code_field)?t
```

```
his.defaultZipCodeField:this.def.zip_code_field;
        this.def.city_field = _.isEmpty(this.def.city_field)?this.defa
ultCityField:this.def.city_field;
    },

    /**
     * Rowaction fields have a default event which calls rowActionSele
ct
     */
    rowActionSelect: function(evt) {
        this.validatePostalCode(evt);
    },

    validatePostalCode: function(evt) {
        var currentCity = this.model.get(this.def.city_field);
        var currentZip = this.model.get(this.def.zip_code_field);

        //jQuery AJAX call to Zippopotamus REST API
        $.ajax({
            url: 'http://api.zippopotam.us/us/' + currentZip,
            success: function(data) {
                this.zipJSON = data;
                var city = this.zipJSON.places[0]['place name'];

                if (city === currentCity)
                {
                    app.alert.show('address-check-msg', {
                        level: 'success',
                        messages: 'City and Zipcode match.',
                        autoClose: true
                    });
                }
                else
                {
                    app.alert.show('address-check-msg', {
                        level: 'error',
                        messages: 'City and Zipcode do not match.',
                        autoClose: false
                    });
                }
            }
        });
    }
})
```

Along with the JavaScript controllers for the buttons, you can copy over the detail.hbs and edit.hbs from the base controllers that each button is extended from into each folder. The folder structure will look as follows:

./clients/base/fields/zipcode-check-button/

- zipcode-check-button.js
- detail.hbs
- edit.hbs

./clients/base/fields/zipcode-check-rowaction/

- zipcode-check-rowaction.js
- detail.hbs
- edit.hbs

In the Handlebar files, you should add a custom CSS class called zip-check-btn to each of the layouts, as a way to find the elements on the page. This is also used for the ZipcodeCheckButton to isolate the event trigger. Example below:

./cilents/base/fields/zipcode-check-button/edit.hbs

```
  <a href="{{#if fullRoute}}#{{fullRoute}}{{else}}{{#if def.route}}#{{
buildRoute context=context model=model action=def.route.action}}{{else
}}javascript:void(0);{{/if}}{{/if}}"
    class="btn{{#if def.primary}} btn-primary{{/if}} zip-check-btn"
    {{#if def.tooltip}}
        rel="tooltip"
        data-placement="bottom"
        title="{{str def.tooltip module}}"
    {{/if}}
    {{#if ariaLabel}}aria-label="{{ariaLabel}}"{{/if}}
    role="button" tabindex="{{tabIndex}}" name="{{name}}">{{#if def.ic
on}}<i class="fa {{def.icon}}"></i> {{/if}}{{label}}</a>
```

Once we have the button controllers and the templates setup, we can add the buttons to the View metadata for particular modules.

Appending Buttons to Metadata

After creating your buttons, you will need to append the buttons to the views metadata. For this, you can use the Extension Framework. The following examples will add a button to the action menu on the Accounts record view and a button to the Contacts record view. Please note that this example assumes that you have

created the rowaction button above.

./custom/Extension/modules/Accounts/Ext/clients/base/views/record/addZipCodeCheckRowaction.php

```
  $buttons = isset($viewdefs['Accounts']['base']['view']['record']['buttons'])?$viewdefs['Accounts']['base']['view']['record']['buttons']:array();

if (!empty($buttons)) {
    foreach ($buttons as $key => $button) {
        if ($button['type'] == 'actiondropdown' && $button['name'] == 'main_dropdown') {
            $viewdefs['Accounts']['base']['view']['record']['buttons'][$key]['buttons'][] = array(
                'type' => 'divider',
            );
            $viewdefs['Accounts']['base']['view']['record']['buttons'][$key]['buttons'][] = array(
                'type' => 'zipcode-check-rowaction',
                'event' => 'button:zipcode_check:click',
                'name' => 'zipcode_check_button',
                'label' => 'LBL_ZIPCODE_CHECK_BUTTON_LABEL',
                'acl_action' => 'edit',
                'showOn' => 'view',
            );
            break;
        }
    }
}
```

./custom/Extension/modules/Contacts/Ext/clients/base/views/record/addZipCodeCheckButton.php

```
  $buttons = isset($viewdefs['Contacts']['base']['view']['record']['buttons'])?$viewdefs['Contacts']['base']['view']['record']['buttons']:array();
$zipCodeButton = array (
    'type' => 'zipcode-check-button',
    'event' => 'button:zipcode_check:click',
    'name' => 'zipcode_check_button',
    'label' => 'LBL_ZIPCODE_CHECK_BUTTON_LABEL',
    'acl_action' => 'edit',
    'zip_code_field' => 'primary_address_postalcode',
```

```
    'city_field' => 'primary_address_city'
);

if (!empty($buttons)){
    foreach($buttons as $key => $button){
        if ($button['type'] == 'actiondropdown' && $button['name'] ==
'main_dropdown'){
            //Get everything from this point down
            $slicedBtns = array_slice($viewdefs['Contacts']['base']['v
iew']['record']['buttons'],$key);
            //Remove everything from this point down
            array_splice($viewdefs['Contacts']['base']['view']['record
']['buttons'],$key);
            //Add Zip Code Button
            $viewdefs['Contacts']['base']['view']['record']['buttons']
[] = $zipCodeButton;
            //Add back the buttons we removed
            foreach($slicedBtns as $oldButton){
                $viewdefs['Contacts']['base']['view']['record']['butto
ns'][] = $oldButton;
            }
            break;
        }
    }
} else {
    $viewdefs['Contacts']['base']['view']['record']['buttons'] = array
(
        $zipCodeButton
    );
}
unset($zipCodeButton);
```

The last thing that is needed now, is to define the button's label.

Defining the Button Labels

Since the button was made to work globally on multiple modules, we can define the label at the application level.

./custom/Extension/application/Ext/Language/en_us.ZipCodeCheckButton.php

```
  $app_strings['LBL_ZIPCODE_CHECK_BUTTON_LABEL'] = 'Verify Zip Code';
```

Once all files are in place, you can run a Quick Repair and Rebuild and the buttons will display and check the Zipcode fields for both the Contacts and Accounts record views without having to extend either modules RecordView controllers.

Adding Field Validation to the Record View

Overview

This page explains how to add additional field validation to the record view. In the following examples, we will extend and override the stock Accounts record view to add custom validation. The custom validation will require the Office Phone field when the account type is set to "Customer" and also require the user to enter at least one email address.

Error Messages

When throwing a validation error, Sugar has several stock messages you may choose to use. They are shown below:

Error Key	Label	Description
maxValue	ERROR_MAXVALUE	This maximum value of this field
minValue	ERROR_MINVALUE	This minimum value of this field
maxLength	ERROR_MAX_FIELD_LENGTH	The max length of this field
minLength	ERROR_MIN_FIELD_LENGTH	The min length of this field
datetime	ERROR_DATETIME	This field requires a valid date
required	ERROR_FIELD_REQUIRED	This field is required
email	ERROR_EMAIL	There is an invalid email address
primaryEmail	ERROR_PRIMARY_EMAIL	No primary email address is set
duplicateEmail	ERROR_DUPLICATE_EMAIL	There is a duplicate email address
number	ERROR_NUMBER	This field requires a valid number

isBefore	ERROR_IS_BEFORE	The date of this field can not be after another date
isAfter	ERROR_IS_AFTER	The date of this field can not be before another date

You also have the option of displaying multiple error messages at a time. The example below would throw an error message notifying the user that the selected field is required and that it is also not a number.

```
errors['<field name>'] = errors['<field name>'] || {};
errors['<field name>'].required = true;
errors['<field name>'].number = true;
```

Custom Error Messages

Custom error message can be used by appending custom language keys to app.error.errorName2Keys when initializing an extended controller. To accomplish this, create a new language key in the $app_strings. An example of this is shown below:

./custom/Extension/application/Ext/Language/en_us.error_custom_message.php

```
<?php

$app_strings['ERROR_CUSTOM_MESSAGE'] = 'My custom error message.';
```

Next, you will need to update your controller to use the new language key. To accomplish this, add your custom language key to the app.error.errorName2Keys array in the initialize method:

```
initialize: function (options) {
    this._super('initialize', [options]);

    //add custom message key
    app.error.errorName2Keys['custom_message'] = 'ERROR_CUSTOM_MESSAGE
';

    ....
},
```

Once completed, you can call your custom message by setting the app.error.errorName2Keys entry to true as shown below:

```
errors['phone_office'] = errors['phone_office'] || {};
errors['phone_office'].custom_message = true;
```

Method 1: Extending the RecordView and CreateView Controllers

One way to validate fields on record view is by creating record and create view controllers. This method requires a duplication of code due to the hierarchy design, however, it does organize the code by module and extend the core functionality. To accomplish this, override and extend the create view controller. This handles the validation when a user is creating a new record. Once the controller has been properly extended, define the validation check and use the model.addValidationTask method to append your function to the save validation.

./custom/modules/Accounts/clients/base/views/create/create.js

```
({
    extendsFrom: 'CreateView',
    initialize: function (options) {

        this._super('initialize', [options]);

        //add validation tasks
        this.model.addValidationTask('check_account_type', _.bind(this
._doValidateCheckType, this));
        this.model.addValidationTask('check_email', _.bind(this._doVal
idateEmail, this));
    },

    _doValidateCheckType: function(fields, errors, callback) {
        //validate type requirements
        if (this.model.get('account_type') == 'Customer' && _.isEmpty(
this.model.get('phone_office')))
        {
            errors['phone_office'] = errors['phone_office'] || {};
            errors['phone_office'].required = true;
        }

        callback(null, fields, errors);
    },
```

```
    _doValidateEmail: function(fields, errors, callback) {
        //validate email requirements
        if (_.isEmpty(this.model.get('email')))
        {
            errors['email'] = errors['email'] || {};
            errors['email'].required = true;
        }

        callback(null, fields, errors);
    },
})
```

Next, duplicate the validation logic for the record view. This handles the validation when editing existing records.

./custom/modules/Accounts/clients/base/views/record/record.js

```
({
    /* because 'accounts' already has a record view, we need to extend
 it */
    extendsFrom: 'AccountsRecordView',

    initialize: function (options) {

        this._super('initialize', [options]);

        //add validation tasks
        this.model.addValidationTask('check_account_type', _.bind(this
._doValidateCheckType, this));
        this.model.addValidationTask('check_email', _.bind(this._doVal
idateEmail, this));
    },

    _doValidateCheckType: function(fields, errors, callback) {

        //validate requirements
        if (this.model.get('account_type') == 'Customer' && _.isEmpty(
this.model.get('phone_office')))
        {
            errors['phone_office'] = errors['phone_office'] || {};
            errors['phone_office'].required = true;
        }
```

```
        callback(null, fields, errors);
    },

    _doValidateEmail: function(fields, errors, callback) {

        //validate email requirements
        if (_.isEmpty(this.model.get('email')))
        {
            errors['email'] = errors['email'] || {};
            errors['email'].required = true;
        }

        callback(null, fields, errors);
    },
})
```

Once the files are in place, navigate to Admin > Repair > Quick Repair and Rebuild. More information on displaying custom error messages can be found in the Error Messages section.

Method 2: Overriding the RecordView and CreateView Layouts

Another method for defining your own custom validation is to override a module's record and create layouts to append a new view with your logic. The benefits of this method are that you can use the single view to house the validation logic, however, this means that you will have to override the layout. When overriding a layout, verify that the layout has not changed when upgrading your instance. Once the layouts are overridden, define the validation check and use the model.addValidationTask method to append the function to the save validation.

First, create the custom view. For the accounts example, create the view validate-account:

./custom/modules/Accounts/clients/base/views/validate-account/validate-account.js

```
({
   className:"hidden",

   _render: function() {
       //No-op, Do nothing here
   },
```

```
    bindDataChange: function() {
        //add validation tasks
        this.model.addValidationTask('check_account_type', _.bind(this.
_doValidateCheckType, this));
        this.model.addValidationTask('check_email', _.bind(this._doVali
dateEmail, this));
    },

    _doValidateCheckType: function(fields, errors, callback) {
        //validate type requirements
        if (this.model.get('account_type') == 'Customer' && _.isEmpty(t
his.model.get('phone_office')))
        {
            errors['phone_office'] = errors['phone_office'] || {};
            errors['phone_office'].required = true;
        }

        callback(null, fields, errors);
    },

    _doValidateEmail: function(fields, errors, callback) {
        //validate email requirements
        if (_.isEmpty(this.model.get('email')))
        {
            errors['email'] = errors['email'] || {};
            errors['email'].required = true;
        }

        callback(null, fields, errors);
    },
})
```

More information on displaying custom error messages can be found in the Error Messages section. Next, depending on the selected module, duplicate its create layout to the modules custom folder to handle record creation. In our Accounts example, we have an existing ./modules/Accounts/clients/base/layouts/create/create.php file so we need to duplicate this file to ./custom/modules/Accounts/clients/base/layouts/create/create.php. After this has been completed, append the new custom view to the components. append:

```
array(
    'view' => 'validate-account',
```

```
),
```

As shown below:

./custom/modules/Accounts/clients/base/layouts/create/create.php

```
<?php

$viewdefs['Accounts']['base']['layout']['create'] = array(
    'components' =>
        array(
            array(
                'layout' =>
                    array(
                        'components' =>
                            array(
                                array(
                                    'layout' =>
                                        array(
                                            'components' =>
                                                array(
                                                    array(
                                                        'view' => 'cre
ate',
                                                    ),
                                                    array(
                                                        'view' => 'val
idate-account',
                                                    ),
                                                ),
                                            'type' => 'simple',
                                            'name' => 'main-pane',
                                            'span' => 8,
                                        ),
                                ),
                                array(
                                    'layout' =>
                                        array(
                                            'components' =>
                                                array(),
                                            'type' => 'simple',
                                            'name' => 'side-pane',
                                            'span' => 4,
                                        ),
```

```
                                ),
                                array(
                                    'layout' =>
                                        array(
                                            'components' =>
                                                array(
                                                    array(
                                                        'view' =>
                                                            array (
                                                                'name'
 => 'dnb-account-create',
                                                                'label
' => 'DNB Account Create',
                                                            ),
                                                        'width' => 12,
                                                    ),
                                            ),
                                            'type' => 'simple',
                                            'name' => 'dashboard-
pane',
                                            'span' => 4,
                                        ),
                                ),
                                array(
                                    'layout' =>
                                        array(
                                            'components' =>
                                                array(
                                                    array(
                                                        'layout' => 'p
review',
                                                    ),
                                            ),
                                            'type' => 'simple',
                                            'name' => 'preview-pane',
                                            'span' => 8,
                                        ),
                                ),
                            ),
                        'type' => 'default',
                        'name' => 'sidebar',
                        'span' => 12,
                    ),
            ),
        ),
    'type' => 'simple',
```

```
    'name' => 'base',
    'span' => 12,
);
```

Lastly, depending on the selected module, duplicate its record layout to the modules custom folder to handle editing a record. In the accounts example, we do not have an existing ./modules/Accounts/clients/base/layouts/record/record.php file so we duplicated the core ./clients/base/layouts/record/record.php to ./custom/modules/Accounts/clients/base/layouts/record/record.php. Since we are copying from the ./clients/ core directory, modify:

```
$viewdefs['base']['layout']['record'] = array(
    ...
);
```

To:

```
$viewdefs['Accounts']['base']['layout']['record'] = array(
    ...
);
```

After this has been completed, append the new custom view to the components:

```
array(
    'view' => 'validate-account',
),
```

The resulting file is shown below:

./custom/modules/Accounts/clients/base/layouts/record/record.php

```
<?php

$viewdefs['Accounts']['base']['layout']['record'] = array(
    'components' => array(
        array(
            'layout' => array(
                'components' => array(
```

```
                    array(
                        'layout' => array(
                            'components' => array(
                                array(
                                    'view' => 'record',
                                    'primary' => true,
                                ),
                                array(
                                    'view' => 'validate-account',
                                ),
                                array(
                                    'layout' => 'extra-info',
                                ),
                                array(
                                    'layout' => array(
                                        'name' => 'filterpanel',
                                        'span' => 12,
                                        'last_state' => array(
                                            'id' => 'record-
filterpanel',
                                            'defaults' => array(
                                                'toggle-
view' => 'subpanels',
                                            ),
                                        ),
                                        'availableToggles' => array(
                                            array(
                                                'name' => 'subpanels',
                                                'icon' => 'icon-
table',
                                                'label' => 'LBL_DATA_V
IEW',
                                            ),
                                            array(
                                                'name' => 'list',
                                                'icon' => 'icon-
table',
                                                'label' => 'LBL_LISTVI
EW',
                                            ),
                                            array(
                                                'name' => 'activitystr
eam',
                                                'icon' => 'icon-th-
list',
                                                'label' => 'LBL_ACTIVI
```

```
TY_STREAM',
                                        ),
                                    ),
                                    'components' => array(
                                        array(
                                            'layout' => 'filter',
                                            'targetEl' => '.filter
',
                                            'position' => 'prepend
'
                                        ),
                                        array(
                                            'view' => 'filter-
rows',
                                            "targetEl" => '.filter-
options'
                                        ),
                                        array(
                                            'view' => 'filter-
actions',
                                            "targetEl" => '.filter-
options'
                                        ),
                                        array(
                                            'layout' => 'activitys
tream',
                                            'context' =>
                                            array(
                                                'module' => 'Activ
ities',
                                            ),
                                        ),
                                        array(
                                            'layout' => 'subpanels
',
                                        ),
                                    ),
                                ),
                            ),
                        ),
                        'type' => 'simple',
                        'name' => 'main-pane',
                        'span' => 8,
                    ),
                ),
                array(
```

```
            'layout' => array(
                'components' => array(
                    array(
                        'layout' => 'sidebar',
                    ),
                ),
                'type' => 'simple',
                'name' => 'side-pane',
                'span' => 4,
            ),
        ),
        array(
            'layout' => array(
                'components' => array(
                    array(
                        'layout' => array(
                            'type' => 'dashboard',
                            'last_state' => array(
                                'id' => 'last-visit',
                            )
                        ),
                        'context' => array(
                            'forceNew' => true,
                            'module' => 'Home',
                        ),
                    ),
                ),
                'type' => 'simple',
                'name' => 'dashboard-pane',
                'span' => 4,
            ),
        ),
        array(
            'layout' => array(
                'components' => array(
                    array(
                        'layout' => 'preview',
                    ),
                ),
                'type' => 'simple',
                'name' => 'preview-pane',
                'span' => 8,
            ),
        ),
    ),
    'type' => 'default',
```

```
                'name' => 'sidebar',
                'span' => 12,
            ),
        ),
    ),
    'type' => 'simple',
    'name' => 'base',
    'span' => 12,
);
```

Once the files are in place, navigate to Admin > Repair > Quick Repair and Rebuild.

Adding an Existing Note to an Email as Attachment

Overview

There may be times when you want to reuse a file attached to one Note record as an attachment for an Email, similar to the ability in the Compose Email view to add an attachment using 'Sugar Document'.

Key Concepts

There are two key things to understand when implementing this functionality:

1. You **can not** relate an existing Note record to an Email record. This will throw an Exception in the API as intended. A note that is used as an attachment can only exist once and can not act as an attachment across multiple emails.
2. You **can** create a new Note record that uses an existing Note's attached file. Doing so essentially requires setting the upload_id field of the new Note record to the id of the existing Note you want to reuse the file from, and setting the file_source field of the new Note to 'Notes'. In addition to needing to set the upload_id, you must also set the filename and name field.

The following examples demonstrate how to do this in three different contexts: server-side (SugarBean/PHP), client-side (sidecar/Javascript), and purely through the API. But all three contexts follow the same essential steps:

1. Fetch the original Note record;

2. Create a new Note record, setting the necessary fields from the original Note record;
3. Link the new Note record to the Email record via the appropriate 'attachments' link.

PHP

This example would be done server-side using the SugarBean object, similar to how you might interact with the records in a custom Logic Hook or Scheduler job.

```
  $original_note_id = '4e226282-8158-11e8-a1b3-439fe19c087a';
$email_id = 'e3e058f4-7f11-11e8-ba11-fcdd97d61bbe';

// 1. Fetch Original Note:
$original_note = BeanFactory::retrieveBean('Notes', $original_note_id)
;

// 2.  Create a new note based on original note, setting upload_id, na
me, and filename:
$new_note = BeanFactory::newBean('Notes');
$new_note->upload_id = $original_note->getUploadId();
$new_note->file_source = 'Notes';
$new_note->filename = $original_note->filename;
$new_note->name = $original_note->filename;

// 3. Relate the new note to Email record using 'attachments' link:
$email = BeanFactory::retrieveBean('Emails', $email_id);
$email->load_relationship('attachments');
$email->attachments->add($email, $new_note);
```

Note that in most contexts, such as a Logic Hook or a custom endpoint, you will not need to call save() on either the $new_note or $email for $new_note to be related as an attachment to $email and for $new_note to save. The $new_note record will save as part of being linked to $email.

Javascript

The following example is totally standalone within Sidecar, demonstrating how to fetch the existing Note using app.data.createBean and bean.fetch(), create the new Note using app.data.createBean and setting the relevant fields, and then adding the new Note to the email records attachments_collection.

```
  var original_note_id = '4e226282-8158-11e8-a1b3-439fe19c087a';
```

```
var email_id = 'e3e058f4-7f11-11e8-ba11-fcdd97d61bbe';

// 1. Fetch Original Note:
var original_note = app.data.createBean('Notes');
original_note.set('id', original_note_id);
original_note.fetch();

// 2.  Create a new note based on original note, setting upload_id, na
me, and filename:

var new_note = app.data.createBean('Notes', {
 _link: 'attachments',
 upload_id: original_note.get('id'),
 file_source: 'Notes',
 filename: original_note.get('filename'),
 name: original_note.get('filename'),
 file_mime_type: original_note.get('file_mime_type'),
 file_size: original_note.get('file_size'),
 file_ext: original_note.get('file_ext'),
});

// 3.  Relate the new note to Email record using 'attachments_collecti
on' link:

var email = app.data.createBean('Emails');
email.set('id', email_id);
email.fetch();

email.get('attachments_collection').add(new_note,{merge:true});
email.save();
```

Note: In the above example, the fields file_mime_type, file_size, and file_ext are also set. This is assuming a context where the customization is adding the attachments to the Compose Email view. Setting these fields makes the new attachments look correct to the user before saving the Email, but these fields are otherwise set automatically on save. If extending the actual compose view for emails, you also wouldn't fetch the email directly. This is added in this example purely for demonstration purposes.

REST API

The following section will outline how to related the attachment using the REST API using the /link/attachments/ endpoint for the Email record.

Fetch Original Note

To fetch the original note, send a GET request to rest/{REST VERSION}/Notes/{$original_note_id}. An example of the response is shown below :

```
  {
  "id": "4e226282-8158-11e8-a1b3-439fe19c087a",
  "name": "Note with Attachment",
  "date_entered": "2018-07-01T12:00:00-00:00",
  "description": "Send to special clients.",
  "file_mime_type": "application/pdf",
  "filename": "special_sales_doc.pdf",
  "_module": "Notes"
}
```

Create and Relate a Note

Create a JSON object based on the response (which we will treat as a JSON object named $original_note) like:

```
  {
 "upload_id": "$original_note.id",
 "file_source" : "Notes"
 "name" : "$original_note.filename",
 "filename" : "$original_note.filename",
}
```

Relate the Note to the Email record using the 'attachments' link. Next, send a POST request to rest/{REST VERSION}/Emails/{$email_id}/link/attachments/ with the JSON shown above as the request body.

Adding the Email Field to a Bean

Overview

This example explains how to add an emails field for modules that don't extend the Person module template. We will create an email field and bring the email functionality module bean. The steps are applicable for stock and custom modules. In this example, we will add the email field into the Opportunities module.

Steps to Complete

This tutorial requires the following steps, which are explained in the sections below:

1. Creating a Custom Vardef file for email Field and Relationships
2. Creating a Custom Bean and Modify the save function
3. Replacing the stock bean with the custom bean
4. Adding Emails Field into Record View

Creating a Custom Vardef file for email Field and Relationships

You will create a file in the custom/Extension/modules/Opportunities/ folder for the field and relationships definitions.

./custom/Extension/modules/Opportunities/Ext/Vardefs/custom_email_field.php

```
  <?php

$module = 'Opportunity';
$table_name = 'opportunities';

$dictionary[$module]['fields']['email'] = array(
    'name' => 'email',
    'type' => 'email',
    'query_type' => 'default',
    'source' => 'non-db',
    'operator' => 'subquery',
    'subquery' => 'SELECT eabr.bean_id FROM email_addr_bean_rel eabr J
OIN email_addresses ea ON (ea.id = eabr.email_address_id) WHERE eabr.d
eleted=0 AND ea.email_address LIKE',
    'db_field' => array(
        'id',
    ),
    'vname' => 'LBL_EMAIL_ADDRESS',
    'studio' => array(
        'visible' => true,
        'searchview' => true,
        'editview' => true,
        'editField' => true,
    ),
    'duplicate_on_record_copy' => 'always',
    'len' => 100,
    'link' => 'email_addresses_primary',
    'rname' => 'email_address',
```

```
    'module' => 'EmailAddresses',
    'full_text_search' => array(
        'enabled' => true,
        'searchable' => true,
        'boost' => 1.50,
    ),
    'audited' => true,
    'pii' => true,
);

$dictionary[$module]['fields']['email1'] = array(
    'name' => 'email1',
    'vname' => 'LBL_EMAIL_ADDRESS',
    'type' => 'varchar',
    'function' => array(
        'name' => 'getEmailAddressWidget',
        'returns' => 'html',
    ),
    'source' => 'non-db',
    'link' => 'email_addresses_primary',
    'rname' => 'email_address',
    'group' => 'email1',
    'merge_filter' => 'enabled',
    'module' => 'EmailAddresses',
    'studio' => false,
    'duplicate_on_record_copy' => 'always',
    'importable' => false,
);

$dictionary[$module]['fields']['email2'] = array(
    'name' => 'email2',
    'vname' => 'LBL_OTHER_EMAIL_ADDRESS',
    'type' => 'varchar',
    'function' => array(
        'name' => 'getEmailAddressWidget',
        'returns' => 'html',
    ),
    'source' => 'non-db',
    'group' => 'email2',
    'merge_filter' => 'enabled',
    'studio' => 'false',
    'duplicate_on_record_copy' => 'always',
    'importable' => false,
    'workflow' => false,
);
```

```
$dictionary[$module]['fields']['invalid_email'] = array(
    'name' => 'invalid_email',
    'vname' => 'LBL_INVALID_EMAIL',
    'source' => 'non-db',
    'type' => 'bool',
    'link' => 'email_addresses_primary',
    'rname' => 'invalid_email',
    'massupdate' => false,
    'studio' => 'false',
    'duplicate_on_record_copy' => 'always',
);

$dictionary[$module]['fields']['email_opt_out'] = array(
    'name' => 'email_opt_out',
    'vname' => 'LBL_EMAIL_OPT_OUT',
    'source' => 'non-db',
    'type' => 'bool',
    'link' => 'email_addresses_primary',
    'rname' => 'opt_out',
    'massupdate' => false,
    'studio' => 'false',
    'duplicate_on_record_copy' => 'always',
);

$dictionary[$module]['fields']['email_addresses_primary'] = array(
    'name' => 'email_addresses_primary',
    'type' => 'link',
    'relationship' => strtolower($table_name) . '_email_addresses_prim
ary',
    'source' => 'non-db',
    'vname' => 'LBL_EMAIL_ADDRESS_PRIMARY',
    'duplicate_merge' => 'disabled',
    'primary_only' => true,
);

$dictionary[$module]['fields']['email_addresses'] = array(
    'name' => 'email_addresses',
    'type' => 'link',
    'relationship' => strtolower($table_name) . '_email_addresses',
    'source' => 'non-db',
    'vname' => 'LBL_EMAIL_ADDRESSES',
    'reportable' => false,
    'unified_search' => true,
    'rel_fields' => array('primary_address' => array('type' => 'bool')
),
);
```

```
// Used for non-primary mail import
$dictionary[$module]['fields']['email_addresses_non_primary'] = array(
    'name' => 'email_addresses_non_primary',
    'type' => 'varchar',
    'source' => 'non-db',
    'vname' => 'LBL_EMAIL_NON_PRIMARY',
    'studio' => false,
    'reportable' => false,
    'massupdate' => false,
);

$dictionary[$module]['relationships'][strtolower($table_name) . '_emai
l_addresses'] = array(
    'lhs_module' => $table_name,
    'lhs_table' => strtolower($table_name),
    'lhs_key' => 'id',
    'rhs_module' => 'EmailAddresses',
    'rhs_table' => 'email_addresses',
    'rhs_key' => 'id',
    'relationship_type' => 'many-to-many',
    'join_table' => 'email_addr_bean_rel',
    'join_key_lhs' => 'bean_id',
    'join_key_rhs' => 'email_address_id',
    'relationship_role_column' => 'bean_module',
    'relationship_role_column_value' => $table_name,
);

$dictionary[$module]['relationships'][strtolower($table_name) . '_emai
l_addresses_primary'] = array(
    'lhs_module' => $table_name,
    'lhs_table' => strtolower($table_name),
    'lhs_key' => 'id',
    'rhs_module' => 'EmailAddresses',
    'rhs_table' => 'email_addresses',
    'rhs_key' => 'id',
    'relationship_type' => 'many-to-many',
    'join_table' => 'email_addr_bean_rel',
    'join_key_lhs' => 'bean_id',
    'join_key_rhs' => 'email_address_id',
    'relationship_role_column' => 'bean_module',
    'relationship_role_column_value' => $module,
    'primary_flag_column' => 'primary_address',
);
```

Creating a Custom Bean and Modifying the Save Function

Next step, you will need to create a custom bean that extends from the stock bean. In our example, we choose the Opportunities.

./custom/modules/Opportunities/CustomOpportunity.php

```
    <?php

class CustomOpportunity extends Opportunity
{
    /**
     * Constructor
     */
    public function __construct()
    {
        parent::__construct();
        $this->emailAddress = BeanFactory::newBean('EmailAddresses');
    }

    /**
     * Populate email address fields here instead of retrieve() so tha
t they are properly available for logic hooks
     *
     * @see parent::fill_in_relationship_fields()
     */
    public function fill_in_relationship_fields()
    {
        parent::fill_in_relationship_fields();
        $this->emailAddress->handleLegacyRetrieve($this);
    }

    /**
     * @see parent::get_list_view_data()
     */
    public function get_list_view_data()
    {
        global $current_user;

        $temp_array = $this->get_list_view_array();

        $temp_array['EMAIL'] = $this->emailAddress->getPrimaryAddress(
$this);

        // Fill in the email1 field only if the user has access to it
        // This is a special case, because getEmailLink() uses email1
```

```
field for making the link
        // Otherwise get_list_view_data() shouldn't set any fields exc
ept fill the template data
        if ($this->ACLFieldAccess('email1', 'read')) {
            $this->email1 = $temp_array['EMAIL'];
        }

        $temp_array['EMAIL_LINK'] = $current_user->getEmailLink('email
1', $this, '', '', 'ListView');

        return $temp_array;
    }

    /**
     *
     * @see parent::save()
     */
    public function save($check_notify = false)
    {
        if (static::inOperation('saving_related')) {
            parent::save($check_notify);

            return $this;
        }

        $ori_in_workflow = empty($this->in_workflow) ? false : true;
        $this->emailAddress->handleLegacySave($this, $this->module_dir
);
        parent::save($check_notify);
        $override_email = array();
        if (!empty($this->email1_set_in_workflow)) {
            $override_email['emailAddress0'] = $this->email1_set_in_wo
rkflow;
        }

        if (!empty($this->email2_set_in_workflow)) {
            $override_email['emailAddress1'] = $this->email2_set_in_wo
rkflow;
        }

        if (!isset($this->in_workflow)) {
            $this->in_workflow = false;
        }

        if ($ori_in_workflow === false || !empty($override_email)) {
            $result = $this->emailAddress->save($this->id, $this->modu
```

```
le_dir, $override_email, '', '', '', '',
                $this->in_workflow);
        }

        return $this;
    }
}
```

Replacing the Stock Bean with a Custom Bean

Once you created the custom bean, you will need to show Sugar to use Custom bean. To do that you will create ./custom/modules/<modulename>/<modulename>.php and update the class name and file path in the $beanList and $beanFiles. (See: Modules $beanList and Customizing Core SugarBeans)

custom/Extension/application/Ext/Include/customOpportunities.php

```
  <?php

$objectList['Opportunities'] = 'Opportunity';
$beanList['Opportunities'] = 'CustomOpportunity';
$beanFiles['CustomOpportunity'] = 'custom/modules/Opportunities/Custom
Opportunity.php';
```

Once you have created these files, you will need to navigate Admin > Repairs and perform Quick Repair and Rebuild.

Adding the Emails Field to the Record View

At the stage, you have already created the field and updated the bean. The last step is placing the email field to record view. You can perform this step using Studio. Once you navigate to Studio you will find an Email field on the left column where available fields are listed. After adding the field the record views of Opportunities will look like this:

./custom/modules/Opportunities/clients/base/views/record/record.php

```
  <?php
$viewdefs['Opportunities'] =
array (
  'base' =>
```

```
array (
  'view' =>
  array (
    'record' =>
    array (
      'buttons' =>
      array (
        0 =>
        array (
          'type' => 'button',
          'name' => 'cancel_button',
          'label' => 'LBL_CANCEL_BUTTON_LABEL',
          'css_class' => 'btn-invisible btn-link',
          'showOn' => 'edit',
          'events' =>
          array (
            'click' => 'button:cancel_button:click',
          ),
        ),
        1 =>
        array (
          'type' => 'rowaction',
          'event' => 'button:save_button:click',
          'name' => 'save_button',
          'label' => 'LBL_SAVE_BUTTON_LABEL',
          'css_class' => 'btn btn-primary',
          'showOn' => 'edit',
          'acl_action' => 'edit',
        ),
        2 =>
        array (
          'type' => 'actiondropdown',
          'name' => 'main_dropdown',
          'primary' => true,
          'showOn' => 'view',
          'buttons' =>
          array (
            0 =>
            array (
              'type' => 'rowaction',
              'event' => 'button:edit_button:click',
              'name' => 'edit_button',
              'label' => 'LBL_EDIT_BUTTON_LABEL',
              'acl_action' => 'edit',
            ),
            1 =>
```

```
array (
  'type' => 'shareaction',
  'name' => 'share',
  'label' => 'LBL_RECORD_SHARE_BUTTON',
  'acl_action' => 'view',
),
2 =>
array (
  'type' => 'pdfaction',
  'name' => 'download-pdf',
  'label' => 'LBL_PDF_VIEW',
  'action' => 'download',
  'acl_action' => 'view',
),
3 =>
array (
  'type' => 'pdfaction',
  'name' => 'email-pdf',
  'label' => 'LBL_PDF_EMAIL',
  'action' => 'email',
  'acl_action' => 'view',
),
4 =>
array (
  'type' => 'divider',
),
5 =>
array (
  'type' => 'rowaction',
  'event' => 'button:find_duplicates_button:click',
  'name' => 'find_duplicates_button',
  'label' => 'LBL_DUP_MERGE',
  'acl_action' => 'edit',
),
6 =>
array (
  'type' => 'rowaction',
  'event' => 'button:duplicate_button:click',
  'name' => 'duplicate_button',
  'label' => 'LBL_DUPLICATE_BUTTON_LABEL',
  'acl_module' => 'Opportunities',
  'acl_action' => 'create',
),
7 =>
array (
  'type' => 'rowaction',
```

```
          'event' => 'button:historical_summary_button:click',
          'name' => 'historical_summary_button',
          'label' => 'LBL_HISTORICAL_SUMMARY',
          'acl_action' => 'view',
        ),
        8 =>
        array (
          'type' => 'rowaction',
          'event' => 'button:audit_button:click',
          'name' => 'audit_button',
          'label' => 'LNK_VIEW_CHANGE_LOG',
          'acl_action' => 'view',
        ),
        9 =>
        array (
          'type' => 'divider',
        ),
        10 =>
        array (
          'type' => 'rowaction',
          'event' => 'button:delete_button:click',
          'name' => 'delete_button',
          'label' => 'LBL_DELETE_BUTTON_LABEL',
          'acl_action' => 'delete',
        ),
      ),
    ),
    3 =>
    array (
      'name' => 'sidebar_toggle',
      'type' => 'sidebartoggle',
    ),
  ),
  'panels' =>
  array (
    0 =>
    array (
      'name' => 'panel_header',
      'header' => true,
      'fields' =>
      array (
        0 =>
        array (
          'name' => 'picture',
          'type' => 'avatar',
          'size' => 'large',
```

```
      'dismiss_label' => true,
      'readonly' => true,
    ),
    1 =>
    array (
      'name' => 'name',
      'related_fields' =>
      array (
        0 => 'total_revenue_line_items',
        1 => 'closed_revenue_line_items',
        2 => 'included_revenue_line_items',
      ),
    ),
    2 =>
    array (
      'name' => 'favorite',
      'label' => 'LBL_FAVORITE',
      'type' => 'favorite',
      'dismiss_label' => true,
    ),
    3 =>
    array (
      'name' => 'follow',
      'label' => 'LBL_FOLLOW',
      'type' => 'follow',
      'readonly' => true,
      'dismiss_label' => true,
    ),
  ),
),
1 =>
array (
  'name' => 'panel_body',
  'label' => 'LBL_RECORD_BODY',
  'columns' => 2,
  'labels' => true,
  'labelsOnTop' => true,
  'placeholders' => true,
  'newTab' => false,
  'panelDefault' => 'expanded',
  'fields' =>
  array (
    0 =>
    array (
      'name' => 'account_name',
      'related_fields' =>
```

```
  array (
    0 => 'account_id',
  ),
),
1 =>
array (
  'name' => 'date_closed',
  'related_fields' =>
  array (
    0 => 'date_closed_timestamp',
  ),
),
2 =>
array (
  'name' => 'amount',
  'type' => 'currency',
  'label' => 'LBL_LIKELY',
  'related_fields' =>
  array (
    0 => 'amount',
    1 => 'currency_id',
    2 => 'base_rate',
  ),
  'currency_field' => 'currency_id',
  'base_rate_field' => 'base_rate',
  'span' => 12,
),
3 =>
array (
  'name' => 'best_case',
  'type' => 'currency',
  'label' => 'LBL_BEST',
  'related_fields' =>
  array (
    0 => 'best_case',
    1 => 'currency_id',
    2 => 'base_rate',
  ),
  'currency_field' => 'currency_id',
  'base_rate_field' => 'base_rate',
),
4 =>
array (
  'name' => 'worst_case',
  'type' => 'currency',
  'label' => 'LBL_WORST',
```

```
        'related_fields' =>
        array (
          0 => 'worst_case',
          1 => 'currency_id',
          2 => 'base_rate',
        ),
        'currency_field' => 'currency_id',
        'base_rate_field' => 'base_rate',
      ),
      5 =>
      array (
        'name' => 'tag',
        'span' => 6,
      ),
      6 =>
      array (
        'name' => 'sales_status',
        'readonly' => true,
        'studio' => true,
        'label' => 'LBL_SALES_STATUS',
        'span' => 6,
      ),
      7 =>
      array (
        'name' => 'email',
        'studio' =>
        array (
          'visible' => true,
          'searchview' => true,
          'editview' => true,
          'editField' => true,
        ),
        'label' => 'LBL_EMAIL_ADDRESS',
        'span' => 12,
      ),
    ),
  ),
  2 =>
  array (
    'name' => 'panel_hidden',
    'label' => 'LBL_RECORD_SHOWMORE',
    'hide' => true,
    'labelsOnTop' => true,
    'placeholders' => true,
    'columns' => 2,
    'newTab' => false,
```

```
'panelDefault' => 'expanded',
'fields' =>
array (
  0 => 'next_step',
  1 => 'opportunity_type',
  2 => 'lead_source',
  3 => 'campaign_name',
  4 =>
  array (
    'name' => 'description',
    'span' => 12,
  ),
  5 => 'assigned_user_name',
  6 => 'team_name',
  7 =>
  array (
    'name' => 'date_entered_by',
    'readonly' => true,
    'type' => 'fieldset',
    'label' => 'LBL_DATE_ENTERED',
    'fields' =>
    array (
      0 =>
      array (
        'name' => 'date_entered',
      ),
      1 =>
      array (
        'type' => 'label',
        'default_value' => 'LBL_BY',
      ),
      2 =>
      array (
        'name' => 'created_by_name',
      ),
    ),
  ),
  8 =>
  array (
    'name' => 'date_modified_by',
    'readonly' => true,
    'type' => 'fieldset',
    'label' => 'LBL_DATE_MODIFIED',
    'fields' =>
    array (
      0 =>
```

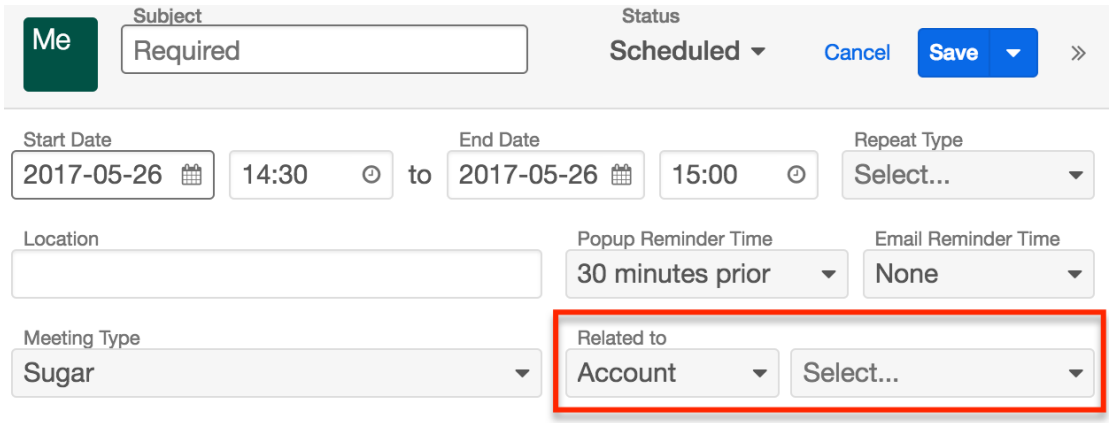
```
                array (
                  'name' => 'date_modified',
                ),
                1 =>
                array (
                  'type' => 'label',
                  'default_value' => 'LBL_BY',
                ),
                2 =>
                array (
                  'name' => 'modified_by_name',
                ),
              ),
            ),
          ),
        ),
        'templateMeta' =>
        array (
          'useTabs' => false,
        ),
      ),
    ),
  ),
);
```

Once the files are in place, navigate to Admin > Repair > Quick Repair & Rebuild to regenerate the metadata.

Changing the Default Module When Logging a New Call or Meeting

Overview

When creating a call or meeting directly from the Calls or Meetings module in Sugar, the default module for the Related To field is Accounts. If your sales team frequently schedules calls and meetings related to records from a module other than Accounts, it may make sense to adjust the behavior so that the Related To field defaults to a more commonly used module. This article covers how to change the default related module for calls and meetings in Sugar.



Prerequisites

This change requires code-level customizations, which requires direct access to the server or familiarity with creating and installing module loadable packages. If you need assistance making these changes and already have a relationship with a Sugar partner, you can work with them to make this customization. If not, please refer to the Partner Page to find a reselling partner to help with your development needs.

Note: Sugar Sell Essentials customers do not have the ability to upload custom file packages to Sugar using Module Loader.

Steps to Complete

For this example, we will change the default "Relates To" module to Contacts for records created in the Calls module and the Meetings module. Please note that, after completing these steps, the Related To field will still default to Accounts when creating a call or meeting from a contact that has an account relationship or to the current module from any other related module's record view.

Calls

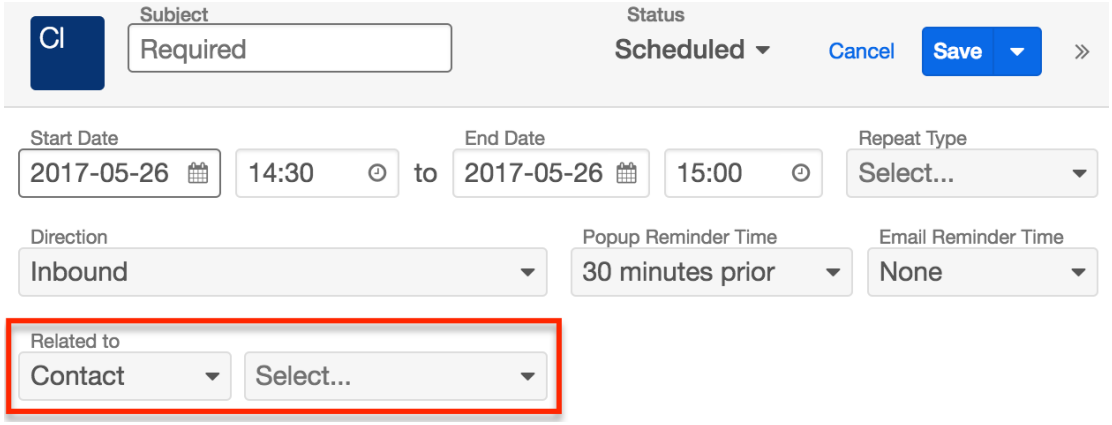
1. Create the following directory path if it does not already exist from the root of your Sugar instance directory: ./custom/Extension/modules/Calls/Ext/Vardefs/
2. Create a file in the directory called sugarfield_parent_type.php with the following contents:

```
<?php

$dictionary['Call']['fields']['parent_type']['default'] = 'Conta
```

```
cts';
```

3. Save the file and ensure that the file has the correct permissions by referring to the Required File System Permissions on Linux and Required File System Permissions on Windows With IIS articles.
4. Log in to Sugar as an administrator and navigate to Admin > Repair and perform a Quick Repair and Rebuild.

Once the quick repair completes, navigate to the Calls module and "Contact" should now be selected by default for the Related To field when logging a new call.



Meetings

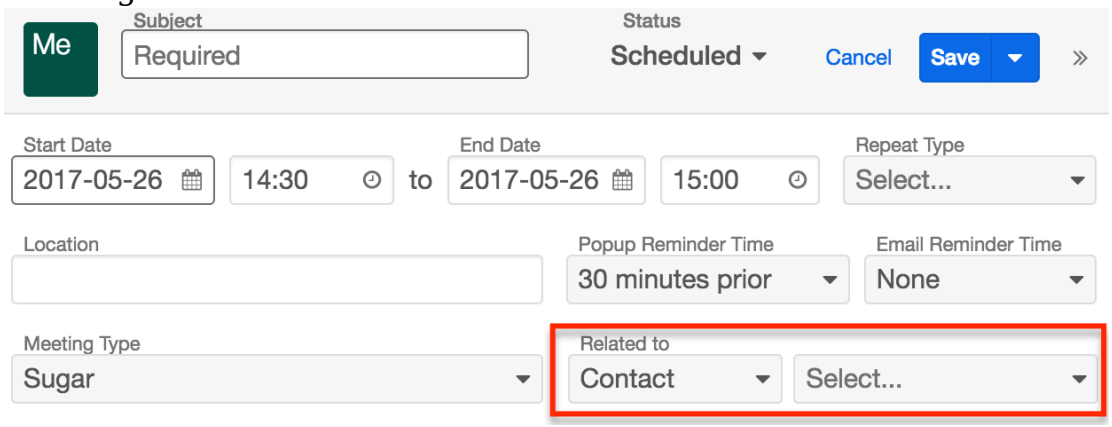
1. Create the following directory path if it does not already exist from the root of your Sugar instance directory: ./custom/Extension/modules/Meetings/Ext/Vardefs/.
2. Create a file in the directory called sugarfield_parent_type.php with the following contents:

```
<?php

$dictionary['Meeting']['fields']['parent_type']['default'] = 'Co
ntacts';
```

3. Save the file and ensure that the file has the correct permissions by referring to the Required File System Permissions on Linux and Required File System Permissions on Windows With IIS articles.
4. Log in to Sugar as an administrator and navigate to Admin > Repair and perform a Quick Repair and Rebuild.

Once the quick repair completes, navigate to the Meetings module and "Contact" should now be selected by default for the Related To field when scheduling a new

meeting.



Converting Address' Country Field to a Dropdown

Overview

Address fields in Sugar® are normally text fields, which allow users to enter in the appropriate information (e.g. street, city, and country) for the record. However, with multiple users working in Sugar, it is possible for data (e.g. country) to be entered in a variety of different ways (e.g. USA, U.S.A, and United States) when creating or editing the record. This can cause some issues when creating a report grouped by the Billing Country field, for example, as records with the same country will be grouped separately based on the different ways the country was entered.

This article will cover how to change the address' Billing Country field to a dropdown list which will allow users to select a single value (e.g. USA) and maintain consistency in data throughout the system.

Note: This article pertains to Sugar versions 6.x and 7.x.

Use Case

In this example, we will convert the Billing Country field in the Accounts module to a dropdown-type field to allow values to be selected from a dropdown list.

Prerequisites

A part of making this change involves mapping the "countries_dom" dropdown list to your existing Billing Country field's values. This dropdown list can be accessed and modified via Admin > Dropdown Editor. For more information on editing dropdown lists, please refer to the Developer Tools documentation. It is very

important that the existing values in the Billing Country field exactly match the Item Name values in the "countries_dom" dropdown list in order for the values to convert properly. If the existing value (e.g. United States) does not match one of the country options (e.g. USA) in the dropdown list, then the new dropdown version of the Billing Country field will most likely default to a blank value for that record record.

To avoid such issues, please review and update all of your existing Billing Country text field values prior to making this change. For more information on updating many records at once via import, please refer to the Updating Records Via Import article.

Steps to Complete

Converting Field to Dropdown

Use the following steps to change the Billing Country field to a dropdown list:

1. First, we will associate the Billing Country field in the Accounts module with the "countries_dom" dropdown. Locate the following directory in your Sugar file system: ./custom/Extension/modules/Accounts/Ext/Vardefs/
2. Locate a file called sugarfield_billing_address_country.php which controls the billing_address_country field and add the following lines to the file. This will set the field type to 'enum' and define the dropdown (countries_dom) to use for the field.
   Note: If this file or location does not exist, then you will need to first create this path and file.
   Your file should look like this:

3.
```
<?php

$dictionary['Account']['fields']['billing_address_country']['type']='enum';
$dictionary['Account']['fields']['billing_address_country']['options']='countries_dom';
```

4. Now, since the address block is handled uniquely in Sugar, we will also need to modify the template file in order to display the field as a dropdown anywhere the layout is being used in backward compatibility mode. To do this in an upgrade safe way, you will need to copy the file located in ./include/SugarFields/Fields/Address/en_us.EditView.tpl and place it in the following directory: ./custom/include/SugarFields/Fields/Address/.
   If you are using multiple languages in Sugar, you will need to make this change for each language-type in your system. For our example, we will be

focusing on the "en_us" language.
Note: You will most likely need to create this directory as it likely does not already exist.

5. Once the en_us.EditView.tpl file is in the custom directory, locate the line for the input html element for the country field. The line of code should look similar to this:

```
<input type="text" name="{{$country}}" id="{{$country}}" size="{
{$displayParams.size|default:30}}" {{if !empty($vardef.len)}}max
length='{{$vardef.len}}'{{/if}} value='{$fields.{{$country}}.val
ue}' tabindex="{{$tabindex}}">
```

6. Directly between the list shown above and the <td> appearing before it, add the following lines to display the field as a dropdown:

```
{if (!isset($config.enable_autocomplete) || $config.enable_autoc
omplete==false) && isset($fields.{{$country}}.options)}
<select name="{{$country}}" id="{{$country}}" title=''>
{if isset($fields.{{$country}}.value) && $fields.{{$country}}.va
lue != ''}
{html_options options=$fields.{{$country}}.options selected=$fie
lds.{{$country}}.value}
{else}
{html_options options=$fields.{{$country}}.options selected=$fie
lds.{{$country}}.default}
{/if}
</select>
{else}
<input type="text" name="{{$country}}" id="{{$country}}" size="{
{$displayParams.size|default:30}}" {{if !empty($vardef.len)}}max
length='{{$vardef.len}}'{{/if}} value='{$fields.{{$country}}.val
ue}' tabindex="{{$tabindex}}">
{/if}
```

7. Save the changes to the file.

Once the necessary change has been made, please navigate to Admin > Repair and perform a Quick Repair and Rebuild in order for the change to take effect. Your Billing Country field will now display as a dropdown field in the Accounts module.

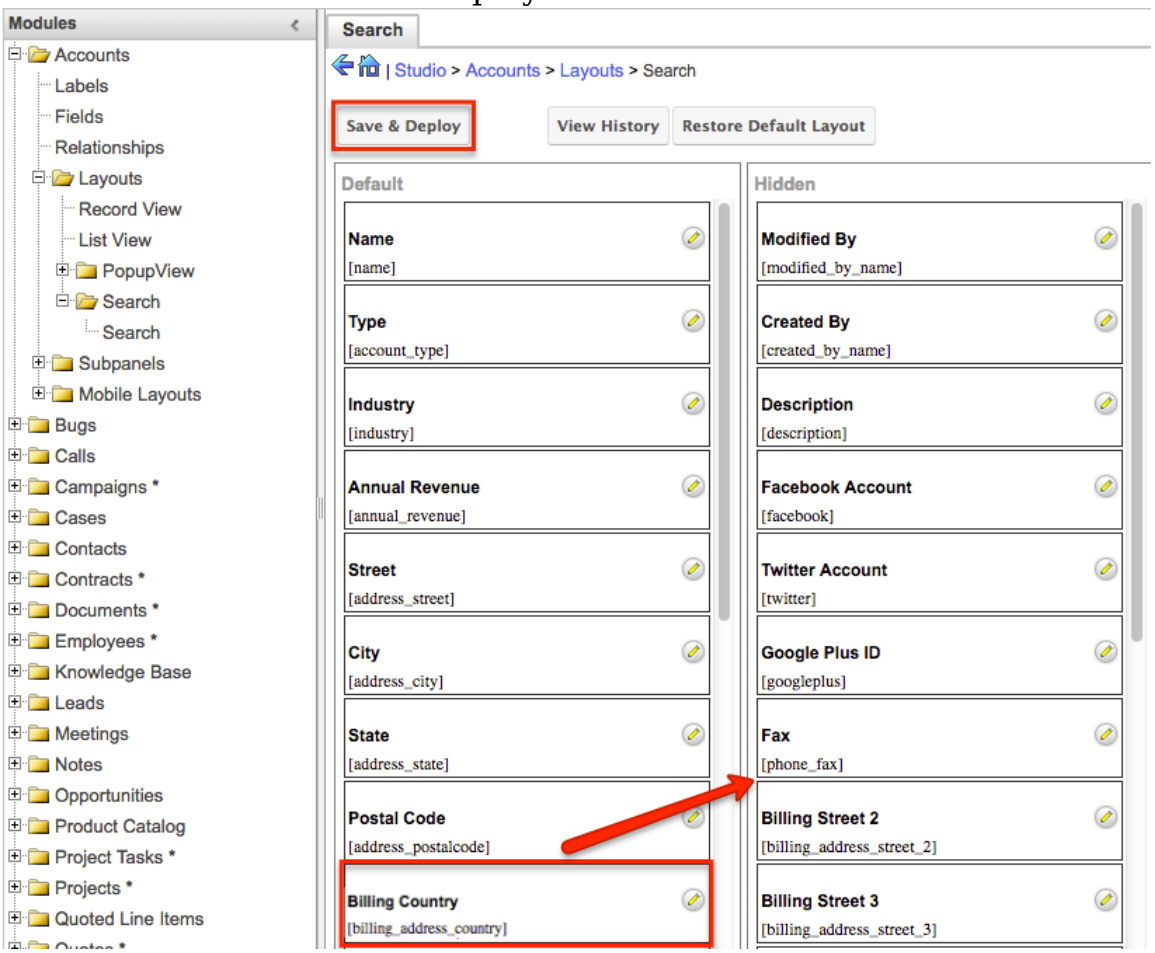
Please note that only the Account module's Billing Country field will be converted to a dropdown field once this change is applied. If you wish to have the Shipping Country field converted to a dropdown as well, please follow the steps above, but be sure to make the change specific to the "shipping_address_country" field. In

addition, you can make this change for other modules (e.g. Contacts and Leads) as well by specifying the appropriate module name (e.g. Contacts) and field name (e.g. Primary Address Country) when going through the steps.

Updating the Field Type for List View Filter

Once the appropriate changes have been made per the section above, use the following steps to get the list view filter to recognize the change:

1. Navigate to Admin > Studio > Accounts > Layouts > Search.
2. Drag the Billing Country field from the Default column to the Hidden column and click "Save & Deploy".



3. Now drag the Billing Country field back to the Default column and click "Save & Deploy" again.

This will retrieve the Billing Country field in its new state and add it to the account's list view filter.

Application

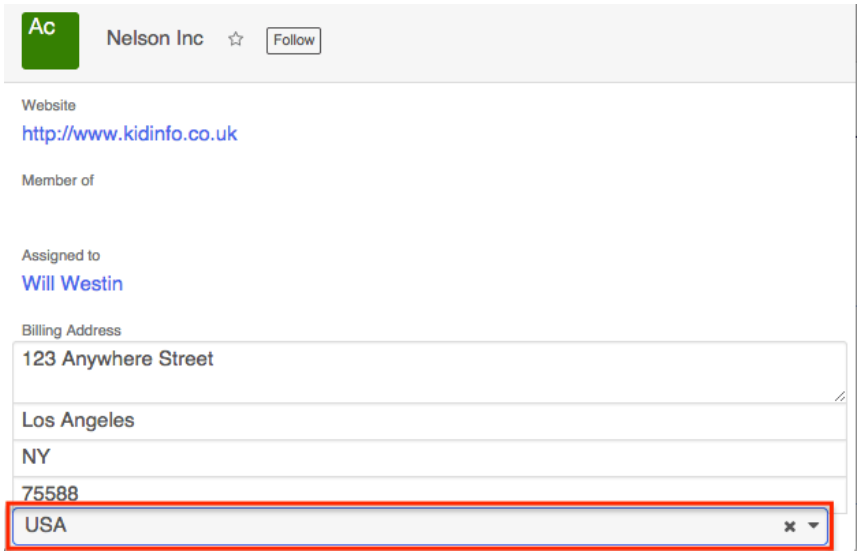
Once the Billing Country field is converted successfully to a dropdown, all records that had an existing value in the field should have the matching country

automatically selected. Going forward, users will simply need to select the appropriate country when entering address information in Sugar.

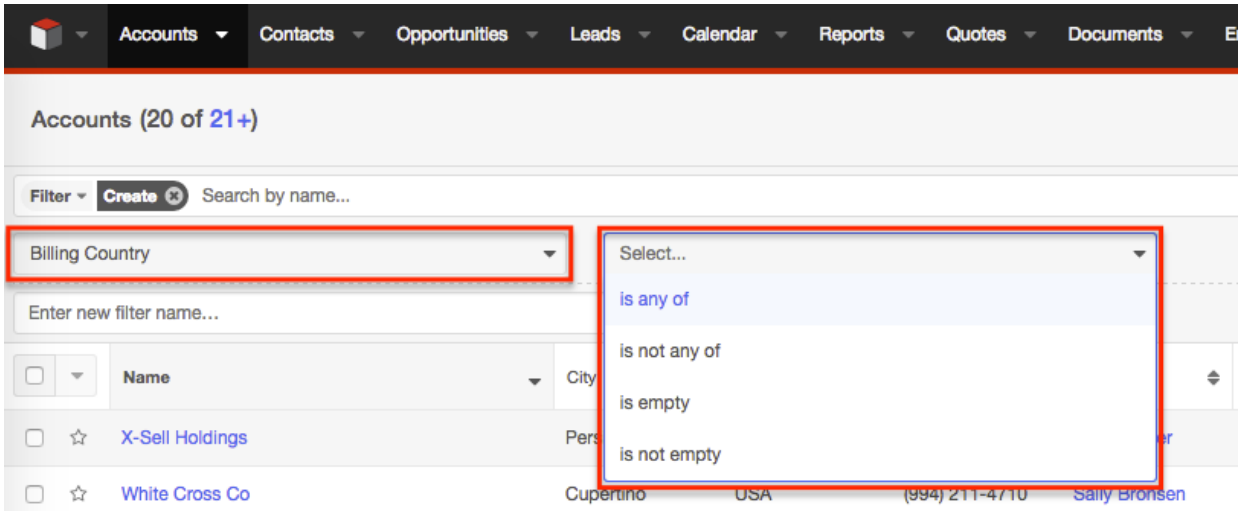
Please note that administrators can add additional country values as necessary to the "countries_dom" dropdown via Admin > Dropdown Editor. For more information on editing dropdown lists, please refer to the Developer Tools documentation.

The Billing Country field will now appear as follows:

For Accounts record view:



For Accounts list view filter:



Creating Custom Field Types

Overview

In this example, we create a custom field type called "Highlightfield", which will mimic the base text field type with the added feature that the displayed text for the field will be highlighted in a color chosen when the field is created in Studio.

This example requires the following steps, which are covered in the sections and subsections below:

1. Creating the JavaScript Controller
2. Defining the Handlebar Templates
3. Adding the Field Type to Studio
4. Enabling Search and Filtering

Note: Custom field types are *not* currently functional for use in Reports and will break Report Wizard.

Naming a Custom Field Type in Sugar

When choosing a name for a custom field type, keep in mind the following rules:

- The first letter of a custom field type's name must be capitalized.
- All subsequent letters in the field type's name must be lowercase.
- A field type's name cannot contain non-letter characters such as 0-9, hyphens, or dashes.

Therefore, in this example, the field type "Highlightfield" cannot be called HighLightField or highlightfield.

Creating the JavaScript Controller

First, create a controller file. Since we are starting from scratch, we need to extend the base field template. To accomplish this, create ./custom/clients/base/fields/Highlightfield/Highlightfield.js. This file will contain the JavaScript needed to render the field and format the values. By default, all fields extend the base template and do not require you to add the extendsFrom property. An example template is shown below:

./custom/clients/base/fields/Highlightfield/Highlightfield.js

```
({
  /**
```

```
     * Called when initializing the field
     * @param options
     */
    initialize: function(options) {
        this._super('initialize', [options]);
    },

    /**
     * Called when rendering the field
     * @private
     */
    _render: function() {
        this._super('_render');
    },

    /**
     * Called when formatting the value for display
     * @param value
     */
    format: function(value) {
        return this._super('format', [value]);
    },

    /**
     * Called when unformatting the value for storage
     * @param value
     */
    unformat: function(value) {
        return this._super('unformat', [value]);
    }
})
```

Note: This controller file example contains methods for initialize, _render, format, and unformat. These are shown for your ease of use but are not necessarily needed for this customization. For example, you could choose to create an empty controller consisting of nothing other than ({}) and the field would render as expected in this example.

Defining the Handlebar Templates

Next, define the handlebar templates. The templates will nearly match the base template found in ./clients/base/fields/base/ with the minor difference of an additional attribute of style="background:{{def.backcolor}};

color:{{def.textcolor}}" for the detail and list templates.

Detail View

The first template to define is the Sidecar detail view. This template handles the display of data on the record view.

./custom/clients/base/fields/Highlightfield/detail.hbs

```
  {{!
    The data for field colors are passed into the handlebars template
through the def array. For this example, the def.backcolor and def.tex
tcolor properties. These indexes are defined in:
    ./custom/modules/DynamicFields/templates/Fields/TemplateHighlightf
ield.php
}}

{{#if value}}
    <div class="ellipsis_inline" data-placement="bottom" style="backgr
ound:{{def.backcolor}}; color:{{def.textcolor}}">
        {{value}}
    </div>
{{/if}}
```

Edit View

Now define the Sidecar edit view. This template handles the display of the field in the edit view.

./custom/clients/base/fields/Highlightfield/edit.hbs.

```
  {{!
    We have not made any edits to this file that differ from stock, ho
wever,
    we could add styling here just as we did for the detail and list t
emplates.
}}

<input type="text"
    name="{{name}}"
    value="{{value}}"
    {{#if def.len}}maxlength="{{def.len}}"{{/if}}
    {{#if def.placeholder}}placeholder="{{str def.placeholder this.mod
el.module}}"{{/if}}
```

```
    class="inherit-width">
<p class="help-block">
```

List View

Finally, define the list view. This template handles the display of the custom field in list views.

./custom/clients/base/fields/Highlightfield/list.hbs

```
  {{!
    The data for field colors are passed into the handlebars template
through the def array. Our case
    being the def.backcolor and def.textcolor properties.  These index
es are defined in:
    ./custom/modules/DynamicFields/templates/Fields/TemplateHighlightf
ield.php
}}

<div class="ellipsis_inline" data-placement="bottom" data-original-
title="{{#unless value}}
    {{#if def.placeholder}}{{str def.placeholder this.model.module}}{{
/if}}{{/unless}}{{value}}"
     style="background:{{def.backcolor}}; color:{{def.textcolor}}">

    {{#if def.link}}
       <a href="{{#if def.events}}javascript:void(0);{{else}}{{href}}{{
/if}}">{{value}}</a>{{else}}{{value}}
    {{/if}}
</div>
```

Adding the Field Type to Studio

To enable the new field type for use in Studio, define the Studio field template. This will also allow us to map any additional properties we need for the templates. For this example, map the ext1 and ext2 fields from the fields_meta_data table to the backcolor and textcolor definitions. Also, set the dbfield definition to varchar so that the correct database field type is created.

Note: We are setting "reportable" to false to avoid issues with Report Wizard.

./custom/modules/DynamicFields/templates/Fields/TemplateHighlightfield.php

```
  <?php

if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry Poin
t');

require_once 'modules/DynamicFields/templates/Fields/TemplateField.php
';

class TemplateHighlightfield extends TemplateField
{
    var $type = 'varchar';
    var $supports_unified_search = true;

    /**
     * TemplateAutoincrement Constructor: Map the ext attribute fields
 to the relevant color properties
     *
     * References:      get_field_def function below
     *
     * @returns field_type
     */
    function __construct()
    {
        $this->vardef_map['ext1'] = 'backcolor';
        $this->vardef_map['ext2'] = 'textcolor';
        $this->vardef_map['backcolor'] = 'ext1';
        $this->vardef_map['textcolor'] = 'ext2';
    }

    //BEGIN BACKWARD COMPATIBILITY
    // AS 7.x does not have EditViews and DetailViews anymore these ar
e here
    // for any modules in backwards compatibility mode.

    function get_xtpl_edit()
    {
        $name = $this->name;
        $returnXTPL = array();

        if (!empty($this->help)) {
            $returnXTPL[strtoupper($this->name . '_help')] = translate
($this->help, $this->bean->module_dir);
        }

        if (isset($this->bean->$name)) {
            $returnXTPL[$this->name] = $this->bean->$name;
```

```
        } else {
            if (empty($this->bean->id)) {
                $returnXTPL[$this->name] = $this->default_value;
            }
        }
        return $returnXTPL;
    }

    function get_xtpl_search()
    {
        if (!empty($_REQUEST[$this->name])) {
            return $_REQUEST[$this->name];
        }
    }

    function get_xtpl_detail()
    {
        $name = $this->name;
        if (isset($this->bean->$name)) {
            return $this->bean->$name;
        }
        return '';
    }

    //END BACKWARD COMPATIBILITY

    /**
     * Function:        get_field_def
     * Description:        Get the field definition attributes that ar
e required for the Highlightfield Field
     *                      the primary reason this function is here i
s to set the dbType to 'varchar',
     *                      otherwise 'Highlightfield' would be used b
y default.
     * References:      __construct function above
     *
     * @return            Field Definition
     */
    function get_field_def()
    {
        $def = parent::get_field_def();

        //set our fields database type
        $def['dbType'] = 'varchar';

        //set our fields to false to avoid issues with Report Wizard.
```

```
        $def['reportable'] = false;

        //set our field as custom type
        $def['custom_type'] = 'varchar';

        //map our extension fields for colorizing the field
        $def['backcolor'] = !empty($this->backcolor) ? $this->backcolo
r : $this->ext1;
        $def['textcolor'] = !empty($this->textcolor) ? $this->textcolo
r : $this->ext2;

        return $def;
    }
}
```

Note: For the custom field type, the ext1, ext2, ext3, and ext4 database fields in the fields_meta_data table offer additional property storage.

Creating the Form Controller

Next, set up the field's form controller. This controller will handle the field's form template in Admin > Studio > Fields and allow us to assign color values to the Smarty template.

./custom/modules/DynamicFields/templates/Fields/Forms/Highlightfield.php

```
  <?php

if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry Poin
t');

require_once 'custom/modules/DynamicFields/templates/Fields/TemplateHi
ghlightfield.php';

/**
 * Implement get_body function to correctly populate the template for
the ModuleBuilder/Studio
 * Add field page.
 *
 * @param Sugar_Smarty $ss
 * @param array $vardef
 *
 */
function get_body(&$ss, $vardef)
```

```
{
    global $app_list_strings, $mod_strings;
    $vars = $ss->get_template_vars();
    $fields = $vars['module']->mbvardefs->vardefs['fields'];
    $fieldOptions = array();
    foreach ($fields as $id => $def) {
        $fieldOptions[$id] = $def['name'];
    }
    $ss->assign('fieldOpts', $fieldOptions);

    //If there are no colors defined, use black text on
    // a white background
    if (isset($vardef['backcolor'])) {
        $backcolor = $vardef['backcolor'];
    } else {
        $backcolor = '#ffffff';
    }
    if (isset($vardef['textcolor'])) {
        $textcolor = $vardef['textcolor'];
    } else {
        $textcolor = '#000000';
    }
    $ss->assign('BACKCOLOR', $backcolor);
    $ss->assign('TEXTCOLOR', $textcolor);

    $colorArray = $app_list_strings['highlightColors'];
    asort($colorArray);

    $ss->assign('highlightColors', $colorArray);
    $ss->assign('textColors', $colorArray);

    $ss->assign('BACKCOLORNAME', $app_list_strings['highlightColors'][
$backcolor]);
    $ss->assign('TEXTCOLORNAME', $app_list_strings['highlightColors'][
$textcolor]);

    return $ss->fetch('custom/modules/DynamicFields/templates/Fields/F
orms/Highlightfield.tpl');
}
?>
```

Creating the Smarty Template

Once the form controller is in place, create the Smarty template. The .tpl below

will define the center content of the field's Studio edit view. The template includes a coreTop.tpl and coreBottom.tpl file. These files add the base field properties such as "Name" and "Display Label" that are common across all field types. This allows us to focus on the fields that are specific to our new field type.

./custom/modules/DynamicFields/templates/Fields/Forms/Highlightfield.tpl

```
  {include file="modules/DynamicFields/templates/Fields/Forms/coreTop.
tpl"}
<tr>
    <td class='mbLBL'>{sugar_translate module="DynamicFields" label="C
OLUMN_TITLE_DEFAULT_VALUE"}:</td>
    <td>
        {if $hideLevel < 5}
            <input type='text' name='default' id='default' value='{$va
rdef.default}'
                   maxlength='{$vardef.len|default:50}'>
        {else}
            <input type='hidden' id='default' name='default' value='{$
vardef.default}'>{$vardef.default}
        {/if}
    </td>
</tr>
<tr>
    <td class='mbLBL'>{sugar_translate module="DynamicFields" label="C
OLUMN_TITLE_MAX_SIZE"}:</td>
    <td>
        {if $hideLevel < 5}
            <input type='text' name='len' id='field_len' value='{$vard
ef.len|default:25}'
                  onchange="forceRange(this,1,255);changeMaxLength(doc
ument.getElementById('default'),this.value);">
            <input type='hidden' id="orig_len" name='orig_len' value='
{$vardef.len}'>
            {if $action=="saveSugarField"}
                <input type='hidden' name='customTypeValidate' id='cus
tomTypeValidate' value='{$vardef.len|default:25}'

                       onchange="if (parseInt(document.getElementById(
'field_len').value) < parseInt(document.getElementById('orig_len').val
ue)) return confirm(SUGAR.language.get('ModuleBuilder', 'LBL_CONFIRM_L
OWER_LENGTH')); return true;">
            {/if}
        {literal}
            <script>
                function forceRange(field, min, max) {
```

```
                    field.value = parseInt(field.value);
                    if (field.value == 'NaN')field.value = max;
                    if (field.value > max) field.value = max;
                    if (field.value < min) field.value = min;
                }
                function changeMaxLength(field, length) {
                    field.maxLength = parseInt(length);
                    field.value = field.value.substr(0, field.maxLengt
h);
                }
            </script>
        {/literal}
        {else}
            <input type='hidden' name='len' value='{$vardef.len}'>{$va
rdef.len}
        {/if}
    </td>
</tr>
<tr>
    <td class='mbLBL'>{sugar_translate module="DynamicFields" label="L
BL_HIGHLIGHTFIELD_BACKCOLOR"}:</td>
    <td>
        {if $hideLevel < 5}
            {html_options name="ext1" id="ext1" selected=$BACKCOLOR op
tions=$highlightColors}
        {else}
            <input type='hidden' id='ext1' name='ext1' value='{$BACKCO
LOR}'>
            {$BACKCOLORNAME}
        {/if}
    </td>
</tr>
<tr>
    <td class='mbLBL'>{sugar_translate module="DynamicFields" label="L
BL_HIGHLIGHTFIELD_TEXTCOLOR"}:</td>
    <td>
        {if $hideLevel < 5}
            {html_options name="ext2" id="ext2" selected=$TEXTCOLOR op
tions=$highlightColors}
        {else}
            <input type='hidden' id='ext2' name='ext2' value='{$TEXTCO
LOR}'>
            {$TEXTCOLORNAME}
        {/if}
    </td>
</tr>
```

```
{include file="modules/DynamicFields/templates/Fields/Forms/coreBottom
.tpl"}
```

Registering the Field Type

Once you have defined the Studio template, register the field as a valid field type. For this example, the field will inherit the base field type as it is a text field. In doing this, we are able to override the core functions. The most used override is the save function shown below.

./custom/include/SugarFields/Fields/Highlightfield/SugarFieldHighlightfield.php

```
  <?php

require_once 'include/SugarFields/Fields/Base/SugarFieldBase.php';
require_once 'data/SugarBean.php';

class SugarFieldHighlightfield extends SugarFieldBase
{
    //this function is called to format the field before saving.  For
example we could put code in here
    // to check spelling or to change the case of all the letters
    public function save(&$bean, $params, $field, $properties, $prefix
 = '')
    {
        $GLOBALS['log']->debug("SugarFieldHighlightfield::save() funct
ion called.");
        parent::save($bean, $params, $field, $properties, $prefix);
    }
}
?>
```

Creating Language Definitions

For the new field, you must define several language extensions to ensure everything is displayed correctly. This section includes three steps:

- Adding the Custom Field to the Type List
- Creating Labels for Studio
- Defining Dropdown Controls

Adding the Custom Field to the Type List

Define the new field type in the field types list. This will allow an Administrator to select the Highlightfield field type in Studio when creating a new field.

./custom/Extension/modules/ModuleBuilder/Ext/Language/en_us.Highlightfield.php

```
  <?php

$mod_strings['fieldTypes']['Highlightfield'] = 'Highlighted Text';
```

Creating Labels for Studio

Once the field type is defined, add the labels for the Studio template.

./custom/Extension/modules/DynamicFields/Ext/Language/en_us.Highlightfield.php

```
  <?php

$mod_strings['LBL_HIGHLIGHTFIELD'] = 'Highlighted Text';
$mod_strings['LBL_HIGHLIGHTFIELD_FORMAT_HELP'] = '';

$mod_strings['LBL_HIGHLIGHTFIELD_BACKCOLOR'] = 'Background Color';
$mod_strings['LBL_HIGHLIGHTFIELD_TEXTCOLOR'] = 'Text Color';
```

Defining Dropdown Controls

For this example, we must define dropdown lists, which will be available in Studio for an administrator to add or remove color options for the field type.

./custom/Extension/application/Ext/Language/en_us.Highlightfield.php

```
  <?php

$app_strings['LBL_HIGHLIGHTFIELD_OPERATOR_CONTAINS'] = 'contains';
$app_strings['LBL_HIGHLIGHTFIELD_OPERATOR_NOT_CONTAINS'] = 'does not c
ontain';
$app_strings['LBL_HIGHLIGHTFIELD_OPERATOR_STARTS_WITH'] = 'starts with
';

$app_list_strings['highlightColors'] = array(
    '#0000FF' => 'Blue',
    '#00ffff' => 'Aqua',
```

```
    '#FF00FF' => 'Fuchsia',
    '#808080' => 'Gray',
    '#ffff00' => 'Olive',
    '#000000' => 'Black',
    '#800000' => 'Maroon',
    '#ff0000' => 'Red',
    '#ffA500' => 'Orange',
    '#ffff00' => 'Yellow',
    '#800080' => 'Purple',
    '#ffffff' => 'White',
    '#00ff00' => 'Lime',
    '#008000' => 'Green',
    '#008080' => 'Teal',
    '#c0c0c0' => 'Silver',
    '#000080' => 'Navy'
);
```

Enabling Search and Filtering

To enable Highlightfield for searching and filtering, define the filter operators and the Sugar widget.

Defining the Filter Operators

The filter operators, defined in ./custom/clients/base/filters/operators/operators.php, allow the custom field be used for searching in Sidecar listviews.

./custom/clients/base/filters/operators/operators.php

```
  <?php

require 'clients/base/filters/operators/operators.php';

$viewdefs['base']['filter']['operators']['Highlightfield'] = array(
    '$contains' => 'LBL_HIGHLIGHTFIELD_OPERATOR_CONTAINS',
    '$not_contains' => 'LBL_HIGHLIGHTFIELD_OPERATOR_NOT_CONTAINS',
    '$starts' => 'LBL_HIGHLIGHTFIELD_OPERATOR_STARTS_WITH',
);
```

Note: The labels for the filters in this example are defined in

./custom/Extension/application/Ext/Language/en_us.Highlightfield.php. For the full list of filters in the system, you can look at ./clients/base/filters/operators/operators.php.

Defining the Sugar Widget

Finally, define the Sugar widget. The widget will help our field for display on Reports and subpanels in backward compatibility. It also controls how search filters are applied to our field.

./custom/include/generic/SugarWidgets/SugarWidgetFieldHighlightfield.php

```
  <?php

if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry Poin
t');

require_once 'include/generic/SugarWidgets/SugarWidgetFieldvarchar.php
';

class SugarWidgetFieldHighlightfield extends SugarWidgetFieldVarchar
{
    function SugarWidgetFieldText(&$layout_manager)
    {
        parent::SugarWidgetFieldVarchar($layout_manager);
    }

    function queryFilterEquals($layout_def)
    {
        return $this->reporter->db->convert($this->_get_column_select(
$layout_def), "text2char") .
        " = " . $this->reporter->db->quoted($layout_def['input_name0']
);
    }

    function queryFilterNot_Equals_Str($layout_def)
    {
        $column = $this->_get_column_select($layout_def);
        return "($column IS NULL OR " . $this->reporter->db->convert($
column, "text2char") . " != " .
        $this->reporter->db->quoted($layout_def['input_name0']) . ")";
    }

    function queryFilterNot_Empty($layout_def)
    {
        $column = $this->_get_column_select($layout_def);
```

```
        return "($column IS NOT NULL AND " . $this->reporter->db->conv
ert($column, "length") . " > 0)";
    }

    function queryFilterEmpty($layout_def)
    {
        $column = $this->_get_column_select($layout_def);
        return "($column IS NULL OR " . $this->reporter->db->convert($
column, "length") . " = 0)";
    }

    function displayList($layout_def)
    {
        return nl2br(parent::displayListPlain($layout_def));
    }
}

?>
```

Once the files are in place, navigate to Admin > Repair > Quick Repair and Rebuild. It is also best practice to clear your browser cache before using the new field type in Studio.

Creating an Auto-Incrementing Field

Overview

This article will cover the two approaches to creating an auto-incrementing field in Sugar.

Case 1: Auto-Incrementing a Field Using a Logic Hook

The benefits of having an auto-incrementing field populated from a logic hook are that it can be incremented based on additional logic. It also allows for multiple auto-incrementing fields, while a database-level auto-incrementing field is only allowed once per table. Auto-incremented fields handled in a logic hook can also be formatted as needed, such as adding leading zeros or alpha characters. You can also control the auto-increment value based on another field, for instance, if you wanted to increase a version_number based on other records related to the same parent. For this example, we will create a simple auto-incrementing field that goes up by 1 whenever the record is saved for the first time.

First, create an integer field incrementing_number_c in Studio.

Next, create a custom class and method that the logic hook will call:

./custom/modules/Accounts/Accounts_Save.php

```
  <?php

class Accounts_Save
{
    function auto_increment($bean, $event, $arguments)
    {
        if (!$arguments['isUpdate']) {
            $sq = new \SugarQuery();
            $sq->from(BeanFactory::newBean('Accounts'), ['team_securit
y' => false]);
            $sq->select->fieldRaw('MAX(' . $sq->getFromAlias() . '_cst
m.incrementing_number_c)', 'current_max_id');
            $current_max_id = $sq->getOne();
            $max_id = (empty($current_max_id)) ? 1 : $current_max_id +
 1;
            $bean->incrementing_number_c = $max_id;
        }
    }
}
```

Next, register the custom logic hook via the LogicHooks Extension framework by creating a file at ./custom/Extension/modules/Accounts/Ext/LogicHooks. An example is shown below:

./custom/Extension/modules/Accounts/Ext/LogicHooks/autoIncrementOnSave.php

```
  <?php

$hook_array['before_save'][] = array(
    2,
    'Custom Logic to Auto-Increment integer_field_c',
    'custom/modules/Accounts/Accounts_Save.php',
    'Accounts_Save',
    'auto_increment'
);
```

After creating these files, run a Quick Repair and Rebuild.

Case 2: Creating an Auto-Incrementing Field on the Database

Creating an auto-incrementing field at the database level has the benefit of being unique and easy to manage. One downside to having an auto-incrementing field at the database level is that it means only being allowed one such field per module, as databases only allow one auto-incrementing field per table. A custom auto-incrementing field also relies on updating a module's core table, which is less upgrade safe. For this reason, a logic hook based auto-incrementing field is recommended over a database-driven auto-incrementing field.

To create an integer field in Sugar which auto-increments at the database level, we will create a custom field via the VarDefs extension as shown below:

./custom/Extension/modules/Accounts/Ext/Vardefs/autoIncrement.php

```
  <?php

$dictionary['Account']['fields']['account_number_auto'] = array (
    'name' => 'account_number_auto',
    'vname' => 'LBL_NUMBER_AUTO',
    'type' => 'int',
    'readonly' => true,
    'len' => 11,
    'required' => true,
    'auto_increment' => true,
    'unified_search' => true,
    'full_text_search' => array(
        'enabled' => true,
        'searchable' => true,
        'boost' => 1.25
    ),
    'comment' => 'Visual unique identifier',
    'duplicate_merge' => 'disabled',
    'disable_num_format' => true,
    'studio' => array('quickcreate' => false),
    'duplicate_on_record_copy' => 'no',
);

$dictionary['Account']['indices']['account_number_auto'] = array(
    'name' =>'accountsnumk_cstm',
    'type' =>'unique',
    'fields'=>array('account_number_auto'),
```

```
);
```

Navigate to Admin > Studio > Repair > Quick Repair and Rebuild. After the repair, you will see a vardef comparison as shown below:

```
  /*COLUMNS*/
/*MISSING IN DATABASE - account_number_auto - ROW*/
ALTER TABLE accounts add COLUMN account_number_auto int(11) NOT NULL a
uto_increment;
/* INDEXES */
/*MISSING INDEX IN DATABASE - accountsnumk_cstm - unique ROW */
ALTER TABLE accounts ADD CONSTRAINT UNIQUE accountsnumk_cstm (account_
number_auto);;
```

Alter this to be the following and execute the changes:

```
  ALTER TABLE accounts add COLUMN account_number_auto int(11) UNIQUE N
OT NULL auto_increment;
```

Note: Since auto-incremented fields are populated at the database level, all existing records in the database will have this field populated as soon as the field is created on the database table. The value assigned will not be based on any logic (such as the record creation date) that can be controlled or predicted. Therefore when sorting by this field, records created after the field was added will be in order that they were created, but records created before will not.

Customizing Prefill Fields When Copying Records

Overview

The copy action on the record view allows for users to duplicate records. This article will cover the various ways to customize the prefill fields on the copy view.

Modifying the Copy Prefill View Using the Vardefs

The following section will outline how to modify the fields that are prefilled when copying a Bug record from the record view using the beans Vardefs. This is helpful when the list of copied fields are static and have no dependencies. You can apply

this to any module in the system.

./custom/Extension/modules/Bugs/Ext/Vardefs/copyPrefill.php

```
  <?php

//remove a field from copy
$dictionary['Bug']['fields']['description']['duplicate_on_record_copy'
] = 'no';

//add a field to copy
$dictionary['Bug']['fields']['priority']['duplicate_on_record_copy'] =
 'always';
```

Once in place, navigate to Admin > Repair > Quick Repair & Rebuild.

Note: You can name the file 'copyPrefill.php' anything you like. We advise against making these changes in the ./custom/Extension/modules/<module>/Ext/Vardefs/sugarfield_<field>.php files as these changes may be removed during studio edits.

Modifying the Copy Prefill View Using JavaScript Controllers

The following section will outline how to modify the fields that are prefilled when copying a Bug record from the record view using the JavaScript Controller. This is helpful when you need to dependently determine the fields to copy by a field on the bean. You can apply this to any module in the system.

./custom/modules/Bugs/clients/base/views/record/record.js

```
  ({
    extendsFrom: 'RecordView',

    setupDuplicateFields: function (prefill) {

        this._super('setupDuplicateFields', prefill);

        var fields = [
            'name',
            'assigned_user_id',
            'priority',
            'type',
            'product_category',
```

```
            'description'
        ];

        //determines whether the field list above is a set of allowlis
ted (allowed) or denylisted (denied) fields
        var denylist = false;

        if (denylist) {
            _.each(fields, function (field) {
                if (field && prefill.has(field)) {
                    //set denylist field to the default value if exist
s
                    if (!_.isUndefined(prefill.fields[field]) && !_.is
Undefined(prefill.fields[field].default)) {
                        prefill.set(field, prefill.fields[field].defau
lt);
                    } else {
                        prefill.unset(field);
                    }
                }
            });
        } else {
            _.each(prefill.fields, function (value, field) {
                if (!_.contains(fields, field)) {
                    if (!_.isUndefined(prefill.fields[field].default))
 {
                        prefill.set(field, prefill.fields[field].defau
lt);
                    } else {
                        prefill.unset(field);
                    }
                }
            });
        }

    }
})
```

Once in place, navigate to Admin > Repair > Quick Repair & Rebuild. The denylist variable will determine whether the list of fields are allowlisted or denylisted from the copy feature.

Customizing the Email Editor Buttons

Overview

The Emails module in Sugar® displays commonly-used buttons in the HTML editor. This article explains how to modify the buttons on the editor's toolbar.

Modifying The View

The following sections outline how to customize the button toolbar presented within the TinyMCE editor. In the example below, we will be adding buttons for Cut, Copy, and Paste to the toolbar.

Using Custom Module Metadata

To override the Emails module compose-email view, we will need to copy ./modules/Emails/clients/base/views/compose-email/compose-email.php to ./custom/modules/Emails/clients/base/views/compose-email/compose-email.php. Once completed, we will then edit the ['tinyConfig']['toolbar'] definition to include | cut copy paste . An example of this is shown below:

./custom/modules/Emails/clients/base/views/compose-email/compose-email.php

```
  $viewdefs['Emails']['base']['view']['compose-email'] = array(
    ...
    'panels' => array(
        array(
            ...
            'fields' => array(
             ...
                array(
                    'name' => 'description_html',
                    'dismiss_label' => true,
                    'span' => 12,
                    'tinyConfig' => array(
                        'toolbar' => 'code | bold italic underline str
ikethrough | bullist numlist | ' .
                            'alignleft aligncenter alignright alignjus
tify | forecolor backcolor | ' .
                            'fontsizeselect | formatselect | fontselec
t | sugarattachment sugarsignature sugartemplate | cut copy paste',
                    ),
                ),
            ),
        ),
```

```
        ...
    ),
);
```

Note: We recommend modifying the modules custom metadata for local deployments when the customization is not part of a distributed package.

Once this file is in place, navigate to Admin > Repair > Quick Repair and Rebuild. Once completed, your changes will be reflected in the system.

Using The Module Extension Framework

To extend the Emails module compose-email view using the extension framework, we will need to loop through the existing views array definition to find and modify the description_html field and update the relevant sub-array ['tinyConfig']['toolbar'] with the additional toolbar buttons we want to display. An example of this is shown below:

./custom/Extension/modules/Emails/Ext/clients/base/views/compose-email/add_toolbar_buttons.php

```
  <?php

$target_view = 'compose-email';
$target_fieldname = 'description_html';

$default_toolbar =  'code | bold italic underline strikethrough | bull
ist numlist | ' .
                            'alignleft aligncenter alignright alignjus
tify | forecolor backcolor | ' .
                            'fontsizeselect | formatselect | fontselec
t | sugarattachment sugarsignature sugartemplate';
$custom_toolbar = ' | cut copy paste';

if (!empty($viewdefs['Emails']['base']['view'][$target_view]['panels']
)) {
 $panels = $viewdefs['Emails']['base']['view'][$target_view]['panels']
;
 foreach ($panels as $i => $panel) {
  if (!empty($panel['fields'])) {
   foreach ($panel['fields'] as $j => $field) {
    if ($field['name'] == $target_fieldname) {
     if(!isset($field['tinyConfig'])){
      $field['tinyConfig'] = array();
```

```
     }
     /* If toolbar already exists, add our custom buttons to the end,
     otherwise set it to the default and append that. */
     if(!isset($field['tinyConfig']['toolbar'])) {
      $field['tinyConfig']['toolbar'] = $default_toolbar;
     }
     $viewdefs['Emails']['base']['view'][$target_view]['panels'][$i]['
fields'][$j]['tinyConfig']['toolbar'] = $field['tinyConfig']['toolbar'
] . $custom_toolbar;
    }
   }
  }
 }
}
```

Note: We recommend using the extension framework when the code will be installed as part of a distributed package.

Once this file is in place, navigate to Admin > Repair > Quick Repair and Rebuild. Once completed, your changes will be reflected in the system.

Toolbar Format

As demonstrated in the examples above, the toolbar value is a string with the button keywords space-separated. The | indicates a divider should be shown in the editor. If the goal were to replace the toolbar buttons with only text-modifier buttons, the value for toolbar would be:

```
'bold italic | underline strikethrough'
```

The toolbar button keywords are documented at TinyMCE Editor Control Identifiers. Note that at this time, only control keywords listed as "Core" are supported by Sugar®.

Customizing the Start Speed of List View Search

Overview

When searching a Sidecar module's list view, Sugar® begins returning results automatically once a predefined number of milliseconds have passed. This article

covers how to customize the start speed of the list view search for Sidecar modules in Sugar.

Prerequisites

This change requires code-level customizations, and you will need direct access to the server in order to make the necessary changes in the file system. If you already have a relationship with a Sugar partner, you can work with them to make this customization. If not, please refer to the Partner Page to find a reselling partner to help with your development needs.

Note: If your instance is hosted in Sugar's cloud environment, you can create a package that can be installed within Admin > Module Loader. For more information, please refer to the Creating an Installable Package That Copies Files article.

Note: Sugar Sell Essentials customers do not have the ability to upload custom file packages to Sugar using Module Loader.

Use Case

By default, the list view search process in Sugar begins to run 400 milliseconds after a user stops typing or pasting values into one of the search fields. For some users or situations, the system-defined start speed for list view search may be considered too fast. We will walk through increasing the length of time it takes before the search starts to run, using 750 milliseconds as an example.

Steps to Complete

The function that controls the list view search speed in Sugar can be found in the following file: ./clients/base/views/filter-quicksearch/filter-quicksearch.js In order to customize this function, please use the following steps to create an extension in the ./custom/ directory:

1. Navigate to the ./custom/ directory in the Sugar file system and create the following directory structure if it does not already exist: ./clients/base/views/filter-quicksearch/.
2. In a text editor application, create a new file.
3. Copy the code provided below into this new file:

```
({
  extendsFrom: 'FilterQuicksearchView',

  /**
```

```
     * @override
     * @param {Object} opts
     */
    initialize: function(opts) {
        this._super('initialize', [opts]);
    },

    /**
     * Fire quick search
     * @param {Event} e
     */
     throttledSearch: _.debounce(function(e) {
       var newSearch = this.$el.val();
       if(this.currentSearch !== newSearch) {
           this.currentSearch = newSearch;
           this.layout.trigger('filter:apply', newSearch);
       }
     }, 750),
})
```

4. For our example, the value of "750" at the end of the code represents the new start speed of list view search. Feel free to replace it with your desired value in milliseconds.
5. Save the new file as./custom/clients/base/views/filter-quicksearch/filter-quicksearch.js.
6. Update the Ownership and Permissions of the directories and file that were created.
    - For more information on Linux based stacks, please refer to the article Required File System Permissions on Linux.
    - For more information on Windows-based stacks, please refer to the article Required File System Permissions on Windows With IIS.
7. Finally, log into Sugar and navigate to Admin > Repair then perform a "Quick Repair and Rebuild".

After completing the steps in this article, Sugar will wait 750 milliseconds before returning search results once the user has stopped entering search criteria.

Disabling RLI Alerts on Opportunities

Overview

How to disable Revenue Line Item (RLI) alerts on Opportunities using a custom JavaScript controller.

Overriding the Record View

First, we must override the stock opportunities record view. This will handle the RLI alerts when navigating to an existing record. This can be done by creating:

./custom/modules/Opportunities/clients/base/views/record/record.js

```
  ({
    extendsFrom: 'OpportunitiesRecordView',
    initialize: function (options) {
        this._super('initialize', [options]);
    },
    /**
     * Hide the warning message about missing RLIs
     * @param string module     The module that we are currently on.
     */
    showRLIWarningMessage: function(module) {
        //here we create an empty override function
    },
    /**
     * @inheritdoc
     */
    _dispose: function() {
        this._super('_dispose', []);
    }
})
```

As you can see, this file extends from 'OpportunitiesRecordView' which points our code to extend the stock ./modules/Opportunities/clients/base/views/record/record.js file.

Disabling Tooltips

Overview

This article will demonstrate how to disable the tooltips in Sugar.

Steps to Complete

Creating a Custom JavaScript File

First, we will need to create a custom JavaScript file. This file can technically exist

anywhere within the root of your Sugar instance.

custom/JavaScript/disable_tooltips.js

```
  (function(app){
     app.events.on('app:init', function(){
          // Clear existing ones
          app.tooltip.clear();
          // Disable all
          app.tooltip._disable();
     });
})(SUGAR.App);
```

Appending to JSGroupings

Second, we need to add our disable_tooltips.js file to sugar_grp7.min.js in our JSGroupings.

custom/Extension/application/Ext/JSGroupings/disable_tooltips.php

```
  <?php
foreach ($js_groupings as $key => $groupings) {
    foreach ($groupings as $file => $target) {
        if ($target == 'include/javascript/sugar_grp7.min.js') {
            $js_groupings[$key]['custom/JavaScript/disable_tooltips.js
'] = 'include/javascript/sugar_grp7.min.js';
        }
        break;
    }
}
```

Note: More information about JSGroupings can be found here.

Quick Repair and Rebuild

After creating the ./custom/JavaScript/disable_tooltips.js and ./custom/Extension/application/Ext/JSGroupings/disable_tooltips.php files, navigate to Admin > Repairs and perform a "Quick Repair and Rebuild". This will rebuild the cached files to fully implement the changes. Once "Quick Repair and Rebuild" finishes, the tooltips will be disabled through Sugar.

Dynamically Hiding Subpanels Based on Record Values

Overview

This article will demonstrate how to dynamically hide subpanels that are dependent on record values by overriding the subpanels layout for a module. This example is for an account record where we will hide the Contacts subpanel when the account type is not 'Customer'.

Example

To accomplish this, we will create a JavaScript controller that extends SubpanelsLayout. This controller will then monitor the model and control the display of the subpanels.

./custom/modules/Accounts/clients/base/layouts/subpanels/subpanels.js

Removed script tag to: https://gist.github.com/3722771.js?file=view.detail.php

```
  ({
    extendsFrom: 'SubpanelsLayout',

    _hiddenSubpanels: [],

    initialize: function(options) {
        this._super('initialize', [options]);
        this.registerModelEvents();
    },

    /**
     * Add the model change events for fields that determine when a su
bpanel should be hidden
     */
    registerModelEvents: function(){
        this.model.on('change:account_type',function(model) {
            var link = 'contacts';
            if (model.get('account_type') == "Customer"){
                this.unhideSubpanel(link);
            }else{
                this.hideSubpanel(link);
            }
        },this);
    },
```

```
    /**
     * Override showSubpanel to re-
hide subpanels when outside changes occur, like reordering subpanel
     * @inheritdoc
     */
    showSubpanel: function(linkName) {
        this._super('showSubpanel',[linkName]);

        _.each(this._hiddenSubpanels, function(link) {
            this.hideSubpanel(link);
        },this);
    },

    /**
     * Helper for getting the Subpanel View for a specific link
     */
    getSubpanelByLink: function(link){
        return this._components.find(function(component){
           return component.context.get('link') === link;
        });
    },

    /**
     * Add to the _hiddenSubpanels array, and hide the subpanel
     */
    hideSubpanel: function(link){
        this._hiddenSubpanels.push(link);
        var component = this.getSubpanelByLink(link);
        if (!_.isUndefined(component)){
            component.hide();
        }
        this._hiddenSubpanels = _.unique(this._hiddenSubpanels);
    },

    /**
     * Unhide the Subpanel and remove from _hiddenSubpanels array
     */
    unhideSubpanel: function(link){
        var index = this._hiddenSubpanels.findIndex(function(l){
            return l == link;
        });
        if (_.isUndefined(index)){
            delete this._hiddenSubpanels[index];
        }
        var component = this.getSubpanelByLink(link);
```

```
        if (!_.isUndefined(component)){
            component.show();
        }
    }
})
```

Once the file is in place, run a Quick Repair and Rebuild. The changes will then be reflected in the Accounts record view.

Related

Enabling Importing for Custom Modules

Overview

When designing a custom module in Module Builder, you have the option to enable importing for the module. If the custom module is deployed without enabling this option, it is not recommended that you redeploy the module since any changes made in Studio and potentially other areas of the application could be lost. This article will cover how to enable importing for custom modules via a code-level change to preserve any additional configurations made to the module since being deployed from Module Builder.

Steps to Complete

To enable importing for your custom module, you must modify certain PHP files depending on the version of Sugar that you have. Please note that the instructions below apply to custom modules created via Admin > Module Builder and are not applicable to any stock modules which come out-of-the-box with Sugar. All of the directory paths are relative to the root directory of Sugar on the web server and require you to replace the <module_key> and <module_name> variables with appropriate values for your situation. For instance, if your module is installed in the directory of ./modules/abc_custom_module/ then the <module_key> would be abc and <module_name> would be custom_module.

1. Edit the ./modules/<module_key>_<module_name>/<module_key>_<module_name>_sugar.php file in your Sugar file system.
2. Around line 24 of the file, locate and change public $importable = false; to public $importable = true;.
3. Save your changes to the file.
4. Next, edit the ./modules/<module_key>_<module_name>/clients/base/menus/header/header.php file which should look similar to this:

```
$viewdefs[$moduleName]['base']['menu']['header'] = array(
  array(
    'route' => "#$moduleName/create",
    'label' => 'LNK_NEW_RECORD',
    'acl_action' => 'create',
    'acl_module' => $moduleName,
    'icon' => 'fa-plus',
  ),
  array(
    'route' => "#$moduleName",
    'label' => 'LNK_LIST',
    'acl_action' => 'list',
    'acl_module' => $moduleName,
    'icon' => 'fa-bars',
  ),
);
```

5. Add the following line to the end of the file after the last), but before the ending);:

```
array(
 'route' => "#bwc/index.php?module=Import&action=Step1&import_mo
dule=$moduleName&return_module=$moduleName&return_action=index",
 'label' => 'LBL_IMPORT',
 'acl_action' => 'import',
 'acl_module' => $moduleName,
 'icon' => 'icon-upload',
 ),
```

6. Save your changes to the file. The updated file should then look similar to this:

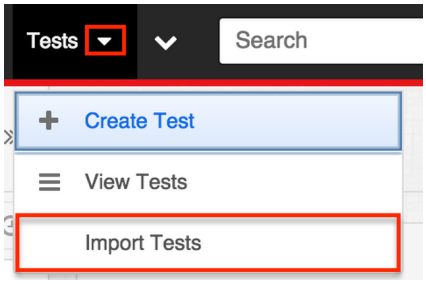
```
$viewdefs[$moduleName]['base']['menu']['header'] = array(
  array(
    'route' => "#$moduleName/create",
    'label' => 'LNK_NEW_RECORD',
    'acl_action' => 'create',
    'acl_module' => $moduleName,
    'icon' => 'fa-plus',
  ),
  array(
    'route' => "#$moduleName",
```

```
        'label' => 'LNK_LIST',
        'acl_action' => 'list',
        'acl_module' => $moduleName,
        'icon' => 'fa-bars',
      ),
      array(
        'route' => "#bwc/index.php?module=Import&action=Step1&import
_module=$moduleName&return_module=$moduleName&return_action=inde
x",
        'label' => 'LBL_IMPORT',
        'acl_action' => 'import',
        'acl_module' => $moduleName,
        'icon' => 'icon-upload',
     ),
    );
```

Once the necessary changes have been made, log into Sugar and navigate to Admin > Repair and perform a "Quick Repair and Rebuild". This will rebuild the cached files to fully implement the changes.

Application

After making these changes, the "Import {Module Name}" option will appear in the Actions menu of the custom module's module tab. Simply click the triangle in the module tab, then select the Import option to create or update records for your custom module. For instructions on using Sugar's Import tool, please refer to the Import documentation.

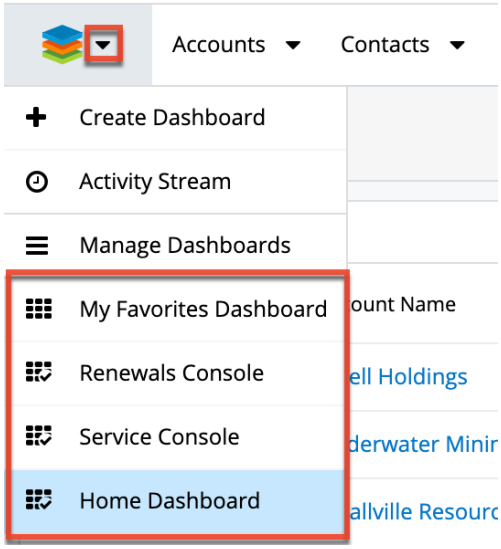


Increasing the Number of Dashboards Displayed in the Home Menu

Overview

By default, the "Home Dashboard" comes out-of-the-box with Sugar® to display on

the home page. In addition, Sugar users with a Sugar Serve, Sugar Sell, or Enterprise license type will have access to specialized Home page dashboards called "Service Console" or "Renewals Console". Users have the ability to create new dashboards on their home page and build out its layout and dashlet set. Currently, Sugar imposes a limit of 50 dashboards that can be displayed under the Home module tab. So, when creating multiple dashboards, keep in mind that having more than 50 in the list of available dashboards will cause the ones in the beginning to be dropped off the list. This article covers how to increase the number of dashboards allowed in the Home menu via a code-level customization.



Use Case

In this example, we will make a code-level change to set the number of dashboards allowed in the Home menu to "80".

Prerequisites

This change requires code-level customizations. You will need direct access to the server as well as administrator access in Sugar in order to perform the necessary actions. If you need assistance making these changes and already have a relationship with a Sugar partner, you can work with them to make this change. If not, please refer to the Partner Page to find a reselling partner to help with your development needs.

You will also need the following capabilities prior to making this code-level customization:

- You should have a working knowledge of PHP development and arrays.
- You should have access to a text editor that can be used for programming.
- If your Sugar instance is hosted on Sugar's cloud service or hosted on a server to which you do not have direct file access, you will need to know

how to create and deploy Module Loader packages. For more information, please refer to the Module Loader section of the Developer Guide.

Note: Sugar Sell Essentials customers do not have the ability to upload custom file packages to Sugar using Module Loader.

Steps to Complete

The following steps cover setting the number of dashboards allowed on the Home menu to "80" as an example:

1. Navigate to ./modules/Home/clients/base/views/module-menu/module-menu.php and copy the file contents to a new file at ./custom/modules/Home/clients/base/views/module-menu/module-menu.php. The contents of the file should look similar to this:

```
<?php

$viewdefs['Home']['base']['view']['module-menu'] = array(
  'settings' => array(
     'favorites' => 0,
     'recently_viewed' => 10,
     'recently_viewed_toggle' => 3,
  ),
);
?>
```

2. Modify the ./custom/modules/Home/clients/base/views/module-menu/module-menu.php file and add the dashboards => 80, element to the array. The modified file should look similar to this:

```
<?php

$viewdefs['Home']['base']['view']['module-menu'] = array(
  'settings' => array(
     'dashboards' => 80,
     'favorites' => 0,
     'recently_viewed' => 10,
     'recently_viewed_toggle' => 3,
  ),
);
?>
```

3. Save the changes to the file and update the ownership and permissions of the files that were created.
    - For more information on Linux-based stacks, please refer to the Required File System Permissions on Linux article.
    - For more information on Windows-based stacks, please refer to the Required File System Permissions on Windows With IIS article.
4. Finally, log into Sugar as an administrator, navigate to Admin > Repair, and perform a "Quick Repair and Rebuild". This will rebuild the cached files to fully implement the changes to your Sugar instance.

Application

Once the appropriate changes have been made, users should be able to create and view up to 80 dashboards on their home page and any previously missing dashboards should now display in the Home menu as well.

Logic Hooks

These pages demonstrate some common examples of working with logic hooks in Sugar.

Comparing Bean Properties Between Logic Hooks

Overview

How to compare the properties of a bean between the before_save and after_save logic hooks

Storing and Comparing Values

When working with a bean in the after_save logic hook, you may notice that the after_save fetched rows no longer match the before_save fetched rows. If your after_save logic needs to be able to compare values that were in the before_save, you can use the following example to help you store and use the values.

./custom/modules/<module>/logic_hooks.php

```
  <?php

$hook_version = 1;
$hook_array = Array();

$hook_array['before_save'] = Array();
$hook_array['before_save'][] = Array(
    1,
    'Store values',
    'custom/modules/<module>/My_Logic_Hooks.php',
    'My_Logic_Hooks',
    'before_save_method'
);

$hook_array['after_save'] = Array();
$hook_array['after_save'][] = Array(
    1,
    'Retrieve and compare values',
    'custom/modules/<module>/My_Logic_Hooks.php',
    'My_Logic_Hooks',
    'after_save_method'
);

?>
```

./custom/modules/<module>/My_Logic_Hooks.php

```
  <?php

if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry Poin
t');

class My_Logic_Hooks
{
    function before_save_method($bean, $event, $arguments)
    {
        //store as a new bean property
        $bean->stored_fetched_row_c = $bean->fetched_row;
    }

    function after_save_method($bean, $event, $arguments)
    {
```

```
        //check if a fields value has changed
        if (
            isset($bean->stored_fetched_row_c)
            && $bean->stored_fetched_row_c['field'] != $bean->field
           )
        {
            //execute logic
        }
    }
}

?>
```

Preventing Infinite Loops with Logic Hooks

Overview

Infinite loops often happen when a logic hook calls save on a bean in a scenario that triggers the same hook again. This example shows how to add a check to a logic hook to eliminate perpetual looping.

Saving in an After Save Hook

Infinite loops can sometimes happen when you have a need to update a record after it has been run through the workflow process in the after_save hook. Here is an example of a looping hook:

./custom/modules/Accounts/logic_hooks.php

```
  <?php

$hook_version = 1;
$hook_array = Array();
$hook_array['after_save'] = Array();
$hook_array['after_save'][] = Array(
    1,
    'Update Account Record',
    'custom/modules/Accounts/Accounts_Hook.php',
    'Accounts_Hook',
    'update_self'
);
```

./custom/modules/Accounts/Accounts_Hook.php

```
  <?php

if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry Poin
t');

class Accounts_Hook
{
    function update_self($bean, $event, $arguments)
    {
        //generic condition
        if ($bean->account_type == 'Analyst')
        {
            //update
            $bean->industry = 'Banking';
            $bean->save();
        }
    }
}
```

In the example above, there is a condition that, when met, will trigger the update of a field on the record. This will cause an infinite loop because calling the save() method will trigger once during the before_save hook and then again during the after_save hook. The solution to this problem is to add a new property on the bean to ignore any unneeded saves. For this example, we will name the property "ignore_update_c" and add a check to the logic hook to eliminate the perpetual loop. An example of this is shown below:

./custom/modules/Accounts/Accounts_Hook.php

```
  <?php

if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry Poin
t');

class Accounts_Hook
{
    function update_self($bean, $event, $arguments)
    {
        //loop prevention check
        if (!isset($bean->ignore_update_c) || $bean->ignore_update_c =
== false)
        {
```

```
            //generic condition
            if ($bean->account_type == 'Analyst')
            {
                //update
                $bean->ignore_update_c = true;
                $bean->industry = 'Banking';
                $bean->save();
            }
        }
    }
}
```

Related Record Save Loops

Sometimes logic hooks on two separate but related modules can cause an infinite loop. The problem arises when the two modules have logic hooks that update one another. This is often done when wanting to keep a field in sync between two modules. The example below will demonstrate this behavior on the accounts:contacts relationship:

./custom/modules/Accounts/logic_hooks.php

```
  <?php

$hook_version = 1;
$hook_array = Array();
$hook_array['before_save'] = Array();
$hook_array['before_save'][] = Array(
    1,
    'Handling associated Contacts records',
    'custom/modules/Accounts/Accounts_Hook.php',
    'Accounts_Hook',
    'update_contacts'
);
```

./custom/modules/Accounts/Accounts_Hook.php

```
  <?php

if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry Poin
t');
```

```
class Accounts_Hook
{
    function update_contacts($bean, $event, $arguments)
    {
        //if relationship is loaded
        if ($bean->load_relationship('contacts'))
        {
            //fetch related beans
            $relatedContacts = $bean->contacts->getBeans();

            foreach ($relatedContacts as $relatedContact)
            {
                //perform any other contact logic
                $relatedContact->save();
            }
        }
    }
}
```

The above example will loop through all contacts related to the account and save them. At this point, the save will then trigger our contacts logic hook shown below:

./custom/modules/Contacts/logic_hooks.php

```
  <?php

$hook_version = 1;
$hook_array = Array();
$hook_array['before_save'] = Array();
$hook_array['before_save'][] = Array(
    1,
    'Handling associated Accounts records',
    'custom/modules/Contacts/Contacts_Hook.php',
    'Contacts_Hook',
    'update_account'
);
```

./custom/modules/Contacts/Contacts_Hook.php

```
  <?php

if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry Poin
```

```
t');

class Contacts_Hook
{
    function update_account($bean, $event, $arguments)
    {

        //if relationship is loaded
        if ($bean->load_relationship('accounts'))
        {
            //fetch related beans
            $relatedAccounts = $bean->accounts->getBeans();

            $parentAccount = false;
            if (!empty($relatedAccounts))
            {
                //order the results
                reset($relatedAccounts);

                //first record in the list is the parent
                $parentAccount = current($relatedAccounts);
            }

            if ($parentAccount !== false && is_object($parentAccount))
            {
                //perform any other account logic
                $parentAccount->save();
            }
        }
    }
}
```

These two logic hooks will continuously trigger one another in this scenario until you run into a max_execution or memory_limit error. The solution to this problem is to add a new property on the bean to ignore any unneeded saves. In our example, we will name this property "ignore_update_c" and add a check to our logic hook to eliminate the perpetual loop. The following code snippets are copies of the two classes with the ignore_update_c property added.

./custom/modules/Accounts/Accounts_Hook.php

```
  <?php

if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry Poin
```

```
t');

class Accounts_Hook
{
    function update_contacts($bean, $event, $arguments)
    {
        //if relationship is loaded
        if ($bean->load_relationship('contacts'))
        {
            //fetch related beans
            $relatedContacts = $bean->contacts->getBeans();

            foreach ($relatedContacts as $relatedContact)
            {
                //check if the bean's ignore_update_c attribute is not
 set
                if (!isset($bean->ignore_update_c) || $bean->ignore_up
date_c === false)
                {
                    //set the ignore update attribute
                    $relatedContact->ignore_update_c = true;

                    //perform any other contact logic
                    $relatedContact->save();
                }
            }
        }
    }
}
```

./custom/modules/Contacts/Contacts_Hook.php

```
  <?php

if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry Poin
t');

class Contacts_Hook
{
    function update_account($bean, $event, $arguments)
    {
        //if relationship is loaded
        if ($bean->load_relationship('accounts'))
        {
```

```
            //fetch related beans
            $relatedAccounts = $bean->accounts->getBeans();

            $parentAccount = false;
            if (!empty($relatedAccounts))
            {
                //order the results
                reset($relatedAccounts);

                //first record in the list is the parent
                $parentAccount = current($relatedAccounts);
            }

            if ($parentAccount !== false && is_object($parentAccount))
            {
                //check if the bean's ignore_update_c element is set
                if (!isset($bean->ignore_update_c) || $bean->ignore_up
date_c === false)
                {
                    //set the ignore update attribute
                    $parentAccount->ignore_update_c = true;

                    //perform any other account logic
                    $parentAccount->save();
                }
            }
        }
    }
}
```

Modifying Calendar Item Colors

Overview

Each calendar event type (Meetings, Calls, and Tasks) has its own distinct color scheme. This article will review how to modify these colors, as well as demonstrate how to set the event color based on the event status.

The color scheme for each calendar activity type is defined in the stock file ./modules/Calendar/CalendarDisplay.php in the $activity_colors array as shown below:

```
  public $activity_colors = array(
 'Meetings' => array(
```

```
  'border' => '#1C5FBD',
  'body' => '#D2E5FC',
 ),
 'Calls' => array(
  'border' => '#DE4040',
  'body' => '#FCDCDC',
 ),
 'Tasks' => array(
  'border' => '#015900',
  'body' => '#B1F5AE',
 ),
);
```

This file can not be overridden or extended in an upgrade-safe way, but this array can still be overridden later in the rendering process allowing for an upgrade safe customization.

Modifying the Calendar Activity Colors

The CalendarDisplay class passes the $activity_colors array to javascript by way of the ./modules/Calendar/tpls/main.tpl Smarty template.

In this template file, you will see the following code where the CAL.activity_colors object is setup:

```
  CAL.activity_colors = [];
{foreach name=colors from=$activity_colors key=module item=v}
 CAL.activity_colors['{$module}'] = [];
 CAL.activity_colors['{$module}']['border'] = '{$v.border}';
 CAL.activity_colors['{$module}']['body'] = '{$v.body}'
{/foreach}
```

This is where we can update the $activity_colors array in an upgrade safe manner. To do this, copy the template file from./modules/Calendar/tpls/main.tpl to ./custom/modules/Calendar/tpls/main.tpl.

If we wanted to change Meetings to have a purple color scheme, we might choose:

border: #580059
body : #F2AEF5

To implement this, we'll update the activity colors array in the custom template file as follows:

./custom/modules/Calendar/tpls/main.tpl

```
  CAL.activity_colors = [];
{foreach name=colors from=$activity_colors key=module item=v}
 CAL.activity_colors['{$module}'] = [];
 CAL.activity_colors['{$module}']['border'] = '{$v.border}';
 CAL.activity_colors['{$module}']['body'] = '{$v.body}'
{/foreach}

{literal}

CAL.activity_colors["Meetings"] = {
 "border": "#580059",
 "body": "#F2AEF5"
};

{/literal}
```

In the above code, the activity_colors entry for "Meetings" has specifically been altered to set the custom colors just for Meetings, leaving the other activity types as they were. Alternatively, the entire activity_colors array could be updated, replacing the original coming from CalendarDisplay. For example:

./custom/modules/Calendar/tpls/main.tpl

```
  CAL.activity_colors = [];
{foreach name=colors from=$activity_colors key=module item=v}
 CAL.activity_colors['{$module}'] = [];
 CAL.activity_colors['{$module}']['border'] = '{$v.border}';
 CAL.activity_colors['{$module}']['body'] = '{$v.body}'
{/foreach}

{literal}

CAL.activity_colors = {
  "Meetings": {
   "border": "#580059",
   "body": "#F2AEF5"
  },
  "Calls": {
   "border": "#DE4040",
```

```
   "body": "#FCDCDC"
  },
  "Tasks": {
   "border": "#015900",
   "body": "#B1F5AE"
  }
 };

{/literal}
```

While Calls and Tasks have not been modified from their original colors, having the entire array defined here may make it easier to make further color customizations later on, since the modified activity_colors will be completely defined in the custom template file.

After making the above changes, run a Quick Repair and Rebuild and reload the Calendar module in your browser to see the changes take effect.

Customizing colors based on event status

If you wanted to go a step further, you might want to customize the colors based on the status of the activity. For example, perhaps for Meetings, the color should reflect if the Meeting status is Scheduled, Held, or Canceled. To achieve this, we will need to customize the javascript that actually renders the activities onto the calendar view, so we will copy ./modules/Calendar/Cal.js to ./custom/modules/Calendar/Cal.js.

Additionally, in order for our custom template to load the custom Cal.js, it will need to be updated to point to the custom Cal.js by changing the line:

```
  {sugar_getscript file="modules/Calendar/Cal.js"}
```

to

```
  {sugar_getscript file="custom/modules/Calendar/Cal.js"}
```

The activity colors from the CAL.activity_colors object get applied in the Cal.js script at:

```
  el.style.backgroundColor = CAL.activity_colors[item.module_name]['bo
```

```
dy'];
el.style.borderColor = CAL.activity_colors[item.module_name]['border']
;
```

The activity type (Meetings, Calls, Tasks) comes from item.module_name, as shown above. The activity's status can be found in the item.status variable. Using these, we can modify the color-setting logic to apply a status-specific color scheme to the Meeting items in the calendar, for example:

```
  if (item.module_name =="Meetings") {
    el.style.backgroundColor = CAL.activity_colors[item.module_name][i
tem.status]['body'];
    el.style.borderColor     = CAL.activity_colors[item.module_name][i
tem.status]['border'];
}
else {
    el.style.backgroundColor = CAL.activity_colors[item.module_name]['
body'];
    el.style.borderColor     = CAL.activity_colors[item.module_name]['
border'];
}
```

This assumes that the 'Meetings' part of CAL.activity_colors has status-specific colors set. Therefore we will need to update our custom template file so that this is the case:

./custom/modules/Calendar/tpls/main.tpl

```
  {literal}

CAL.activity_colors = {
    "Meetings": {
        "Planned": {
            "border": "#FF6666",
            "body": "#FF6666"
        },
        "Held": {
            "border": "#FF9999",
            "body": "#FF9999"
        },
        "Not Held": {
            "border": "#6C8CD5",
```

```
            "body": "#6C8CD5"
        }
    },
    "Calls": {
        "border": "#DE4040",
        "body": "#FCDCDC"
    },
    "Tasks": {
        "border": "#015900",
        "body": "#B1F5AE"
    },
    "Planned": {
        "border": "#ff6666",
        "body": "#ff6666"
    },
    "Held": {
        "border": "#FF9999",
        "body": "#FF9999"
    },
    "Not Held": {
        "border": "#6C8CD5",
        "body": "#6C8CD5"
    }
};

{/literal}
```

After making the above changes, run Rebuild JS Grouping Files followed by a Quick Repair and Rebuild, then reload the Calendar module in your browser to see the changes take effect.

Modifying Layouts to Display Additional Columns

Overview

By default, the record view layout for each module displays two columns of fields. The number of columns to display can be customized on a per-module basis with the following steps.

Resolution

First, you will want to ensure your layouts are deployed in the custom directory. If

you have not previously customized your layouts via Studio, go to Admin > Studio > {Module Name} > Layouts. From there, select each layout you wish to add additional columns to and click 'Save & Deploy'. This action will create a corresponding layout file under the ./custom/modules/{Module Name}/clients/base/views/record/ directory. The file will be named record.php.

In your custom record.php file, locate the maxColumns value under templateMeta array, and change it to the number of columns you would like to have on screen:

```
  'maxColumns' => '3',
```

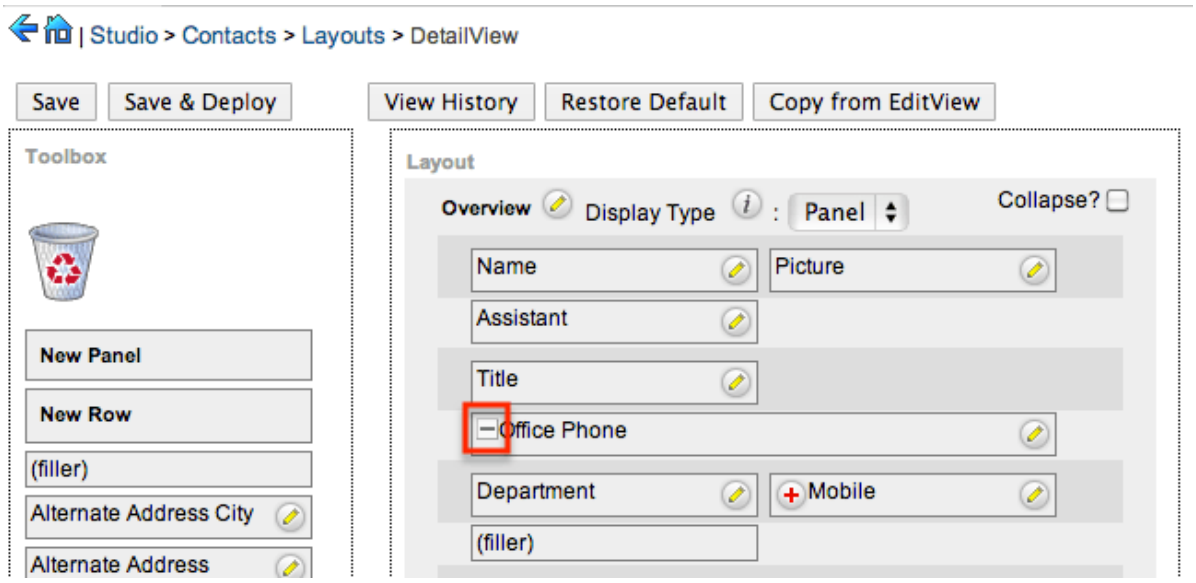
Once that is updated, locate the widths array to define the spacing for your each column(s). You should have a label and field entry for each column in your layout:

```
  'widths' => array (
    0 => array (
        'label' => '10',
        'field' => '30',
    ),
    1 => array (
        'label' => '10',
        'field' => '30',
    ),
    2 => array (
        'label' => '10',
        'field' => '30',
    ),
),
```

Next, under the existing panels array, each entry will already have a columns property, with a value of 2. Update each of these to the number of columns you set for maxColumns :

```
  'columns' => 3,
```

Once the file has been modified, you can navigate to Studio and add fields to your new column in the layout. For any rows that already contain two fields, the second field will automatically span the second and third column. Simply click the minus (-) icon to contract the field to one column and expose the new column space:



After you have added the desired fields in Studio, click 'Save & Deploy' to finalize your changes

Column Widths

Studio does not support expanding a field beyond two column-widths. If you want a field to span the full layout (all three columns), you would need to manually update this in the custom record.php like so:

```
  array (
    'name' => 'description',
    'span' => 12,
)
```

Changing the span value from 8 to 12. After saving the changes on the file, you would then run a Quick Repair and Rebuild. However, any updates in Studio will revert the span back to 8.

Modifying Subpanel Buttons

Overview

Several buttons exist on each subpanel by default such as "Create" (+ Plus symbol) and "Link Existing Record". These buttons are controlled by the panel-top view for

each module. To make changes to the buttons included in a subpanel layout, you will need to create an override for the panel-top view in ./custom/Extension/modules/<module>/Ext/clients/base/views/panel-top/ and update the corresponding buttons property to include or exclude a button.

If you need to make changes to a specific relationships subpanel, then you will need to modify the subpanels layout definition for the relationship to point to a custom panel-top view that you define.

Steps to Complete

Editing Subpanel Buttons

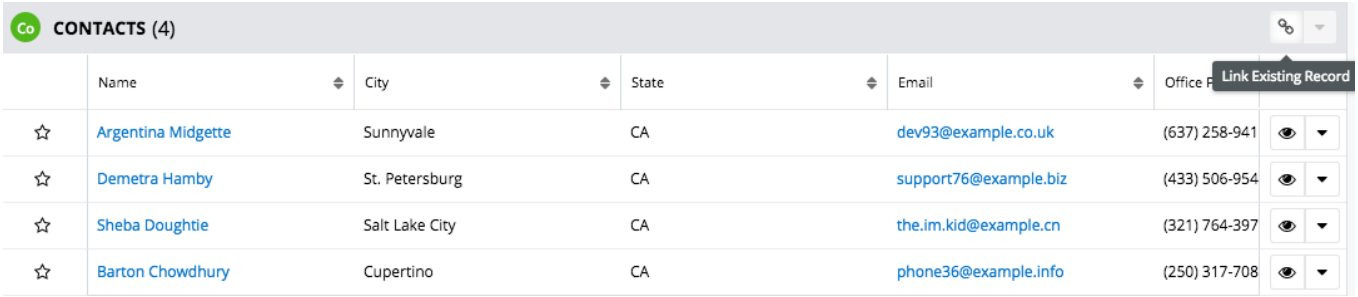
In this example, we will remove the "Create" button from the Contacts subpanel for all modules. This code snippet uses the same button definition that the create button uses. It has been changed to trigger the "Link Existing Record" action by setting the button type to link-action. The blank array after the initial button is just to assure that the drop-down button displays next to it as disabled for placement purposes.

1. Create the file custom/Extension/modules/Contacts/Ext/clients/base/views/panel-top/panel-top.php as follows:

```
<?php

$viewdefs['Contacts']['base']['view']['panel-
top']['buttons'] = array(
    array(
        'type' => 'actiondropdown',
        'name' => 'panel_dropdown',
        'css_class' => 'pull-right',
        'buttons' => array(
            array(
                'type' => 'link-action',
                'icon' => 'fa-link',
                'name' => 'select_button',
                'label' => ' ',
                'tooltip' => 'LBL_ASSOC_RELATED_RECORD',
            ),
            array(
            ),
        ),
    ),
);
```

2. Navigate to Admin > Repair and click "Quick Repair and Rebuild".

The subpanel will appear as shown below for all Contacts subpanels.



Using a Custom Panel-Top View for a Specific Relationship

In this example, we will only remove the "Create" action from a specific relationships subpanel. Specifically, the example will look at Contacts subpanel on the Accounts module and implement the above changes explicitly for that relationship.

1. Create the custom panel-top view in ./custom/modules/Contacts/clients/base/views/panel-top-for-accounts/panel-top-for-accounts.php as follows:

```
<?php

$viewdefs['Contacts']['base']['view']['panel-top-for-
accounts'] = array(
    'type' => 'panel-top',
    'template' => 'panel-top',
    'buttons' => array(
        array(
            'type' => 'actiondropdown',
            'name' => 'panel_dropdown',
            'css_class' => 'pull-right',
            'buttons' => array(
                array(
                    'type' => 'link-action',
                    'icon' => 'fa-link',
                    'name' => 'select_button',
                    'label' => ' ',
```

```
                    'tooltip' => 'LBL_ASSOC_RELATED_RECORD',
                ),
                array(
                ),
            ),
        ),
    ),
);
```

2. Create the subpanel panel-top override definition for Accounts in ./custom/modules/Accounts/clients/base/layouts/subpanels/subpanels.php as follows:

```
<?php

require 'modules/Accounts/clients/base/layouts/subpanels/subpanels.php';

foreach($viewdefs['Accounts']['base']['layout']['subpanels']['components'] as $key => $component){
    if ($component['context']['link'] == 'contacts'){
        $viewdefs['Accounts']['base']['layout']['subpanels']['components'][$key]['override_paneltop_view']  = 'panel-top-for-accounts';
        break;
    }
}
```

3. Navigate to Admin > Repair and click "Quick Repair and Rebuild".

Your changes will now be reflected in the system.

Module Loadable Packages

Examples working with the module loadable packages.

Creating an Installable Package for a Logic Hook

Overview

These examples will demonstrate how to create module loadable packages for logic hooks based on different scenarios.

Logic Hook File vs. Logic Hook Definition

No matter the deployment method, all logic hooks require being registered (meaning they are attached to the $hook_array array), defining the logic hook event, module, file to load, and class and function to run for that logic hook event. For all deployment options, the logic hook definition points to a file where the class and function to run is located, and this file must be copied to the instance via the copy installdefs of the manifest. The bulk of this article will focus on the various options for registering the logic hook definition via the package manifest. But all deployment strategies will assume the same logic hook definition, the same logic hook file, and will use the same copy array to move the logic hook file into the instance.

Note: More information on the $manifest or $installdef can be found in the Introduction to the Manifest .

The Logic Hook

This logic hook will install a before_save hook to the accounts module that will default the account name to 'My New Account Name ({time stamp})'. This various ways of installing this hook will be shown in the following sections.

Logic Hook Definition

<basepath>/Files/custom/Extension/Accounts/Ext/LogicHooks/account_name_hook.php

```
  <?php

$hook_array['before_save'][] = Array(
    99,
    'Example Logic Hook - Updates account name',
    'custom/modules/Accounts/accounts_save.php',
    'Accounts_Save',
    'updateAccountName'
);
```

Logic Hook File

<basepath>/Files/custom/modules/Accounts/accounts_save.php

```
  <?php

if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry Poin
```

```
t');

class Accounts_Save
{
 function updateAccountName($bean, $event, $arguments)
 {
  $bean->name = "My New Account Name (" . time() . ")";
 }
}
```

Distributed Plugin

If the package being built is intended to be distributed to various Sugar instances as part of a distributed plugin, your logic hook definition should be created as part of the extension framework using the LogicHooks Extension, which can be done using $installdefs['logic_hooks'] in the package manifest. When using this extension, the logic hook will be installed and uninstalled along with the plugin.

Note: More information on the $installdefs['logic_hooks'] can be found in LogicHooks Extensions.

Package Example

The following files are relative to the root of the .zip archive which is referred to at the basepath. This example package can be downloaded here.

<basepath>/manifest.php

```
  <?php

$manifest = array(
    'acceptable_sugar_flavors' => array(
        'PRO',
        'ENT',
        'ULT'
    ),
    'acceptable_sugar_versions' => array(
        'exact_matches' => array(),
        'regex_matches' => array(
            '12.0.*
        '),
    ),
    'author' => 'SugarCRM',
```

```
    'description' => 'Installs a sample logic hook - using extension f
ramework',
    'icon' => '',
    'is_uninstallable' => true,
    'name' => 'Example Logic Hook Installer - Using Extension Framewor
k',
    'published_date' => '2018-01-01 00:00:00',
    'type' => 'module',
    'version' => '1.0.0',
);

$installdefs = array(
 'id' => 'package_123',
 'copy' => array(
  array(
   'from' => '<basepath>/Files/custom/modules/Accounts/accounts_save.p
hp',
   'to' => 'custom/modules/Accounts/accounts_save.php',
  ),
 ),
 'hookdefs' => array(
  array(
   'from' => '<basepath>/Files/custom/Extension/Accounts/Ext/LogicHook
s/account_name_hook.php',
   'to_module' => 'Accounts',
  )
 ),
);
```

<basepath>/Files/custom/Extension/Accounts/Ext/LogicHooks/account_name_hook.php

```
  <?php

$hook_array['before_save'][] = Array(
    99,
    'Example Logic Hook - Updates account name',
    'custom/modules/Accounts/accounts_save.php',
    'Accounts_Save',
    'updateAccountName'
);
```

<basepath>/Files/custom/modules/Accounts/accounts_save.php

```
  <?php

if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry Poin
t');

class Accounts_Save
{
 function updateAccountName($bean, $event, $arguments)
 {
  $bean->name = "My New Account Name (" . time() . ")";
 }
}
```

Promotion Process

If the package being built is intended to be part of a code promotion process, you are welcome to use the LogicHooks Extension as mentioned above or you may find it easier to update the ./custom/modules/<module>/logic_hooks.php file, which can be done using the $installdefs['logic_hooks'] in the package manifest.

Package Example

The following files are relative to the root of the .zip archive which is referred to at the basepath. This example package can be downloaded here.

<basepath>/manifest.php

```
  <?php

$manifest = array(
        'acceptable_sugar_flavors' => array(
            'PRO',
            'ENT',
            'ULT'
        ),
        'acceptable_sugar_versions' => array(
            'exact_matches' => array(),
            'regex_matches' => array(
                '12.0.*
            '),
        ),
```

```
 'author' => 'SugarCRM',
 'description' => 'Installs a sample logic hook - using logic_hooks in
stalldefs',
 'icon' => '',
 'is_uninstallable' => true,
 'name' => 'Example Logic Hook Installer - Using logic_hooks installde
fs',
 'published_date' => '2018-01-01 00:00:00',
 'type' => 'module',
 'version' => '1.0.0',
);

$installdefs = array(
 'id' => 'package_456',
 'copy' => array(
  array(
   'from' => '<basepath>/Files/custom/modules/Accounts/accounts_save.p
hp',
   'to' => 'custom/modules/Accounts/accounts_save.php',
  ),
 ),
 'logic_hooks' => array(
  array(
   'module' => 'Accounts',
   'hook' => 'before_save',
   'order' => 99,
   'description' => 'Example Logic Hook - Updates account name',
   'file' => 'custom/modules/Accounts/accounts_save.php',
   'class' => 'Accounts_Save',
   'function' => 'updateAccountName',
  ),
 ),
);
```

<basepath>/Files/custom/modules/Accounts/accounts_save.php

```
  <?php

if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry Poin
t');

class Accounts_Save
{
 function updateAccountName($bean, $event, $arguments)
```

```
 {
  $bean->name = "My New Account Name (" . time() . ")";
 }
}
```

Conditionally Add or Remove Logic Hook via post_execute script

A developer may want to be able to trigger the addition or removal of logic hooks from a customization. This can be done using a post execute script with the check_logic_hook_file and remove_logic_hook_file functions.

It is recommended to be cautious when doing this and limit any updates to off-hours as not to affect system behavior for users.

Note: remove_logic_hook_file only removes hooks defined in the given module's logic_hooks.php file. check_logic_hook_file only checks hooks defined in the given module's logic_hooks.php file. Neither check or remove hooks defined via the Logic Hook Extension framework. Thus this deployment option should be considered a subset of the Promotion Process deployment option listed earlier.

Package Example

Note: For demonstration purposes, in the post_execute script, the Accounts_Save logic hook from the earlier example package is unregistered (if it exists) and a second logic hook Accounts_Save_Enterprise is registered. The second logic hook points to a different logic hook file, which is moved to the instance using copy in the manifest.

The following files are relative to the root of the .zip archive which is referred to at the basepath. This example package can be downloaded here.

<basepath>/manifest.php

```
  <?php

$manifest = array(
        'acceptable_sugar_flavors' => array(
            'PRO',
            'ENT',
            'ULT'
        ),
        'acceptable_sugar_versions' => array(
```

```
            'exact_matches' => array(),
            'regex_matches' => array(
                '12.0.*
            '),
        ),
 'author' => 'SugarCRM',
 'description' => 'Installs a sample logic hook - using post_execute c
heck_logic_hook_file',
 'icon' => '',
 'is_uninstallable' => true,
 'name' => 'Example Logic Hook Installer - Using post_execute check_lo
gic_hook_file',
 'published_date' => '2018-01-01 00:00:00',
 'type' => 'module',
 'version' => '1.0.0',
);

$installdefs = array(
 'id' => 'package_789',
 'copy' => array(
  array(
   'from' => '<basepath>/Files/custom/modules/Accounts/accounts_save_e
nterprise.php',
   'to' => 'custom/modules/Accounts/accounts_save_enterprise.php',
  ),
 ),
 'post_execute' => array(
   '<basepath>/change_hooks.php',
 ),

);
```

<basepath>/Files/custom/modules/Accounts/accounts_save_enterprise.php

```
  <?php

if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry Poin
t');

class Accounts_Save_Enterprise
{
 function updateAccountName($bean, $event, $arguments)
 {
  $bean->name = "My New Account Name (" . time() . ") -- Enterprise";
```

```
 }
}
```

<basepath>/change_hooks.php

```
  <?php

if(!defined('sugarEntry')) define('sugarEntry', true);

require_once 'include/entryPoint.php';

$accounts_hook_pro = array(
 99,
 'Example Logic Hook - Updates account name',
 'custom/modules/Accounts/accounts_save.php',
 'Accounts_Save',
 'updateAccountName'
);

$accounts_hook_ent = array(
 99,
 'Example Logic Hook - Updates account name - Enterprise',
 'custom/modules/Accounts/accounts_save_enterprise.php',
 'Accounts_Save_Enterprise',
 'updateAccountName'
);

if( $GLOBALS['sugar_flavor'] == 'ENT' ) {

 //unregister Accounts_Save logichook
 remove_logic_hook("Accounts", "before_save", $accounts_hook_pro);

 //register Accounts_Save_Enterprise logichook
 check_logic_hook_file("Accounts", "before_save", $accounts_hook_ent);
}
```

Creating an Installable Package That Copies Files

Overview

This is an overview of how to create a module loadable package that will copy files into your instance of Sugar. This is most helpful when your instance is hosted in

Sugar's cloud environment or by a third party. For more details on the $manifest or $installdef options, you can visit the Introduction to the Manifest. This example package can be downloaded here.

Manifest Example

```
  <?php

$manifest = array(
    'acceptable_sugar_flavors' => array('PRO','ENT','ULT'),
    'acceptable_sugar_versions' => array(
        'exact_matches' => array(),
        'regex_matches' => array('(.*?)\\.(.*?)\\.(.*?)$'),
    ),
    'author' => 'SugarCRM',
    'description' => 'Copies my files to custom/src/',
    'icon' => '',
    'is_uninstallable' => true,
    'name' => 'Example File Installer',
    'published_date' => '2018-06-29 00:00:00',
    'type' => 'module',
    'version' => '1.0.0',
);

$installdefs = array(
    'id' => 'package_1530305622',
    'copy' => array(
        0 => array(
            'from' => '<basepath>/Files/custom/src/MyLibrary/MyCustomC
lass.php',
            'to' => 'custom/src/MyLibrary/MyCustomClass.php',
        ),
    ),
);
```

Copied File Example

This example copies a file to ./custom/src/MyLibrary/MyCustomClass.php, which adds the custom
namespaced class Sugarcrm\Sugarcrm\custom\MyLibrary\MyCustomClass to the application.

```
  <?php
```

```
namespace Sugarcrm\Sugarcrm\custom\MyLibrary;

use \Sugarcrm\Sugarcrm\Logger\Factory;

class MyCustomClass
{
    public function MyCustomMethod($message = 'relax')
    {
        $logger = Factory::getLogger('default');
        $logger->info('MyCustomClass says' . "'{$message}'");
    }
}
```

Creating an Installable Package that Creates New Fields

Overview

This is an overview of how to create a Module Loader package that will install custom fields to a module. This example will install a set of fields to the accounts module. The full package is downloadable here for your reference. For more details on the $manifest or $installdef options, you can visit the Introduction to the Manifest File.

Note: Sugar Sell Essentials customers do not have the ability to upload custom file packages to Sugar using Module Loader.

Manifest Example

<basepath>/manifest.php

```
  <?php

$manifest = array(
    'acceptable_sugar_flavors' => array('CE', 'PRO', 'CORP', 'ENT', 'U
LT'),
    'acceptable_sugar_versions' => array(
        'exact_matches' => array(),
        'regex_matches' => array(
            0 => '6\\.5\\.(.*?)',
            1 => '7\\.8\\.(.*?)\\.(.*?)',
            2 => '7\\.9\\.(.*?)\\.(.*?)',
```

```
            3 => '7\\.10\\.(.*?)\\.(.*?)'
        ),
    'author' => 'SugarCRM',
    'description' => 'Installs a sample set of custom fields to the ac
counts module',
    'icon' => '',
    'is_uninstallable' => true,
    'name' => 'Example Custom Field Installer',
    'published_date' => '2015-05-11 20:45:04',
    'type' => 'module',
    'version' => '1.0.0',
);

$installdefs = array(
    'id' => 'package_1341607504',
    'language' => array(
        array(
            'from' => '<basepath>/Files/Language/Accounts/en_us.lang.p
hp',
            'to_module' => 'Accounts',
            'language' => 'en_us'
        ),
    ),
    'custom_fields' => array(
        //Text
        array(
            'name' => 'text_field_example_c',
            'label' => 'LBL_TEXT_FIELD_EXAMPLE',
            'type' => 'varchar',
            'module' => 'Accounts',
            'help' => 'Text Field Help Text',
            'comment' => 'Text Field Comment Text',
            'default_value' => '',
            'max_size' => 255,
            'required' => false, // true or false
            'reportable' => true, // true or false
            'audited' => false, // true or false
            'importable' => 'true', // 'true', 'false', 'required'
            'duplicate_merge' => false, // true or false
        ),
        //DropDown
        array(
            'name' => 'dropdown_field_example_c',
            'label' => 'LBL_DROPDOWN_FIELD_EXAMPLE',
            'type' => 'enum',
            'module' => 'Accounts',
```

```
            'help' => 'Enum Field Help Text',
            'comment' => 'Enum Field Comment Text',
            'ext1' => 'account_type_dom', //maps to options - specify
list name
            'default_value' => 'Analyst', //key of entry in specified
list
            'mass_update' => false, // true or false
            'required' => false, // true or false
            'reportable' => true, // true or false
            'audited' => false, // true or false
            'importable' => 'true', // 'true', 'false' or 'required'
            'duplicate_merge' => false, // true or false
        ),
        //MultiSelect
        array(
            'name' => 'multiselect_field_example_c',
            'label' => 'LBL_MULTISELECT_FIELD_EXAMPLE',
            'type' => 'multienum',
            'module' => 'Accounts',
            'help' => 'Multi-Enum Field Help Text',
            'comment' => 'Multi-Enum Field Comment Text',
            'ext1' => 'account_type_dom', //maps to options - specify
list name
            'default_value' => 'Analyst', //key of entry in specified
list
            'mass_update' => false, // true or false
            'required' => false, // true or false
            'reportable' => true, // true or false
            'audited' => false, // true or false
            'importable' => 'true', // 'true', 'false' or 'required'
            'duplicate_merge' => false, // true or false
        ),
        //Checkbox
        array(
            'name' => 'checkbox_field_example_c',
            'label' => 'LBL_CHECKBOX_FIELD_EXAMPLE',
            'type' => 'bool',
            'module' => 'Accounts',
            'default_value' => true, // true or false
            'help' => 'Bool Field Help Text',
            'comment' => 'Bool Field Comment',
            'audited' => false, // true or false
            'mass_update' => false, // true or false
            'duplicate_merge' => false, // true or false
            'reportable' => true, // true or false
            'importable' => 'true', // 'true', 'false' or 'required'
```

```
),
//Date
array(
    'name' => 'date_field_example_c',
    'label' => 'LBL_DATE_FIELD_EXAMPLE',
    'type' => 'date',
    'module' => 'Accounts',
    'default_value' => '',
    'help' => 'Date Field Help Text',
    'comment' => 'Date Field Comment',
    'mass_update' => false, // true or false
    'required' => false, // true or false
    'reportable' => true, // true or false
    'audited' => false, // true or false
    'duplicate_merge' => false, // true or false
    'importable' => 'true', // 'true', 'false' or 'required'
),
//DateTime
array(
    'name' => 'datetime_field_example_c',
    'label' => 'LBL_DATETIME_FIELD_EXAMPLE',
    'type' => 'datetime',
    'module' => 'Accounts',
    'default_value' => '',
    'help' => 'DateTime Field Help Text',
    'comment' => 'DateTime Field Comment',
    'mass_update' => false, // true or false
    'enable_range_search' => false, // true or false
    'required' => false, // true or false
    'reportable' => true, // true or false
    'audited' => false, // true or false
    'duplicate_merge' => false, // true or false
    'importable' => 'true', // 'true', 'false' or 'required'
),
//Encrypt
array(
    'name' => 'encrypt_field_example_c',
    'label' => 'LBL_ENCRYPT_FIELD_EXAMPLE',
    'type' => 'encrypt',
    'module' => 'Accounts',
    'default_value' => '',
    'help' => 'Encrypt Field Help Text',
    'comment' => 'Encrypt Field Comment',
    'reportable' => true, // true or false
    'audited' => false, // true or false
    'duplicate_merge' => false, // true or false
```

```
            'importable' => 'true', // 'true', 'false' or 'required'
        ),
    ),
);

?>
```

Language Example

<basepath>/Files/Language/Accounts/en_us.lang.php

```
  <?php

$mod_strings['LBL_TEXT_FIELD_EXAMPLE'] = 'Text Field Example';
$mod_strings['LBL_DROPDOWN_FIELD_EXAMPLE'] = 'DropDown Field Example';
$mod_strings['LBL_CHECKBOX_FIELD_EXAMPLE'] = 'Checkbox Field Example';
$mod_strings['LBL_MULTISELECT_FIELD_EXAMPLE'] = 'Multi-
Select Field Example';
$mod_strings['LBL_DATE_FIELD_EXAMPLE'] = 'Date Field Example';
$mod_strings['LBL_DATETIME_FIELD_EXAMPLE'] = 'DateTime Field Example';
$mod_strings['LBL_ENCRYPT_FIELD_EXAMPLE'] = 'Encrypt Field Example';
```

Removing Files with an Installable Package

Overview

This is an overview of how to create a module loadable package that will remove files from the custom directory that may have been left over from a previous package installation or legacy versions of your code customization.

Note: For SugarCloud customers, this method will not work because removing files is prohibited by Package Scanner. To have files removed, you can submit a case to Sugar Support with a list of files to be removed or request a package approval and provide your module loadable package.

This example will install a custom file and use a pre-execute script to remove files from a previously installed customization. Typically, uninstalling a package would remove the files of the customization, however, in some scenarios, it is desired to install a new package on top of a previous version, and removing files with the new package might be necessary. The full package for this example is downloadable here for your reference.

For more details on the $manifest or $installdef options, you can visit the Introduction to the Manifest documentation.

Manifest Example

<basepath>/manifest.php

```
  <?php

    $manifest = array(
        'acceptable_sugar_flavors' => array(
            'PRO',
            'ENT',
            'ULT'
        ),
        'acceptable_sugar_versions' => array(
            'exact_matches' => array(),
            'regex_matches' => array(
                '12.0.*
            '),
        ),
        'author' => 'SugarCRM',
        'description' => 'Installs a custom class file',
        'icon' => '',
        'is_uninstallable' => true,
        'name' => 'Example Removing File Installer',
        'published_date' => '2018-12-06 00:00:00',
        'type' => 'module',
        'version' => '1.2.0'
    );

    $installdefs = array(
        'id' => 'package_1341607504',
        'copy' => array(
            array(
                'from' => '<basepath>/Files/custom/src/CustomClass.php
',
                'to' => 'custom/src/CustomClass.php',
            ),
        ),
        'pre_execute' => array(
            '<basepath>/removeLegacyFiles.php',
        ),
        'remove_files' => array(
            'custom/include/CustomClass.php'
```

```
        )
    );
```

<basepath>/removeLegacyFiles.php

```
  <?php

if (isset($this->installdefs['remove_files'])) {
    foreach($this->installdefs['remove_files'] as $relpath){
        if (SugarAutoloader::fileExists($relpath)) {
            SugarAutoloader::unlink($relpath);
        }
    }
}
```

The full package is downloadable here for your reference.

Passing Data to Templates

Overview

This page explains how to create a custom view component that passes data to the Handlebars template.

Steps to Complete

The view component will render the actual content on the page. The view below will display data passed to it from the controller.

./custom/clients/base/views/my-dataview/my-dataview.js

```
  ({
    className: 'tcenter',
    loadData: function (options) {
        //populate your data
        myData=new Object();
        myData.myProperty = "My Value";
        this.myData = myData;

        /*
```

```
            //alternatively, you can pass in a JSON array to populate
your data
            myData = $.parseJSON( '{"myData":{"myProperty":"My Value"}
}' );
            _.extend(this, myData);
        */

        this.render();

        //reset flags on reload
        if (options && _.isFunction(options.complete)) {
            options.complete();
        }

        // Another Alternative you can get the date from an end point
        let self = this;
        app.api.call("read", app.api.buildURL("Accounts/4bbd3b4e-755d-
11e9-9a7b-f45c89a8598f"), null, {
            success: function (accountResponse) {
                self.myData['account'] = accountResponse;
                self.render();
            },
        })

    },
})
```

Next, add the Handlebars template to render the data. An example is shown below:

./custom/clients/base/views/my-dataview/my-dataview.hbs

```
  {{#with myData}}
 {{account.name}} - {{account.assigned_user_name}} <br>
 {{myProperty}}
{{/with}}
```

Once the files are in place, add the view to a layout and then navigate to Admin > Repair > Quick Repair and Rebuild.

Prepopulating the Compose Email View

Overview

When composing an email in Sugar, it may be useful to modify the compose view to better suit your common business practices. In this article, we will use JavaScript to create a code-level customization which causes the To, CC, and Subject fields of the email to prepopulate with data from a related Opportunity.

Steps to Complete

Create a Custom Compose Email View

To modify the default compose email functionality, we will need to create our own compose-email view that extends from the base compose-email view. To accomplish this, we will need to create the following file:

./custom/modules/Emails/clients/base/views/compose-email/compose-email.js

```
({
  extendsFrom: 'EmailsComposeEmailView',

  initialize: function(options){
      this._super('initialize',[options]);
      this.configureDefaults();
  },

  /**
   * Configure the default data
   */
  configureDefaults: function(){
      var model = this.context.parent.get('model');
      var module = this.context.parent.get('module');
      if (module == 'Opportunities'){
          var contacts;
          var oppName = model.get('name');
          if (this.model.get('to_collection').length === 0) {
              var email = this.model;
              //Get the related contacts to the Opportunity
              contacts = model.getRelatedCollection('contacts');
              contacts.fetch({
                  relate: true,
                  success: function (data) {
                      var ccRecords = [],
                          toRecords = [];
                      data.forEach(function (record) {
                          var parentName = app.utils.getRecordName(r
```

```
ecord);
                            if (record.attributes.opportunity_role ==
'Primary Decision Maker') {
                                //Primary decisions makers are added t
o the TO Recipients
                                toRecords.push(app.data.createBean('Em
ailParticipants', {
                                    _link: 'to',
                                    parent: _.extend({type: record.mod
ule}, record.attributes),
                                    parent_type: record.module,
                                    parent_id: record.get('id'),
                                    parent_name: parentName
                                }));
                            } else {
                                //All other contacts are added to the
CC Recipients
                                ccRecords.push(app.data.createBean('Em
ailParticipants', {
                                    _link: 'cc',
                                    parent: _.extend({type: record.mod
ule}, record.attributes),
                                    parent_type: record.module,
                                    parent_id: record.get('id'),
                                    parent_name: parentName
                                }));
                            }
                        });

                        email.get('to_collection').add(toRecords);
                        email.get('cc_collection').add(ccRecords);
                    },
                    fields: ['id', 'full_name', 'email', 'opportunity_
role']
                });
            }
            //Default the Subject of Email to the Opportunity Name
            email.set('name',oppName);
        }
    }
})
```

Some key things to note about this customization:

- The custom view should extend from EmailsComposeEmailView
- When adding recipients to an Email Bean Model, you should create an EmailParticipants Bean Model and specify the _link property that will be used.
- This view will be used when clicking on the "Create" button in the Emails subpanel, or when clicking on Email Address on a record when the user is configured to use the Sugar Email Client

Once you have added the custom code, execute a Quick Repair and rebuild via Admin > Repair > Quick Repair and Rebuild, and your changes should now be reflected in your instance.

Refreshing Subpanels on the RecordView

Overview

How to refresh specific subpanels on the Record View.

Refreshing Subpanels

When Working with the Record View, it is sometimes necessary to force the refresh of a subpanel. The following example will demonstrate how to add buttons to force refresh a specific subpanel or all subpanels on the Accounts RecordView.

Adding the Button Metadata

For our example, we will first create a metadata extension file to append our custom refresh buttons to the Accounts RecordView action menu.

./custom/Extension/modules/Accounts/Ext/clients/base/views/record/refreshButtons.php

```
<?php

//module name
$module = 'Accounts';

//buttons to append
$addButtons = array(
    array(
        'type' => 'divider',
    ),
    array (
        'type' => 'rowaction',
```

```
        'event' => 'button:refresh_specific_subpanel:click',
        'name' => 'refresh_specific_subpanel',
        'label' => 'LBL_REFRESH_SPECIFIC_SUBPANEL',
        'acl_action' => 'view',
    ),
    array (
        'type' => 'rowaction',
        'event' => 'button:refresh_all_subpanels:click',
        'name' => 'refresh_all_subpanels',
        'label' => 'LBL_REFRESH_ALL_SUBPANELS',
        'acl_action' => 'view',
    )
);

//if the buttons are missing in our base modules metadata, include cor
e buttons
if (!isset($viewdefs[$module]['base']['view']['record']['buttons']))
{
    require('clients/base/views/record/record.php');
    $viewdefs[$module]['base']['view']['record']['buttons'] = $viewdef
s['base']['view']['record']['buttons'];
    unset($viewdefs['base']);
}

foreach($viewdefs[$module]['base']['view']['record']['buttons'] as $ou
terKey => $outerButton)
{
    if (
        isset($outerButton['type'])
        && $outerButton['type'] == 'actiondropdown'
        && isset($outerButton['name'])
        && $outerButton['name'] == 'main_dropdown'
        && isset($outerButton['buttons'])
    )
    {
        /*
        //removing buttons by name
        foreach($viewdefs[$module]['base']['view']['record']['buttons'
][$outerKey]['buttons'] as $innerKey => $innerButton)
        {
            if (
                isset($innerButton['name'])
                && $innerButton['name'] == 'button_name'
            )
            {
                unset($viewdefs[$module]['base']['view']['record']['bu
```

```
ttons'][$outerKey]['buttons'][$innerKey]);
            }
        }
        */

        //appending buttons
        foreach ($addButtons as $addButton)
        {
            $viewdefs[$module]['base']['view']['record']['buttons'][$o
uterKey]['buttons'][]=$addButton;
        }
    }
}
```

Next, we will create our language labels for the buttons.

./custom/Extension/modules/Accounts/Ext/Language/en_us.refreshButtons.php

```
<?php

$mod_strings['LBL_REFRESH_SPECIFIC_SUBPANEL'] = 'Refresh Specific Subp
anel';
$mod_strings['LBL_REFRESH_ALL_SUBPANELS'] = 'Refresh All Subpanels';
```

Our next step is to extend the Accounts controller file. This is where we will add our code to refresh the subpanels when the buttons are clicked.

./custom/modules/Accounts/clients/base/views/record/record.js

```
({
    extendsFrom: 'RecordView',

    initialize: function (options) {
       this._super('initialize', [options]);

        //add listeners for custom buttons
        this.context.on('button:refresh_specific_subpanel:click', this
.refresh_specific_subpanel, this);
        this.context.on('button:refresh_all_subpanels:click', this.ref
resh_all_subpanels, this);
    },
```

```
    /**
     * Refreshes a specific subpanel given a link
     */
    refresh_specific_subpanel: function() {
        var linkName = 'contacts';
        var subpanelCollection = this.model.getRelatedCollection(linkN
ame);
        subpanelCollection.fetch({relate: true});
    },

    /**
     * Refreshes all subpanels
     */
    refresh_all_subpanels: function() {
        _.each(this.model._relatedCollections, function(collection){
            collection.fetch({relate: true});
        });
    }
})
```

Note: To refresh a specific subpanel, you will need to pass the linkname of the relationship to the this.model.getRelatedCollection() method.

Finally, we will then need to navigate to Admin > Repair > Quick Repair and Rebuild. This will rebuild our extensions and make the refresh buttons available on our RecordView.

Removing the Account Requirement on Opportunities

Overview

This article covers how to remove the Account field from being required on the Opportunities module.

Removing the Requirement in Configuration

By default, Sugar requires the accounts field to be populated on several modules. These modules include

- Opportunities

- Cases
- Contracts

In order to remove this dependency, you will need to modify the require_accounts configuration in your ./config_override.php.

```
$sugar_config['require_accounts'] = false;
```

The resulting file should look similar to:

```
$sugar_config['disable_count_query'] = true;
$sugar_config['http_referer']['weak'] = true;
$sugar_config['hide_full_text_engine_config'] = false;
$sugar_config['fts_disable_notification'] = true;

$sugar_config['require_accounts'] = false;

/***CONFIGURATOR***/
```

Removing the Requirement in Vardefs

Once you have updated your configuration, you may find the field is still required. This may be due to the module's vardefs array having the field's required attribute set to true. This will also need to be updated. Using the Opportunities vardefs as an example, we will need to do the following:

- Create a custom file in the Opportunities Extension directory like ./custom/Extension/modules/Opportunities/Ext/Vardefs/no_account_required.php
- Set the content of this file so that the account_name field has required set to false

```
<?php

$dictionary['Opportunity']['fields']['account_name']['required']
 = false;
```

Once completed, you will need to navigate to Admin > Repairs and run a Quick Repair & Rebuild

Removing the Requirement in View Metadata

If the Account field is still required after making these changes, this may be due to the views metadata having the field's required attribute set to true. By default,

account_name should not be set to required, but if you have created custom views, these may need to be updated. Using the Opportunities record view as an example, we will need to do the following:

- Edit ./custom/modules/Opportunities/clients/base/views/record/record.php with a text editor application.
- Search for the account_name field in panel definitions of your record view.
- Remove 'required' => true, from the account_name definition. If it doesn't already exist, you don't need to modify anything.

```
...
  'fields' =>
    array (
      0 =>
        array (
          'name' => 'account_name',
          'required' => false,
          ...
```

Your resulting file should be:

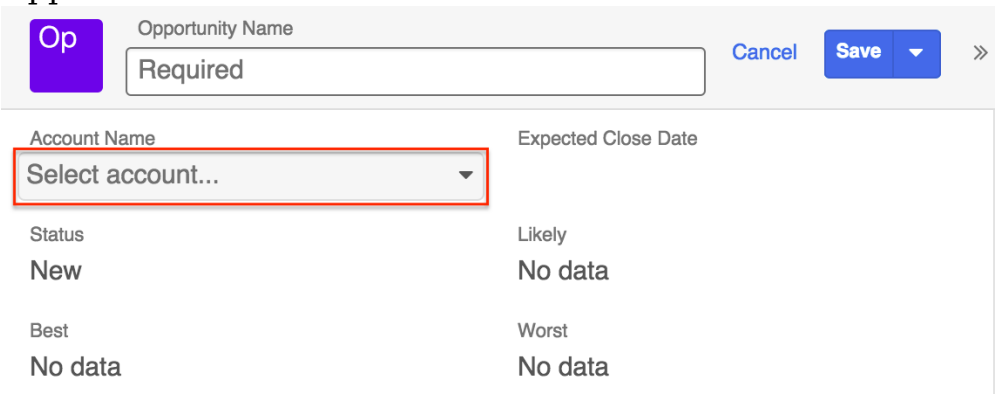
```
...
  'fields' =>
    array (
      0 =>
        array (
          'name' => 'account_name',
          ...
```

- These changes will only affect the record view for Opportunities. You may need to modify additional module layouts based on your use case. A list of other views you many need to modify for your module are shown below.
    - ./custom/modules/<module>/clients/base/views/list/list.php
    - ./custom/modules/<module>/clients/base/views/record/record.php
    - ./custom/modules/<module>/clients/base/views/dupecheck-list/dupecheck-list.php
    - ./custom/modules/<module>/clients/base/views/resolve-conflicts-list/resolve-conflicts-list.php
    - ./custom/modules/<module>/clients/base/views/selection-list/selection-list.php
    - ./custom/modules/<module>/clients/base/views/subpanel-list/subpanel-list.php

Once completed, you will need to navigate to Admin > Repairs and run a Quick Repair & Rebuild

Application

Now that the appropriate changes have been made, navigate to the Opportunities module and create a new opportunity record. You will notice that the Account Name field no longer indicates "Required" in the field. In addition, the Account Name field will no longer be marked as required when importing records into the Opportunities module.



Web Services

Examples when working with Sugar's web service endpoints.

REST API

Examples of integrating with Sugar REST APIs.

Bash

Bash cURL examples of integrating with the REST v11 API.

How to Export a List of Records

Overview

An example in bash script demonstrating how to export a list of records using the v11 /<module>/export/:record_list_id REST GET endpoint.

Exporting Records

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  curl -X POST -H Cache-Control:no-cache -H "Content-
Type: application/json" -d '{
 "grant_type":"password",
 "client_id":"sugar",
 "client_secret":"",
 "username":"admin",
 "password":"password",
 "platform":"custom_api"
}' https://{site_url}/rest/v11/oauth2/token
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Filtering Records

Next, we will need to identify the records we want to export using the /<module>/filter endpoint.

```
  curl -s -X POST -H OAuth-Token:{access_token} -H "Content-
Type: application/json" -H Cache-Control:no-cache -d '{
   "filter":[
      {
         "$or":[
            {
               "name":{
                  "$starts":"A"
               }
            },
            {
               "name":{
                  "$starts":"b"
               }
            }
         ]
      }
   ],
   "max_num":2,
```

```
   "offset":0,
   "fields":"id",
   "order_by":"date_entered",
   "favorites":false,
   "my_items":false
}' https://{site_url}/rest/v11/Accounts/filter
```

More information on the filter API can be found in the /<module>/filter documentation.

Response

The data received from the server is shown below:

```
  {
   "next_offset":2,
   "records":[
      {
         "id":"f16760a4-3a52-f77d-1522-5703ca28925f",
         "date_modified":"2016-04-05T10:23:28-04:00",
         "_acl":{
            "fields":{

            }
         },
         "_module":"Accounts"
      },
      {
         "id":"ec409fbb-2b22-4f32-7fa1-5703caf78dc3",
         "date_modified":"2016-04-05T10:23:28-04:00",
         "_acl":{
            "fields":{

            }
         },
         "_module":"Accounts"
      }
   ]
}
```

Creating a Record List

Once we have the list of ids, we then need to create a record list in Sugar that consists of those ids.

```
  curl -s -X POST -H OAuth-Token:{access_token} -H "Content-
Type: application/json" -H Cache-Control:no-cache -d '{
   "records":[
      "f16760a4-3a52-f77d-1522-5703ca28925f",
      "ec409fbb-2b22-4f32-7fa1-5703caf78dc3"
   ]
}' https://{site_url}/rest/v11/Accounts/record_list
```

Response

The data received from the server is shown below:

```
  {
   "id":"ef963176-4845-bc55-b03e-570430b4173c",
   "assigned_user_id":"1",
   "module_name":"Accounts",
   "records":[
      "f16760a4-3a52-f77d-1522-5703ca28925f",
      "ec409fbb-2b22-4f32-7fa1-5703caf78dc3"
   ],
   "date_modified":"2016-04-05 21:39:19"
}
```

Exporting Records

Once we have the record list created, we can then export the CSV file.

```
  curl -i -s -X GET -H OAuth-Token:{access_token} -H Cache-Control:no-
cache https://{site_url}/rest/v11/Accounts/export/ef963176-4845-bc55-b
03e-570430b4173c
```

More information on exporting records can be found in the /<module>/export/:record_list_id documentation.

Response

The data received from the server is shown below:

```
  HTTP/1.1 200 OK
Date: Tue, 05 Apr 2016 21:50:32
GMT Server: Apache/2.2.29 (Unix)
DAV/2 PHP/5.3.29 mod_ssl/2.2.29
OpenSSL/0.9.8zg X-Powered-By:
PHP/5.3.29 Expires:
Cache-Control: max-age=10,
private Pragma:
Content-Disposition: attachment; filename=Accounts.csv
Content-transfer-encoding: binary
Last-Modified: Tue, 05 Apr 2016 21:50:32
GMT ETag: 9b34f5d74e0298aaf7fd1f27d02e14f2
Content-Length: 1703
Connection: close
Content-Type: application/octet-stream; charset=ISO-8859-1
"Name","ID","Website","Office Phone","Alternate Phone","Fax","Billing
Street","Billing City","Billing State","Billing Postal Code","Billing
Country","Shipping Street","Shipping City","Shipping State","Shipping
Postal Code","Shipping Country","Description","Type","Industry","Annua
l Revenue","Employees","SIC Code","Ticker Symbol","Parent Account ID",
"Ownership","Campaign ID","Rating","Assigned User Name","Assigned User
 ID","Team ID","Teams","Team Set ID","Date Created","Date Modified","M
odified By Name","Modified By ID","Created By","Created By ID","Delete
d","test","Facebook Account","Twitter Account","Google Plus ID","DUNS"
,"Email","Invalid Email","Email Opt Out","Non-primary emails"
"Arts & Crafts Inc","ec409fbb-2b22-4f32-7fa1-5703caf78dc3","www.hrinfo
.tw","(252) 456-8602","","","777 West Filmore Ln","Los Angeles","CA","
77076","USA","777 West Filmore Ln","Los Angeles","CA","77076","USA",""
,"Customer","Hospitality","","","","","","","","","","Max Jensen","seed_m
ax_id","West","West, East, Global","dec43cb2-5273-8be2-968a-5703cadee7
5f","2016-04-05 10:23","2016-04-05 10:23","Administrator","1","Adminis
trator","1","0","","","","","","sugar.sugar.section@example.org","0","
0","dev.phone@example.biz,0,0"
"B.H. Edwards Inc","f16760a4-3a52-f77d-1522-5703ca28925f","www.section
dev.edu","(361) 765-0216","","","111 Silicon Valley Road","Persistance
","CA","29709","USA","111 Silicon Valley Road","Persistance","CA","297
09","USA","","Customer","Apparel","","","","","","","","","Sally Brons
en","seed_sally_id","West","West","West","2016-04-05 10:23","2016-04-0
5 10:23","Administrator","1","Administrator","1","0","","","","","","i
nfo.sugar@example.de","0","1","phone.sales.section@example.tv,0,0"
```

You can also output the results directly to a CSV file by omitting the header data and output the results directly to a new CSV file.

```
  curl -s -X GET -H OAuth-Token:{access_token} -H -H Cache-Control:no-
cache https://{site_url}/rest/v11/Accounts/export/ef963176-4845-bc55-b
03e-570430b4173c > Output.csv
```

How to Filter a List of Records

Overview

An example in bash script demonstrating how to filter records using the v11 /<module>/filter REST POST endpoints.

Filtering Records

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  curl -X POST -H Cache-Control:no-cache -H "Content-
Type: application/json" -d '{
 "grant_type":"password",
 "client_id":"sugar",
 "client_secret":"",
 "username":"admin",
 "password":"password",
 "platform":"custom_api"
}' https://{site_url}/rest/v11/oauth2/token
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Filtering Records

Next we can filter the records we want to return using the /<module>/filter endpoint with a POST request.

```
  curl -s -X POST -H OAuth-Token:{access_token} -H "Content-
Type: application/json" -H Cache-Control:no-cache -d '{
   "filter":[
      {
         "$or":[
            {
               "name":{
```

```
                "$starts":"A"
            }
         },
         {
            "name":{
               "$starts":"b"
            }
         }
      ]
   }
],
"max_num":2,
"offset":0,
"fields":"id",
"order_by":"date_entered",
"favorites":false,
"my_items":false
}' https://{site_url}/rest/v11/Accounts/filter
```

More information on the filter API can be found in the /<module>/filter documentation.

Note : The /<module>/filter endpoint can be called using a GET request as well, though long URL requests can have issues so the POST request is recommended.

Response

The data received from the server is shown below:

```
{
 "next_offset":2,
 "records":[
    {
       "id":"f16760a4-3a52-f77d-1522-5703ca28925f",
       "date_modified":"2016-04-05T10:23:28-04:00",
       "_acl":{
          "fields":{

          }
       },
       "_module":"Accounts"
    },
    {
       "id":"ec409fbb-2b22-4f32-7fa1-5703caf78dc3",
```

```
         "date_modified":"2016-04-05T10:23:28-04:00",
         "_acl":{
            "fields":{

            }
         },
         "_module":"Accounts"
      }
   ]
}
```

How to Favorite a Record

Overview

An example in bash script demonstrating how to favorite a record using the v11 <module>/:record/favorite REST PUT API endpoint.

Favoriting a Record

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  curl -X POST -H Cache-Control:no-cache -H "Content-
Type: application/json" -d '{
 "grant_type":"password",
 "client_id":"sugar",
 "client_secret":"",
 "username":"admin",
 "password":"password",
 "platform":"custom_api"
}' https://{site_url}/rest/v11/oauth2/token
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Favoriting a Record

Next, we can favorite a specific record using the /<module>/:record/favorite endpoint.

```
  curl -s -X PUT -H OAuth-Token:{access_token} -H Cache-Control:no-cac
he https://{site_url}/rest/v11/Accounts/ae8b1783-404a-
fcb8-1e1e-56f1cc52cd1a/favorite
```

More information on the unfavorite API can be found in the /<module>/:record/favorite PUT documentation.

Response

The data received from the server is shown below:

```
  {
    "id": "ae8b1783-404a-fcb8-1e1e-56f1cc52cd1a",
    "name": "Union Bank",
    "date_entered": "2016-03-22T17:49:50-05:00",
    "date_modified": "2016-03-30T17:44:20-05:00",
    "modified_user_id": "1",
    "modified_by_name": "Administrator",
    "modified_user_link": {
        "full_name": "Administrator",
        "id": "1",
        "_acl": {
            "fields": [],
            "delete": "no",
            "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
        }
    },
    "created_by": "1",
    "created_by_name": "Administrator",
    "created_by_link": {
        "full_name": "Administrator",
        "id": "1",
        "_acl": {
            "fields": [],
            "delete": "no",
            "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
        }
    },
    "description": "",
    "deleted": false,
    "facebook": "",
    "twitter": "",
    "googleplus": "",
    "account_type": "Customer",
```

```
"industry": "Banking",
"annual_revenue": "",
"phone_fax": "",
"billing_address_street": "67321 West Siam St.",
"billing_address_street_2": "",
"billing_address_street_3": "",
"billing_address_street_4": "",
"billing_address_city": "Ohio",
"billing_address_state": "CA",
"billing_address_postalcode": "25159",
"billing_address_country": "USA",
"rating": "",
"phone_office": "(065) 489-6104",
"phone_alternate": "",
"website": "www.qahr.edu",
"ownership": "",
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "67321 West Siam St.",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "Ohio",
"shipping_address_state": "CA",
"shipping_address_postalcode": "25159",
"shipping_address_country": "USA",
"parent_id": "",
"sic_code": "",
"duns_num": "",
"parent_name": "",
"member_of": {
    "name": "",
    "id": "",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"campaign_id": "",
"campaign_name": "",
"campaign_accounts": {
    "name": "",
    "id": "",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
```

```
    }
},
"following": true,
"my_favorite": true,
"tag": [],
"assigned_user_id": "seed_sarah_id",
"assigned_user_name": "Sarah Smith",
"assigned_user_link": {
    "full_name": "Sarah Smith",
    "id": "seed_sarah_id",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"team_count": "",
"team_count_link": {
    "team_count": "",
    "id": "West",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"team_name": [{
    "id": "East",
    "name": "East",
    "name_2": "",
    "primary": false
}, {
    "id": 1,
    "name": "Global",
    "name_2": "",
    "primary": false
}, {
    "id": "West",
    "name": "West",
    "name_2": "",
    "primary": true
}],
"email": [{
    "email_address": "hr.support.kid@example.info",
    "invalid_email": false,
    "opt_out": false,
    "primary_address": true,
    "reply_to_address": false
```

```
    }, {
        "email_address": "info.support.the@example.com",
        "invalid_email": false,
        "opt_out": false,
        "primary_address": false,
        "reply_to_address": false
    }],
    "email1": "hr.support.kid@example.info",
    "email2": "info.support.the@example.com",
    "invalid_email": false,
    "email_opt_out": false,
    "email_addresses_non_primary": "",
    "_acl": {
        "fields": {}
    },
    "_module": "Accounts"
}
```

How to Manipulate File Attachments

Overview

An example in bash script demonstrating how to attach a file to a record using the v11 <module>/:record/file/:field REST POST API endpoint, then retrieve it with the GET endpoint.

Manipulating File Attachments

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  curl -X POST -H Cache-Control:no-cache -H "Content-
Type: application/json" -d '{
 "grant_type":"password",
 "client_id":"sugar",
 "client_secret":"",
 "username":"admin",
 "password":"password",
 "platform":"custom_api"
}' https://{site_url}/rest/v11/oauth2/token
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Submitting a File Attachment

Next, we can create a Note record using the /<module endpoint, and then submit a File to the Note record using the /<module>/:record/file/:field endpoint.

Create Note

```
  curl -s -X POST -H OAuth-Token:{access_token}-H "Content-
Type: application/json" -H Cache-Control:no-cache -d '{
   "name":"Test Note"
}' https://{site_url}/rest/v11/Notes
```

Add An Attachment to the Note

```
  curl -X POST -H OAuth-Token:{access_token} -H Cache-Control:no-cache
 -F "filename=@/path/to/file.txt" https://{site_url}/rest/v11/Notes/7b
49aebd-8734-9773-8ef1-53553fa369c7/file/filename
```

Response

```
  {
  "filename":{
     "content-type":"text\/plain",
     "content-length":13,
     "name":"file.txt",
     "uri":"http:\/\/<site url>\/rest\/v11\/Notes\/7b49aebd-8734-9773-
8ef1-53553fa369c7\/file\/filename"
  },
  "record":{
     "my_favorite":false,
     "following":true,
     "id":"7b49aebd-8734-9773-8ef1-53553fa369c7",
     "name":"My Note",
     "date_modified":"2014-04-21T11:53:53-04:00",
     "modified_user_id":"1",
     "modified_by_name":"admin",
     "created_by":"1",
     "created_by_name":"Administrator",
     "doc_owner":"",
```

```
      "user_favorites":[

      ],
      "description":"",
      "deleted":false,
      "assigned_user_id":"",
      "assigned_user_name":"",
      "team_count":"",
      "team_name":[
         {
            "id":1,
            "name":"Global",
            "name_2":"",
            "primary":true
         }
      ],
      "file_mime_type":"text\/plain",
      "file_url":"",
      "filename":"file.txt",
      "parent_type":"",
      "parent_id":"",
      "contact_id":"",
      "portal_flag":false,
      "embed_flag":false,
      "parent_name":"",
      "contact_name":"",
      "contact_phone":"",
      "contact_email":"",
      "account_id":"",
      "opportunity_id":"",
      "acase_id":"",
      "lead_id":"",
      "product_id":"",
      "quote_id":"",
      "_acl":{
         "fields":{

         }
      },
      "_module":"Notes"
   }
}
```


Getting a File Attachment

Next, we can retrieve the file attachment stored in Sugar by utilizing the /<module>/:record/file/:field GET endpoint.

```
  curl -i -s -X GET -H OAuth-Token:{access_token} -H Cache-Control:no-cache https://{site_url}/rest/v11/Notes/7b49aebd-8734-9773-8ef1-53553fa369c7/file/filename
```

Response

```
  HTTP/1.1 200 OK

Date: Wed, 12 Mar 2014 15:15:03 GMT

Server: Apache/2.2.22 (Unix) PHP/5.3.14 mod_ssl/2.2.22 OpenSSL/0.9.8o

X-Powered-By: PHP/5.3.14 ZendServer/5.0

Set-Cookie: ZDEDebuggerPresent=php,phtml,php3; path=/

Expires:

Cache-Control: max-age=0, private

Pragma:

Content-Disposition: attachment; filename="file.txt"

X-Content-Type-Options: nosniff

ETag: d41d8cd98f00b204e9800998ecf8427e

Content-Length: 16

Connection: close

Content-Type: application/octet-stream


This is the file contents.
```

You can also output the results directly to a file by omitting the header data and

output the results directly to a new file.

```
  curl -s -X GET -H OAuth-Token:{access_token} -H Cache-Control:no-cac
he https://{site_url}/rest/v11/Notes/7b49aebd-8734-9773-8ef1-53553fa36
9c7/file/filename > file.txt
```

How to Fetch Related Records

Overview

An example in bash script demonstrating how to fetch related records using the v11 /<module>/:record/link/:link REST GET endpoints.

Fetching Related Records

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  curl -X POST -H Cache-Control:no-cache -H "Content-
Type: application/json" -d '{
 "grant_type":"password",
 "client_id":"sugar",
 "client_secret":"",
 "username":"admin",
 "password":"password",
 "platform":"custom_api"
}' https://{site_url}/rest/v11/oauth2/token
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Fetching Related Records

Next, we can fetch the related records we want to return using the /<module>/:record/link/:link endpoint with a GET request where

Element	Meaning
<module>	The parent module name
:record	The parent records ID

Element	Meaning
link	the actual word "link"
:link	The name of the relationship to fetch

In this example we will fetch the related Contacts for an Account :

```
  curl -s -X GET -H OAuth-Token:{access_token} -H Cache-Control:no-cac
he https://{site_url}/rest/v11/Accounts/e8c641ca-1b8c-74c1-d08d-56fedb
dd3187/link/contacts
```

Response

The data received from the server is shown below:


```
{
   "next_offset":-1,
   "records":[
      {
         "my_favorite":false,
         "following":false,
         "id":"819f4149-b007-a6da-a5fa-56fedbf2de77",
         "name":"Florine Marcus",
         "date_entered":"2016-04-01T15:34:00-05:00",
         "date_modified":"2016-04-01T15:34:00-05:00",
         "modified_user_id":"1",
         "modified_by_name":"Administrator",
         "created_by":"1",
         "created_by_name":"Administrator",
         "doc_owner":"",
         "user_favorites":"",
         "description":"",
         "deleted":false,
         "assigned_user_id":"seed_will_id",
         "assigned_user_name":"Will Westin",
         "team_count":"",
         "team_name":[
            {
               "id":"East",
               "name":"East",
               "name_2":"",
               "primary":true
            },
```

```
      {
         "id":"West",
         "name":"West",
         "name_2":"",
         "primary":false
      }
   ],
   "email":[
      {
         "email_address":"support27@example.tv",
         "primary_address":true,
         "reply_to_address":false,
         "invalid_email":false,
         "opt_out":false
      },
      {
         "email_address":"support.support.kid@example.net",
         "primary_address":false,
         "reply_to_address":false,
         "invalid_email":false,
         "opt_out":true
      }
   ],
   "email1":"support27@example.tv",
   "email2":"",
   "invalid_email":false,
   "email_opt_out":false,
   "email_addresses_non_primary":"",
   "salutation":"",
   "first_name":"Florine",
   "last_name":"Marcus",
   "full_name":"Florine Marcus",
   "title":"President",
   "facebook":"",
   "twitter":"",
   "googleplus":"",
   "department":"",
   "do_not_call":false,
   "phone_home":"(746) 162-2314",
   "phone_mobile":"(941) 088-2874",
   "phone_work":"(827) 541-9614",
   "phone_other":"",
   "phone_fax":"",
   "primary_address_street":"1715 Scott Dr",
   "primary_address_street_2":"",
   "primary_address_street_3":"",
```

```
"primary_address_city":"Alabama",
"primary_address_state":"NY",
"primary_address_postalcode":"70187",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
"assistant":"",
"assistant_phone":"",
"picture":"",
"email_and_name1":"",
"lead_source":"Employee",
"account_name":"MTM Investment Bank F S B",
"account_id":"e8c641ca-1b8c-74c1-d08d-56fedbdd3187",
"dnb_principal_id":"",
"opportunity_role_fields":"",
"opportunity_role_id":"",
"opportunity_role":"",
"reports_to_id":"",
"report_to_name":"",
"birthdate":"",
"portal_name":"FlorineMarcus119",
"portal_active":true,
"portal_password":true,
"portal_password1":null,
"portal_app":"",
"preferred_language":"en_us",
"campaign_id":"",
"campaign_name":"",
"c_accept_status_fields":"",
"m_accept_status_fields":"",
"accept_status_id":"",
"accept_status_name":"",
"accept_status_calls":"",
"accept_status_meetings":"",
"sync_contact":false,
"mkto_sync":false,
"mkto_id":null,
"mkto_lead_score":null,
"_acl":{
   "fields":{
```

```
            }
        },
        "_module":"Contacts"
    },
    {
        "my_favorite":false,
        "following":false,
        "id":"527cc1a9-7984-91fe-4148-56fedbc356aa",
        "name":"Shaneka Aceto",
        "date_entered":"2016-04-01T15:34:00-05:00",
        "date_modified":"2016-04-01T15:34:00-05:00",
        "modified_user_id":"1",
        "modified_by_name":"Administrator",
        "created_by":"1",
        "created_by_name":"Administrator",
        "doc_owner":"",
        "user_favorites":"",
        "description":"",
        "deleted":false,
        "assigned_user_id":"seed_will_id",
        "assigned_user_name":"Will Westin",
        "team_count":"",
        "team_name":[
            {
                "id":"East",
                "name":"East",
                "name_2":"",
                "primary":true
            },
            {
                "id":"West",
                "name":"West",
                "name_2":"",
                "primary":false
            }
        ],
        "email":[
            {
                "email_address":"support17@example.cn",
                "primary_address":true,
                "reply_to_address":false,
                "invalid_email":false,
                "opt_out":false
            },
            {
                "email_address":"section.sugar.the@example.tv",
```

```
      "primary_address":false,
      "reply_to_address":false,
      "invalid_email":false,
      "opt_out":true
   }
],
"email1":"support17@example.cn",
"email2":"",
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"",
"salutation":"",
"first_name":"Shaneka",
"last_name":"Aceto",
"full_name":"Shaneka Aceto",
"title":"IT Developer",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(502) 528-5151",
"phone_mobile":"(816) 719-3739",
"phone_work":"(994) 769-5855",
"phone_other":"",
"phone_fax":"",
"primary_address_street":"123 Anywhere Street",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Denver",
"primary_address_state":"NY",
"primary_address_postalcode":"15128",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
"assistant":"",
"assistant_phone":"",
"picture":"",
"email_and_name1":"",
"lead_source":"Email",
"account_name":"MTM Investment Bank F S B",
```

```
      "account_id":"e8c641ca-1b8c-74c1-d08d-56fedbdd3187",
      "dnb_principal_id":"",
      "opportunity_role_fields":"",
      "opportunity_role_id":"",
      "opportunity_role":"",
      "reports_to_id":"",
      "report_to_name":"",
      "birthdate":"",
      "portal_name":"ShanekaAceto151",
      "portal_active":true,
      "portal_password":true,
      "portal_password1":null,
      "portal_app":"",
      "preferred_language":"en_us",
      "campaign_id":"",
      "campaign_name":"",
      "c_accept_status_fields":"",
      "m_accept_status_fields":"",
      "accept_status_id":"",
      "accept_status_name":"",
      "accept_status_calls":"",
      "accept_status_meetings":"",
      "sync_contact":false,
      "mkto_sync":false,
      "mkto_id":null,
      "mkto_lead_score":null,
      "_acl":{
         "fields":{

         }
      },
      "_module":"Contacts"
   },
   {
      "my_favorite":false,
      "following":false,
      "id":"42d703ed-f834-f87c-967d-56fedb044051",
      "name":"Johnnie Pina",
      "date_entered":"2016-04-01T15:34:00-05:00",
      "date_modified":"2016-04-01T15:34:00-05:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "doc_owner":"",
      "user_favorites":"",
```

```
"description":"",
"deleted":false,
"assigned_user_id":"seed_will_id",
"assigned_user_name":"Will Westin",
"team_count":"",
"team_name":[
   {
      "id":"East",
      "name":"East",
      "name_2":"",
      "primary":true
   },
   {
      "id":"West",
      "name":"West",
      "name_2":"",
      "primary":false
   }
],
"email":[
   {
      "email_address":"sugar.support.hr@example.co.uk",
      "primary_address":true,
      "reply_to_address":false,
      "invalid_email":false,
      "opt_out":false
   },
   {
      "email_address":"support.im@example.tw",
      "primary_address":false,
      "reply_to_address":false,
      "invalid_email":false,
      "opt_out":true
   }
],
"email1":"sugar.support.hr@example.co.uk",
"email2":"",
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"",
"salutation":"",
"first_name":"Johnnie",
"last_name":"Pina",
"full_name":"Johnnie Pina",
"title":"VP Operations",
"facebook":"",
```

```
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(159) 335-1423",
"phone_mobile":"(580) 140-4050",
"phone_work":"(418) 792-9611",
"phone_other":"",
"phone_fax":"",
"primary_address_street":"345 Sugar Blvd.",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Denver",
"primary_address_state":"NY",
"primary_address_postalcode":"70648",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
"assistant":"",
"assistant_phone":"",
"picture":"",
"email_and_name1":"",
"lead_source":"Direct Mail",
"account_name":"MTM Investment Bank F S B",
"account_id":"e8c641ca-1b8c-74c1-d08d-56fedbdd3187",
"dnb_principal_id":"",
"opportunity_role_fields":"",
"opportunity_role_id":"",
"opportunity_role":"",
"reports_to_id":"",
"report_to_name":"",
"birthdate":"",
"portal_name":"JohnniePina194",
"portal_active":true,
"portal_password":true,
"portal_password1":null,
"portal_app":"",
"preferred_language":"en_us",
"campaign_id":"",
"campaign_name":"",
"c_accept_status_fields":"",
```

```
         "m_accept_status_fields":"",
         "accept_status_id":"",
         "accept_status_name":"",
         "accept_status_calls":"",
         "accept_status_meetings":"",
         "sync_contact":false,
         "mkto_sync":false,
         "mkto_id":null,
         "mkto_lead_score":null,
         "_acl":{
            "fields":{

            }
         },
         "_module":"Contacts"
      }
   ]
}
```

How to Manipulate Records (CRUD)

Overview

The following page will provide examples in bash script demonstrating how to use the CRUD (Create, Read, Update, Delete) endpoints in the REST v11 API.

CRUD Operations

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  curl -X POST -H Cache-Control:no-cache -H "Content-
Type: application/json" -d '{
 "grant_type":"password",
 "client_id":"sugar",
 "client_secret":"",
 "username":"admin",
 "password":"password",
 "platform":"custom_api"
}' https://{site_url}/rest/v11/oauth2/token
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Creating a Record

Next, we need to submit the record to the Sugar instance using the /<module> endpoint. In this example, we are going to create an Account record with a Name of 'Test Record' and an email of 'test@sugar.com'.

```
  curl -X POST -H OAuth-Token:<access_token> -H Cache-Control:no-
cache -H "Content-Type: application/json -d '{
   "name":"Test Record",
   "email1":"test@sugar.com"
}' http://<site_url>/rest/v11/Accounts
```

More information on this API endpoint can be found in the /<module> - POST documentation.

Response

The data received from the server is shown below:

```
  {
   "id":"ab2222df-73da-0e92-6887-5705428f4d68",
   "name":"Test Record",
   "date_entered":"2016-04-06T13:07:41-04:00",
   "date_modified":"2016-04-06T13:07:41-04:00",
   "modified_user_id":"1",
   "modified_by_name":"Administrator",
   "modified_user_link":{
      "full_name":"Administrator",
      "id":"1",
      "_acl":{
         "fields":[

         ],
         "delete":"no",
         "_hash":"8e11bf9be8f04daddee9d08d44ea891e"
      }
   },
   "created_by":"1",
   "created_by_name":"Administrator",
   "created_by_link":{
      "full_name":"Administrator",
```

```
      "id":"1",
      "_acl":{
         "fields":[

         ],
         "delete":"no",
         "_hash":"8e11bf9be8f04daddee9d08d44ea891e"
      }
   },
   "description":"",
   "deleted":false,
   "facebook":"",
   "twitter":"",
   "googleplus":"",
   "account_type":"",
   "industry":"",
   "annual_revenue":"",
   "phone_fax":"",
   "billing_address_street":"",
   "billing_address_street_2":"",
   "billing_address_street_3":"",
   "billing_address_street_4":"",
   "billing_address_city":"",
   "billing_address_state":"",
   "billing_address_postalcode":"",
   "billing_address_country":"",
   "rating":"",
   "phone_office":"",
   "phone_alternate":"",
   "website":"",
   "ownership":"",
   "employees":"",
   "ticker_symbol":"",
   "shipping_address_street":"",
   "shipping_address_street_2":"",
   "shipping_address_street_3":"",
   "shipping_address_street_4":"",
   "shipping_address_city":"",
   "shipping_address_state":"",
   "shipping_address_postalcode":"",
   "shipping_address_country":"",
   "parent_id":"",
   "sic_code":"",
   "duns_num":"",
   "parent_name":"",
   "member_of":{
```

```
      "name":"",
      "id":"",
      "_acl":{
         "fields":[

         ],
         "_hash":"654d337e0e912edaa00dbb0fb3dc3c17"
      }
   },
   "campaign_id":"",
   "campaign_name":"",
   "campaign_accounts":{
      "name":"",
      "id":"",
      "_acl":{
         "fields":[

         ],
         "_hash":"654d337e0e912edaa00dbb0fb3dc3c17"
      }
   },
   "following":true,
   "my_favorite":false,
   "tag":[

   ],
   "assigned_user_id":"",
   "assigned_user_name":"",
   "assigned_user_link":{
      "full_name":"",
      "id":"",
      "_acl":{
         "fields":[

         ],
         "_hash":"654d337e0e912edaa00dbb0fb3dc3c17"
      }
   },
   "team_count":"",
   "team_count_link":{
      "team_count":"",
      "id":"1",
      "_acl":{
         "fields":[

         ],
```

```
            "_hash":"654d337e0e912edaa00dbb0fb3dc3c17"
         }
      },
      "team_name":[
         {
            "id":1,
            "name":"Global",
            "name_2":"",
            "primary":true
         }
      ],
      "email":[
         {
            "email_address":"test@sugar.com",
            "invalid_email":false,
            "opt_out":false,
            "primary_address":true,
            "reply_to_address":false
         }
      ],
      "email1":"test@sugar.com",
      "email2":"",
      "invalid_email":false,
      "email_opt_out":false,
      "email_addresses_non_primary":"",
      "_acl":{
         "fields":{

         }
      },
      "_module":"Accounts"
}
```


Getting a Record

Next we can get the created record from the Sugar instance using the /<module>/:record endpoint. In this example, we are going to get an Account record by it's ID, but only request the Name, Email, and Industry fields.

```
curl -H OAuth-Token:<access_token> -H Cache-Control:no-cache http://<s
ite_url>/rest/v11/Accounts/<record_id>?fields=name,email1,industry
```

More information on this API endpoint can be found in the /<module>/:record - GET documentation.

Response

The data received from the server is shown below:

```
  {
   "id":"ab2222df-73da-0e92-6887-5705428f4d68",
   "name":"Test Record",
   "date_modified":"2016-04-06T15:03:21-04:00",
   "industry":"",
   "email1":"test@sugar.com",
   "_acl":{
      "fields":{

      }
   },
   "_module":"Accounts"
}
```

Updating a Record

Next we can update the record in the Sugar instance using the /<module>/:record endpoint, and the PUT Http method. In this example we are going to update the Account record and change it's name to "Updated Test Record".

```
curl -X PUT -H OAuth-Token:<access_token> -H Cache-Control:no-
cache -d '{
   "name":"Updated Record"
}' http://<site_url>/rest/v11/Accounts/<record_id>
```

More information on this API endpoint can be found in the /<module>/:record - PUT documentation.

Response

The data received from the server is shown below:

```
  {
```

```
"id":"ab2222df-73da-0e92-6887-5705428f4d68",
"name":"Updated Test Record",
"date_entered":"2016-04-06T15:03:21-04:00",
"date_modified":"2016-04-06T15:03:22-04:00",
"modified_user_id":"1",
"modified_by_name":"Administrator",
"modified_user_link":{
   "full_name":"Administrator",
   "id":"1",
   "_acl":{
      "fields":[

      ],
      "delete":"no",
      "_hash":"8e11bf9be8f04daddee9d08d44ea891e"
   }
},
"created_by":"1",
"created_by_name":"Administrator",
"created_by_link":{
   "full_name":"Administrator",
   "id":"1",
   "_acl":{
      "fields":[

      ],
      "delete":"no",
      "_hash":"8e11bf9be8f04daddee9d08d44ea891e"
   }
},
"description":"",
"deleted":false,
"facebook":"",
"twitter":"",
"googleplus":"",
"account_type":"",
"industry":"",
"annual_revenue":"",
"phone_fax":"",
"billing_address_street":"",
"billing_address_street_2":"",
"billing_address_street_3":"",
"billing_address_street_4":"",
"billing_address_city":"",
"billing_address_state":"",
"billing_address_postalcode":"",
```

```
"billing_address_country":"",
"rating":"",
"phone_office":"",
"phone_alternate":"",
"website":"",
"ownership":"",
"employees":"",
"ticker_symbol":"",
"shipping_address_street":"",
"shipping_address_street_2":"",
"shipping_address_street_3":"",
"shipping_address_street_4":"",
"shipping_address_city":"",
"shipping_address_state":"",
"shipping_address_postalcode":"",
"shipping_address_country":"",
"parent_id":"",
"sic_code":"",
"duns_num":"",
"parent_name":"",
"member_of":{
   "name":"",
   "id":"",
   "_acl":{
      "fields":[

      ],
      "_hash":"654d337e0e912edaa00dbb0fb3dc3c17"
   }
},
"campaign_id":"",
"campaign_name":"",
"campaign_accounts":{
   "name":"",
   "id":"",
   "_acl":{
      "fields":[

      ],
      "_hash":"654d337e0e912edaa00dbb0fb3dc3c17"
   }
},
"following":true,
"my_favorite":false,
"tag":[
```

```
],
"assigned_user_id":"",
"assigned_user_name":"",
"assigned_user_link":{
   "full_name":"",
   "id":"",
   "_acl":{
      "fields":[

      ],
      "_hash":"654d337e0e912edaa00dbb0fb3dc3c17"
   }
},
"team_count":"",
"team_count_link":{
   "team_count":"",
   "id":"1",
   "_acl":{
      "fields":[

      ],
      "_hash":"654d337e0e912edaa00dbb0fb3dc3c17"
   }
},
"team_name":[
   {
      "id":1,
      "name":"Global",
      "name_2":"",
      "primary":true
   }
],
"email":[
   {
      "email_address":"test@sugar.com",
      "invalid_email":false,
      "opt_out":false,
      "primary_address":true,
      "reply_to_address":false
   }
],
"email1":"test@sugar.com",
"email2":"",
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"",
```

```
   "_acl":{
      "fields":{

      }
   },
   "_module":"Accounts"
}
```

Deleting a Record

Next, we can delete the record from the Sugar instance using the /<module>/:record endpoint, by using the DELETE Http Method.

```
  curl -X DELETE -H OAuth-Token:<access_token> -H Cache-Control:no-
cache http://<site_url>/rest/v11/Accounts/<record_id>
```

More information on this API endpoint can be found in the /<module>/:record - DELETE documentation.

Response

The data received from the server is shown below:

```
  {
   "id":"ab2222df-73da-0e92-6887-5705428f4d68"
}
```

How to Follow a Record

Overview

An example in bash script demonstrating how to follow a record using the v11 /<module>/:record/subscribe REST POST endpoint.

Following a Record

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  curl -X POST -H Cache-Control:no-cache -H "Content-
Type: application/json" -d '{
 "grant_type":"password",
 "client_id":"sugar",
 "client_secret":"",
 "username":"admin",
 "password":"password",
 "platform":"custom_api"
}' https://{site_url}/rest/v11/oauth2/token
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Following a Record

Next, we can follow a specific record using the /<module>/:record/subscribe endpoint.

```
  curl -s -X POST -H OAuth-Token:{access_token} -H Cache-Control:no-ca
che https://{site_url}/rest/v11/Accounts/e3a59dcd-
ff5e-8a78-cd7f-570811366792/subscribe
```

More information on the subscribe API can be found in the /<module>/:record/subscribe POST documentation.

Response

The data received from the server is shown below:

```
  "58f96315-9e75-6562-42e9-5705917d2cdc"
```

How to Unfavorite a Record

Overview

An example in bash script demonstrating how to unfavorite a record using the v11 /<module>/:record/unfavorite REST PUT endpoint.

Unfavoriting a Record

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  curl -X POST -H Cache-Control:no-cache -H "Content-
Type: application/json" -d '{
 "grant_type":"password",
 "client_id":"sugar",
 "client_secret":"",
 "username":"admin",
 "password":"password",
 "platform":"custom_api"
}' https://{site_url}/rest/v11/oauth2/token
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Unfavoriting a Record

Next, we can unfavorite a specific record using the /<module>/:record/unfavorite endpoint.

```
  curl -s -X PUT -H OAuth-Token:{access_token} -H Cache-Control:no-cac
he https://{site_url}/rest/v11/Accounts/ae8b1783-404a-
fcb8-1e1e-56f1cc52cd1a/unfavorite
```

More information on the unfavorite API can be found in the /<module>/:record/unfavorite PUT documentation.

Response

The data received from the server is shown below:


```
  {
    "id": "ae8b1783-404a-fcb8-1e1e-56f1cc52cd1a",
    "name": "Union Bank",
    "date_entered": "2016-03-22T17:49:50-05:00",
    "date_modified": "2016-03-30T17:44:20-05:00",
    "modified_user_id": "1",
    "modified_by_name": "Administrator",
    "modified_user_link": {
```

```
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
        "fields": [],
        "delete": "no",
        "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
    }
},
"created_by": "1",
"created_by_name": "Administrator",
"created_by_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
        "fields": [],
        "delete": "no",
        "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
    }
},
"description": "",
"deleted": false,
"facebook": "",
"twitter": "",
"googleplus": "",
"account_type": "Customer",
"industry": "Banking",
"annual_revenue": "",
"phone_fax": "",
"billing_address_street": "67321 West Siam St.",
"billing_address_street_2": "",
"billing_address_street_3": "",
"billing_address_street_4": "",
"billing_address_city": "Ohio",
"billing_address_state": "CA",
"billing_address_postalcode": "25159",
"billing_address_country": "USA",
"rating": "",
"phone_office": "(065) 489-6104",
"phone_alternate": "",
"website": "www.qahr.edu",
"ownership": "",
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "67321 West Siam St.",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
```

```
"shipping_address_street_4": "",
"shipping_address_city": "Ohio",
"shipping_address_state": "CA",
"shipping_address_postalcode": "25159",
"shipping_address_country": "USA",
"parent_id": "",
"sic_code": "",
"duns_num": "",
"parent_name": "",
"member_of": {
    "name": "",
    "id": "",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"campaign_id": "",
"campaign_name": "",
"campaign_accounts": {
    "name": "",
    "id": "",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"following": true,
"my_favorite": false,
"tag": [],
"assigned_user_id": "seed_sarah_id",
"assigned_user_name": "Sarah Smith",
"assigned_user_link": {
    "full_name": "Sarah Smith",
    "id": "seed_sarah_id",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"team_count": "",
"team_count_link": {
    "team_count": "",
    "id": "West",
    "_acl": {
        "fields": [],
```

```
            "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
        }
    },
    "team_name": [{
        "id": "East",
        "name": "East",
        "name_2": "",
        "primary": false
    }, {
        "id": 1,
        "name": "Global",
        "name_2": "",
        "primary": false
    }, {
        "id": "West",
        "name": "West",
        "name_2": "",
        "primary": true
    }],
    "email": [{
        "email_address": "hr.support.kid@example.info",
        "invalid_email": false,
        "opt_out": false,
        "primary_address": true,
        "reply_to_address": false
    }, {
        "email_address": "info.support.the@example.com",
        "invalid_email": false,
        "opt_out": false,
        "primary_address": false,
        "reply_to_address": false
    }],
    "email1": "hr.support.kid@example.info",
    "email2": "info.support.the@example.com",
    "invalid_email": false,
    "email_opt_out": false,
    "email_addresses_non_primary": "",
    "_acl": {
        "fields": {}
    },
    "_module": "Accounts"
}
```


How to Unfollow a Record

Overview

An example in bash script demonstrating how to unfollow a record using the v11 /<module>/:record/unsubscribe REST DELETE endpoint.

Unfollowing a Record

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  curl -X POST -H Cache-Control:no-cache -H "Content-
Type: application/json" -d '{
 "grant_type":"password",
 "client_id":"sugar",
 "client_secret":"",
 "username":"admin",
 "password":"password",
 "platform":"custom_api"
}' https://{site_url}/rest/v11/oauth2/token
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Unfollowing a Record

Next, we can unfollow a specific record using the /<module>/:record/unsubscribe endpoint.

```
  curl -s -X DELETE -H OAuth-Token:{access_token} -H Cache-Control:no-
cache https://{site_url}/rest/v11/Accounts/e7b0d745-0a3e-
c896-5358-5708113786d7/unsubscribe
```

More information on the unsubscribe API can be found in the /:record/unsubscribe DELETE documentation.

Response

The data received from the server is shown below:

```
  true
```

How to Get the Most Active Users

Overview

An example in bash script demonstrating how to fetch the most active users for meetings, calls, inbound emails, and outbound emails using the v11 /mostactiveusers REST GET endpoint.

Get Most Active Users

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  curl -X POST -H Cache-Control:no-cache -H "Content-
Type: application/json" -d '{
 "grant_type":"password",
 "client_id":"sugar",
 "client_secret":"",
 "username":"admin",
 "password":"password",
 "platform":"custom_api"
}' https://{site_url}/rest/v11/oauth2/token
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Active Users

Next, we can retrieve the most active users using the /mostactiveusers endpoint.

```
  curl -s -X GET -H OAuth-Token:{access_token} -H Cache-Control:no-
cache https://{site_url}/rest/v11/mostactiveusers?days=30
```

More information on the mostactiveusers API can be found in the /mostactiveusers documentation.

Response

The data received from the server is shown below:

```
  {
    "meetings": {
        "user_id": "seed_max_id",
        "count": "21",
        "first_name": "Max",
        "last_name": "Jensen"
    },
    "calls": {
        "user_id": "seed_chris_id",
        "count": "4",
        "first_name": "Chris",
        "last_name": "Olliver"
    },
    "inbound_emails": {
        "user_id": "seed_chris_id",
        "count": "23",
        "first_name": "Chris",
        "last_name": "Olliver"
    },
    "outbound_emails": {
        "user_id": "seed_sarah_id",
        "count": "20",
        "first_name": "Sarah",
        "last_name": "Smith"
    }
}
```

How to Authenticate and Log Out

Overview

An example in bash script on how to authenticate and logout of the v11 REST API using the /oauth2/token and /oauth2/logout POST endpoints.

Authenticating

The example below demonstrates how to authenticate to the REST v11 API. It is important to note that the oauth2 token arguments takes in several parameters that you should be aware of. The platform argument tends to cause confusion in that it is used to authenticate a user to a specific platform. Since Sugar only allows 1 user in the system at a time per platform, authenticating an integration script with a platform type of "base" will logout any current users in the system using those credentials. To work around this, your custom scripts should have a new platform type specified such as "custom_api" or any other static text you prefer.

The client_id and client_secret parameters can be used for additional security based on client types. You can create your own client type in Admin > OAuth Keys. More information can be found in the /oauth2/token documentation. An example script is shown below:

```
  curl -X POST -H Cache-Control:no-cache -H "Content-
Type: application/json" -d '{
 "grant_type":"password",
 "client_id":"sugar",
 "client_secret":"",
 "username":"admin",
 "password":"password",
 "platform":"custom_api"
}' https://{site_url}/rest/v11/oauth2/token
```

Response

The data received from the server is shown below:

```
  {
   "access_token":"c6d495c9-bb25-81d2-5f81-533ef6479f9b",
   "expires_in":3600,
   "token_type":"bearer",
   "scope":null,
   "refresh_token":"cbc40e67-12bc-4b56-a1d9-533ef62f2601",
   "refresh_expires_in":1209600,
   "download_token":"cc5d1a9f-6627-3349-96e5-533ef6b1a493"
}
```

Logout

To log out of a session, a request can be made to the /oauth2/logout POST endpoint. More information can be found in the /oauth2/logout documentation. An example extending off of the above authentication example is shown below:

```
  curl -s -X POST -H OAuth-Token:<access_token> -H Cache-Control:no-
cache https://{site_url}/rest/v11/oauth2/logout
```

Response

The data received from the server is shown below:

```
  {
    "success":true
}
```

How to Ping

Overview

An example in bash script demonstrating how to ping using the REST v11 /ping GET endpoint.

Pinging

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  curl -X POST -H Cache-Control:no-cache -H "Content-
Type: application/json" -d '{
 "grant_type":"password",
 "client_id":"sugar",
 "client_secret":"",
 "username":"admin",
 "password":"password",
 "platform":"custom_api"
}' https://{site_url}/rest/v11/oauth2/token
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Pinging

Once we have authenticated, we can now ping the system as shown below.

```
  curl -s -X GET -H OAuth-Token:{access_token} -H Cache-Control:no-
cache https://{site_url}/rest/v11/ping
```

More information on pinging can be found in the /ping documentation.

Response

```
"pong"
```

How to Fetch the Current Users Time

Overview

An example in bash script demonstrating how to fetch the current users time with the v11 /ping/whattimeisit REST GET endpoint.

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  curl -X POST -H Cache-Control:no-cache -H "Content-
Type: application/json" -d '{
 "grant_type":"password",
 "client_id":"sugar",
 "client_secret":"",
 "username":"admin",
 "password":"password",
 "platform":"custom_api"
}' https://{site_url}/rest/v11/oauth2/token
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Getting the Current Time and Date for the User

Next, we can get the current time and date using the /ping/whattimeisit endpoint.

```
  curl -s -X GET -H OAuth-Token:{access_token} -H Cache-Control:no-
cache https://{site_url}/rest/v11/ping/whattimeisit
```

Response

```
  "2017-11-03T06:45:59-07:00"
```

How to Fetch Recently Viewed Records

Overview

An example in bash script demonstrating how to retrieve recently viewed records using the v11 /recent REST GET endpoint.

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  curl -X POST -H Cache-Control:no-cache -H "Content-
Type: application/json" -d '{
 "grant_type":"password",
 "client_id":"sugar",
 "client_secret":"",
 "username":"admin",
 "password":"password",
 "platform":"custom_api"
}' https://{site_url}/rest/v11/oauth2/token
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Recently Viewed Records

Next, we will need to identify the records we want to see using the /recent endpoint. In this case, we are going to request Accounts, Contacts and Leads.

```
  curl -s -X GET -H OAuth-Token:{access_token} -H Cache-Control:no-cac
he https://{site_url}/rest/v11/recent?module_list=Accounts%2CContacts%
2CLeads
```

More information on the search API can be found in the /recent documentation.

Response

The data received from the server is shown below:

```
  {
```

```
"next_offset":-1,
"records":[
   {
      "my_favorite":false,
      "following":false,
      "id":"f3951f4d-2d17-7939-c5ec-56fedbb9e92f",
      "name":"Talia Knupp",
      "date_entered":"2016-04-01T15:34:00-05:00",
      "date_modified":"2016-04-06T10:33:24-05:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "doc_owner":"",
      "user_favorites":"",
      "description":"",
      "deleted":false,
      "assigned_user_id":"seed_will_id",
      "assigned_user_name":"Will Westin",
      "team_count":"",
      "team_name":[
         {
            "id":"East",
            "name":"East",
            "name_2":"",
            "primary":true
         },
         {
            "id":"West",
            "name":"West",
            "name_2":"",
            "primary":false
         }
      ],
      "email":[
         {
            "email_address":"jsmith@sugar.com",
            "invalid_email":false,
            "opt_out":false,
            "primary_address":true,
            "reply_to_address":false
         },
         {
            "email_address":"cjsmith@sugar.com",
            "invalid_email":false,
            "opt_out":false,
```

```
      "primary_address":false,
      "reply_to_address":false
   }
],
"email1":"jsmith@sugar.com",
"email2":"cjsmith@sugar.com",
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"",
"salutation":"",
"first_name":"Talia",
"last_name":"Knupp",
"full_name":"Talia Knupp",
"title":"Senior Product Manager",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(963) 741-3689",
"phone_mobile":"(600) 831-9872",
"phone_work":"(680) 991-2837",
"phone_other":"",
"phone_fax":"",
"primary_address_street":"111 Silicon Valley Road",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Sunnyvale",
"primary_address_state":"NY",
"primary_address_postalcode":"99452",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
"assistant":"",
"assistant_phone":"",
"picture":"",
"converted":false,
"refered_by":"",
"lead_source":"Word of mouth",
"lead_source_description":"",
"status":"In Process",
```

```
"status_description":"",
"reports_to_id":"",
"report_to_name":"",
"dnb_principal_id":"",
"account_name":"First National S\/B",
"account_description":"",
"contact_id":"",
"contact_name":"",
"account_id":"",
"opportunity_id":"",
"opportunity_name":"",
"opportunity_amount":"",
"campaign_id":"",
"campaign_name":"",
"c_accept_status_fields":"",
"m_accept_status_fields":"",
"accept_status_id":"",
"accept_status_name":"",
"accept_status_calls":"",
"accept_status_meetings":"",
"webtolead_email1":[
   {
      "email_address":"jsmith@sugar.com",
      "invalid_email":false,
      "opt_out":false,
      "primary_address":true,
      "reply_to_address":false
   },
   {
      "email_address":"cjsmith@sugar.com",
      "invalid_email":false,
      "opt_out":false,
      "primary_address":false,
      "reply_to_address":false
   }
],
"webtolead_email2":[
   {
      "email_address":"jsmith@sugar.com",
      "invalid_email":false,
      "opt_out":false,
      "primary_address":true,
      "reply_to_address":false
   },
   {
      "email_address":"cjsmith@sugar.com",
```

```
            "invalid_email":false,
            "opt_out":false,
            "primary_address":false,
            "reply_to_address":false
         }
      ],
      "webtolead_email_opt_out":"",
      "webtolead_invalid_email":"",
      "birthdate":"",
      "portal_name":"",
      "portal_app":"",
      "website":"",
      "preferred_language":"",
      "mkto_sync":false,
      "mkto_id":null,
      "mkto_lead_score":null,
      "fruits_c":"Apples",
      "_acl":{
         "fields":{

         }
      },
      "_module":"Leads",
      "_last_viewed_date":"2016-04-06T10:33:24-05:00"
   },
   {
      "my_favorite":false,
      "following":false,
      "id":"e8c641ca-1b8c-74c1-d08d-56fedbdd3187",
      "name":"MTM Investment Bank F S B",
      "date_entered":"2016-04-01T15:34:00-05:00",
      "date_modified":"2016-04-06T10:16:52-05:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "doc_owner":"",
      "user_favorites":"",
      "description":"",
      "deleted":false,
      "assigned_user_id":"seed_will_id",
      "assigned_user_name":"Will Westin",
      "team_count":"",
      "team_name":[
         {
            "id":"East",
```

```
            "name":"East",
            "name_2":"",
            "primary":true
        },
        {
            "id":"West",
            "name":"West",
            "name_2":"",
            "primary":false
        }
    ],
    "email":[
        {
            "email_address":"jsmith@sugar.com",
            "invalid_email":false,
            "opt_out":false,
            "primary_address":true,
            "reply_to_address":false
        },
        {
            "email_address":"the60@example.us",
            "invalid_email":false,
            "opt_out":false,
            "primary_address":false,
            "reply_to_address":false
        }
    ],
    "email1":"jsmith@sugar.com",
    "email2":"the60@example.us",
    "invalid_email":false,
    "email_opt_out":false,
    "email_addresses_non_primary":"",
    "facebook":"",
    "twitter":"",
    "googleplus":"",
    "account_type":"Customer",
    "industry":"a",
    "annual_revenue":"",
    "phone_fax":"",
    "billing_address_street":"67321 West Siam St.",
    "billing_address_street_2":"",
    "billing_address_street_3":"",
    "billing_address_street_4":"",
    "billing_address_city":"Alabama",
    "billing_address_state":"NY",
    "billing_address_postalcode":"52272",
```

```
      "billing_address_country":"USA",
      "rating":"",
      "phone_office":"(012) 704-8075",
      "phone_alternate":"",
      "website":"www.salesqa.it",
      "ownership":"",
      "employees":"",
      "ticker_symbol":"",
      "shipping_address_street":"67321 West Siam St.",
      "shipping_address_street_2":"",
      "shipping_address_street_3":"",
      "shipping_address_street_4":"",
      "shipping_address_city":"Alabama",
      "shipping_address_state":"NY",
      "shipping_address_postalcode":"52272",
      "shipping_address_country":"USA",
      "parent_id":"",
      "sic_code":"",
      "duns_num":"",
      "parent_name":"",
      "campaign_id":"",
      "campaign_name":"",
      "_acl":{
         "fields":{

         }
      },
      "_module":"Accounts",
      "_last_viewed_date":"2016-04-06T10:16:52-05:00"
   },
   {
      "my_favorite":false,
      "following":false,
      "id":"f31b2f00-468c-3d35-1e88-56fedbd3921d",
      "name":"Kaycee Gibney",
      "date_entered":"2016-04-01T15:34:00-05:00",
      "date_modified":"2016-04-06T10:16:24-05:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "doc_owner":"",
      "user_favorites":"",
      "description":"",
      "deleted":false,
      "assigned_user_id":"seed_jim_id",
```

```
"assigned_user_name":"Jim Brennan",
"team_count":"",
"team_name":[
   {
      "id":"East",
      "name":"East",
      "name_2":"",
      "primary":true
   }
],
"email":[
   {
      "email_address":"jsmith@sugar.com",
      "invalid_email":false,
      "opt_out":false,
      "primary_address":true,
      "reply_to_address":false
   },
   {
      "email_address":"sales.kid.dev@example.info",
      "invalid_email":false,
      "opt_out":true,
      "primary_address":false,
      "reply_to_address":false
   }
],
"email1":"jsmith@sugar.com",
"email2":"sales.kid.dev@example.info",
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"",
"salutation":"",
"first_name":"Kaycee",
"last_name":"Gibney",
"full_name":"Kaycee Gibney",
"title":"Mgr Operations",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(599) 165-2396",
"phone_mobile":"(215) 591-9574",
"phone_work":"(771) 945-3648",
"phone_other":"",
"phone_fax":"",
```

```
"primary_address_street":"321 University Ave.",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Santa Monica",
"primary_address_state":"NY",
"primary_address_postalcode":"96154",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
"assistant":"",
"assistant_phone":"",
"picture":"",
"email_and_name1":"",
"lead_source":"Existing Customer",
"account_name":"Tracker Com LP",
"account_id":"72ad6f00-e345-1cab-b370-56fedbd23deb",
"dnb_principal_id":"",
"opportunity_role_fields":"",
"opportunity_role_id":"",
"opportunity_role":"",
"reports_to_id":"",
"report_to_name":"",
"birthdate":"",
"portal_name":"KayceeGibney33",
"portal_active":true,
"portal_password":true,
"portal_password1":null,
"portal_app":"",
"preferred_language":"en_us",
"campaign_id":"",
"campaign_name":"",
"c_accept_status_fields":"",
"m_accept_status_fields":"",
"accept_status_id":"",
"accept_status_name":"",
"accept_status_calls":"",
"accept_status_meetings":"",
"sync_contact":false,
"mkto_sync":false,
"mkto_id":null,
"mkto_lead_score":null,
```

```
         "_acl":{
            "fields":{

            }
         },
         "_module":"Contacts",
         "_last_viewed_date":"2016-04-06T10:16:24-05:00"
      }
   ]
}
```

There are 3 records shown above, the _module field tells you if the record is from Accounts, Contacts or Leads. The _last_viewed_date tells you when the record was last seen.

How to Use the Global Search

Overview

An example in bash script demonstrating how to globally search for records using the REST v11 /search GET endpoint.

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  curl -X POST -H Cache-Control:no-cache -H "Content-
Type: application/json" -d '{
 "grant_type":"password",
 "client_id":"sugar",
 "client_secret":"",
 "username":"admin",
 "password":"password",
 "platform":"custom_api"
}' https://{site_url}/rest/v11/oauth2/token
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Searching Records

Next, we will need to identify the records we want to see using the /search endpoint. In this case, we are going to search for all records that have the email address of 'jsmith@sugar.com'. In this example, there are 3 records, an Account, a Contact and a Lead.

```
  curl -s -X GET -H OAuth-Token:{access_token} -H Cache-Control:no-
cache https://{site_url}/rest/v11/search?
 q=jsmith@sugar.com&
 max_num=3&
 offset=0&
 fields=&
 order_by=&
 favorites=false&
 my_items=false
```

More information on the search API can be found in the /search documentation.

Response

The data received from the server is shown below:


```
  {
   "next_offset":-1,
   "records":[
      {
         "my_favorite":false,
         "following":false,
         "id":"f31b2f00-468c-3d35-1e88-56fedbd3921d",
         "name":"Kaycee Gibney",
         "date_entered":"2016-04-01T20:34:00+00:00",
         "date_modified":"2016-04-06T15:16:24+00:00",
         "modified_user_id":"1",
         "modified_by_name":"Administrator",
         "created_by":"1",
         "created_by_name":"Administrator",
         "doc_owner":"",
         "user_favorites":"",
         "description":"",
         "deleted":false,
         "assigned_user_id":"seed_jim_id",
         "assigned_user_name":"Jim Brennan",
         "team_count":"",
         "team_name":[
```

```
      {
         "id":"East",
         "name":"East",
         "name_2":"",
         "primary":true
      }
   ],
   "email":[
      {
         "email_address":"jsmith@sugar.com",
         "invalid_email":false,
         "opt_out":false,
         "primary_address":true,
         "reply_to_address":false
      },
      {
         "email_address":"sales.kid.dev@example.info",
         "invalid_email":false,
         "opt_out":true,
         "primary_address":false,
         "reply_to_address":false
      }
   ],
   "email1":"jsmith@sugar.com",
   "email2":"sales.kid.dev@example.info",
   "invalid_email":false,
   "email_opt_out":false,
   "email_addresses_non_primary":"",
   "salutation":"",
   "first_name":"Kaycee",
   "last_name":"Gibney",
   "full_name":"Kaycee Gibney",
   "title":"Mgr Operations",
   "facebook":"",
   "twitter":"",
   "googleplus":"",
   "department":"",
   "do_not_call":false,
   "phone_home":"(599) 165-2396",
   "phone_mobile":"(215) 591-9574",
   "phone_work":"(771) 945-3648",
   "phone_other":"",
   "phone_fax":"",
   "primary_address_street":"321 University Ave.",
   "primary_address_street_2":"",
   "primary_address_street_3":"",
```

```
"primary_address_city":"Santa Monica",
"primary_address_state":"NY",
"primary_address_postalcode":"96154",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
"assistant":"",
"assistant_phone":"",
"picture":"",
"email_and_name1":"",
"lead_source":"Existing Customer",
"account_name":"Tracker Com LP",
"account_id":"72ad6f00-e345-1cab-b370-56fedbd23deb",
"dnb_principal_id":"",
"opportunity_role_fields":"",
"opportunity_role_id":"",
"opportunity_role":"",
"reports_to_id":"",
"report_to_name":"",
"birthdate":"",
"portal_name":"KayceeGibney33",
"portal_active":true,
"portal_password":true,
"portal_password1":null,
"portal_app":"",
"preferred_language":"en_us",
"campaign_id":"",
"campaign_name":"",
"c_accept_status_fields":"",
"m_accept_status_fields":"",
"accept_status_id":"",
"accept_status_name":"",
"accept_status_calls":"",
"accept_status_meetings":"",
"sync_contact":false,
"mkto_sync":false,
"mkto_id":null,
"mkto_lead_score":null,
"_acl":{
   "fields":{
```

```
            }
         },
         "_module":"Contacts",
         "_search":{
            "score":0.70710677,
            "highlighted":{
               "email1":{
                  "text":"\u003Cstrong\u003Ejsmith@sugar.com\u003C\/st
rong\u003E",
                  "module":"Contacts",
                  "label":"LBL_EMAIL_ADDRESS"
               }
            }
         }
      },
      {
         "my_favorite":false,
         "following":false,
         "id":"e8c641ca-1b8c-74c1-d08d-56fedbdd3187",
         "name":"MTM Investment Bank F S B",
         "date_entered":"2016-04-01T20:34:00+00:00",
         "date_modified":"2016-04-06T15:16:52+00:00",
         "modified_user_id":"1",
         "modified_by_name":"Administrator",
         "created_by":"1",
         "created_by_name":"Administrator",
         "doc_owner":"",
         "user_favorites":"",
         "description":"",
         "deleted":false,
         "assigned_user_id":"seed_will_id",
         "assigned_user_name":"Will Westin",
         "team_count":"",
         "team_name":[
            {
               "id":"East",
               "name":"East",
               "name_2":"",
               "primary":true
            },
            {
               "id":"West",
               "name":"West",
               "name_2":"",
               "primary":false
            }
```

```
],
"email":[
   {
      "email_address":"jsmith@sugar.com",
      "invalid_email":false,
      "opt_out":false,
      "primary_address":true,
      "reply_to_address":false
   },
   {
      "email_address":"the60@example.us",
      "invalid_email":false,
      "opt_out":false,
      "primary_address":false,
      "reply_to_address":false
   }
],
"email1":"jsmith@sugar.com",
"email2":"the60@example.us",
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"account_type":"Customer",
"industry":"a",
"annual_revenue":"",
"phone_fax":"",
"billing_address_street":"67321 West Siam St.",
"billing_address_street_2":"",
"billing_address_street_3":"",
"billing_address_street_4":"",
"billing_address_city":"Alabama",
"billing_address_state":"NY",
"billing_address_postalcode":"52272",
"billing_address_country":"USA",
"rating":"",
"phone_office":"(012) 704-8075",
"phone_alternate":"",
"website":"www.salesqa.it",
"ownership":"",
"employees":"",
"ticker_symbol":"",
"shipping_address_street":"67321 West Siam St.",
"shipping_address_street_2":"",
```

```
         "shipping_address_street_3":"",
         "shipping_address_street_4":"",
         "shipping_address_city":"Alabama",
         "shipping_address_state":"NY",
         "shipping_address_postalcode":"52272",
         "shipping_address_country":"USA",
         "parent_id":"",
         "sic_code":"",
         "duns_num":"",
         "parent_name":"",
         "campaign_id":"",
         "campaign_name":"",
         "_acl":{
            "fields":{

            }
         },
         "_module":"Accounts",
         "_search":{
            "score":0.70710677,
            "highlighted":{
               "email1":{
                  "text":"\u003Cstrong\u003Ejsmith@sugar.com\u003C\/st
rong\u003E",
                  "module":"Accounts",
                  "label":"LBL_EMAIL_ADDRESS"
               }
            }
         }
      },
      {
         "my_favorite":false,
         "following":false,
         "id":"f3951f4d-2d17-7939-c5ec-56fedbb9e92f",
         "name":"Talia Knupp",
         "date_entered":"2016-04-01T20:34:00+00:00",
         "date_modified":"2016-04-06T15:33:24+00:00",
         "modified_user_id":"1",
         "modified_by_name":"Administrator",
         "created_by":"1",
         "created_by_name":"Administrator",
         "doc_owner":"",
         "user_favorites":"",
         "description":"",
         "deleted":false,
         "assigned_user_id":"seed_will_id",
```

```
"assigned_user_name":"Will Westin",
"team_count":"",
"team_name":[
   {
      "id":"East",
      "name":"East",
      "name_2":"",
      "primary":true
   },
   {
      "id":"West",
      "name":"West",
      "name_2":"",
      "primary":false
   }
],
"email":[
   {
      "email_address":"jsmith@sugar.com",
      "invalid_email":false,
      "opt_out":false,
      "primary_address":true,
      "reply_to_address":false
   },
   {
      "email_address":"cjsmith@sugar.com",
      "invalid_email":false,
      "opt_out":false,
      "primary_address":false,
      "reply_to_address":false
   }
],
"email1":"jsmith@sugar.com",
"email2":"cjsmith@sugar.com",
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"",
"salutation":"",
"first_name":"Talia",
"last_name":"Knupp",
"full_name":"Talia Knupp",
"title":"Senior Product Manager",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
```

```
"do_not_call":false,
"phone_home":"(963) 741-3689",
"phone_mobile":"(600) 831-9872",
"phone_work":"(680) 991-2837",
"phone_other":"",
"phone_fax":"",
"primary_address_street":"111 Silicon Valley Road",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Sunnyvale",
"primary_address_state":"NY",
"primary_address_postalcode":"99452",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
"assistant":"",
"assistant_phone":"",
"picture":"",
"converted":false,
"refered_by":"",
"lead_source":"Word of mouth",
"lead_source_description":"",
"status":"In Process",
"status_description":"",
"reports_to_id":"",
"report_to_name":"",
"dnb_principal_id":"",
"account_name":"First National S\/B",
"account_description":"",
"contact_id":"",
"contact_name":"",
"account_id":"",
"opportunity_id":"",
"opportunity_name":"",
"opportunity_amount":"",
"campaign_id":"",
"campaign_name":"",
"c_accept_status_fields":"",
"m_accept_status_fields":"",
"accept_status_id":"",
"accept_status_name":"",
```

```
"accept_status_calls":"",
"accept_status_meetings":"",
"webtolead_email1":[
   {
      "email_address":"jsmith@sugar.com",
      "invalid_email":false,
      "opt_out":false,
      "primary_address":true,
      "reply_to_address":false
   },
   {
      "email_address":"cjsmith@sugar.com",
      "invalid_email":false,
      "opt_out":false,
      "primary_address":false,
      "reply_to_address":false
   }
],
"webtolead_email2":[
   {
      "email_address":"jsmith@sugar.com",
      "invalid_email":false,
      "opt_out":false,
      "primary_address":true,
      "reply_to_address":false
   },
   {
      "email_address":"cjsmith@sugar.com",
      "invalid_email":false,
      "opt_out":false,
      "primary_address":false,
      "reply_to_address":false
   }
],
"webtolead_email_opt_out":"",
"webtolead_invalid_email":"",
"birthdate":"",
"portal_name":"",
"portal_app":"",
"website":"",
"preferred_language":"",
"mkto_sync":false,
"mkto_id":null,
"mkto_lead_score":null,
"fruits_c":"Apples",
"_acl":{
```

```
            "fields":{

            }
         },
         "_module":"Leads",
         "_search":{
            "score":0.70710677,
            "highlighted":{
               "email1":{
                  "text":"\u003Cstrong\u003Ejsmith@sugar.com\u003C\/st
rong\u003E",
                  "module":"Leads",
                  "label":"LBL_EMAIL_ADDRESS"
               }
            }
         }
      }
   ]
}
```


There are 3 records shown above, the _module field tells you if the record is from Accounts, Contacts or Leads. The _search field is an array that tells you where in the record it found the search string. It gives you back a highlighted string that you can use for display.

How to Check for Duplicate Records

Overview

An example in bash script demonstrating how to check for duplicate records using the v11 /<module>/duplicateCheck REST POST endpoint.

Duplicate Records

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  curl -X POST -H Cache-Control:no-cache -H "Content-
Type: application/json" -d '{
 "grant_type":"password",
 "client_id":"sugar",
```

```
 "client_secret":"",
 "username":"admin",
 "password":"password",
 "platform":"custom_api"
}' https://{site_url}/rest/v11/oauth2/token
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Retrieving Duplicates

Next, we will need to identify the records that are duplicates using the /<module>/duplicateCheck endpoint.

```
  curl -s -X POST -H OAuth-Token:{access_token} -H Cache-Control:no-
cache -H "Content-Type: application/json" -d '{
    "name":"Test Record"
}' https://{site_url}/rest/v11/Accounts/duplicateCheck
```

More information on the duplicate check API can be found in the /<module>/duplicateCheck documentation.

Response

The data received from the server is shown below:

```
  {
    "next_offset": -1,
    "records": [{
        "id": "7f6ea7be-60d6-11e6-8885-a0999b033b33",
        "name": "Test Record",
        "date_entered": "2016-08-12T14:48:25-07:00",
        "date_modified": "2016-08-12T14:48:25-07:00",
        "modified_user_id": "1",
        "modified_by_name": "Administrator",
        "modified_user_link": {
            "full_name": "Administrator",
            "id": "1",
            "_acl": {
                "fields": [],
                "delete": "no",
                "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
```

```
        }
    },
    "created_by": "1",
    "created_by_name": "Administrator",
    "created_by_link": {
        "full_name": "Administrator",
        "id": "1",
        "_acl": {
            "fields": [],
            "delete": "no",
            "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
        }
    },
    "description": "Test Data 1",
    "deleted": false,
    "facebook": "",
    "twitter": "",
    "googleplus": "",
    "account_type": "",
    "industry": "",
    "annual_revenue": "",
    "phone_fax": "",
    "billing_address_street": "",
    "billing_address_street_2": "",
    "billing_address_street_3": "",
    "billing_address_street_4": "",
    "billing_address_city": "",
    "billing_address_state": "",
    "billing_address_postalcode": "",
    "billing_address_country": "",
    "rating": "",
    "phone_office": "",
    "phone_alternate": "",
    "website": "",
    "ownership": "",
    "employees": "",
    "ticker_symbol": "",
    "shipping_address_street": "",
    "shipping_address_street_2": "",
    "shipping_address_street_3": "",
    "shipping_address_street_4": "",
    "shipping_address_city": "",
    "shipping_address_state": "",
    "shipping_address_postalcode": "",
    "shipping_address_country": "",
    "parent_id": "",
```

```
"sic_code": "",
"duns_num": "",
"parent_name": "",
"member_of": {
    "name": "",
    "id": "",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"campaign_id": "",
"campaign_name": "",
"campaign_accounts": {
    "name": "",
    "id": "",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"following": true,
"my_favorite": false,
"tag": [],
"assigned_user_id": "1",
"assigned_user_name": "Administrator",
"assigned_user_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
        "fields": [],
        "delete": "no",
        "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
    }
},
"team_count": "",
"team_count_link": {
    "team_count": "",
    "id": "1",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"team_name": [{
    "id": "1",
```

```
            "name": "Global",
            "name_2": "",
            "primary": true
        }],
        "email": [],
        "email1": "",
        "email2": "",
        "invalid_email": "",
        "email_opt_out": "",
        "email_addresses_non_primary": "",
        "_acl": {
            "fields": {}
        },
        "_module": "Accounts",
        "duplicate_check_rank": 8
    }, {
        "id": "868b4f16-60d6-11e6-bdfc-a0999b033b33",
        "name": "Test Record",
        "date_entered": "2016-08-12T14:48:37-07:00",
        "date_modified": "2016-08-12T14:48:37-07:00",
        "modified_user_id": "1",
        "modified_by_name": "Administrator",
        "modified_user_link": {
            "full_name": "Administrator",
            "id": "1",
            "_acl": {
                "fields": [],
                "delete": "no",
                "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
            }
        },
        "created_by": "1",
        "created_by_name": "Administrator",
        "created_by_link": {
            "full_name": "Administrator",
            "id": "1",
            "_acl": {
                "fields": [],
                "delete": "no",
                "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
            }
        },
        "description": "Test Data 2",
        "deleted": false,
        "facebook": "",
        "twitter": "",
```

```
"googleplus": "",
"account_type": "",
"industry": "",
"annual_revenue": "",
"phone_fax": "",
"billing_address_street": "",
"billing_address_street_2": "",
"billing_address_street_3": "",
"billing_address_street_4": "",
"billing_address_city": "",
"billing_address_state": "",
"billing_address_postalcode": "",
"billing_address_country": "",
"rating": "",
"phone_office": "",
"phone_alternate": "",
"website": "",
"ownership": "",
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "",
"shipping_address_state": "",
"shipping_address_postalcode": "",
"shipping_address_country": "",
"parent_id": "",
"sic_code": "",
"duns_num": "",
"parent_name": "",
"member_of": {
    "name": "",
    "id": "",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"campaign_id": "",
"campaign_name": "",
"campaign_accounts": {
    "name": "",
    "id": "",
    "_acl": {
```

```
                "fields": [],
                "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
            }
        },
        "following": true,
        "my_favorite": false,
        "tag": [],
        "assigned_user_id": "1",
        "assigned_user_name": "Administrator",
        "assigned_user_link": {
            "full_name": "Administrator",
            "id": "1",
            "_acl": {
                "fields": [],
                "delete": "no",
                "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
            }
        },
        "team_count": "",
        "team_count_link": {
            "team_count": "",
            "id": "1",
            "_acl": {
                "fields": [],
                "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
            }
        },
        "team_name": [{
            "id": "1",
            "name": "Global",
            "name_2": "",
            "primary": true
        }],
        "email": [],
        "email1": "",
        "email2": "",
        "invalid_email": "",
        "email_opt_out": "",
        "email_addresses_non_primary": "",
        "_acl": {
            "fields": {}
        },
        "_module": "Accounts",
        "duplicate_check_rank": 8
    }]
}
```

How to Manipulate Quotes

Overview

A Bash example demonstrating how to manipulate Quotes and related record data such as ProductBundles, Products, and ProductBundleNotes.

Manipulating Quotes

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  curl -X POST -H Cache-Control:no-cache -H "Content-
Type: application/json" -d '{
 "grant_type":"password",
 "client_id":"sugar",
 "client_secret":"",
 "username":"admin",
 "password":"password",
 "platform":"custom_api"
}' https://{site_url}/rest/v11/oauth2/token
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Creating a Quote

Once authenticated, we can submit a Quote record using the <module> endpoint. Since a Quote record is a sum of its parts (i.e. ProductBundles and Products) you can submit the related ProductBundles and Products in the same request payload using the relationship Links and the "create" property.

```
  curl -X POST -H OAuth-Token:<access_token> -H Cache-Control:no-
cache -H "Content-Type: application/json" -d '{
  "name": "Test Quote",
  "quote_stage": "Draft",
  "date_quote_expected_closed": "2017-06-12T11:51:57-0400",
  "product_bundles": {
    "create": [
      {
        "name": "Product Bundle 1",
```

```
    "bundle_stage": "Draft",
    "currency_id": ":-99",
    "base_rate": "1.0",
    "shipping": "0.00",
    "products": {
      "create": [
        {
          "tax_class": "Taxable",
          "quantity": 1000,
          "name": "Test Product 1",
          "description": "My Test Product",
          "mft_part_num": "mft100012021",
          "cost_price": "100.00",
          "list_price": "200.00",
          "discount_price": "175.00",
          "discount_amount": "0.00",
          "discount_select": 0,
          "product_template_id": "",
          "type_id": "",
          "status": "Quotes"
        }
      ]
    },
    "product_bundle_notes": {
      "create": [
        {
          "description": "Free shipping",
          "position": 1
        }
      ]
    }
  },
  {
    "name": "Product Bundle 2",
    "bundle_stage": "Draft",
    "currency_id": ":-99",
    "base_rate": "1.0",
    "shipping": "25.00",
    "products": {
      "create": [
        {
          "quantity": 1000,
          "name": "Test Product 2",
          "description": "My Other Product",
          "mft_part_num": "mft100012234",
          "cost_price": "150.00",
```

```
              "list_price": "300.00",
              "discount_price": "275.00",
              "discount_amount": "0.00",
              "discount_select": 0,
              "product_template_id": "",
              "type_id": "",
              "status": "Quotes"
            },
            {
              "quantity": 500,
              "name": "Test Product 3",
              "description": "My Other Other Product",
              "mft_part_num": "mft100012123",
              "cost_price": "10.00",
              "list_price": "500.00",
              "discount_price": "400.00",
              "discount_amount": "0.00",
              "discount_select": 0,
              "product_template_id": "",
              "type_id": "",
              "status": "Quotes"
            }
          ]
        }
      }
    ]
  }
}' http://<site_url>/rest/v11/Quotes
```

Response

The data received from the server is shown below:

```
  {
  "id": "ee1e1ae8-372a-11e7-8bf4-3c15c2c94fb0",
  "name": "Test Quote",
  "date_entered": "2017-05-12T11:51:57-04:00",
  "date_modified": "2017-05-12T11:51:57-04:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "modified_user_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
```

```
    "fields": {
      "pwd_last_changed": {
        "write": "no",
        "create": "no"
      },
      "last_login": {
        "write": "no",
        "create": "no"
      }
    },
    "delete": "no",
    "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
  }
},
"created_by": "1",
"created_by_name": "Administrator",
"created_by_link": {
  "full_name": "Administrator",
  "id": "1",
  "_acl": {
    "fields": {
      "pwd_last_changed": {
        "write": "no",
        "create": "no"
      },
      "last_login": {
        "write": "no",
        "create": "no"
      }
    },
    "delete": "no",
    "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
  }
},
"description": "",
"deleted": false,
"shipper_id": "",
"shipper_name": "",
"shippers": {
  "name": "",
  "id": "",
  "_acl": {
    "fields": [

    ],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
```

```
  }
},
"taxrate_id": "",
"taxrate_name": "",
"taxrates": {
  "name": "",
  "id": "",
  "_acl": {
    "fields": [

    ],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"taxrate_value": "0.000000",
"show_line_nums": true,
"quote_type": "Quotes",
"date_quote_expected_closed": "2017-06-12",
"original_po_date": "",
"payment_terms": "",
"date_quote_closed": "",
"date_order_shipped": "",
"order_stage": "",
"quote_stage": "Draft",
"purchase_order_num": "",
"quote_num": 2,
"subtotal": "650000.000000",
"subtotal_usdollar": "650000.000000",
"shipping": "0.000000",
"shipping_usdollar": "0.000000",
"discount": "",
"deal_tot": "0.00",
"deal_tot_discount_percentage": "0.00",
"deal_tot_usdollar": "0.00",
"new_sub": "650000.000000",
"new_sub_usdollar": "650000.000000",
"taxable_subtotal": "650000.000000",
"tax": "0.000000",
"tax_usdollar": "0.000000",
"total": "650000.000000",
"total_usdollar": "650000.000000",
"billing_address_street": "",
"billing_address_city": "",
"billing_address_state": "",
"billing_address_postalcode": "",
"billing_address_country": "",
```

```
"shipping_address_street": "",
"shipping_address_city": "",
"shipping_address_state": "",
"shipping_address_postalcode": "",
"shipping_address_country": "",
"system_id": 1,
"shipping_account_name": "",
"shipping_accounts": {
  "name": "",
  "id": "",
  "_acl": {
    "fields": [

    ],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"shipping_account_id": "",
"shipping_contact_name": "",
"shipping_contacts": {
  "full_name": "",
  "id": "",
  "_acl": {
    "fields": [

    ],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  },
  "last_name": ""
},
"shipping_contact_id": "",
"account_name": "",
"billing_accounts": {
  "name": "",
  "id": "",
  "_acl": {
    "fields": [

    ],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"account_id": "",
"billing_account_name": "",
"billing_account_id": "",
"billing_contact_name": "",
```

```
"billing_contacts": {
  "full_name": "",
  "id": "",
  "_acl": {
    "fields": [

    ],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  },
  "last_name": ""
},
"billing_contact_id": "",
"opportunity_name": "",
"opportunities": {
  "name": "",
  "id": "",
  "_acl": {
    "fields": [

    ],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"opportunity_id": "",
"following": "",
"my_favorite": false,
"tag": [

],
"locked_fields": [

],
"assigned_user_id": "",
"assigned_user_name": "",
"assigned_user_link": {
  "full_name": "",
  "id": "",
  "_acl": {
    "fields": [

    ],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"team_count": "",
"team_count_link": {
```

```
    "team_count": "",
    "id": "1",
    "_acl": {
      "fields": [

      ],
      "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
  },
  "team_name": [
    {
      "id": "a0512788-3680-11e7-b42f-3c15c2c94fb0",
      "name": "Administrator",
      "name_2": "",
      "primary": false,
      "selected": false
    },
    {
      "id": "1",
      "name": "Global",
      "name_2": "",
      "primary": true,
      "selected": false
    }
  ],
  "currency_id": "-99",
  "base_rate": "1.000000",
  "currency_name": "",
  "currencies": {
    "name": "",
    "id": "-99",
    "_acl": {
      "fields": [

      ],
      "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    },
    "symbol": ""
  },
  "currency_symbol": "",
  "_acl": {
    "fields": {

    }
  },
  "_module": "Quotes"
```

```
}
```

You might notice that the Response does not contain the related data. To view the related data use the <module>/<record_id>/link/<link_name> - GET Endpoint.

Modifying the Quote's Product Bundles

Once the quote is created, you might need to add or remove Product Bundles from the Quote. This can be done using the /<module>/<record> - PUT endpoint.

```
  //Create a new ProductBundle
curl -X PUT -H OAuth-Token:<access_token> -H Cache-Control:no-
cache -d '{
  "product_bundles": {
    "delete": [
      "<related_bundle_id>"
    ],
    "add": [
      "<new_bundle_id>"
    ]
  }
}' http://<site_url>/rest/v11/Quotes/<record_id>
```

The above script removes a previously related Product Bundle from the Quote and adds a different already created Product Bundle to the Quote

Response Payload

The response payload will match the standard /<module>/<record> - PUT Endpoint Response which is the entire values of the updated record. The previous Response for Creating the quote is the same as shown above.

How to Manipulate Tags (CRUD)

Overview

The following page will provide examples of bash script demonstrating how to use the CRUD (Create, Read, Update, Delete) endpoints for tags in the REST v11 API.

Authentication

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  curl -X POST -H Cache-Control:no-cache -d '{
 "grant_type":"password",
 "client_id":"sugar",
 "client_secret":"",
 "username":"admin",
 "password":"password",
 "platform":"custom_api"
}' http://<site_url>/rest/v11/oauth2/token
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout POST endpoint documentation.

Creating Tags

Once you get oauth_token you would need to use it in the following API Calls to create tags.

```
  curl -X POST -H OAuth-Token:<access_token> -H Cache-Control:no-
cache -d '{
   "name":"Test Name",
}' http://<site_url>/rest/v11/Tags
```

More information on this API endpoint can be found in the /<module> POST documentation.

Response

```
  {
    "id": "12c6ee48-1000-11e8-8838-6a0001bcacb0",
    "name": "Tag Name",
    "date_entered": "2018-02-12T15:21:52+01:00",
    "date_modified": "2018-02-12T15:21:52+01:00",
    "modified_user_id": "1",
    "modified_by_name": "Administrator",
    "modified_user_link": {
        "full_name": "Administrator",
        "id": "1",
        "_acl": {
            "fields": {
```

```
            "pwd_last_changed": { "write": "no", "create": "no" },
            "last_login": { "write": "no", "create": "no" }
        },
        "delete": "no",
        "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
    }
},
"created_by": "1",
"created_by_name": "Administrator",
"created_by_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
        "fields": {
            "pwd_last_changed": { "write": "no", "create": "no" },
            "last_login": { "write": "no", "create": "no" }
        },
        "delete": "no",
        "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
    }
},
"description": "",
"deleted": false,
"name_lower": "tag name",
"following": "",
"my_favorite": false,
"locked_fields": [],
"source_id": "",
"source_type": "",
"source_meta": "",
"assigned_user_id": "1",
"assigned_user_name": "Administrator",
"assigned_user_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
        "fields": {
            "pwd_last_changed": { "write": "no", "create": "no" },
            "last_login": { "write": "no", "create": "no" }
        },
        "delete": "no",
        "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
    }
},
"_acl": { "fields": {} },
"_module": "Tags"
```

```
}
```

Creating Records with Tags

You can also create tags on initial record creation. All you need to do populate the tag field as an array when creating a record. Here is an example that demonstrates using the /<module> POST endpoint.

```
  curl -X POST -H OAuth-Token:<access_token> -H Cache-Control:no-
cache -d '{
    name: "Test Record",
    tag:["First Tag", "Second Tag"]
}' http://<site_url>/rest/v11/Accounts
```

More information on this API endpoint can be found in the /<module> POST documentation.

Response

The data sent to the server is shown below:

```
  {
    "id": "ea507760-0ffd-11e8-bcf5-6a0001bcacb0",
    "name": "Test Record",
    "date_entered": "2018-02-12T15:06:25+01:00",
    "date_modified": "2018-02-12T15:06:25+01:00",
    "modified_user_id": "1",
    "modified_by_name": "Administrator",
    "modified_user_link": {
        "full_name": "Administrator",
        "id": "1",
        "_acl": {
            "fields": {
                "pwd_last_changed": { "write": "no", "create": "no" },
                "last_login": { "write": "no", "create": "no" }
            },
            "delete": "no",
            "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
        }
    },
    "created_by": "1",
    "created_by_name": "Administrator",
```

```
"created_by_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
        "fields": {
            "pwd_last_changed": { "write": "no", "create": "no" },
            "last_login": { "write": "no", "create": "no" }
        },
        "delete": "no",
        "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
    }
},
"description": "",
"deleted": false,
"facebook": "",
"twitter": "",
"googleplus": "",
"account_type": "",
"industry": "",
"annual_revenue": "",
"phone_fax": "",
"billing_address_street": "",
"billing_address_street_2": "",
"billing_address_street_3": "",
"billing_address_street_4": "",
"billing_address_city": "",
"billing_address_state": "",
"billing_address_postalcode": "",
"billing_address_country": "",
"rating": "",
"phone_office": "",
"phone_alternate": "",
"website": "",
"ownership": "",
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "",
"shipping_address_state": "",
"shipping_address_postalcode": "",
"shipping_address_country": "",
"parent_id": "",
"sic_code": "",
```

```
    "duns_num": "",
    "parent_name": "",
    "member_of": {
        "name": "",
        "id": "",
        "_acl": { "fields": [], "_hash": "654d337e0e912edaa00dbb0fb3dc
3c17" }
    },
    "campaign_id": "",
    "campaign_name": "",
    "campaign_accounts": {
        "name": "",
        "id": "",
        "_acl": { "fields": [], "_hash": "654d337e0e912edaa00dbb0fb3dc
3c17" }
    },
    "following": true,
    "my_favorite": false,
    "tag": [
        {
            "id": "ea69c120-0ffd-11e8-b5f6-6a0001bcacb0",
            "name": "First Tag",
            "tags__name_lower": "first tag"
        },
        {
            "id": "eafb28e0-0ffd-11e8-8d80-6a0001bcacb0",
            "name": "Second Tag",
            "tags__name_lower": "second tag"
        }
    ],
    "locked_fields": [],
    "assigned_user_id": "",
    "assigned_user_name": "",
    "assigned_user_link": {
        "full_name": "",
        "id": "",
        "_acl": { "fields": [], "_hash": "654d337e0e912edaa00dbb0fb3dc
3c17" }
    },
    "team_count": "",
    "team_count_link": {
        "team_count": "",
        "id": "1",
        "_acl": { "fields": [], "_hash": "654d337e0e912edaa00dbb0fb3dc
3c17" }
    },
```

```
    "team_name": [
        {
            "id": "1",
            "name": "Global",
            "name_2": "",
            "primary": true,
            "selected": false
        }
    ],
    "email": [],
    "email1": "",
    "email2": "",
    "invalid_email": "",
    "email_opt_out": "",
    "email_addresses_non_primary": "",
    "calculated_c": "",
    "_acl": { "fields": {} },
    "_module": "Accounts"
}
```


Reading/Retrieving Tags

Next, we can retrieve the records using /Tags/:record GET endpoint. Here is the example of retrieving the tag record that we created.

```
  curl -H OAuth-Token: -H Cache-Control:no-cache http://<site_url>/res
t/v11/Tags/12c6ee48-1000-11e8-8838-6a0001bcacb0
```

More information on this API endpoint can be found in the /<module>/:record GET documentation.

Response


```
  {
    "id": "12c6ee48-1000-11e8-8838-6a0001bcacb0",
    "name": "Tag Name",
    "date_entered": "2018-02-12T15:21:52+01:00",
    "date_modified": "2018-02-12T15:21:52+01:00",
    "modified_user_id": "1",
    "modified_by_name": "Administrator",
    "modified_user_link": {
        "full_name": "Administrator",
```

```
    "id": "1",
    "_acl": {
        "fields": {
            "pwd_last_changed": { "write": "no", "create": "no" },
            "last_login": { "write": "no", "create": "no" }
        },
        "delete": "no",
        "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
    }
},
"created_by": "1",
"created_by_name": "Administrator",
"created_by_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
        "fields": {
            "pwd_last_changed": { "write": "no", "create": "no" },
            "last_login": { "write": "no", "create": "no" }
        },
        "delete": "no",
        "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
    }
},
"description": "",
"deleted": false,
"name_lower": "tag name",
"following": "",
"my_favorite": false,
"locked_fields": [],
"source_id": "",
"source_type": "",
"source_meta": "",
"assigned_user_id": "1",
"assigned_user_name": "Administrator",
"assigned_user_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
        "fields": {
            "pwd_last_changed": { "write": "no", "create": "no" },
            "last_login": { "write": "no", "create": "no" }
        },
        "delete": "no",
        "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
    }
```

```
    },
    "_acl": { "fields": {} },
    "_module": "Tags"
}
```

Updating Tags

You can update a tag using the /Tags/:record PUT endpoint. In this example, we are going to update the Tag record that we created in the Creating Tags section and change its name to "Renamed Tag Name".

```
  curl -X PUT -H OAuth-Token:<access_token> -H Cache-Control:no-
cache -d '{
    "name":"Renamed Tag Name"
}' http://<site_url>/rest/v11/Tags/12c6ee48-1000-11e8-8838-6a0001bcacb
0
```

More information on this API endpoint can be found in the /<module>/:record PUT documentation.

Response

```
  {
    "id": "12c6ee48-1000-11e8-8838-6a0001bcacb0",
    "name": "Renamed Tag Name",
    "date_entered": "2018-02-12T15:21:52+01:00",
    "date_modified": "2018-02-12T16:07:18+01:00",
    "modified_user_id": "1",
    "modified_by_name": "Administrator",
    "modified_user_link": {
        "full_name": "Administrator",
        "id": "1",
        "_acl": {
            "fields": {
                "pwd_last_changed": { "write": "no", "create": "no" },
                "last_login": { "write": "no", "create": "no" }
            },
            "delete": "no",
            "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
        }
    },
    "created_by": "1",
```

```
    "created_by_name": "Administrator",
    "created_by_link": {
        "full_name": "Administrator",
        "id": "1",
        "_acl": {
            "fields": {
                "pwd_last_changed": { "write": "no", "create": "no" },
                "last_login": { "write": "no", "create": "no" }
            },
            "delete": "no",
            "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
        }
    },
    "description": "",
    "deleted": false,
    "name_lower": "renamed tag name",
    "following": "",
    "my_favorite": false,
    "locked_fields": [],
    "source_id": "",
    "source_type": "",
    "source_meta": "",
    "assigned_user_id": "1",
    "assigned_user_name": "Administrator",
    "assigned_user_link": {
        "full_name": "Administrator",
        "id": "1",
        "_acl": {
            "fields": {
                "pwd_last_changed": { "write": "no", "create": "no" },
                "last_login": { "write": "no", "create": "no" }
            },
            "delete": "no",
            "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
        }
    },
    "_acl": { "fields": {} },
    "_module": "Tags"
}
```


Deleting Tags

Finally, we will delete the record we created using /Tags/:record DELETE API request.

```
  curl -X DELETE -H OAuth-Token:<access_token> -H Cache-Control:no-cac
he http://<site_url>/rest/v11/Tags/12c6ee48-1000-11e8-8838-6a0001bcacb
0
```

More information on this API endpoint can be found in the /<module>/:record DELETE documentation.

Response

```
  {"id":"12c6ee48-1000-11e8-8838-6a0001bcacb0"}
```

PHP

PHP Examples interacting with the v11 REST API.

How to Export a List of Records

Overview

An PHP example demonstrating how to export a list of records using the v11 /<module>/export/:record_list_id REST GET endpoint.

Exporting Records

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  <?php

$instance_url = "http://{site_url}/rest/v11";
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
```

```
    //It is recommended to create your own in Admin > OAuth Keys
    "client_id" => "sugar",
    "client_secret" => "",
    "username" => $username,
    "password" => $password,
    //platform type - default is base.
    //It is recommend to change the platform to a custom name such as
"custom_api" to avoid authentication conflicts.
    "platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Filtering Records

Next, we will need to identify the records we want to export using the /<module>/filter endpoint.

```
  //Identify records to export - POST /<module>/filter

$filter_url = $instance_url . "/Accounts/filter";
```

```
$filter_arguments = array(
    "filter" => array(
        array(
            '$or' => array(
                array(
                    //name starts with 'a'
                    "name" => array(
                        '$starts'=>"A",
                    )
                ),
                array(
                    //name starts with 'b'
                    "name" => array(
                        '$starts'=>"b",
                    )
                )
            ),
        ),
    ),
    "max_num" => 2,
    "offset" => 0,
    "fields" => "id",
    "order_by" => "date_entered",
    "favorites" => false,
    "my_items" => false,
);

$filter_request = curl_init($filter_url);
curl_setopt($filter_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1
_0);
curl_setopt($filter_request, CURLOPT_HEADER, false);
curl_setopt($filter_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($filter_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($filter_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($filter_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//convert arguments to json
$json_arguments = json_encode($filter_arguments);
curl_setopt($filter_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$filter_response = curl_exec($filter_request);
```

```
//decode json
$filter_response_obj = json_decode($filter_response);

//store ids of records to export
$export_ids = array();
foreach ($filter_response_obj->records as $record)
{
    $export_ids[] = $record->id;
}
```

More information on the filter API can be found in the /<module>/filter documentation.

Request Payload

The data sent to the server is shown below:

```
  {
   "filter":[
      {
         "$or":[
            {
               "name":{
                  "$starts":"A"
               }
            },
            {
               "name":{
                  "$starts":"b"
               }
            }
         ]
      }
   ],
   "max_num":2,
   "offset":0,
   "fields":"id",
   "order_by":"date_entered",
   "favorites":false,
   "my_items":false
}
```

Response

The data received from the server is shown below:

```
  {
   "next_offset":2,
   "records":[
      {
         "id":"f16760a4-3a52-f77d-1522-5703ca28925f",
         "date_modified":"2016-04-05T10:23:28-04:00",
         "_acl":{
            "fields":{

            }
         },
         "_module":"Accounts"
      },
      {
         "id":"ec409fbb-2b22-4f32-7fa1-5703caf78dc3",
         "date_modified":"2016-04-05T10:23:28-04:00",
         "_acl":{
            "fields":{

            }
         },
         "_module":"Accounts"
      }
   ]
}
```

Creating a Record List

Once we have the list of ids, we then need to create a record list in Sugar that consists of those ids.

```
  //Create a record list - POST /<module>/record_list

$record_list_url = $instance_url . "/Accounts/record_list";

$record_list_arguments = array(
    "records" => $export_ids,
);

$record_list_request = curl_init($record_list_url);
```

```
curl_setopt($record_list_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERS
ION_1_0);
curl_setopt($record_list_request, CURLOPT_HEADER, false);
curl_setopt($record_list_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($record_list_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($record_list_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($record_list_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//convert arguments to json
$json_arguments = json_encode($record_list_arguments);
curl_setopt($record_list_request, CURLOPT_POSTFIELDS, $json_arguments)
;

//execute request
$record_list_response = curl_exec($record_list_request);
```

Request Payload

The data sent to the server is shown below:

```
  {
   "records":[
      "f16760a4-3a52-f77d-1522-5703ca28925f",
      "ec409fbb-2b22-4f32-7fa1-5703caf78dc3"
   ]
}
```

Response

The data received from the server is shown below:

```
  {
   "id":"ef963176-4845-bc55-b03e-570430b4173c",
   "assigned_user_id":"1",
   "module_name":"Accounts",
   "records":[
      "f16760a4-3a52-f77d-1522-5703ca28925f",
      "ec409fbb-2b22-4f32-7fa1-5703caf78dc3"
   ],
   "date_modified":"2016-04-05 21:39:19"
```

```
}
```

Exporting Records

Once we have the record list created, we can then export the CSV file.

```
  //Export Records - GET /<module>/export/:record_list_id

$export_url = $instance_url . "/Accounts/export/" . $record_list_respo
nse_obj->id;

$export_request = curl_init($export_url);
curl_setopt($export_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1
_0);
curl_setopt($export_request, CURLOPT_HEADER, true); //needed to return
 file headers
curl_setopt($export_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($export_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($export_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($export_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

$export_response = curl_exec($export_request);

//set headers from response
list($headers, $content) = explode("\r\n\r\n", $export_response ,2);
foreach (explode("\r\n",$headers) as $header) {
    header($header);
}

$content = trim($content);
echo $content;
```

More information on exporting records can be found in the /<module>/export/:record_list_id documentation.

Response

The data received from the server is shown below:

```
HTTP/1.1 200 OK
Date: Tue, 05 Apr 2016 21:50:32
GMT Server: Apache/2.2.29 (Unix)
DAV/2 PHP/5.3.29 mod_ssl/2.2.29
OpenSSL/0.9.8zg X-Powered-By:
PHP/5.3.29 Expires:
Cache-Control: max-age=10,
private Pragma:
Content-Disposition: attachment; filename=Accounts.csv
Content-transfer-encoding: binary
Last-Modified: Tue, 05 Apr 2016 21:50:32
GMT ETag: 9b34f5d74e0298aaf7fd1f27d02e14f2
Content-Length: 1703
Connection: close
Content-Type: application/octet-stream; charset=ISO-8859-1
"Name","ID","Website","Office Phone","Alternate Phone","Fax","Billing
Street","Billing City","Billing State","Billing Postal Code","Billing
Country","Shipping Street","Shipping City","Shipping State","Shipping
Postal Code","Shipping Country","Description","Type","Industry","Annua
l Revenue","Employees","SIC Code","Ticker Symbol","Parent Account ID",
"Ownership","Campaign ID","Rating","Assigned User Name","Assigned User
 ID","Team ID","Teams","Team Set ID","Date Created","Date Modified","M
odified By Name","Modified By ID","Created By","Created By ID","Delete
d","test","Facebook Account","Twitter Account","Google Plus ID","DUNS"
,"Email","Invalid Email","Email Opt Out","Non-primary emails"
"Arts & Crafts Inc","ec409fbb-2b22-4f32-7fa1-5703caf78dc3","www.hrinfo
.tw","(252) 456-8602","","","777 West Filmore Ln","Los Angeles","CA","
77076","USA","777 West Filmore Ln","Los Angeles","CA","77076","USA",""
,"Customer","Hospitality","","","","","","","","","Max Jensen","seed_m
ax_id","West","West, East, Global","dec43cb2-5273-8be2-968a-5703cadee7
5f","2016-04-05 10:23","2016-04-05 10:23","Administrator","1","Adminis
trator","1","0","","","","","","sugar.sugar.section@example.org","0","
0","dev.phone@example.biz,0,0"
"B.H. Edwards Inc","f16760a4-3a52-f77d-1522-5703ca28925f","www.section
dev.edu","(361) 765-0216","","","111 Silicon Valley Road","Persistance
","CA","29709","USA","111 Silicon Valley Road","Persistance","CA","297
09","USA","","Customer","Apparel","","","","","","","","","Sally Brons
en","seed_sally_id","West","West","West","2016-04-05 10:23","2016-04-0
5 10:23","Administrator","1","Administrator","1","0","","","","","","i
nfo.sugar@example.de","0","1","phone.sales.section@example.tv,0,0"
```

Download

You can download the full API example here.

How to Filter a List of Records

Overview

A PHP example demonstrating how to filter records using the v11 /<module>/filter REST POST endpoints.

Filtering Records

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  <?php

$instance_url = "http://{site_url}/rest/v11";
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
    //It is recommended to create your own in Admin > OAuth Keys
    "client_id" => "sugar",
    "client_secret" => "",
    "username" => $username,
    "password" => $password,
    //platform type - default is base.
    //It is recommend to change the platform to a custom name such as
"custom_api" to avoid authentication conflicts.
    "platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
```

```
    "Content-Type: application/json"
));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Filtering Records

Next, we can filter the records we want to return using the /<module>/filter endpoint with a POST request.

```
  //Identify records to export - POST /<module>/filter

$filter_url = $instance_url . "/Accounts/filter";

$filter_arguments = array(
    "filter" => array(
        array(
            '$or' => array(
                array(
                    //name starts with 'a'
                    "name" => array(
                        '$starts'=>"A",
                    )
                ),
                array(
                    //name starts with 'b'
                    "name" => array(
                        '$starts'=>"b",
                    )
                )
            ),
        ),
```

```
    ),
    "max_num" => 2,
    "offset" => 0,
    "fields" => "id",
    "order_by" => "date_entered",
    "favorites" => false,
    "my_items" => false,
);

$filter_request = curl_init($filter_url);
curl_setopt($filter_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1
_0);
curl_setopt($filter_request, CURLOPT_HEADER, false);
curl_setopt($filter_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($filter_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($filter_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($filter_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//convert arguments to json
$json_arguments = json_encode($filter_arguments);
curl_setopt($filter_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$filter_response = curl_exec($filter_request);

//decode json
$filter_response_obj = json_decode($filter_response);
```

More information on the filter API can be found in the /<module>/filter documentation.

Note : The /<module>/filter endpoint can be called using a GET request as well, though long URL requests can have issues so the POST request is recommended.

Request Payload

The data sent to the server is shown below:

```
  {
   "filter":[
      {
```

```
         "$or":[
            {
               "name":{
                  "$starts":"A"
               }
            },
            {
               "name":{
                  "$starts":"b"
               }
            }
         ]
      }
   ],
   "max_num":2,
   "offset":0,
   "fields":"id",
   "order_by":"date_entered",
   "favorites":false,
   "my_items":false
}
```

Response

The data received from the server is shown below:

```
  {
   "next_offset":2,
   "records":[
      {
         "id":"f16760a4-3a52-f77d-1522-5703ca28925f",
         "date_modified":"2016-04-05T10:23:28-04:00",
         "_acl":{
            "fields":{

            }
         },
         "_module":"Accounts"
      },
      {
         "id":"ec409fbb-2b22-4f32-7fa1-5703caf78dc3",
         "date_modified":"2016-04-05T10:23:28-04:00",
         "_acl":{
            "fields":{
```

```
            }
         },
         "_module":"Accounts"
      }
   ]
}
```

Download

You can download the full API example using POST here and using GET here.

How to Favorite a Record

Overview

A PHP example demonstrating how to favorite a record using the v11 <module>/:record/favorite REST PUT API endpoint.

Favoriting a Record

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  <?php

$instance_url = "http://{site_url}/rest/v11";
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
    //It is recommended to create your own in Admin > OAuth Keys
    "client_id" => "sugar",
    "client_secret" => "",
    "username" => $username,
    "password" => $password,
```

```
    //platform type - default is base.
    //It is recommend to change the platform to a custom name such as
"custom_api" to avoid authentication conflicts.
    "platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Favoriting a Record

Next, we can favorite a specific record using the /<module>/:record/favorite endpoint.

```
  //Favorite record - PUT //:record/favorite

$favorite_url = $instance_url . "/Accounts/ae8b1783-404a-
fcb8-1e1e-56f1cc52cd1a/favorite";

$favorite_request = curl_init($favorite_url);

curl_setopt($favorite_request, CURLOPT_CUSTOMREQUEST, "PUT");
```

```
curl_setopt($favorite_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION
_1_0);
curl_setopt($favorite_request, CURLOPT_HEADER, false);
curl_setopt($favorite_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($favorite_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($favorite_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($favorite_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//execute request
$favorite_response = curl_exec($favorite_request);
```

More information on theunfavorite API can be found in the /<module>/:record/favorite PUT documentation.

Request Payload

The data sent to the server is shown below:

```
This endpoint does not accept any request arguments.
```

Response

The data received from the server is shown below:


```
  {
    "id": "ae8b1783-404a-fcb8-1e1e-56f1cc52cd1a",
    "name": "Union Bank",
    "date_entered": "2016-03-22T17:49:50-05:00",
    "date_modified": "2016-03-30T17:44:20-05:00",
    "modified_user_id": "1",
    "modified_by_name": "Administrator",
    "modified_user_link": {
        "full_name": "Administrator",
        "id": "1",
        "_acl": {
            "fields": [],
            "delete": "no",
            "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
        }
```

```
},
"created_by": "1",
"created_by_name": "Administrator",
"created_by_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
        "fields": [],
        "delete": "no",
        "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
    }
},
"description": "",
"deleted": false,
"facebook": "",
"twitter": "",
"googleplus": "",
"account_type": "Customer",
"industry": "Banking",
"annual_revenue": "",
"phone_fax": "",
"billing_address_street": "67321 West Siam St.",
"billing_address_street_2": "",
"billing_address_street_3": "",
"billing_address_street_4": "",
"billing_address_city": "Ohio",
"billing_address_state": "CA",
"billing_address_postalcode": "25159",
"billing_address_country": "USA",
"rating": "",
"phone_office": "(065) 489-6104",
"phone_alternate": "",
"website": "www.qahr.edu",
"ownership": "",
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "67321 West Siam St.",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "Ohio",
"shipping_address_state": "CA",
"shipping_address_postalcode": "25159",
"shipping_address_country": "USA",
"parent_id": "",
"sic_code": "",
```

```
"duns_num": "",
"parent_name": "",
"member_of": {
    "name": "",
    "id": "",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"campaign_id": "",
"campaign_name": "",
"campaign_accounts": {
    "name": "",
    "id": "",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"following": true,
"my_favorite": true,
"tag": [],
"assigned_user_id": "seed_sarah_id",
"assigned_user_name": "Sarah Smith",
"assigned_user_link": {
    "full_name": "Sarah Smith",
    "id": "seed_sarah_id",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"team_count": "",
"team_count_link": {
    "team_count": "",
    "id": "West",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"team_name": [{
    "id": "East",
    "name": "East",
    "name_2": "",
```

```
        "primary": false
    }, {
        "id": 1,
        "name": "Global",
        "name_2": "",
        "primary": false
    }, {
        "id": "West",
        "name": "West",
        "name_2": "",
        "primary": true
    }],
    "email": [{
        "email_address": "hr.support.kid@example.info",
        "invalid_email": false,
        "opt_out": false,
        "primary_address": true,
        "reply_to_address": false
    }, {
        "email_address": "info.support.the@example.com",
        "invalid_email": false,
        "opt_out": false,
        "primary_address": false,
        "reply_to_address": false
    }],
    "email1": "hr.support.kid@example.info",
    "email2": "info.support.the@example.com",
    "invalid_email": false,
    "email_opt_out": false,
    "email_addresses_non_primary": "",
    "_acl": {
        "fields": {}
    },
    "_module": "Accounts"
}
```


Download

You can download the full API example here.

How to Manipulate File Attachments

Overview

A PHP example demonstrating how to attach a file to a record using the v11 <module>/:record/file/:field REST POST API endpoint, then retrieve it with the GET endpoint.

Manipulating File Attachments

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  <?php

$instance_url = "http://{site_url}/rest/v11";
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
    //It is recommended to create your own in Admin > OAuth Keys
    "client_id" => "sugar",
    "client_secret" => "",
    "username" => $username,
    "password" => $password,
    //platform type - default is base.
    //It is recommend to change the platform to a custom name such as
"custom_api" to avoid authentication conflicts.
    "platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));
```

```
//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Submitting a File Attachment

Next, we can create a Note record using the /<module endpoint, and then submit a File to the Note record using the /<module>/:record/file/:field endpoint.

```
  //Create Note - POST /
$url = $instance_url . "/Notes";
//Set up the Record details
$record = array(
    'name' => 'Test Note'
);

$curl_request = curl_init($url);
curl_setopt($curl_request, CURLOPT_POST, 1);
curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($curl_request, CURLOPT_HEADER, false);
curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($curl_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//convert arguments to json
$json_arguments = json_encode($record);
curl_setopt($curl_request, CURLOPT_POSTFIELDS, $json_arguments);
//execute request
```

```
$curl_response = curl_exec($curl_request);
//decode json
$noteRecord = json_decode($curl_response);

//display the created record
echo "Created Record: ". $noteRecord->id;

curl_close($curl_request);

//Add An Attachment to the Note
$url = $instance_url . "/Notes/{$noteRecord->id}/file/filename";

$file_arguments = array(
    "format" => "sugar-html-json",
    "delete_if_fails" => true,
    "oauth_token" => $oauth_token,
);

if ((version_compare(PHP_VERSION, '5.5') >= 0)) {
    $file_arguments['filename'] = new CURLFile($path);
} else {
    $file_arguments['filename'] = '@'.$path;
}

$curl_request = curl_init($url);
curl_setopt($curl_request, CURLOPT_POST, 1);
curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($curl_request, CURLOPT_HEADER, false);
curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);
//Do NOT set Content Type Header to JSON
curl_setopt($curl_request, CURLOPT_HTTPHEADER, array(
    "oauth-token: {$oauth_token}"
));

curl_setopt($curl_request, CURLOPT_POSTFIELDS, $file_arguments);
//execute request
$curl_response = curl_exec($curl_request);
//decode json
$noteRecord = json_decode($curl_response);
//print Note with attachment details
print_r($noteRecord);

curl_close($curl_request);
```

Note: As of PHP 5.6, the '@' upload modifier is disabled for security reasons by the CURLOPT_SAFE_UPLOAD option. We recommending using CURLFILE if using PHP >= 5.5, If you would prefer to use the upload modifier, you can set the CURLOPT_SAFE_UPLOAD option to false.

```
  curl_setopt($ch, CURLOPT_SAFE_UPLOAD, false);
```

Request

```
  //Raw Post - not json encoded
Array
(
    [format] => sugar-html-json
    [delete_if_fails] => 1
    [oauth_token] => 09eac950-c99e-4786-8b10-a3670f38fb3f
    [filename] => CURLFile Object
        (
            [name] => /Library/WebServer/Documents/file_attachment_man
ipulation/testfile.txt
            [mime] =>
            [postname] =>
        )
)
```

Response

```
  {
  "filename":{
     "content-type":"text\/plain",
     "content-length":13,
     "name":"upload.txt",
     "uri":"http:\/\/<site url>\/rest\/v11\/Notes\/7b49aebd-8734-9773-
8ef1-53553fa369c7\/file\/filename"
  },
  "record":{
     "my_favorite":false,
     "following":true,
     "id":"7b49aebd-8734-9773-8ef1-53553fa369c7",
     "name":"My Note",
     "date_modified":"2014-04-21T11:53:53-04:00",
     "modified_user_id":"1",
```

```
      "modified_by_name":"admin",
      "created_by":"1",
      "created_by_name":"Administrator",
      "doc_owner":"",
      "user_favorites":[

      ],
      "description":"",
      "deleted":false,
      "assigned_user_id":"",
      "assigned_user_name":"",
      "team_count":"",
      "team_name":[
         {
            "id":1,
            "name":"Global",
            "name_2":"",
            "primary":true
         }
      ],
      "file_mime_type":"text\/plain",
      "file_url":"",
      "filename":"upload.txt",
      "parent_type":"",
      "parent_id":"",
      "contact_id":"",
      "portal_flag":false,
      "embed_flag":false,
      "parent_name":"",
      "contact_name":"",
      "contact_phone":"",
      "contact_email":"",
      "account_id":"",
      "opportunity_id":"",
      "acase_id":"",
      "lead_id":"",
      "product_id":"",
      "quote_id":"",
      "_acl":{
         "fields":{

         }
      },
      "_module":"Notes"
   }
}
```

Getting a File Attachment

Next, we can retrieve the file attachment stored in Sugar by utilizing the /<module>/:record/file/:field GET endpoint. The following code example, works when being accessed via a web browser, as it receives the response from Sugar, and sets the Headers received from Sugar on itself, so that the browser knows to download a file.

```
  //Get An Attachment on a Note
$url = $instance_url . "/Notes/{$noteRecord->id}/file/filename";

$curl_request = curl_init($url);
curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
//Return Headers
curl_setopt($curl_request, CURLOPT_HEADER, true);
curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);
//DO NOT set Content Type Header to JSON
curl_setopt($curl_request, CURLOPT_HTTPHEADER, array(
    "oauth-token: {$oauth_token}"
));

//execute request
$curl_response = curl_exec($curl_request);

//Get Return Headers
$header_size = curl_getinfo($curl_request,CURLINFO_HEADER_SIZE);
$headers = substr($curl_response, 0, $header_size);

//Outputting the file contents
echo 'File saved to download.txt';
$file = substr($curl_response, $header_size);
file_put_contents('download.txt', $file);

curl_close($curl_request);
```

Request

```
http://{site_url}/rest/v11/Notes/bd490e66-2ea7-9349-19cf-535569400cca/
file/filename
```

Note: GET request arguments are passed in the form of a query string.

Response

```
  HTTP/1.1 200 OK

Date: Wed, 12 Mar 2014 15:15:03 GMT

Server: Apache/2.2.22 (Unix) PHP/5.3.14 mod_ssl/2.2.22 OpenSSL/0.9.8o

X-Powered-By: PHP/5.3.14 ZendServer/5.0

Set-Cookie: ZDEDebuggerPresent=php,phtml,php3; path=/

Expires:

Cache-Control: max-age=0, private

Pragma:

Content-Disposition: attachment; filename="upload.txt"

X-Content-Type-Options: nosniff

ETag: d41d8cd98f00b204e9800998ecf8427e

Content-Length: 16

Connection: close

Content-Type: application/octet-stream


This is the file contents.
```

Download

You can download the full API example here.

How to Fetch Related Records

Overview

A PHP example demonstrating how to fetch related records using the v11 /<module>/:record/link/:link REST GET endpoints.

Fetching Related Records

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  <?php

$instance_url = "http://{site_url}/rest/v11";
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
    //It is recommended to create your own in Admin > OAuth Keys
    "client_id" => "sugar",
    "client_secret" => "",
    "username" => $username,
    "password" => $password,
    //platform type - default is base.
    //It is recommend to change the platform to a custom name such as
"custom_api" to avoid authentication conflicts.
    "platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
```

```
));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Fetching Related Records

Next, we can fetch the related records we want to return using the /<module>/:record/link/:link endpoint with a GET request where

Element	Meaning
<module>	The parent module name
:record	The parent records ID
link	the actual word "link"
:link	The name of the relationship to fetch

In this example, we will fetch the related Contacts for an Account

```
  //Identify records to fetch - POST /<module>/:record/link/:link

$fetch_url = $instance_url . "/Accounts/d8f05e67-dee3-553d-0040-5342e8
8f2fd1/link/contacts";

$fetch_request = curl_init($fetch_url);
curl_setopt($fetch_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_
0);
curl_setopt($fetch_request, CURLOPT_HEADER, false);
curl_setopt($fetch_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($fetch_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($fetch_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($fetch_request, CURLOPT_HTTPHEADER, array(
```

```
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//execute request
$fetch_response = curl_exec($fetch_request);

//decode json
$fetch_response_obj = json_decode($fetch_response);
```

Request

```
  http://{site_url}/rest/v11/Accounts/d8f05e67-dee3-553d-0040-5342e88f
2fd1/link/contacts
```

Response

The data received from the server is shown below:


```
{
   "next_offset":-1,
   "records":[
      {
         "my_favorite":false,
         "following":false,
         "id":"819f4149-b007-a6da-a5fa-56fedbf2de77",
         "name":"Florine Marcus",
         "date_entered":"2016-04-01T15:34:00-05:00",
         "date_modified":"2016-04-01T15:34:00-05:00",
         "modified_user_id":"1",
         "modified_by_name":"Administrator",
         "created_by":"1",
         "created_by_name":"Administrator",
         "doc_owner":"",
         "user_favorites":"",
         "description":"",
         "deleted":false,
         "assigned_user_id":"seed_will_id",
         "assigned_user_name":"Will Westin",
         "team_count":"",
         "team_name":[
```

```
      {
         "id":"East",
         "name":"East",
         "name_2":"",
         "primary":true
      },
      {
         "id":"West",
         "name":"West",
         "name_2":"",
         "primary":false
      }
   ],
   "email":[
      {
         "email_address":"support27@example.tv",
         "primary_address":true,
         "reply_to_address":false,
         "invalid_email":false,
         "opt_out":false
      },
      {
         "email_address":"support.support.kid@example.net",
         "primary_address":false,
         "reply_to_address":false,
         "invalid_email":false,
         "opt_out":true
      }
   ],
   "email1":"support27@example.tv",
   "email2":"",
   "invalid_email":false,
   "email_opt_out":false,
   "email_addresses_non_primary":"",
   "salutation":"",
   "first_name":"Florine",
   "last_name":"Marcus",
   "full_name":"Florine Marcus",
   "title":"President",
   "facebook":"",
   "twitter":"",
   "googleplus":"",
   "department":"",
   "do_not_call":false,
   "phone_home":"(746) 162-2314",
   "phone_mobile":"(941) 088-2874",
```

```
"phone_work":"(827) 541-9614",
"phone_other":"",
"phone_fax":"",
"primary_address_street":"1715 Scott Dr",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Alabama",
"primary_address_state":"NY",
"primary_address_postalcode":"70187",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
"assistant":"",
"assistant_phone":"",
"picture":"",
"email_and_name1":"",
"lead_source":"Employee",
"account_name":"MTM Investment Bank F S B",
"account_id":"e8c641ca-1b8c-74c1-d08d-56fedbdd3187",
"dnb_principal_id":"",
"opportunity_role_fields":"",
"opportunity_role_id":"",
"opportunity_role":"",
"reports_to_id":"",
"report_to_name":"",
"birthdate":"",
"portal_name":"FlorineMarcus119",
"portal_active":true,
"portal_password":true,
"portal_password1":null,
"portal_app":"",
"preferred_language":"en_us",
"campaign_id":"",
"campaign_name":"",
"c_accept_status_fields":"",
"m_accept_status_fields":"",
"accept_status_id":"",
"accept_status_name":"",
"accept_status_calls":"",
"accept_status_meetings":"",
"sync_contact":false,
```

```
      "mkto_sync":false,
      "mkto_id":null,
      "mkto_lead_score":null,
      "_acl":{
         "fields":{

         }
      },
      "_module":"Contacts"
   },
   {
      "my_favorite":false,
      "following":false,
      "id":"527cc1a9-7984-91fe-4148-56fedbc356aa",
      "name":"Shaneka Aceto",
      "date_entered":"2016-04-01T15:34:00-05:00",
      "date_modified":"2016-04-01T15:34:00-05:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "doc_owner":"",
      "user_favorites":"",
      "description":"",
      "deleted":false,
      "assigned_user_id":"seed_will_id",
      "assigned_user_name":"Will Westin",
      "team_count":"",
      "team_name":[
         {
            "id":"East",
            "name":"East",
            "name_2":"",
            "primary":true
         },
         {
            "id":"West",
            "name":"West",
            "name_2":"",
            "primary":false
         }
      ],
      "email":[
         {
            "email_address":"support17@example.cn",
            "primary_address":true,
```

```
            "reply_to_address":false,
            "invalid_email":false,
            "opt_out":false
        },
        {
            "email_address":"section.sugar.the@example.tv",
            "primary_address":false,
            "reply_to_address":false,
            "invalid_email":false,
            "opt_out":true
        }
    ],
    "email1":"support17@example.cn",
    "email2":"",
    "invalid_email":false,
    "email_opt_out":false,
    "email_addresses_non_primary":"",
    "salutation":"",
    "first_name":"Shaneka",
    "last_name":"Aceto",
    "full_name":"Shaneka Aceto",
    "title":"IT Developer",
    "facebook":"",
    "twitter":"",
    "googleplus":"",
    "department":"",
    "do_not_call":false,
    "phone_home":"(502) 528-5151",
    "phone_mobile":"(816) 719-3739",
    "phone_work":"(994) 769-5855",
    "phone_other":"",
    "phone_fax":"",
    "primary_address_street":"123 Anywhere Street",
    "primary_address_street_2":"",
    "primary_address_street_3":"",
    "primary_address_city":"Denver",
    "primary_address_state":"NY",
    "primary_address_postalcode":"15128",
    "primary_address_country":"USA",
    "alt_address_street":"",
    "alt_address_street_2":"",
    "alt_address_street_3":"",
    "alt_address_city":"",
    "alt_address_state":"",
    "alt_address_postalcode":"",
    "alt_address_country":"",
```

```
      "assistant":"",
      "assistant_phone":"",
      "picture":"",
      "email_and_name1":"",
      "lead_source":"Email",
      "account_name":"MTM Investment Bank F S B",
      "account_id":"e8c641ca-1b8c-74c1-d08d-56fedbdd3187",
      "dnb_principal_id":"",
      "opportunity_role_fields":"",
      "opportunity_role_id":"",
      "opportunity_role":"",
      "reports_to_id":"",
      "report_to_name":"",
      "birthdate":"",
      "portal_name":"ShanekaAceto151",
      "portal_active":true,
      "portal_password":true,
      "portal_password1":null,
      "portal_app":"",
      "preferred_language":"en_us",
      "campaign_id":"",
      "campaign_name":"",
      "c_accept_status_fields":"",
      "m_accept_status_fields":"",
      "accept_status_id":"",
      "accept_status_name":"",
      "accept_status_calls":"",
      "accept_status_meetings":"",
      "sync_contact":false,
      "mkto_sync":false,
      "mkto_id":null,
      "mkto_lead_score":null,
      "_acl":{
         "fields":{

         }
      },
      "_module":"Contacts"
   },
   {
      "my_favorite":false,
      "following":false,
      "id":"42d703ed-f834-f87c-967d-56fedb044051",
      "name":"Johnnie Pina",
      "date_entered":"2016-04-01T15:34:00-05:00",
      "date_modified":"2016-04-01T15:34:00-05:00",
```

```
"modified_user_id":"1",
"modified_by_name":"Administrator",
"created_by":"1",
"created_by_name":"Administrator",
"doc_owner":"",
"user_favorites":"",
"description":"",
"deleted":false,
"assigned_user_id":"seed_will_id",
"assigned_user_name":"Will Westin",
"team_count":"",
"team_name":[
   {
      "id":"East",
      "name":"East",
      "name_2":"",
      "primary":true
   },
   {
      "id":"West",
      "name":"West",
      "name_2":"",
      "primary":false
   }
],
"email":[
   {
      "email_address":"sugar.support.hr@example.co.uk",
      "primary_address":true,
      "reply_to_address":false,
      "invalid_email":false,
      "opt_out":false
   },
   {
      "email_address":"support.im@example.tw",
      "primary_address":false,
      "reply_to_address":false,
      "invalid_email":false,
      "opt_out":true
   }
],
"email1":"sugar.support.hr@example.co.uk",
"email2":"",
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"",
```

```
"salutation":"",
"first_name":"Johnnie",
"last_name":"Pina",
"full_name":"Johnnie Pina",
"title":"VP Operations",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(159) 335-1423",
"phone_mobile":"(580) 140-4050",
"phone_work":"(418) 792-9611",
"phone_other":"",
"phone_fax":"",
"primary_address_street":"345 Sugar Blvd.",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Denver",
"primary_address_state":"NY",
"primary_address_postalcode":"70648",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
"assistant":"",
"assistant_phone":"",
"picture":"",
"email_and_name1":"",
"lead_source":"Direct Mail",
"account_name":"MTM Investment Bank F S B",
"account_id":"e8c641ca-1b8c-74c1-d08d-56fedbdd3187",
"dnb_principal_id":"",
"opportunity_role_fields":"",
"opportunity_role_id":"",
"opportunity_role":"",
"reports_to_id":"",
"report_to_name":"",
"birthdate":"",
"portal_name":"JohnniePina194",
"portal_active":true,
"portal_password":true,
```

```
         "portal_password1":null,
         "portal_app":"",
         "preferred_language":"en_us",
         "campaign_id":"",
         "campaign_name":"",
         "c_accept_status_fields":"",
         "m_accept_status_fields":"",
         "accept_status_id":"",
         "accept_status_name":"",
         "accept_status_calls":"",
         "accept_status_meetings":"",
         "sync_contact":false,
         "mkto_sync":false,
         "mkto_id":null,
         "mkto_lead_score":null,
         "_acl":{
            "fields":{

            }
         },
         "_module":"Contacts"
      }
   ]
}
```


Download

You can download the full API example here

How to Manipulate Records (CRUD)

Overview

A PHP example demonstrating how to use the CRUD (Create, Read, Update, Delete) endpoints in the REST v11 API.

CRUD Operations

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  <?php

$instance_url = "http://{site_url}/rest/v11";
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
    //It is recommended to create your own in Admin > OAuth Keys
    "client_id" => "sugar",
    "client_secret" => "",
    "username" => $username,
    "password" => $password,
    //platform type - default is base.
    //It is recommend to change the platform to a custom name such as
"custom_api" to avoid authentication conflicts.
    "platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Creating a Record

Next, we need to submit the record to the Sugar instance using the /<module> endpoint. In this example we are going to create an Account record with a Name of 'Test Record' and an email of 'test@sugar.com'.

```
  //Create Records - POST /<module>
$url = $instance_url . "/Accounts";
//Set up the Record details
$record = array(
    'name' => 'Test Record',
    'email' => array(
        array(
            'email_address' => 'test@sugar.com',
            'primary_address' => true
        )
    ),
);

$curl_request = curl_init($url);
curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($curl_request, CURLOPT_HEADER, false);
curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($curl_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//convert arguments to json
$json_arguments = json_encode($record);
curl_setopt($curl_request, CURLOPT_POSTFIELDS, $json_arguments);
//execute request
$curl_response = curl_exec($curl_request);
//decode json
$createdRecord = json_decode($curl_response);

//display the created record
print_r($createdRecord);
curl_close($curl_request);
```

More information on this API endpoint can be found in the /<module> - POST documentation.

Request Payload

The data sent to the server is shown below:

```
  {
    "name": "Test Record",
    "email": [{ "email_address": "test@sugar.com", "primary_address":
true }]
}
```

Response

The data received from the server is shown below:

```
  {
    "id": "ae78a068-7a0c-11e8-8b9e-6a0001bcacb0",
    "name": "Test Record",
    "date_entered": "2018-06-27T15:19:11+02:00",
    "date_modified": "2018-06-27T15:19:11+02:00",
    "modified_user_id": "1",
    "modified_by_name": "Administrator",
    "modified_user_link": {
        "full_name": "Administrator",
        "id": "1",
        "_acl": {
            "fields": {
                "pwd_last_changed": { "write": "no", "create": "no" },
                "last_login": { "write": "no", "create": "no" }
            },
            "delete": "no",
            "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
        }
    },
    "created_by": "1",
    "created_by_name": "Administrator",
    "created_by_link": {
        "full_name": "Administrator",
        "id": "1",
        "_acl": {
            "fields": {
                "pwd_last_changed": { "write": "no", "create": "no" },
```

```
            "last_login": { "write": "no", "create": "no" }
        },
        "delete": "no",
        "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
    }
},
"description": "",
"deleted": false,
"facebook": "",
"twitter": "",
"googleplus": "",
"account_type": "",
"industry": "",
"annual_revenue": "",
"phone_fax": "",
"billing_address_street": "",
"billing_address_street_2": "",
"billing_address_street_3": "",
"billing_address_street_4": "",
"billing_address_city": "",
"billing_address_state": "",
"billing_address_postalcode": "",
"billing_address_country": "",
"rating": "",
"phone_office": "",
"phone_alternate": "",
"website": "",
"ownership": "",
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "",
"shipping_address_state": "",
"shipping_address_postalcode": "",
"shipping_address_country": "",
"parent_id": "",
"sic_code": "",
"duns_num": "",
"parent_name": "",
"member_of": {
    "name": "",
    "id": "",
    "_acl": { "fields": [], "_hash": "654d337e0e912edaa00dbb0fb3dc
```

```
3c17" }
    },
    "campaign_id": "",
    "campaign_name": "",
    "campaign_accounts": {
        "name": "",
        "id": "",
        "_acl": { "fields": [], "_hash": "654d337e0e912edaa00dbb0fb3dc
3c17" }
    },
    "following": true,
    "my_favorite": false,
    "tag": [],
    "locked_fields": [],
    "assigned_user_id": "",
    "assigned_user_name": "",
    "assigned_user_link": {
        "full_name": "",
        "id": "",
        "_acl": { "fields": [], "_hash": "654d337e0e912edaa00dbb0fb3dc
3c17" }
    },
    "team_count": "",
    "team_count_link": {
        "team_count": "",
        "id": "1",
        "_acl": { "fields": [], "_hash": "654d337e0e912edaa00dbb0fb3dc
3c17" }
    },
    "team_name": [
        {
            "id": "1",
            "name": "Global",
            "name_2": "",
            "primary": true,
            "selected": false
        }
    ],
    "email": [
        {
            "email_address": "test@sugar.com",
            "invalid_email": false,
            "opt_out": false,
            "email_address_id": "85125194-7a0a-11e8-9c17-6a0001bcacb0"
,
            "primary_address": true,
```

```
            "reply_to_address": false
        }
    ],
    "email1": "test@sugar.com",
    "email2": "",
    "invalid_email": false,
    "email_opt_out": false,
    "email_addresses_non_primary": "",
    "test_c": "",
    "dri_workflow_template_id": "",
    "dri_workflow_template_name": "",
    "dri_workflow_template_link": {
        "name": "",
        "id": "",
        "_acl": { "fields": [], "_hash": "654d337e0e912edaa00dbb0fb3dc
3c17" }
    },
    "_acl": { "fields": {} },
    "_module": "Accounts"
}
```

Getting a Record

Next, we can get the created record from the Sugar instance using the /<module>/:record endpoint. In this example, we are going to get an Account record by it's ID, but only request the Name, Email, and Industry fields.

```
  $id = $createdRecord->id;

//Get Record - GET //:record
$url = $instance_url . "/Accounts/$id";

//Setup request to only return some fields on module
$data = array(
    'fields' => 'name,email1,industry'
);

//Add data to the URL
$url = $url."?".http_build_query($data);

$curl_request = curl_init($url);
curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($curl_request, CURLOPT_HEADER, false);
```

```
curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($curl_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//execute request
$curl_response = curl_exec($curl_request);
//decode json
$record = json_decode($curl_response);

//display the created record
print_r($record);
curl_close($curl_request);
```

Updating a Record

Next, we can update the record in the Sugar instance using the /<module>/:record endpoint, and the PUT Http method. In this example, we are going to update the Account record and change it's name to "Updated Test Record".

```
  $id = $record->id;
//Update Record - PUT /<module>/:record
$url = $instance_url . "/Accounts/$id";
//Set up the Record details
$record->name = 'Updated Test Record';

$curl_request = curl_init($url);
curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "PUT");
curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($curl_request, CURLOPT_HEADER, false);
curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($curl_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//convert arguments to json
```

```
$json_arguments = json_encode($record);
curl_setopt($curl_request, CURLOPT_POSTFIELDS, $json_arguments);
//execute request
$curl_response = curl_exec($curl_request);
//decode json
$updatedRecord = json_decode($curl_response);

//display the created record
echo "Updated Record Name:".$updatedRecord->name;
curl_close($curl_request);
```

More information on this API endpoint can be found in the /<module>/:record - PUT documentation.

Request Payload

The URL sent to the server is shown below:

```
{"name":"Updated Test Record"}
```

Response

The data received from the server is shown below:

```
{
  "id": "ae78a068-7a0c-11e8-8b9e-6a0001bcacb0",
  "name": "Updated Test Record",
  "date_entered": "2018-06-27T15:19:11+02:00",
  "date_modified": "2018-06-27T15:23:19+02:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "modified_user_link": {
      "full_name": "Administrator",
      "id": "1",
      "_acl": {
          "fields": {
              "pwd_last_changed": { "write": "no", "create": "no" },
              "last_login": { "write": "no", "create": "no" }
          },
          "delete": "no",
          "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
      }
```

```
},
"created_by": "1",
"created_by_name": "Administrator",
"created_by_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
        "fields": {
            "pwd_last_changed": { "write": "no", "create": "no" },
            "last_login": { "write": "no", "create": "no" }
        },
        "delete": "no",
        "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
    }
},
"description": "",
"deleted": false,
"facebook": "",
"twitter": "",
"googleplus": "",
"account_type": "",
"industry": "",
"annual_revenue": "",
"phone_fax": "",
"billing_address_street": "",
"billing_address_street_2": "",
"billing_address_street_3": "",
"billing_address_street_4": "",
"billing_address_city": "",
"billing_address_state": "",
"billing_address_postalcode": "",
"billing_address_country": "",
"rating": "",
"phone_office": "",
"phone_alternate": "",
"website": "",
"ownership": "",
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "",
"shipping_address_state": "",
"shipping_address_postalcode": "",
```

```
    "shipping_address_country": "",
    "parent_id": "",
    "sic_code": "",
    "duns_num": "",
    "parent_name": "",
    "member_of": {
        "name": "",
        "id": "",
        "_acl": { "fields": [], "_hash": "654d337e0e912edaa00dbb0fb3dc
3c17" }
    },
    "campaign_id": "",
    "campaign_name": "",
    "campaign_accounts": {
        "name": "",
        "id": "",
        "_acl": { "fields": [], "_hash": "654d337e0e912edaa00dbb0fb3dc
3c17" }
    },
    "following": true,
    "my_favorite": false,
    "tag": [],
    "locked_fields": [],
    "assigned_user_id": "",
    "assigned_user_name": "",
    "assigned_user_link": {
        "full_name": "",
        "id": "",
        "_acl": { "fields": [], "_hash": "654d337e0e912edaa00dbb0fb3dc
3c17" }
    },
    "team_count": "",
    "team_count_link": {
        "team_count": "",
        "id": "1",
        "_acl": { "fields": [], "_hash": "654d337e0e912edaa00dbb0fb3dc
3c17" }
    },
    "team_name": [
        {
            "id": "1",
            "name": "Global",
            "name_2": "",
            "primary": true,
            "selected": false
        }
```

```
    ],
    "email": [
        {
            "email_address": "test@sugar.com",
            "invalid_email": false,
            "opt_out": false,
            "email_address_id": "85125194-7a0a-11e8-9c17-6a0001bcacb0"
,
            "primary_address": true,
            "reply_to_address": false
        }
    ],
    "email1": "test@sugar.com",
    "email2": "",
    "invalid_email": false,
    "email_opt_out": false,
    "email_addresses_non_primary": "",
    "test_c": "",
    "dri_workflow_template_id": "",
    "dri_workflow_template_name": "",
    "dri_workflow_template_link": {
        "name": "",
        "id": "",
        "_acl": { "fields": [], "_hash": "654d337e0e912edaa00dbb0fb3dc
3c17" }
    },
    "_acl": { "fields": {} },
    "_module": "Accounts"
}
```

Deleting a Record

Next, we can delete the record from the Sugar instance using the /<module>/:record endpoint, by using the DELETE Http Method.

```
  $id = $updatedRecord->id;
//Delete Record - DELETE /<module>/:record
$url = $instance_url . "/Accounts/$id";

$curl_request = curl_init($url);
curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "DELETE");
curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($curl_request, CURLOPT_HEADER, false);
```

```
curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($curl_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//execute request
$curl_response = curl_exec($curl_request);
//decode json
$deletedRecord = json_decode($curl_response);

//display the created record
echo "Deleted Record:".$deletedRecord->id;
curl_close($curl_request);
```

More information on this API endpoint can be found in the /<module>/:record - DELETE documentation.

Request Payload

The URL sent to the server is shown below:

```
No payload is sent for this request.
```

Response

The data received from the server is shown below:

```
{"id":"ae78a068-7a0c-11e8-8b9e-6a0001bcacb0"}
```

Download

You can download the full API example here.

How to Follow a Record

Overview

A PHP example demonstrating how to follow a record using the v11 /<module>/:record/subscribe REST POST endpoint.

Following a Record

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  <?php

$instance_url = "http://{site_url}/rest/v11";
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
    //It is recommended to create your own in Admin > OAuth Keys
    "client_id" => "sugar",
    "client_secret" => "",
    "username" => $username,
    "password" => $password,
    //platform type - default is base.
    //It is recommend to change the platform to a custom name such as
"custom_api" to avoid authentication conflicts.
    "platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));
```

```
//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Following a Record

Next, we can follow a specific record using the /<module>/:record/subscribe endpoint.

```
  //Subscribe to record - POST //:record/subscribe

$subscribe_url = $instance_url . "/Accounts/ae8b1783-404a-
fcb8-1e1e-56f1cc52cd1a/subscribe";

$subscribe_request = curl_init($subscribe_url);

curl_setopt($subscribe_request, CURLOPT_POST, 1);
curl_setopt($subscribe_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSIO
N_1_0);
curl_setopt($subscribe_request, CURLOPT_HEADER, false);
curl_setopt($subscribe_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($subscribe_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($subscribe_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($subscribe_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//execute request
$subscribe_response = curl_exec($subscribe_request);
```

More information on the subscribe API can be found in the

/<module>/:record/subscribe POST documentation.

Request Payload

The data sent to the server is shown below:

```
This endpoint does not accept any request arguments.
```

Response

The data received from the server is shown below:

```
  "58f96315-9e75-6562-42e9-5705917d2cdc"
```

Download

You can download the full API example here.

How to Unfavorite a Record

Overview

A PHP example demonstrating how to unfavorite a record using the v11 /<module>/:record/unfavorite REST PUT endpoint.

Unfavoriting a Record

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  <?php

$instance_url = "http://{site_url}/rest/v11";
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";
```

```
$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
    //It is recommended to create your own in Admin > OAuth Keys
    "client_id" => "sugar",
    "client_secret" => "",
    "username" => $username,
    "password" => $password,
    //platform type - default is base.
    //It is recommend to change the platform to a custom name such as
"custom_api" to avoid authentication conflicts.
    "platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Unfavoriting a Record

Next, we can unfavorite a specific record using the /<module>/:record/unfavorite endpoint.

```
  //Unfavorite record - PUT //:record/unfavorite

$unfavorite_url = $instance_url . "/Accounts/ae8b1783-404a-
fcb8-1e1e-56f1cc52cd1a/unfavorite";

$unfavorite_request = curl_init($unfavorite_url);

curl_setopt($unfavorite_request, CURLOPT_CUSTOMREQUEST, "PUT");
curl_setopt($unfavorite_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSI
ON_1_0);
curl_setopt($unfavorite_request, CURLOPT_HEADER, false);
curl_setopt($unfavorite_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($unfavorite_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($unfavorite_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($unfavorite_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//execute request
$unfavorite_response = curl_exec($unfavorite_request);
```

More information on the unfavorite API can be found in the /<module>/:record/unfavorite PUT documentation.

Request Payload

The data sent to the server is shown below:

```
This endpoint does not accept any request arguments.
```

Response

The data received from the server is shown below:

```
  {
    "id": "ae8b1783-404a-fcb8-1e1e-56f1cc52cd1a",
    "name": "Union Bank",
    "date_entered": "2016-03-22T17:49:50-05:00",
    "date_modified": "2016-03-30T17:44:20-05:00",
    "modified_user_id": "1",
    "modified_by_name": "Administrator",
```

```
"modified_user_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
        "fields": [],
        "delete": "no",
        "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
    }
},
"created_by": "1",
"created_by_name": "Administrator",
"created_by_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
        "fields": [],
        "delete": "no",
        "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
    }
},
"description": "",
"deleted": false,
"facebook": "",
"twitter": "",
"googleplus": "",
"account_type": "Customer",
"industry": "Banking",
"annual_revenue": "",
"phone_fax": "",
"billing_address_street": "67321 West Siam St.",
"billing_address_street_2": "",
"billing_address_street_3": "",
"billing_address_street_4": "",
"billing_address_city": "Ohio",
"billing_address_state": "CA",
"billing_address_postalcode": "25159",
"billing_address_country": "USA",
"rating": "",
"phone_office": "(065) 489-6104",
"phone_alternate": "",
"website": "www.qahr.edu",
"ownership": "",
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "67321 West Siam St.",
"shipping_address_street_2": "",
```

```
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "Ohio",
"shipping_address_state": "CA",
"shipping_address_postalcode": "25159",
"shipping_address_country": "USA",
"parent_id": "",
"sic_code": "",
"duns_num": "",
"parent_name": "",
"member_of": {
    "name": "",
    "id": "",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"campaign_id": "",
"campaign_name": "",
"campaign_accounts": {
    "name": "",
    "id": "",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"following": true,
"my_favorite": false,
"tag": [],
"assigned_user_id": "seed_sarah_id",
"assigned_user_name": "Sarah Smith",
"assigned_user_link": {
    "full_name": "Sarah Smith",
    "id": "seed_sarah_id",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"team_count": "",
"team_count_link": {
    "team_count": "",
    "id": "West",
    "_acl": {
```

```
            "fields": [],
            "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
        }
    },
    "team_name": [{
        "id": "East",
        "name": "East",
        "name_2": "",
        "primary": false
    }, {
        "id": 1,
        "name": "Global",
        "name_2": "",
        "primary": false
    }, {
        "id": "West",
        "name": "West",
        "name_2": "",
        "primary": true
    }],
    "email": [{
        "email_address": "hr.support.kid@example.info",
        "invalid_email": false,
        "opt_out": false,
        "primary_address": true,
        "reply_to_address": false
    }, {
        "email_address": "info.support.the@example.com",
        "invalid_email": false,
        "opt_out": false,
        "primary_address": false,
        "reply_to_address": false
    }],
    "email1": "hr.support.kid@example.info",
    "email2": "info.support.the@example.com",
    "invalid_email": false,
    "email_opt_out": false,
    "email_addresses_non_primary": "",
    "_acl": {
        "fields": {}
    },
    "_module": "Accounts"
}
```

Download

You can download the full API example here.

How to Unfollow a Record

Overview

A PHP example demonstrating how to unfollow a record using the v11 /<module>/:record/unsubscribe REST DELETE endpoint.

Unfollowing a Record

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  <?php

$instance_url = "http://{site_url}/rest/v11";
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
    //It is recommended to create your own in Admin > OAuth Keys
    "client_id" => "sugar",
    "client_secret" => "",
    "username" => $username,
    "password" => $password,
    //platform type - default is base.
    //It is recommend to change the platform to a custom name such as
"custom_api" to avoid authentication conflicts.
    "platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
```

```
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Unfollowing a Record

Next, we can unfollow a specific record using the /<module>/:record/unsubscribe endpoint.

```
  //Unfollow a record - DELETE /<module>/:record/unsubscribe

$unsubscribe_url = $instance_url . "/Accounts/ae8b1783-404a-
fcb8-1e1e-56f1cc52cd1a/unsubscribe";

$unsubscribe_request = curl_init($unsubscribe_url);

curl_setopt($unsubscribe_request, CURLOPT_CUSTOMREQUEST, "DELETE");
curl_setopt($unsubscribe_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERS
ION_1_0);
curl_setopt($unsubscribe_request, CURLOPT_HEADER, false);
curl_setopt($unsubscribe_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($unsubscribe_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($unsubscribe_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($unsubscribe_request, CURLOPT_HTTPHEADER, array(
 "Content-Type: application/json",
 "oauth-token: {$oauth_token}"
```

```
));

//execute request
$unsubscribe_response = curl_exec($unsubscribe_request);
```

More information on the unsubscribe API can be found in the /<module>/:record/unsubscribe DELETE documentation.

Request Payload

The data sent to the server is shown below:

```
This endpoint does not accept any request arguments.
```

Response

The data received from the server is shown below:

```
  true
```

Download

You can download the full API example here.

How to Get the Most Active Users

Overview

A PHP example demonstrating how to fetch the most active users for meetings, calls, inbound emails, and outbound emails using the v11 /mostactiveusers REST GET endpoint.

Get Most Active Users

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  <?php

$instance_url = "http://{site_url}/rest/v11";
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
    //It is recommended to create your own in Admin > OAuth Keys
    "client_id" => "sugar",
    "client_secret" => "",
    "username" => $username,
    "password" => $password,
    //platform type - default is base.
    //It is recommend to change the platform to a custom name such as
"custom_api" to avoid authentication conflicts.
    "platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Active Users

Next, we can retrieve the most active users using the /mostactiveusers endpoint.

```
  //Fetch users - GET /mostactiveusers

$most_active_arguments = array(
    "days" => 30,
);

$most_active_url = $instance_url . "/mostactiveusers";

$most_active_url .= "?" . http_build_query($most_active_arguments);

$most_active_request = curl_init($most_active_url);

curl_setopt($most_active_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERS
ION_1_0);
curl_setopt($most_active_request, CURLOPT_HEADER, false);
curl_setopt($most_active_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($most_active_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($most_active_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($most_active_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//execute request
$most_active_response = curl_exec($most_active_request);

//decode json
$most_active_response_obj = json_decode($most_active_response);
```

More information on the mostactiveusers API can be found in the /mostactiveusers documentation.

Request

The URL sent to the server is shown below:

```
  http://{site_url}/rest/v11/mostactiveusers?days=30
```

Response

The data received from the server is shown below:


```
  {
    "meetings": {
        "user_id": "seed_max_id",
        "count": "21",
        "first_name": "Max",
        "last_name": "Jensen"
    },
    "calls": {
        "user_id": "seed_chris_id",
        "count": "4",
        "first_name": "Chris",
        "last_name": "Olliver"
    },
    "inbound_emails": {
        "user_id": "seed_chris_id",
        "count": "23",
        "first_name": "Chris",
        "last_name": "Olliver"
    },
    "outbound_emails": {
        "user_id": "seed_sarah_id",
        "count": "20",
        "first_name": "Sarah",
        "last_name": "Smith"
    }
}
```


Download

You can download the full API example here.

How to Authenticate and Log Out

Overview

A PHP example on how to authenticate and logout of the v11 REST API using the

/oauth2/token and /oauth2/logout POST endpoints.

Authenticating

The example below demonstrates how to authenticate to the REST v11 API. It is important to note that the oauth2 token arguments takes in several parameters that you should be aware of. The platform argument tends to cause confusion in that it is used to authenticate a user to a specific platform. Since Sugar only allows 1 user in the system at a time per platform, authenticating an integration script with a platform type of "base" will logout any current users in the system using those credentials. To work around this, your custom scripts should have a new platform type specified such as "custom_api" or any other static text you prefer. The client_id and client_secret parameters can be used for additional security based on client types. You can create your own client type in Admin > OAuth Keys. More information can be found in the /oauth2/token documentation. An example script is shown below:

```
  <?php

$instance_url = "http://{site_url}/rest/v11";
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
    //It is recommended to create your own in Admin > OAuth Keys
    "client_id" => "sugar",
    "client_secret" => "",
    "username" => $username,
    "password" => $password,
    //platform type - default is base.
    //It is recommend to change the platform to a custom name such as
"custom_api" to avoid authentication conflicts.
    "platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
```

```
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);
```

Request Payload

The data sent to the server is shown below:

```
  {
   "grant_type":"password",
   "client_id":"sugar",
   "client_secret":"",
   "username":"admin",
   "password":"password",
   "platform":"custom_api"
}
```

Response

The data received from the server is shown below:

```
  {
   "access_token":"c6d495c9-bb25-81d2-5f81-533ef6479f9b",
   "expires_in":3600,
   "token_type":"bearer",
   "scope":null,
   "refresh_token":"cbc40e67-12bc-4b56-a1d9-533ef62f2601",
   "refresh_expires_in":1209600,
   "download_token":"cc5d1a9f-6627-3349-96e5-533ef6b1a493"
}
```

Logout

To log out of a session, a request can be made to the /oauth2/logout POST endpoint. More information can be found in the /oauth2/logout documentation. An example extending off of the above authentication example is shown below:

```
  //Logout - POST /oauth2/logout

$logout_url = $instance_url . "/oauth2/logout";

//decode oauth2 repsonse to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;

$logout_request = curl_init($logout_url);
curl_setopt($logout_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1
_0);
curl_setopt($logout_request, CURLOPT_HEADER, false);
curl_setopt($logout_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($logout_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($logout_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($logout_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//set empty post fields
curl_setopt($logout_request, CURLOPT_POSTFIELDS, array());

//execute request
$oauth2_logout_response = curl_exec($logout_request);
```

Response

The data received from the server is shown below:

```
  {
    "success":true
}
```

Download

You can download the full API example here.

How to Ping

Overview

A PHP example demonstrating how to ping using the REST v11 /ping GET endpoint.

Pinging

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  <?php

$instance_url = "http://{site_url}/rest/v11";
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
    //It is recommended to create your own in Admin > OAuth Keys
    "client_id" => "sugar",
    "client_secret" => "",
    "username" => $username,
    "password" => $password,
    //platform type - default is base.
    //It is recommend to change the platform to a custom name such as
"custom_api" to avoid authentication conflicts.
    "platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
```

```
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Pinging

Once we have authenticated, we can now ping the system as shown below.

```
  //Ping - GET /ping

$ping_url = $instance_url . "/ping";

$ping_request = curl_init($ping_url);
curl_setopt($ping_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($ping_request, CURLOPT_HEADER, false); //needed to return
file headers
curl_setopt($ping_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($ping_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ping_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($ping_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

$ping_response = curl_exec($ping_request);
```

More information on pinging can be found in the /ping documentation.

Response

```
"pong"
```

Download

You can download the full API example here.

How to Fetch the Current Users Time

Overview

A PHP example demonstrating how to fetch the current users time with the v11 /ping/whattimeisit REST GET endpoint.

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
<?php

$instance_url = "http://{site_url}/rest/v11";
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
    //It is recommended to create your own in Admin > OAuth Keys
    "client_id" => "sugar",
    "client_secret" => "",
    "username" => $username,
    "password" => $password,
    //platform type - default is base.
    //It is recommend to change the platform to a custom name such as
"custom_api" to avoid authentication conflicts.
    "platform" => "custom_api"
```

```
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Getting the Current Time and Date for the User

```
  //Set up the time parameters - GET /ping/whattimeisit

$time_url = $instance_url . "/ping/whattimeisit";

$time_request = curl_init($search_url);
curl_setopt($time_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($time_request, CURLOPT_HEADER, false);
curl_setopt($time_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($time_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($time_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($time_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
```

```
));

//execute request
$time_response = curl_exec($time_request);

//decode json
$time_response_obj = json_decode($time_response);
```

Request

```
http://{site_url}/rest/v11/ping/whattimeisit
```

Response

```
  "2014-04-08T14:59:13-04:00"
```

Downloads

You can download the full API example here

How to Fetch Recently Viewed Records

Overview

A PHP example demonstrating how to retrieve recently viewed records using the v11 /recent REST GET endpoint.

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  <?php

$instance_url = "http://{site_url}/rest/v11";
$username = "admin";
$password = "password";
```

```
//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
    //It is recommended to create your own in Admin > OAuth Keys
    "client_id" => "sugar",
    "client_secret" => "",
    "username" => $username,
    "password" => $password,
    //platform type - default is base.
    //It is recommend to change the platform to a custom name such as
"custom_api" to avoid authentication conflicts.
    "platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Recently Viewed Records

Next, we will need to identify the records we want to see using the /recent endpoint. In this case, we are going to request Accounts, Contacts and Leads.

```
  //Set up the search parameters - GET /recent

$recent_url = $instance_url . "/recent";

$recent_arguments = array(
    "module_list" => 'Accounts,Contacts,Leads',
);

//As this request is a GET we will add the arguments to the URL
$recent_url .= "?" . http_build_query($recent_arguments);

$recent_request = curl_init($recent_url);
curl_setopt($recent_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1
_0);
curl_setopt($recent_request, CURLOPT_HEADER, false);
curl_setopt($recent_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($recent_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($recent_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($recent_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//execute request
$recent_response = curl_exec($recent_request);

//decode json
$recent_response_obj = json_decode($recent_response);
```

More information on the search API can be found in the /recent documentation.

Request

```
http://{site_url}/rest/v11/recent?module_list=Accounts%2CContacts%2CLe
ads
```

Response

The data received from the server is shown below:


```
{
   "next_offset":-1,
   "records":[
      {
         "my_favorite":false,
         "following":false,
         "id":"f3951f4d-2d17-7939-c5ec-56fedbb9e92f",
         "name":"Talia Knupp",
         "date_entered":"2016-04-01T15:34:00-05:00",
         "date_modified":"2016-04-06T10:33:24-05:00",
         "modified_user_id":"1",
         "modified_by_name":"Administrator",
         "created_by":"1",
         "created_by_name":"Administrator",
         "doc_owner":"",
         "user_favorites":"",
         "description":"",
         "deleted":false,
         "assigned_user_id":"seed_will_id",
         "assigned_user_name":"Will Westin",
         "team_count":"",
         "team_name":[
            {
               "id":"East",
               "name":"East",
               "name_2":"",
               "primary":true
            },
            {
               "id":"West",
               "name":"West",
               "name_2":"",
               "primary":false
            }
         ],
         "email":[
            {
               "email_address":"jsmith@sugar.com",
               "invalid_email":false,
               "opt_out":false,
               "primary_address":true,
               "reply_to_address":false
            },
```

```
      {
         "email_address":"cjsmith@sugar.com",
         "invalid_email":false,
         "opt_out":false,
         "primary_address":false,
         "reply_to_address":false
      }
   ],
   "email1":"jsmith@sugar.com",
   "email2":"cjsmith@sugar.com",
   "invalid_email":false,
   "email_opt_out":false,
   "email_addresses_non_primary":"",
   "salutation":"",
   "first_name":"Talia",
   "last_name":"Knupp",
   "full_name":"Talia Knupp",
   "title":"Senior Product Manager",
   "facebook":"",
   "twitter":"",
   "googleplus":"",
   "department":"",
   "do_not_call":false,
   "phone_home":"(963) 741-3689",
   "phone_mobile":"(600) 831-9872",
   "phone_work":"(680) 991-2837",
   "phone_other":"",
   "phone_fax":"",
   "primary_address_street":"111 Silicon Valley Road",
   "primary_address_street_2":"",
   "primary_address_street_3":"",
   "primary_address_city":"Sunnyvale",
   "primary_address_state":"NY",
   "primary_address_postalcode":"99452",
   "primary_address_country":"USA",
   "alt_address_street":"",
   "alt_address_street_2":"",
   "alt_address_street_3":"",
   "alt_address_city":"",
   "alt_address_state":"",
   "alt_address_postalcode":"",
   "alt_address_country":"",
   "assistant":"",
   "assistant_phone":"",
   "picture":"",
   "converted":false,
```

```
"refered_by":"",
"lead_source":"Word of mouth",
"lead_source_description":"",
"status":"In Process",
"status_description":"",
"reports_to_id":"",
"report_to_name":"",
"dnb_principal_id":"",
"account_name":"First National S\/B",
"account_description":"",
"contact_id":"",
"contact_name":"",
"account_id":"",
"opportunity_id":"",
"opportunity_name":"",
"opportunity_amount":"",
"campaign_id":"",
"campaign_name":"",
"c_accept_status_fields":"",
"m_accept_status_fields":"",
"accept_status_id":"",
"accept_status_name":"",
"accept_status_calls":"",
"accept_status_meetings":"",
"webtolead_email1":[
   {
      "email_address":"jsmith@sugar.com",
      "invalid_email":false,
      "opt_out":false,
      "primary_address":true,
      "reply_to_address":false
   },
   {
      "email_address":"cjsmith@sugar.com",
      "invalid_email":false,
      "opt_out":false,
      "primary_address":false,
      "reply_to_address":false
   }
],
"webtolead_email2":[
   {
      "email_address":"jsmith@sugar.com",
      "invalid_email":false,
      "opt_out":false,
      "primary_address":true,
```

```
            "reply_to_address":false
         },
         {
            "email_address":"cjsmith@sugar.com",
            "invalid_email":false,
            "opt_out":false,
            "primary_address":false,
            "reply_to_address":false
         }
      ],
      "webtolead_email_opt_out":"",
      "webtolead_invalid_email":"",
      "birthdate":"",
      "portal_name":"",
      "portal_app":"",
      "website":"",
      "preferred_language":"",
      "mkto_sync":false,
      "mkto_id":null,
      "mkto_lead_score":null,
      "fruits_c":"Apples",
      "_acl":{
         "fields":{

         }
      },
      "_module":"Leads",
      "_last_viewed_date":"2016-04-06T10:33:24-05:00"
   },
   {
      "my_favorite":false,
      "following":false,
      "id":"e8c641ca-1b8c-74c1-d08d-56fedbdd3187",
      "name":"MTM Investment Bank F S B",
      "date_entered":"2016-04-01T15:34:00-05:00",
      "date_modified":"2016-04-06T10:16:52-05:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "doc_owner":"",
      "user_favorites":"",
      "description":"",
      "deleted":false,
      "assigned_user_id":"seed_will_id",
      "assigned_user_name":"Will Westin",
```

```
"team_count":"",
"team_name":[
   {
      "id":"East",
      "name":"East",
      "name_2":"",
      "primary":true
   },
   {
      "id":"West",
      "name":"West",
      "name_2":"",
      "primary":false
   }
],
"email":[
   {
      "email_address":"jsmith@sugar.com",
      "invalid_email":false,
      "opt_out":false,
      "primary_address":true,
      "reply_to_address":false
   },
   {
      "email_address":"the60@example.us",
      "invalid_email":false,
      "opt_out":false,
      "primary_address":false,
      "reply_to_address":false
   }
],
"email1":"jsmith@sugar.com",
"email2":"the60@example.us",
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"account_type":"Customer",
"industry":"a",
"annual_revenue":"",
"phone_fax":"",
"billing_address_street":"67321 West Siam St.",
"billing_address_street_2":"",
"billing_address_street_3":"",
```

```
      "billing_address_street_4":"",
      "billing_address_city":"Alabama",
      "billing_address_state":"NY",
      "billing_address_postalcode":"52272",
      "billing_address_country":"USA",
      "rating":"",
      "phone_office":"(012) 704-8075",
      "phone_alternate":"",
      "website":"www.salesqa.it",
      "ownership":"",
      "employees":"",
      "ticker_symbol":"",
      "shipping_address_street":"67321 West Siam St.",
      "shipping_address_street_2":"",
      "shipping_address_street_3":"",
      "shipping_address_street_4":"",
      "shipping_address_city":"Alabama",
      "shipping_address_state":"NY",
      "shipping_address_postalcode":"52272",
      "shipping_address_country":"USA",
      "parent_id":"",
      "sic_code":"",
      "duns_num":"",
      "parent_name":"",
      "campaign_id":"",
      "campaign_name":"",
      "_acl":{
         "fields":{

         }
      },
      "_module":"Accounts",
      "_last_viewed_date":"2016-04-06T10:16:52-05:00"
   },
   {
      "my_favorite":false,
      "following":false,
      "id":"f31b2f00-468c-3d35-1e88-56fedbd3921d",
      "name":"Kaycee Gibney",
      "date_entered":"2016-04-01T15:34:00-05:00",
      "date_modified":"2016-04-06T10:16:24-05:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "doc_owner":"",
```

```
"user_favorites":"",
"description":"",
"deleted":false,
"assigned_user_id":"seed_jim_id",
"assigned_user_name":"Jim Brennan",
"team_count":"",
"team_name":[
   {
      "id":"East",
      "name":"East",
      "name_2":"",
      "primary":true
   }
],
"email":[
   {
      "email_address":"jsmith@sugar.com",
      "invalid_email":false,
      "opt_out":false,
      "primary_address":true,
      "reply_to_address":false
   },
   {
      "email_address":"sales.kid.dev@example.info",
      "invalid_email":false,
      "opt_out":true,
      "primary_address":false,
      "reply_to_address":false
   }
],
"email1":"jsmith@sugar.com",
"email2":"sales.kid.dev@example.info",
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"",
"salutation":"",
"first_name":"Kaycee",
"last_name":"Gibney",
"full_name":"Kaycee Gibney",
"title":"Mgr Operations",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(599) 165-2396",
```

```
"phone_mobile":"(215) 591-9574",
"phone_work":"(771) 945-3648",
"phone_other":"",
"phone_fax":"",
"primary_address_street":"321 University Ave.",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Santa Monica",
"primary_address_state":"NY",
"primary_address_postalcode":"96154",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
"assistant":"",
"assistant_phone":"",
"picture":"",
"email_and_name1":"",
"lead_source":"Existing Customer",
"account_name":"Tracker Com LP",
"account_id":"72ad6f00-e345-1cab-b370-56fedbd23deb",
"dnb_principal_id":"",
"opportunity_role_fields":"",
"opportunity_role_id":"",
"opportunity_role":"",
"reports_to_id":"",
"report_to_name":"",
"birthdate":"",
"portal_name":"KayceeGibney33",
"portal_active":true,
"portal_password":true,
"portal_password1":null,
"portal_app":"",
"preferred_language":"en_us",
"campaign_id":"",
"campaign_name":"",
"c_accept_status_fields":"",
"m_accept_status_fields":"",
"accept_status_id":"",
"accept_status_name":"",
"accept_status_calls":"",
"accept_status_meetings":"",
```

```
         "sync_contact":false,
         "mkto_sync":false,
         "mkto_id":null,
         "mkto_lead_score":null,
         "_acl":{
            "fields":{

            }
         },
         "_module":"Contacts",
         "_last_viewed_date":"2016-04-06T10:16:24-05:00"
      }
   ]
}
```

There are 3 records shown above, the _module field tells you if the record is from Accounts, Contacts or Leads. The _last_viewed_date will tell you when the record was last seen.

Downloads

You can download the full API example here

How to Use the Global Search

Overview

A PHP example demonstrating how to globally search for records using the REST v11 /search GET endpoint.

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  <?php

$instance_url = "http://{site_url}/rest/v11";
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
```

```
$auth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
    //It is recommended to create your own in Admin > OAuth Keys
    "client_id" => "sugar",
    "client_secret" => "",
    "username" => $username,
    "password" => $password,
    //platform type - default is base.
    //It is recommend to change the platform to a custom name such as
"custom_api" to avoid authentication conflicts.
    "platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Searching Records

So, we will need to identify the records we want to see using the /search endpoint. In this case, we are going to search for all records that have the email address of 'jsmith@sugar.com'. In this example, there are 3 records, an Account, a Contact and a Lead.

```
  //Set up the search parameters - GET /search

$search_url = $instance_url . "/search";

$search_arguments = array(
    "q" => 'jsmith@sugar.com',
    "max_num" => 3,
    "offset" => 0,
    "fields" => "",
    "order_by" => "",
    "favorites" => false,
    "my_items" => false,
);

//As this request is a GET we will add the arguments to the URL
$search_url .= "?" . http_build_query($search_arguments);

$search_request = curl_init($search_url);
curl_setopt($search_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1
_0);
curl_setopt($search_request, CURLOPT_HEADER, false);
curl_setopt($search_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($search_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($search_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($search_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//execute request
$search_response = curl_exec($search_request);

//decode json
$search_response_obj = json_decode($search_response);
```

More information on the search API can be found in the /search documentation.

Request

```
http://{site_url}/rest/v11/search?q=jsmith%40sugar.com&max_num=3&offset=0&fields=&order_by=&favorites=0&my_items=0
```

Response

The data received from the server is shown below:


```
{
   "next_offset":-1,
   "records":[
      {
         "my_favorite":false,
         "following":false,
         "id":"f31b2f00-468c-3d35-1e88-56fedbd3921d",
         "name":"Kaycee Gibney",
         "date_entered":"2016-04-01T20:34:00+00:00",
         "date_modified":"2016-04-06T15:16:24+00:00",
         "modified_user_id":"1",
         "modified_by_name":"Administrator",
         "created_by":"1",
         "created_by_name":"Administrator",
         "doc_owner":"",
         "user_favorites":"",
         "description":"",
         "deleted":false,
         "assigned_user_id":"seed_jim_id",
         "assigned_user_name":"Jim Brennan",
         "team_count":"",
         "team_name":[
            {
               "id":"East",
               "name":"East",
               "name_2":"",
               "primary":true
            }
         ],
         "email":[
            {
               "email_address":"jsmith@sugar.com",
               "invalid_email":false,
               "opt_out":false,
               "primary_address":true,
               "reply_to_address":false
```

```
      },
      {
         "email_address":"sales.kid.dev@example.info",
         "invalid_email":false,
         "opt_out":true,
         "primary_address":false,
         "reply_to_address":false
      }
   ],
   "email1":"jsmith@sugar.com",
   "email2":"sales.kid.dev@example.info",
   "invalid_email":false,
   "email_opt_out":false,
   "email_addresses_non_primary":"",
   "salutation":"",
   "first_name":"Kaycee",
   "last_name":"Gibney",
   "full_name":"Kaycee Gibney",
   "title":"Mgr Operations",
   "facebook":"",
   "twitter":"",
   "googleplus":"",
   "department":"",
   "do_not_call":false,
   "phone_home":"(599) 165-2396",
   "phone_mobile":"(215) 591-9574",
   "phone_work":"(771) 945-3648",
   "phone_other":"",
   "phone_fax":"",
   "primary_address_street":"321 University Ave.",
   "primary_address_street_2":"",
   "primary_address_street_3":"",
   "primary_address_city":"Santa Monica",
   "primary_address_state":"NY",
   "primary_address_postalcode":"96154",
   "primary_address_country":"USA",
   "alt_address_street":"",
   "alt_address_street_2":"",
   "alt_address_street_3":"",
   "alt_address_city":"",
   "alt_address_state":"",
   "alt_address_postalcode":"",
   "alt_address_country":"",
   "assistant":"",
   "assistant_phone":"",
   "picture":"",
```

```
        "email_and_name1":"",
        "lead_source":"Existing Customer",
        "account_name":"Tracker Com LP",
        "account_id":"72ad6f00-e345-1cab-b370-56fedbd23deb",
        "dnb_principal_id":"",
        "opportunity_role_fields":"",
        "opportunity_role_id":"",
        "opportunity_role":"",
        "reports_to_id":"",
        "report_to_name":"",
        "birthdate":"",
        "portal_name":"KayceeGibney33",
        "portal_active":true,
        "portal_password":true,
        "portal_password1":null,
        "portal_app":"",
        "preferred_language":"en_us",
        "campaign_id":"",
        "campaign_name":"",
        "c_accept_status_fields":"",
        "m_accept_status_fields":"",
        "accept_status_id":"",
        "accept_status_name":"",
        "accept_status_calls":"",
        "accept_status_meetings":"",
        "sync_contact":false,
        "mkto_sync":false,
        "mkto_id":null,
        "mkto_lead_score":null,
        "_acl":{
           "fields":{

           }
        },
        "_module":"Contacts",
        "_search":{
           "score":0.70710677,
           "highlighted":{
              "email1":{
                 "text":"\u003Cstrong\u003Ejsmith@sugar.com\u003C\/st
rong\u003E",
                 "module":"Contacts",
                 "label":"LBL_EMAIL_ADDRESS"
              }
           }
        }
```

```
},
{
   "my_favorite":false,
   "following":false,
   "id":"e8c641ca-1b8c-74c1-d08d-56fedbdd3187",
   "name":"MTM Investment Bank F S B",
   "date_entered":"2016-04-01T20:34:00+00:00",
   "date_modified":"2016-04-06T15:16:52+00:00",
   "modified_user_id":"1",
   "modified_by_name":"Administrator",
   "created_by":"1",
   "created_by_name":"Administrator",
   "doc_owner":"",
   "user_favorites":"",
   "description":"",
   "deleted":false,
   "assigned_user_id":"seed_will_id",
   "assigned_user_name":"Will Westin",
   "team_count":"",
   "team_name":[
      {
         "id":"East",
         "name":"East",
         "name_2":"",
         "primary":true
      },
      {
         "id":"West",
         "name":"West",
         "name_2":"",
         "primary":false
      }
   ],
   "email":[
      {
         "email_address":"jsmith@sugar.com",
         "invalid_email":false,
         "opt_out":false,
         "primary_address":true,
         "reply_to_address":false
      },
      {
         "email_address":"the60@example.us",
         "invalid_email":false,
         "opt_out":false,
         "primary_address":false,
```

```
      "reply_to_address":false
   }
],
"email1":"jsmith@sugar.com",
"email2":"the60@example.us",
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"account_type":"Customer",
"industry":"a",
"annual_revenue":"",
"phone_fax":"",
"billing_address_street":"67321 West Siam St.",
"billing_address_street_2":"",
"billing_address_street_3":"",
"billing_address_street_4":"",
"billing_address_city":"Alabama",
"billing_address_state":"NY",
"billing_address_postalcode":"52272",
"billing_address_country":"USA",
"rating":"",
"phone_office":"(012) 704-8075",
"phone_alternate":"",
"website":"www.salesqa.it",
"ownership":"",
"employees":"",
"ticker_symbol":"",
"shipping_address_street":"67321 West Siam St.",
"shipping_address_street_2":"",
"shipping_address_street_3":"",
"shipping_address_street_4":"",
"shipping_address_city":"Alabama",
"shipping_address_state":"NY",
"shipping_address_postalcode":"52272",
"shipping_address_country":"USA",
"parent_id":"",
"sic_code":"",
"duns_num":"",
"parent_name":"",
"campaign_id":"",
"campaign_name":"",
"_acl":{
   "fields":{
```

```
            }
         },
         "_module":"Accounts",
         "_search":{
            "score":0.70710677,
            "highlighted":{
               "email1":{
                  "text":"\u003Cstrong\u003Ejsmith@sugar.com\u003C\/st
rong\u003E",
                  "module":"Accounts",
                  "label":"LBL_EMAIL_ADDRESS"
               }
            }
         }
      },
      {
         "my_favorite":false,
         "following":false,
         "id":"f3951f4d-2d17-7939-c5ec-56fedbb9e92f",
         "name":"Talia Knupp",
         "date_entered":"2016-04-01T20:34:00+00:00",
         "date_modified":"2016-04-06T15:33:24+00:00",
         "modified_user_id":"1",
         "modified_by_name":"Administrator",
         "created_by":"1",
         "created_by_name":"Administrator",
         "doc_owner":"",
         "user_favorites":"",
         "description":"",
         "deleted":false,
         "assigned_user_id":"seed_will_id",
         "assigned_user_name":"Will Westin",
         "team_count":"",
         "team_name":[
            {
               "id":"East",
               "name":"East",
               "name_2":"",
               "primary":true
            },
            {
               "id":"West",
               "name":"West",
               "name_2":"",
               "primary":false
```

```
      }
   ],
   "email":[
      {
         "email_address":"jsmith@sugar.com",
         "invalid_email":false,
         "opt_out":false,
         "primary_address":true,
         "reply_to_address":false
      },
      {
         "email_address":"cjsmith@sugar.com",
         "invalid_email":false,
         "opt_out":false,
         "primary_address":false,
         "reply_to_address":false
      }
   ],
   "email1":"jsmith@sugar.com",
   "email2":"cjsmith@sugar.com",
   "invalid_email":false,
   "email_opt_out":false,
   "email_addresses_non_primary":"",
   "salutation":"",
   "first_name":"Talia",
   "last_name":"Knupp",
   "full_name":"Talia Knupp",
   "title":"Senior Product Manager",
   "facebook":"",
   "twitter":"",
   "googleplus":"",
   "department":"",
   "do_not_call":false,
   "phone_home":"(963) 741-3689",
   "phone_mobile":"(600) 831-9872",
   "phone_work":"(680) 991-2837",
   "phone_other":"",
   "phone_fax":"",
   "primary_address_street":"111 Silicon Valley Road",
   "primary_address_street_2":"",
   "primary_address_street_3":"",
   "primary_address_city":"Sunnyvale",
   "primary_address_state":"NY",
   "primary_address_postalcode":"99452",
   "primary_address_country":"USA",
   "alt_address_street":"",
```

```
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
"assistant":"",
"assistant_phone":"",
"picture":"",
"converted":false,
"refered_by":"",
"lead_source":"Word of mouth",
"lead_source_description":"",
"status":"In Process",
"status_description":"",
"reports_to_id":"",
"report_to_name":"",
"dnb_principal_id":"",
"account_name":"First National S\/B",
"account_description":"",
"contact_id":"",
"contact_name":"",
"account_id":"",
"opportunity_id":"",
"opportunity_name":"",
"opportunity_amount":"",
"campaign_id":"",
"campaign_name":"",
"c_accept_status_fields":"",
"m_accept_status_fields":"",
"accept_status_id":"",
"accept_status_name":"",
"accept_status_calls":"",
"accept_status_meetings":"",
"webtolead_email1":[
   {
      "email_address":"jsmith@sugar.com",
      "invalid_email":false,
      "opt_out":false,
      "primary_address":true,
      "reply_to_address":false
   },
   {
      "email_address":"cjsmith@sugar.com",
      "invalid_email":false,
      "opt_out":false,
```

```
            "primary_address":false,
            "reply_to_address":false
         }
      ],
      "webtolead_email2":[
         {
            "email_address":"jsmith@sugar.com",
            "invalid_email":false,
            "opt_out":false,
            "primary_address":true,
            "reply_to_address":false
         },
         {
            "email_address":"cjsmith@sugar.com",
            "invalid_email":false,
            "opt_out":false,
            "primary_address":false,
            "reply_to_address":false
         }
      ],
      "webtolead_email_opt_out":"",
      "webtolead_invalid_email":"",
      "birthdate":"",
      "portal_name":"",
      "portal_app":"",
      "website":"",
      "preferred_language":"",
      "mkto_sync":false,
      "mkto_id":null,
      "mkto_lead_score":null,
      "fruits_c":"Apples",
      "_acl":{
         "fields":{

         }
      },
      "_module":"Leads",
      "_search":{
         "score":0.70710677,
         "highlighted":{
            "email1":{
               "text":"\u003Cstrong\u003Ejsmith@sugar.com\u003C\/st
rong\u003E",
               "module":"Leads",
               "label":"LBL_EMAIL_ADDRESS"
            }
```

```
            }
         }
      }
   ]
}
```

There are 3 records shown above, the _module field tells you if the record is from Accounts, Contacts or Leads. The _search field is an array that tells you where in the record it found the search string. It gives you back a highlighted string that you can use for display.

Downloads

You can download the full API example here

How to Check for Duplicate Records

Overview

An PHP example demonstrating how to check for duplicate records using the v11 /<module>/duplicateCheck REST POST endpoint.

Duplicate Records

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  <?php

$instance_url = "http://{site_url}/rest/v11";
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
    //It is recommended to create your own in Admin > OAuth Keys
```

```
    "client_id" => "sugar",
    "client_secret" => "",
    "username" => $username,
    "password" => $password,
    //platform type - default is base.
    //It is recommend to change the platform to a custom name such as
"custom_api" to avoid authentication conflicts.
    "platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Retrieving Duplicates

Next, we will need to identify the records that are duplicates using the /<module>/duplicateCheck endpoint.

```
  //Check for duplicate records - POST /<module>/duplicateCheck

$url = $instance_url . "/Accounts/duplicateCheck";
//Set up the Record details
```

```
$record = array(
    'name' => 'Test Record',
);

$curl_request = curl_init($url);
curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($curl_request, CURLOPT_HEADER, false);
curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($curl_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//convert arguments to json
$json_arguments = json_encode($record);
curl_setopt($curl_request, CURLOPT_POSTFIELDS, $json_arguments);
//execute request
$curl_response = curl_exec($curl_request);
//decode json
$createdRecord = json_decode($curl_response);

//display the created record
print_r($createdRecord);
curl_close($curl_request);
```

More information on the filter API can be found in the /<module>/duplicateCheck documentation.

Request Payload

The data sent to the server is shown below:

```
  {
   "name":"Test Record"
}
```

Response

The data received from the server is shown below:

```
{
  "next_offset": -1,
  "records": [{
      "id": "7f6ea7be-60d6-11e6-8885-a0999b033b33",
      "name": "Test Record",
      "date_entered": "2016-08-12T14:48:25-07:00",
      "date_modified": "2016-08-12T14:48:25-07:00",
      "modified_user_id": "1",
      "modified_by_name": "Administrator",
      "modified_user_link": {
          "full_name": "Administrator",
          "id": "1",
          "_acl": {
              "fields": [],
              "delete": "no",
              "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
          }
      },
      "created_by": "1",
      "created_by_name": "Administrator",
      "created_by_link": {
          "full_name": "Administrator",
          "id": "1",
          "_acl": {
              "fields": [],
              "delete": "no",
              "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
          }
      },
      "description": "Test Data 1",
      "deleted": false,
      "facebook": "",
      "twitter": "",
      "googleplus": "",
      "account_type": "",
      "industry": "",
      "annual_revenue": "",
      "phone_fax": "",
      "billing_address_street": "",
      "billing_address_street_2": "",
      "billing_address_street_3": "",
      "billing_address_street_4": "",
      "billing_address_city": "",
      "billing_address_state": "",
      "billing_address_postalcode": "",
      "billing_address_country": "",
```

```
"rating": "",
"phone_office": "",
"phone_alternate": "",
"website": "",
"ownership": "",
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "",
"shipping_address_state": "",
"shipping_address_postalcode": "",
"shipping_address_country": "",
"parent_id": "",
"sic_code": "",
"duns_num": "",
"parent_name": "",
"member_of": {
    "name": "",
    "id": "",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"campaign_id": "",
"campaign_name": "",
"campaign_accounts": {
    "name": "",
    "id": "",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"following": true,
"my_favorite": false,
"tag": [],
"assigned_user_id": "1",
"assigned_user_name": "Administrator",
"assigned_user_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
```

```
            "fields": [],
            "delete": "no",
            "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
        }
    },
    "team_count": "",
    "team_count_link": {
        "team_count": "",
        "id": "1",
        "_acl": {
            "fields": [],
            "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
        }
    },
    "team_name": [{
        "id": "1",
        "name": "Global",
        "name_2": "",
        "primary": true
    }],
    "email": [],
    "email1": "",
    "email2": "",
    "invalid_email": "",
    "email_opt_out": "",
    "email_addresses_non_primary": "",
    "_acl": {
        "fields": {}
    },
    "_module": "Accounts",
    "duplicate_check_rank": 8
}, {
    "id": "868b4f16-60d6-11e6-bdfc-a0999b033b33",
    "name": "Test Record",
    "date_entered": "2016-08-12T14:48:37-07:00",
    "date_modified": "2016-08-12T14:48:37-07:00",
    "modified_user_id": "1",
    "modified_by_name": "Administrator",
    "modified_user_link": {
        "full_name": "Administrator",
        "id": "1",
        "_acl": {
            "fields": [],
            "delete": "no",
            "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
        }
```

```
},
"created_by": "1",
"created_by_name": "Administrator",
"created_by_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
        "fields": [],
        "delete": "no",
        "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
    }
},
"description": "Test Data 2",
"deleted": false,
"facebook": "",
"twitter": "",
"googleplus": "",
"account_type": "",
"industry": "",
"annual_revenue": "",
"phone_fax": "",
"billing_address_street": "",
"billing_address_street_2": "",
"billing_address_street_3": "",
"billing_address_street_4": "",
"billing_address_city": "",
"billing_address_state": "",
"billing_address_postalcode": "",
"billing_address_country": "",
"rating": "",
"phone_office": "",
"phone_alternate": "",
"website": "",
"ownership": "",
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "",
"shipping_address_state": "",
"shipping_address_postalcode": "",
"shipping_address_country": "",
"parent_id": "",
"sic_code": "",
```

```
"duns_num": "",
"parent_name": "",
"member_of": {
    "name": "",
    "id": "",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"campaign_id": "",
"campaign_name": "",
"campaign_accounts": {
    "name": "",
    "id": "",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"following": true,
"my_favorite": false,
"tag": [],
"assigned_user_id": "1",
"assigned_user_name": "Administrator",
"assigned_user_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
        "fields": [],
        "delete": "no",
        "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
    }
},
"team_count": "",
"team_count_link": {
    "team_count": "",
    "id": "1",
    "_acl": {
        "fields": [],
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"team_name": [{
    "id": "1",
    "name": "Global",
```

```
            "name_2": "",
            "primary": true
        }],
        "email": [],
        "email1": "",
        "email2": "",
        "invalid_email": "",
        "email_opt_out": "",
        "email_addresses_non_primary": "",
        "_acl": {
            "fields": {}
        },
        "_module": "Accounts",
        "duplicate_check_rank": 8
    }]
}
```

Download

You can download the full API example here.

How to Manipulate Quotes

Overview

A PHP example demonstrating how to manipulate Quotes and related record data such as ProductBundles, Products, and ProductBundleNotes.

Manipulating Quotes

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  <?php

$instance_url = "http://{site_url}/rest/v11";
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";
```

```
$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
    //It is recommended to create your own in Admin > OAuth Keys
    "client_id" => "sugar",
    "client_secret" => "",
    "username" => $username,
    "password" => $password,
    //platform type - default is base.
    //It is recommend to change the platform to a custom name such as
"custom_api" to avoid authentication conflicts.
    "platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Creating a Quote

Once authenticated, we can submit a Quote record using the <module> endpoint. Since a Quote record is a sum of its parts (i.e. ProductBundles and Products) you

can submit the related ProductBundles and Products in the same request payload using the relationship Links and the "create" property.

```
  //Create Records - POST /<module>
$url = $instance_url . "/Quotes";
//Set up the Record details
$DateTime = new DateTime();
//Expected Close Date in 1 Month
$DateTime->add(new DateInterval("P1M"));
//Quote Record
$quote = array(
    'name' => 'Test Quote',
    'quote_stage' => 'Draft',
    'date_quote_expected_closed' => $DateTime->format(DateTime::ISO860
1),
    //Create Product Bundles
    'product_bundles' => array(
        'create' => array(
            array(
                "name" => "Product Bundle 1",
                "bundle_stage" => "Draft",
                "currency_id" => ":-99",
                "base_rate" => "1.0",
                "shipping" => "0.00",
                "products" => array(
                    //Create Product in Bundle 1
                    "create" => array(
                        array(
                            "tax_class" => "Taxable",
                            "quantity" => 1000.00,
                            "name" => "Test Product 1",
                            "description" => "My Test Product",
                            "mft_part_num" => "mft100012021",
                            "cost_price" =>  "100.00",
                            "list_price" =>  "200.00",
                            "discount_price" =>  "175.00",
                            "discount_amount" =>  "0.00",
                            "discount_select" =>  0,
                            "product_template_id" =>  "",
                            "type_id"  =>  "",
                            "status"  =>  "Quotes"
                        )
                    )
                ),
                //Create Product Bundle Note in Bundle 1
                "product_bundle_notes" => array(
```

```
            "create" => array(
                array(
                    "description" => "Free shipping",
                    "position" => 1
                )
            )
        )
    ),
    array(
        "name" => "Product Bundle 2",
        "bundle_stage" => "Draft",
        "currency_id" => ":-99",
        "base_rate" => "1.0",
        "shipping" => "25.00",
        "products" => array(
            //Create Products in Bundle 2
            "create" => array(
                array(
                    "quantity" => 1000.00,
                    "name" => "Test Product 2",
                    "description" => "My Other Product",
                    "mft_part_num" => "mft100012234",
                    "cost_price" =>  "150.00",
                    "list_price" =>  "300.00",
                    "discount_price" =>  "275.00",
                    "discount_amount" =>  "0.00",
                    "discount_select" =>  0,
                    "product_template_id" =>  "",
                    "type_id"  =>  "",
                    "status"  =>  "Quotes"
                ),
                array(
                    "quantity" => 500.00,
                    "name" => "Test Product 3",
                    "description" => "My Other Other Product",
                    "mft_part_num" => "mft100012123",
                    "cost_price" =>  "10.00",
                    "list_price" =>  "500.00",
                    "discount_price" =>  "400.00",
                    "discount_amount" =>  "0.00",
                    "discount_select" =>  0,
                    "product_template_id" =>  "",
                    "type_id"  =>  "",
                    "status"  =>  "Quotes"
                )
            )
```

```
                )
            )
        ),
    )
);

$curl_request = curl_init($url);
curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($curl_request, CURLOPT_HEADER, false);
curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($curl_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//convert arguments to json
$json_arguments = json_encode($quote);
curl_setopt($curl_request, CURLOPT_POSTFIELDS, $json_arguments);
//execute request
$curl_response = curl_exec($curl_request);
//decode json
$createdQuote = json_decode($curl_response);

//display the created record
print_r($createdQuote);
curl_close($curl_request);
```

Request Payload

The data sent to the server is shown below:

```
  {
  "name": "Test Quote",
  "quote_stage": "Draft",
  "date_quote_expected_closed": "2017-06-12T11:51:57-0400",
  "product_bundles": {
    "create": [
      {
        "name": "Product Bundle 1",
        "bundle_stage": "Draft",
        "currency_id": ":-99",
```

```
    "base_rate": "1.0",
    "shipping": "0.00",
    "products": {
      "create": [
        {
          "tax_class": "Taxable",
          "quantity": 1000,
          "name": "Test Product 1",
          "description": "My Test Product",
          "mft_part_num": "mft100012021",
          "cost_price": "100.00",
          "list_price": "200.00",
          "discount_price": "175.00",
          "discount_amount": "0.00",
          "discount_select": 0,
          "product_template_id": "",
          "type_id": "",
          "status": "Quotes"
        }
      ]
    },
    "product_bundle_notes": {
      "create": [
        {
          "description": "Free shipping",
          "position": 1
        }
      ]
    }
  },
  {
    "name": "Product Bundle 2",
    "bundle_stage": "Draft",
    "currency_id": ":-99",
    "base_rate": "1.0",
    "shipping": "25.00",
    "products": {
      "create": [
        {
          "quantity": 1000,
          "name": "Test Product 2",
          "description": "My Other Product",
          "mft_part_num": "mft100012234",
          "cost_price": "150.00",
          "list_price": "300.00",
          "discount_price": "275.00",
```

```
            "discount_amount": "0.00",
            "discount_select": 0,
            "product_template_id": "",
            "type_id": "",
            "status": "Quotes"
          },
          {
            "quantity": 500,
            "name": "Test Product 3",
            "description": "My Other Other Product",
            "mft_part_num": "mft100012123",
            "cost_price": "10.00",
            "list_price": "500.00",
            "discount_price": "400.00",
            "discount_amount": "0.00",
            "discount_select": 0,
            "product_template_id": "",
            "type_id": "",
            "status": "Quotes"
          }
        ]
      }
    }
  ]
}
}
```

Response

The data received from the server is shown below:

```
  {
  "id": "ee1e1ae8-372a-11e7-8bf4-3c15c2c94fb0",
  "name": "Test Quote",
  "date_entered": "2017-05-12T11:51:57-04:00",
  "date_modified": "2017-05-12T11:51:57-04:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "modified_user_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
      "fields": {
        "pwd_last_changed": {
```

```
        "write": "no",
        "create": "no"
      },
      "last_login": {
        "write": "no",
        "create": "no"
      }
    },
    "delete": "no",
    "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
  }
},
"created_by": "1",
"created_by_name": "Administrator",
"created_by_link": {
  "full_name": "Administrator",
  "id": "1",
  "_acl": {
    "fields": {
      "pwd_last_changed": {
        "write": "no",
        "create": "no"
      },
      "last_login": {
        "write": "no",
        "create": "no"
      }
    },
    "delete": "no",
    "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
  }
},
"description": "",
"deleted": false,
"shipper_id": "",
"shipper_name": "",
"shippers": {
  "name": "",
  "id": "",
  "_acl": {
    "fields": [

    ],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
```

```
"taxrate_id": "",
"taxrate_name": "",
"taxrates": {
  "name": "",
  "id": "",
  "_acl": {
    "fields": [

    ],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"taxrate_value": "0.000000",
"show_line_nums": true,
"quote_type": "Quotes",
"date_quote_expected_closed": "2017-06-12",
"original_po_date": "",
"payment_terms": "",
"date_quote_closed": "",
"date_order_shipped": "",
"order_stage": "",
"quote_stage": "Draft",
"purchase_order_num": "",
"quote_num": 2,
"subtotal": "650000.000000",
"subtotal_usdollar": "650000.000000",
"shipping": "0.000000",
"shipping_usdollar": "0.000000",
"discount": "",
"deal_tot": "0.00",
"deal_tot_discount_percentage": "0.00",
"deal_tot_usdollar": "0.00",
"new_sub": "650000.000000",
"new_sub_usdollar": "650000.000000",
"taxable_subtotal": "650000.000000",
"tax": "0.000000",
"tax_usdollar": "0.000000",
"total": "650000.000000",
"total_usdollar": "650000.000000",
"billing_address_street": "",
"billing_address_city": "",
"billing_address_state": "",
"billing_address_postalcode": "",
"billing_address_country": "",
"shipping_address_street": "",
"shipping_address_city": "",
```

```
"shipping_address_state": "",
"shipping_address_postalcode": "",
"shipping_address_country": "",
"system_id": 1,
"shipping_account_name": "",
"shipping_accounts": {
  "name": "",
  "id": "",
  "_acl": {
    "fields": [

    ],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"shipping_account_id": "",
"shipping_contact_name": "",
"shipping_contacts": {
  "full_name": "",
  "id": "",
  "_acl": {
    "fields": [

    ],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  },
  "last_name": ""
},
"shipping_contact_id": "",
"account_name": "",
"billing_accounts": {
  "name": "",
  "id": "",
  "_acl": {
    "fields": [

    ],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"account_id": "",
"billing_account_name": "",
"billing_account_id": "",
"billing_contact_name": "",
"billing_contacts": {
  "full_name": "",
```

```
    "id": "",
    "_acl": {
      "fields": [

      ],
      "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    },
    "last_name": ""
  },
  "billing_contact_id": "",
  "opportunity_name": "",
  "opportunities": {
    "name": "",
    "id": "",
    "_acl": {
      "fields": [

      ],
      "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
  },
  "opportunity_id": "",
  "following": "",
  "my_favorite": false,
  "tag": [

  ],
  "locked_fields": [

  ],
  "assigned_user_id": "",
  "assigned_user_name": "",
  "assigned_user_link": {
    "full_name": "",
    "id": "",
    "_acl": {
      "fields": [

      ],
      "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
  },
  "team_count": "",
  "team_count_link": {
    "team_count": "",
    "id": "1",
```

```
    "_acl": {
      "fields": [

      ],
      "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
  },
  "team_name": [
    {
      "id": "a0512788-3680-11e7-b42f-3c15c2c94fb0",
      "name": "Administrator",
      "name_2": "",
      "primary": false,
      "selected": false
    },
    {
      "id": "1",
      "name": "Global",
      "name_2": "",
      "primary": true,
      "selected": false
    }
  ],
  "currency_id": "-99",
  "base_rate": "1.000000",
  "currency_name": "",
  "currencies": {
    "name": "",
    "id": "-99",
    "_acl": {
      "fields": [

      ],
      "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    },
    "symbol": ""
  },
  "currency_symbol": "",
  "_acl": {
    "fields": {

    }
  },
  "_module": "Quotes"
}
```

You might notice that the Response does not contain the related data. To view the related data use the <module>/<record_id>/link/<link_name> - GET Endpoint.

Modifying the Quote's Product Bundles

Once the quote is created, you might need to add or remove Product Bundles from the Quote. This can be done using the /<module>/<record> - PUT endpoint.

```
  //Create a new ProductBundle
$url = $instance_url . "/ProductBundles";
$productBundle = array(
        "name" => "Product Bundle 3",
        "bundle_stage" => "Draft",
        "currency_id" => ":-99",
        "base_rate" => "1.0",
        "shipping" => "0.00",
        "products" => array(
            //Create Product in Bundle 3
            "create" => array(
                array(
                    "tax_class" => "Taxable",
                    "quantity" => 100.00,
                    "name" => "Test Product 3",
                    "description" => "Test Product 3",
                    "mft_part_num" => "mft100012021",
                    "cost_price" =>  "100.00",
                    "list_price" =>  "250.00",
                    "discount_price" =>  "175.00",
                    "discount_amount" =>  "0.00",
                    "discount_select" =>  0,
                    "product_template_id" =>  "",
                    "type_id"  =>  "",
                    "status"  =>  "Quotes",
                    "position" => 0
                )
            )
        ),
        //Create Product Bundle Note in Bundle 3
        "product_bundle_notes" => array(
            "create" => array(
                array(
                    "description" => "Free shipping",
                    "position" => 1
                )
            )
        )
```

```
    );
$curl_request = curl_init($url);
curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($curl_request, CURLOPT_HEADER, false);
curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($curl_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//convert arguments to json
$json_arguments = json_encode($productBundle);
curl_setopt($curl_request, CURLOPT_POSTFIELDS, $json_arguments);
//execute request
$curl_response = curl_exec($curl_request);
//decode json
$createdBundle = json_decode($curl_response);

//display the created record
print_r($createdBundle);
curl_close($curl_request);

//Add Bundle to Previously Created Quote
//PUT to /Quotes/<record_id>
$url = $instance_url . "/Quotes/".$createdQuote->id;
$quote = array(
    'product_bundles' => array(
        'delete' => array(
            'some_bundle_id'
        ),
        'add' => array(
            $createdBundle->id
        )
    )
);

$curl_request = curl_init($url);
curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($curl_request, CURLOPT_HEADER, false);
curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);
```

```
curl_setopt($curl_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//convert arguments to json
$json_arguments = json_encode($quote);
curl_setopt($curl_request, CURLOPT_POSTFIELDS, $json_arguments);
//PUT Request
curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "PUT");
//execute request
$curl_response = curl_exec($curl_request);
//decode json
$updatedQuote = json_decode($curl_response);

//display the updated quote record
print_r($updatedQuote);
curl_close($curl_request);
```

The above script removes a previously related Product Bundle from the Quote and adds the Product Bundle that was created before it in the script.

Request Payload

The data sent to the server to alter the Quotes Product Bundles is shown below:

```
{
  "product_bundles": {
    "delete": [
      "some_bundle_id"
    ],
    "add": [
      "803972ea-3741-11e7-8edc-3c15c2c94fb0"
    ]
  }
}
```

Response Payload

The response payload will match the standard /<module>/<record> - PUT Endpoint Response which is the entire values of the updated record. The previous Response for Creating the quote is the same as shown above.

Download

You can download the full API example here.

How to Manipulate Tags (CRUD)

Overview

A PHP example demonstrating how to work with tags using the v11
REST endpoints.

Authentication

First, you will need to authenticate to the Sugar API. An example is shown below:

```
  <?php

$instance_url = "http://{site_url}/rest/v11";
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
    //It is recommended to create your own in Admin > OAuth Keys
    "client_id" => "sugar",
    "client_secret" => "",
    "username" => $username,
    "password" => $password,
    //platform type - default is base.
    //It is recommend to change the platform to a custom name such as
"custom_api" to avoid authentication conflicts.
    "platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
```

```
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout POST endpoint documentation.

Creating Tags

Once you get oauth_token you would need to use it in the following API Calls to create tags.

```
  //Create Tags - /Tags POST
$url = $instance_url . "/Tags";
//Set up the tag name
$record = array(
    'name' => 'Tag Name',
);

$curl_request = curl_init($url);
curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($curl_request, CURLOPT_HEADER, false);
curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($curl_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));
```

```
//convert arguments to json
$json_arguments = json_encode($record);
curl_setopt($curl_request, CURLOPT_POSTFIELDS, $json_arguments);
//execute request
$curl_response = curl_exec($curl_request);
//decode json
$createdRecord = json_decode($curl_response);

//display the created record
print_r($createdRecord);
curl_close($curl_request);
```

More information on this API endpoint can be found in the /<module> POST documentation.

Request Payload

```
{"name":"Tag Name"}
```

Response

```
{
  "id": "12c6ee48-1000-11e8-8838-6a0001bcacb0",
  "name": "Tag Name",
  "date_entered": "2018-02-12T15:21:52+01:00",
  "date_modified": "2018-02-12T15:21:52+01:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "modified_user_link": {
      "full_name": "Administrator",
      "id": "1",
      "_acl": {
          "fields": {
              "pwd_last_changed": { "write": "no", "create": "no" },
              "last_login": { "write": "no", "create": "no" }
          },
          "delete": "no",
          "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
      }
  },
  "created_by": "1",
  "created_by_name": "Administrator",
```

```
    "created_by_link": {
        "full_name": "Administrator",
        "id": "1",
        "_acl": {
            "fields": {
                "pwd_last_changed": { "write": "no", "create": "no" },
                "last_login": { "write": "no", "create": "no" }
            },
            "delete": "no",
            "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
        }
    },
    "description": "",
    "deleted": false,
    "name_lower": "tag name",
    "following": "",
    "my_favorite": false,
    "locked_fields": [],
    "source_id": "",
    "source_type": "",
    "source_meta": "",
    "assigned_user_id": "1",
    "assigned_user_name": "Administrator",
    "assigned_user_link": {
        "full_name": "Administrator",
        "id": "1",
        "_acl": {
            "fields": {
                "pwd_last_changed": { "write": "no", "create": "no" },
                "last_login": { "write": "no", "create": "no" }
            },
            "delete": "no",
            "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
        }
    },
    "_acl": { "fields": {} },
    "_module": "Tags"
}
```


Creating Records with Tags

You can also create tags when creating new records. All you need to do populate the tag field as an array when creating a record. Here is an example that demonstrates using the /<module> POST endpoint.

```
  //Create Records with Tags - / POST
$url = $instance_url . "/Accounts";
//Set up the Record details with Tags
$record = array(
    'name' => 'Test Record',
    'tag' => array(
        'First Tag',
        'Second Tag'
    )
);

$curl_request = curl_init($url);
curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($curl_request, CURLOPT_HEADER, false);
curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($curl_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//convert arguments to json
$json_arguments = json_encode($record);
curl_setopt($curl_request, CURLOPT_POSTFIELDS, $json_arguments);
//execute request
$curl_response = curl_exec($curl_request);
//decode json
$createdRecord = json_decode($curl_response);

//display the created record
print_r($createdRecord);
curl_close($curl_request);
```

More information on this API endpoint can be found in the /<module> POST documentation.

Request Payload

The data sent to the server is shown below:

```
  {
    "name": "Test Record",
```

```
    "tag": [
        "First Tag",
        "Second Tag"
    ]
}
```

Response

The data sent to the server is shown below:

```
  {
    "id": "ea507760-0ffd-11e8-bcf5-6a0001bcacb0",
    "name": "Test Record",
    "date_entered": "2018-02-12T15:06:25+01:00",
    "date_modified": "2018-02-12T15:06:25+01:00",
    "modified_user_id": "1",
    "modified_by_name": "Administrator",
    "modified_user_link": {
        "full_name": "Administrator",
        "id": "1",
        "_acl": {
            "fields": {
                "pwd_last_changed": { "write": "no", "create": "no" },
                "last_login": { "write": "no", "create": "no" }
            },
            "delete": "no",
            "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
        }
    },
    "created_by": "1",
    "created_by_name": "Administrator",
    "created_by_link": {
        "full_name": "Administrator",
        "id": "1",
        "_acl": {
            "fields": {
                "pwd_last_changed": { "write": "no", "create": "no" },
                "last_login": { "write": "no", "create": "no" }
            },
            "delete": "no",
            "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
        }
    },
    "description": "",
```

```
    "deleted": false,
    "facebook": "",
    "twitter": "",
    "googleplus": "",
    "account_type": "",
    "industry": "",
    "annual_revenue": "",
    "phone_fax": "",
    "billing_address_street": "",
    "billing_address_street_2": "",
    "billing_address_street_3": "",
    "billing_address_street_4": "",
    "billing_address_city": "",
    "billing_address_state": "",
    "billing_address_postalcode": "",
    "billing_address_country": "",
    "rating": "",
    "phone_office": "",
    "phone_alternate": "",
    "website": "",
    "ownership": "",
    "employees": "",
    "ticker_symbol": "",
    "shipping_address_street": "",
    "shipping_address_street_2": "",
    "shipping_address_street_3": "",
    "shipping_address_street_4": "",
    "shipping_address_city": "",
    "shipping_address_state": "",
    "shipping_address_postalcode": "",
    "shipping_address_country": "",
    "parent_id": "",
    "sic_code": "",
    "duns_num": "",
    "parent_name": "",
    "member_of": {
        "name": "",
        "id": "",
        "_acl": { "fields": [], "_hash": "654d337e0e912edaa00dbb0fb3dc
3c17" }
    },
    "campaign_id": "",
    "campaign_name": "",
    "campaign_accounts": {
        "name": "",
        "id": "",
```

```
        "_acl": { "fields": [], "_hash": "654d337e0e912edaa00dbb0fb3dc
3c17" }
    },
    "following": true,
    "my_favorite": false,
    "tag": [
        {
            "id": "ea69c120-0ffd-11e8-b5f6-6a0001bcacb0",
            "name": "First Tag",
            "tags__name_lower": "first tag"
        },
        {
            "id": "eafb28e0-0ffd-11e8-8d80-6a0001bcacb0",
            "name": "Second Tag",
            "tags__name_lower": "second tag"
        }
    ],
    "locked_fields": [],
    "assigned_user_id": "",
    "assigned_user_name": "",
    "assigned_user_link": {
        "full_name": "",
        "id": "",
        "_acl": { "fields": [], "_hash": "654d337e0e912edaa00dbb0fb3dc
3c17" }
    },
    "team_count": "",
    "team_count_link": {
        "team_count": "",
        "id": "1",
        "_acl": { "fields": [], "_hash": "654d337e0e912edaa00dbb0fb3dc
3c17" }
    },
    "team_name": [
        {
            "id": "1",
            "name": "Global",
            "name_2": "",
            "primary": true,
            "selected": false
        }
    ],
    "email": [],
    "email1": "",
    "email2": "",
    "invalid_email": "",
```

```
    "email_opt_out": "",
    "email_addresses_non_primary": "",
    "calculated_c": "",
    "_acl": { "fields": {} },
    "_module": "Accounts"
}
```

Reading/Retrieving Tags

Next, we can retrieve the records using the /Tags/:record GET endpoint.
An example for retrieving the tag records is shown below.

```
  //Reading/Retrieving Tags - /Tags GET
$url = $instance_url . "/Tags/12c6ee48-1000-11e8-8838-6a0001bcacb0";

$curl_request = curl_init($url);
curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($curl_request, CURLOPT_HEADER, false);
curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($curl_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//convert arguments to json
$json_arguments = json_encode($record);
curl_setopt($curl_request, CURLOPT_POSTFIELDS, $json_arguments);
//execute request
$curl_response = curl_exec($curl_request);
//decode json
$createdRecord = json_decode($curl_response);

//display the created record
print_r($createdRecord);
curl_close($curl_request);
```

More information on this API endpoint can be found in the /<module>/:record GET documentation.

Request Payload

No payload is sent for this request.

Response


```
{
  "id": "12c6ee48-1000-11e8-8838-6a0001bcacb0",
  "name": "Tag Name",
  "date_entered": "2018-02-12T15:21:52+01:00",
  "date_modified": "2018-02-12T15:21:52+01:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "modified_user_link": {
      "full_name": "Administrator",
      "id": "1",
      "_acl": {
          "fields": {
              "pwd_last_changed": { "write": "no", "create": "no" },
              "last_login": { "write": "no", "create": "no" }
          },
          "delete": "no",
          "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
      }
  },
  "created_by": "1",
  "created_by_name": "Administrator",
  "created_by_link": {
      "full_name": "Administrator",
      "id": "1",
      "_acl": {
          "fields": {
              "pwd_last_changed": { "write": "no", "create": "no" },
              "last_login": { "write": "no", "create": "no" }
          },
          "delete": "no",
          "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
      }
  },
  "description": "",
  "deleted": false,
  "name_lower": "tag name",
  "following": "",
  "my_favorite": false,
```

```
    "locked_fields": [],
    "source_id": "",
    "source_type": "",
    "source_meta": "",
    "assigned_user_id": "1",
    "assigned_user_name": "Administrator",
    "assigned_user_link": {
        "full_name": "Administrator",
        "id": "1",
        "_acl": {
            "fields": {
                "pwd_last_changed": { "write": "no", "create": "no" },
                "last_login": { "write": "no", "create": "no" }
            },
            "delete": "no",
            "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
        }
    },
    "_acl": { "fields": {} },
    "_module": "Tags"
}
```

Updating Tags

You can update a tag using the /Tags/:record PUT endpoint. In this example, we are going to update the Tag record that we created in Creating Tags section and change its name to "Renamed Tag Name".

```
  //Update Tags - /Tags PUT
$url = $instance_url . "/Tags/12c6ee48-1000-11e8-8838-6a0001bcacb0";
//Set up the new tag name
$record = array(
    'name' => 'Renamed Tag Name',
);

$curl_request = curl_init($url);
curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "PUT");
curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($curl_request, CURLOPT_HEADER, false);
curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($curl_request, CURLOPT_HTTPHEADER, array(
```

```
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//convert arguments to json
$json_arguments = json_encode($record);
curl_setopt($curl_request, CURLOPT_POSTFIELDS, $json_arguments);
//execute request
$curl_response = curl_exec($curl_request);
//decode json
$updatedRecord = json_decode($curl_response);

//display the created record
echo "Updated Record Name:" . $updatedRecord->name;
curl_close($curl_request);
```

More information on this API endpoint can be found in the /<module>/:record PUT documentation.

Request Payload

```
{"name": "Renamed Tag Name"}
```

Response

```
{
  "id": "12c6ee48-1000-11e8-8838-6a0001bcacb0",
  "name": "Renamed Tag Name",
  "date_entered": "2018-02-12T15:21:52+01:00",
  "date_modified": "2018-02-12T16:07:18+01:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "modified_user_link": {
      "full_name": "Administrator",
      "id": "1",
      "_acl": {
          "fields": {
              "pwd_last_changed": { "write": "no", "create": "no" },
              "last_login": { "write": "no", "create": "no" }
          },
          "delete": "no",
          "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
```

```
        }
    },
    "created_by": "1",
    "created_by_name": "Administrator",
    "created_by_link": {
        "full_name": "Administrator",
        "id": "1",
        "_acl": {
            "fields": {
                "pwd_last_changed": { "write": "no", "create": "no" },
                "last_login": { "write": "no", "create": "no" }
            },
            "delete": "no",
            "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
        }
    },
    "description": "",
    "deleted": false,
    "name_lower": "renamed tag name",
    "following": "",
    "my_favorite": false,
    "locked_fields": [],
    "source_id": "",
    "source_type": "",
    "source_meta": "",
    "assigned_user_id": "1",
    "assigned_user_name": "Administrator",
    "assigned_user_link": {
        "full_name": "Administrator",
        "id": "1",
        "_acl": {
            "fields": {
                "pwd_last_changed": { "write": "no", "create": "no" },
                "last_login": { "write": "no", "create": "no" }
            },
            "delete": "no",
            "_hash": "08b99a97c2e8d792f7a44d8882b5af6d"
        }
    },
    "_acl": { "fields": {} },
    "_module": "Tags"
}
```


Deleting Tags

Finally, we can delete the record we created using the /Tags/:record DELETE API request.

```
  //Delete Tags - /Tags DELETE
$url = $instance_url . "/Tags/12c6ee48-1000-11e8-8838-6a0001bcacb0";

$curl_request = curl_init($url);
curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "DELETE");
curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_0
);
curl_setopt($curl_request, CURLOPT_HEADER, false);
curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($curl_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//execute request
$curl_response = curl_exec($curl_request);
//decode json
$deletedRecord = json_decode($curl_response);

//display the created record
echo "Deleted Record:" . $deletedRecord->id;
curl_close($curl_request);
```

More information on this API endpoint can be found in the /<module>/:record DELETE documentation.

Request Payload

```
  No payload is sent for this request.
```

Response

```
  {"id":"12c6ee48-1000-11e8-8838-6a0001bcacb0"}
```

Legacy API

Examples of the legacy v4_1 web service endpoints.

REST

Examples of v4.1 REST API calls.

PHP

PHP v4_1 REST Examples.

Creating Documents

Overview

A PHP example demonstrating how to create a document using set_entry and a document revision with the set_document_revision method using cURL and the v4_1 REST API.

Example

```
<?php

    $url = "http://{site_url}/service/v4_1/rest.php";

    $username = "admin";
    $password = "password";

    //function to make cURL request
    function call($method, $parameters, $url)
    {
        ob_start();
        $curl_request = curl_init();

        curl_setopt($curl_request, CURLOPT_URL, $url);
        curl_setopt($curl_request, CURLOPT_POST, 1);
        curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VER
SION_1_0);
        curl_setopt($curl_request, CURLOPT_HEADER, 1);
        curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
        curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
```

```
    curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

    $jsonEncodedData = json_encode($parameters);

    $post = array(
         "method" => $method,
         "input_type" => "JSON",
         "response_type" => "JSON",
         "rest_data" => $jsonEncodedData
    );

    curl_setopt($curl_request, CURLOPT_POSTFIELDS, $post);
    $result = curl_exec($curl_request);
    curl_close($curl_request);

    $result = explode("\r\n\r\n", $result, 2);
    $response = json_decode($result[1]);
    ob_end_flush();

    return $response;
}

//login ----------------------------------------------------
$login_parameters = array(
     "user_auth" => array(
          "user_name" => $username,
          "password" => md5($password),
          "version" => "1"
     ),
     "application_name" => "RestTest",
     "name_value_list" => array(),
);

$login_result = call("login", $login_parameters, $url);

/*
echo "<pre>";
print_r($login_result);
echo "</pre>";
*/

//get session id
$session_id = $login_result->id;

//create document ---------------------------------------------
$set_entry_parameters = array(
```

```
        //session id
        "session" => $session_id,

        //The name of the module
        "module_name" => "Documents",

        //Record attributes
        "name_value_list" => array(
            //to update a record, pass in a record id as commented bel
ow
            //array("name" => "id", "value" => "9b170af9-3080-e22b-
fbc1-4fea74def88f"),
            array("name" => "document_name", "value" => "Example Docum
ent"),
            array("name" => "revision", "value" => "1"),
        ),
    );

    $set_entry_result = call("set_entry", $set_entry_parameters, $url)
;

    echo "<pre>";
    print_r($set_entry_result);
    echo "</pre>";

    $document_id = $set_entry_result->id;

    //create document revision -------------------------------------
    $contents = file_get_contents ("/path/to/example_document.txt");

    $set_document_revision_parameters = array(
        //session id
        "session" => $session_id,

        //The attachment details
        "note" => array(
            //The ID of the parent document.
            'id' => $document_id,

            //The binary contents of the file.
            'file' => base64_encode($contents),

            //The name of the file
            'filename' => 'example_document.txt',

            //The revision number
```

```
            'revision' => '1',
        ),
    );

    $set_document_revision_result = call("set_document_revision", $set
_document_revision_parameters, $url);

    echo "<pre>";
    print_r($set_document_revision_result);
    echo "</pre>";

?>
```

Result

```
//set_entry result
stdClass Object
(
    [id] => b769cf46-7881-a369-314d-50abaa238c62
    [entry_list] => stdClass Object
        (
            [document_name] => stdClass Object
                (
                    [name] => document_name
                    [value] => Example Document
                )

            [revision] => stdClass Object
                (
                    [name] => revision
                    [value] => 1
                )

        )

)

//set_document_revision result
stdClass Object
(
    [id] => e83f97b9-b818-2d04-1aeb-50abaa8303b5
)
```

Creating Notes with Attachments

Overview

A PHP example demonstrating how to create a note using set_entry and add an attachment with the set_note_attachment method using cURL and the v4_1 REST API.

Example

```
<?php

    $url = "http://{site_url}/service/v4_1/rest.php";
    $username = "admin";
    $password = "password";

    //function to make cURL request
    function call($method, $parameters, $url)
    {
        ob_start();
        $curl_request = curl_init();

        curl_setopt($curl_request, CURLOPT_URL, $url);
        curl_setopt($curl_request, CURLOPT_POST, 1);
        curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VER
SION_1_0);
        curl_setopt($curl_request, CURLOPT_HEADER, 1);
        curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
        curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
        curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

        $jsonEncodedData = json_encode($parameters);

        $post = array(
             "method" => $method,
             "input_type" => "JSON",
             "response_type" => "JSON",
             "rest_data" => $jsonEncodedData
        );

        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $post);
        $result = curl_exec($curl_request);
```

```
        curl_close($curl_request);

        $result = explode("\r\n\r\n", $result, 2);
        $response = json_decode($result[1]);
        ob_end_flush();

        return $response;
    }

    //login ----------------------------------------------------

    $login_parameters = array(
         "user_auth" => array(
              "user_name" => $username,
              "password" => md5($password),
              "version" => "1"
         ),
         "application_name" => "RestTest",
         "name_value_list" => array(),
    );

    $login_result = call("login", $login_parameters, $url);

    /*
    echo "<pre>";
    print_r($login_result);
    echo "</pre>";
    */

    //get session id
    $session_id = $login_result->id;

    //create note ----------------------------------------------

    $set_entry_parameters = array(
         //session id
         "session" => $session_id,

         //The name of the module
         "module_name" => "Notes",

         //Record attributes
         "name_value_list" => array(
              //to update a record, you will nee to pass in a record i
d as commented below
              //array("name" => "id", "value" => "9b170af9-3080-e22b-
```

```
fbc1-4fea74def88f"),
              array("name" => "name", "value" => "Example Note"),
         ),
    );

    $set_entry_result = call("set_entry", $set_entry_parameters, $url)
;

    echo "<pre>";
    print_r($set_entry_result);
    echo "</pre>";

    $note_id = $set_entry_result->id;

    //create note attachment -----------------------------------

    $contents = file_get_contents ("/path/to/example_file.php");

    $set_note_attachment_parameters = array(
        //session id
        "session" => $session_id,

        //The attachment details
        "note" => array(
            //The ID of the note containing the attachment.
            'id' => $note_id,

            //The file name of the attachment.
            'filename' => 'example_file.php',

            //The binary contents of the file.
            'file' => base64_encode($contents),
        ),
    );

    $set_note_attachment_result = call("set_note_attachment", $set_not
e_attachment_parameters, $url);

    echo "<pre>";
    print_r($set_note_attachment_result);
    echo "</pre>";

?>
```

Result

```
//set_entry result
stdClass Object
(
    [id] => 72508938-db19-3b5c-b7a8-50abc7ec3fdb
    [entry_list] => stdClass Object
        (
            [name] => stdClass Object
                (
                    [name] => name
                    [value] => Example Note
                )

        )

)

//set_note_attachment result
stdClass Object
(
    [id] => 72508938-db19-3b5c-b7a8-50abc7ec3fdb
)
```

Creating or Updating a Record

Overview

A PHP example demonstrating how to create or update an Account with the set_entry method using cURL and the v4_1 REST API.

Example

```
<?php

    $url = "http://{site_url}/service/v4_1/rest.php";
    $username = "admin";
    $password = "password";

    //function to make cURL request
    function call($method, $parameters, $url)
    {
        ob_start();
```

```
        $curl_request = curl_init();

        curl_setopt($curl_request, CURLOPT_URL, $url);
        curl_setopt($curl_request, CURLOPT_POST, 1);
        curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VER
SION_1_0);
        curl_setopt($curl_request, CURLOPT_HEADER, 1);
        curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
        curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
        curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

        $jsonEncodedData = json_encode($parameters);

        $post = array(
             "method" => $method,
             "input_type" => "JSON",
             "response_type" => "JSON",
             "rest_data" => $jsonEncodedData
        );

        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $post);
        $result = curl_exec($curl_request);
        curl_close($curl_request);

        $result = explode("\r\n\r\n", $result, 2);
        $response = json_decode($result[1]);
        ob_end_flush();

        return $response;
    }

    //login ------------------------------------------

    $login_parameters = array(
         "user_auth" => array(
              "user_name" => $username,
              "password" => md5($password),
              "version" => "1"
         ),
         "application_name" => "RestTest",
         "name_value_list" => array(),
    );

    $login_result = call("login", $login_parameters, $url);

    /*
```

```
    echo "<pre>";
    print_r($login_result);
    echo "</pre>";
    */

    //get session id
    $session_id = $login_result->id;

    //create account -------------------------------------

    $set_entry_parameters = array(
         //session id
         "session" => $session_id,

         //The name of the module from which to retrieve records.
         "module_name" => "Accounts",

         //Record attributes
         "name_value_list" => array(
              //to update a record, you will nee to pass in a record i
d as commented below
              //array("name" => "id", "value" => "9b170af9-3080-e22b-
fbc1-4fea74def88f"),
              array("name" => "name", "value" => "Test Account"),
         ),
    );

    $set_entry_result = call("set_entry", $set_entry_parameters, $url)
;

    echo "<pre>";
    print_r($set_entry_result);
    echo "</pre>";

?>
```

Result

```
stdClass Object
(
    [id] => 9b170af9-3080-e22b-fbc1-4fea74def88f
    [entry_list] => stdClass Object
        (
            [name] => stdClass Object
```

```
                (
                    [name] => name
                    [value] => Test Account
                )

        )

)
```

Creating or Updating Multiple Records

Overview

A PHP example demonstrating how to create or update multiple contacts with the set_entries method using cURL and the v4_1 REST API.

Example

```
<?php

    $url = "http://{site_url}/service/v4_1/rest.php";
    $username = "admin";
    $password = "password";

    //function to make cURL request
    function call($method, $parameters, $url)
    {
        ob_start();
        $curl_request = curl_init();

        curl_setopt($curl_request, CURLOPT_URL, $url);
        curl_setopt($curl_request, CURLOPT_POST, 1);
        curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VER
SION_1_0);
        curl_setopt($curl_request, CURLOPT_HEADER, 1);
        curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
        curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
        curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

        $jsonEncodedData = json_encode($parameters);

        $post = array(
             "method" => $method,
             "input_type" => "JSON",
```

```
        "response_type" => "JSON",
        "rest_data" => $jsonEncodedData
    );

    curl_setopt($curl_request, CURLOPT_POSTFIELDS, $post);
    $result = curl_exec($curl_request);
    curl_close($curl_request);

    $result = explode("\r\n\r\n", $result, 2);
    $response = json_decode($result[1]);
    ob_end_flush();

    return $response;
}

//login ------------------------------------------
$login_parameters = array(
     "user_auth" => array(
          "user_name" => $username,
          "password" => md5($password),
          "version" => "1"
     ),
     "application_name" => "RestTest",
     "name_value_list" => array(),
);

$login_result = call("login", $login_parameters, $url);

/*
echo "<pre>";
print_r($login_result);
echo "</pre>";
*/

//get session id
$session_id = $login_result->id;

//create contacts ----------------------------------
$set_entries_parameters = array(
     //session id
     "session" => $session_id,

     //The name of the module from which to retrieve records.
     "module_name" => "Contacts",

     //Record attributes
```

```
        "name_value_list" => array(
            array(
               //to update a record, you will nee to pass in a record
 id as commented below
               //array("name" => "id", "value" => "912e58c0-73e9-9cb6
-c84e-4ff34d62620e"),
               array("name" => "first_name", "value" => "John"),
               array("name" => "last_name", "value" => "Smith"),
            ),
            array(
               //to update a record, you will nee to pass in a record
 id as commented below
               //array("name" => "id", "value" => "99d6ddfd-7d52-d45b-
eba8-4ff34d684964"),
               array("name" => "first_name", "value" => "Jane"),
               array("name" => "last_name", "value" => "Doe"),
            ),
        ),
    );

    $set_entries_result = call("set_entries", $set_entries_parameters,
 $url);

    echo "<pre>";
    print_r($set_entries_result);
    echo "</pre>";

?>
```

Result

```
stdClass Object
(
    [ids] => Array
        (
            [0] => 912e58c0-73e9-9cb6-c84e-4ff34d62620e
            [1] => 99d6ddfd-7d52-d45b-eba8-4ff34d684964
        )

)
```

Creating or Updating Teams

Overview

A PHP example demonstrating how to manipulate teams using cURL and the v4_1 REST API.

Note: If you are creating a private team for a user, you will need to set private to true and populate the associated_user_id populated. You should also populate the name and name_2 properties with the users first and last name.

Example

```
<?php

    $url = "http://{site_url}/service/v4_1/rest.php";
    $username = "admin";
    $password = "password";

    //function to make cURL request
    function call($method, $parameters, $url)
    {
        ob_start();
        $curl_request = curl_init();

        curl_setopt($curl_request, CURLOPT_URL, $url);
        curl_setopt($curl_request, CURLOPT_POST, 1);
        curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VER
SION_1_0);
        curl_setopt($curl_request, CURLOPT_HEADER, 1);
        curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
        curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
        curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

        $jsonEncodedData = json_encode($parameters);

        $post = array(
            "method" => $method,
            "input_type" => "JSON",
            "response_type" => "JSON",
            "rest_data" => $jsonEncodedData
        );

        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $post);
```

```
    $result = curl_exec($curl_request);
    curl_close($curl_request);

    $result = explode("\r\n\r\n", $result, 2);
    $response = json_decode($result[1]);
    ob_end_flush();

    return $response;
}

//login ------------------------------------------

$login_parameters = array(
    "user_auth" => array(
        "user_name" => $username,
        "password" => md5($password),
        "version" => "1"
    ),
    "application_name" => "RestTest",
    "name_value_list" => array(),
);

$login_result = call("login", $login_parameters, $url);

/*
echo "<pre>";
print_r($login_result);
echo "</pre>";
*/

//get session id
$session_id = $login_result->id;

//create team ------------------------------------------------

$set_entry_parameters = array(

    // session id
    "session" => $session_id,

    // The name of the module that the record will be create in.
    "module_name" => "Teams",

    // array of arrays for the record attributes
    "name_value_list" => array(
        /* Setting the id with a valid record id will update the r
```

```
ecord.
            array(
                "name" => "id",
                "value" => "47dbab1d-bd78-09e8-4392-5256b4501d90"
            ),
            */
            array(
                "name" => "name",
                "value" => "My Team"
            ),
            array(
                "name" => "description",
                "value" => "My new team"
            ),
            //Whether the team is private.
            //Private teams will have the associated_user_id populated
.
            array(
                "name" => "private",
                "value" => 0
            ),
        ),
    );

    $set_entry_result = call("set_entry", $set_entry_parameters, $url)
;

    echo "<pre>";
    print_r($set_entry_result);
    echo "</pre>";

?>
```

Result

```
stdClass Object
(
    [id] => 5c35a3be-4601-fb45-3afd-52ab78b03f89
    [entry_list] => stdClass Object
        (
            [name] => stdClass Object
                (
                    [name] => name
```

```
                    [value] => My Team
                )

            [description] => stdClass Object
                (
                    [name] => description
                    [value] => My new team
                )

            [private] => stdClass Object
                (
                    [name] => private
                    [value] =>
                )

        )

)
```

Logging In

Overview

A PHP example demonstrating how to log in and retrieve a session key using cURL and the v4_1 REST API.

Standard Authentication Example

```
<?php

    $url = "http://{site_url}/service/v4_1/rest.php";
    $username = "admin";
    $password = "password";

    //function to make cURL request
    function call($method, $parameters, $url)
    {
        ob_start();
        $curl_request = curl_init();

        curl_setopt($curl_request, CURLOPT_URL, $url);
        curl_setopt($curl_request, CURLOPT_POST, 1);
        curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VER
```

```
SION_1_0);
        curl_setopt($curl_request, CURLOPT_HEADER, 1);
        curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
        curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
        curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

        $jsonEncodedData = json_encode($parameters);

        $post = array(
             "method" => $method,
             "input_type" => "JSON",
             "response_type" => "JSON",
             "rest_data" => $jsonEncodedData
        );

        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $post);
        $result = curl_exec($curl_request);
        curl_close($curl_request);

        $result = explode("\r\n\r\n", $result, 2);
        $response = json_decode($result[1]);
        ob_end_flush();

        return $response;
    }

    //login ------------------------------
    $login_parameters = array(
         "user_auth" => array(
              "user_name" => $username,
              "password" => md5($password),
              "version" => "1"
         ),
         "application_name" => "RestTest",
         "name_value_list" => array(),
    );

    $login_result = call("login", $login_parameters, $url);

    echo "<pre>";
    print_r($login_result);
    echo "</pre>";

    //get session id
    $session_id = $login_result->id;
```

```
?>
```

LDAP Authentication Example

```
<?php

    $url = "http://{site_url}/service/v4_1/rest.php";
    $username = "admin";
    $password = "password";
    $ldap_enc_key = 'LDAP_ENCRYPTION_KEY';

    //function to make cURL request
    function call($method, $parameters, $url)
    {
        ob_start();
        $curl_request = curl_init();

        curl_setopt($curl_request, CURLOPT_URL, $url);
        curl_setopt($curl_request, CURLOPT_POST, 1);
        curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VER
SION_1_0);
        curl_setopt($curl_request, CURLOPT_HEADER, 1);
        curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
        curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
        curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

        $jsonEncodedData = json_encode($parameters);

        $post = array(
             "method" => $method,
             "input_type" => "JSON",
             "response_type" => "JSON",
             "rest_data" => $jsonEncodedData
        );

        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $post);
        $result = curl_exec($curl_request);
        curl_close($curl_request);

        $result = explode("\r\n\r\n", $result, 2);
        $response = json_decode($result[1]);
        ob_end_flush();

        return $response;
```

```
    }

    //login ---------------------------------
    $ldap_enc_key = substr(md5($ldap_enc_key), 0, 24);
    $login_parameters = array(
         "user_auth" => array(
              "user_name" => $username,
              "password" => bin2hex(mcrypt_encrypt(MCRYPT_3DES, $ldap_
enc_key, $password, MCRYPT_MODE_CBC, "password")),
              "version" => "1"
         ),
         "application_name" => "RestTest",
         "name_value_list" => array(),
    );

    /*
        The mcrypt extension is deprecated as of PHP 7.1.
        If on 7.1+, LDAP passwords will need to be sent unencrypted, w
ith
        $login_parameters["user_auth"]["encryption"] set to "PLAIN".
        An example is below:
        $login_parameters = array(
            "user_auth" => array(
                "user_name" => $username,
                "password" => $password,
                "version" => "1",
                "encryption" => "PLAIN"
            ),
            "application_name" => "RestTest",
            "name_value_list" => array(),
        );
    */

    $login_result = call("login", $login_parameters, $url);

    echo "<pre>";
    print_r($login_result);
    echo "</pre>";

    //get session id
    $session_id = $login_result->id;

?>
```

Result

```
stdClass Object
(
    [id] => lb7479svj8pjtf57ipmshepo80
    [module_name] => Users
    [name_value_list] => stdClass Object
        (
            [user_id] => stdClass Object
                (
                    [name] => user_id
                    [value] => 1
                )

            [user_name] => stdClass Object
                (
                    [name] => user_name
                    [value] => admin
                )

            [user_language] => stdClass Object
                (
                    [name] => user_language
                    [value] => en_us
                )

            [user_currency_id] => stdClass Object
                (
                    [name] => user_currency_id
                    [value] => -99
                )

            [user_is_admin] => stdClass Object
                (
                    [name] => user_is_admin
                    [value] => 1
                )

            [user_default_team_id] => stdClass Object
                (
                    [name] => user_default_team_id
                    [value] => 1
                )

            [user_default_dateformat] => stdClass Object
```

```
            (
                [name] => user_default_dateformat
                [value] => m/d/Y
            )

        [user_default_timeformat] => stdClass Object
            (
                [name] => user_default_timeformat
                [value] => h:ia
            )

        [user_number_seperator] => stdClass Object
            (
                [name] => user_number_seperator
                [value] => ,
            )

        [user_decimal_seperator] => stdClass Object
            (
                [name] => user_decimal_seperator
                [value] => .
            )

        [mobile_max_list_entries] => stdClass Object
            (
                [name] => mobile_max_list_entries
                [value] => 10
            )

        [mobile_max_subpanel_entries] => stdClass Object
            (
                [name] => mobile_max_subpanel_entries
                [value] => 3
            )

        [user_currency_name] => stdClass Object
            (
                [name] => user_currency_name
                [value] => US Dollars
            )

    )

)
```

Relating Quotes and Products

Overview

A PHP example demonstrating how to create and relate Products to Quotes using cURL and the v4_1 REST API.

Example

```
<?php

    $url = "http://{site_url}/service/v4_1/rest.php";
    $username = "admin";
    $password = "password";

    //function to make cURL request
    function call($method, $parameters, $url)
    {
        ob_start();
        $curl_request = curl_init();

        curl_setopt($curl_request, CURLOPT_URL, $url);
        curl_setopt($curl_request, CURLOPT_POST, 1);
        curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VER
SION_1_0);
        curl_setopt($curl_request, CURLOPT_HEADER, 1);
        curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
        curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
        curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

        $jsonEncodedData = json_encode($parameters);

        $post = array(
            "method" => $method,
            "input_type" => "JSON",
            "response_type" => "JSON",
            "rest_data" => $jsonEncodedData
        );

        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $post);
        $result = curl_exec($curl_request);
        curl_close($curl_request);
```

```
    $result = explode("\r\n\r\n", $result, 2);
    $response = json_decode($result[1]);
    ob_end_flush();

    return $response;
}

//login -------------------------------------------
$login_parameters = array(
    "user_auth" => array(
        "user_name" => $username,
        "password" => md5($password),
        "version" => "1"
    ),
    "application_name" => "RestTest",
    "name_value_list" => array(),
);

$login_result = call("login", $login_parameters, $url);

/*
echo "<pre>";
print_r($login_result);
echo "</pre>";
*/

//get session id
$session_id = $login_result->id;

//create quote -------------------------------------------
$createQuoteParams = array(
    'session' => $session_id,
    'module_name' => 'Quotes',
    'name_value_list' => array(
        array(
            'name' => 'name',
            'value' => 'Widget Quote'
        ),
        array(
            'name' => 'team_count',
            'value' => ''
        ),
        array(
            'name' => 'team_name',
            'value' => ''
```

```
            ),
            array(
                'name' => 'date_quote_expected_closed',
                'value' => date('Y-m-
d', mktime(0, 0, 0, date('m')  , date('d')+7, date('Y')))
            ),
            array(
                'name' => 'quote_stage',
                'value' => 'Negotiation'
            ),
            array(
                'name' => 'quote_num',
                'value' => ''
            ),
            array(
                'name' => 'quote_type',
                'value' => 'Quotes'
            ),
            array(
                'name' => 'subtotal',
                'value' => '1230.23'
            ),
            array(
                'name' => 'subtotal_usdollar',
                'value' => '1230.23'
            ),
        ),
    );

    $createQuoteResult = call('set_entry', $createQuoteParams, $url);

    echo "Create Quote Result<br />";
    echo "<pre>";
    print_r($createQuoteResult);
    echo "</pre>";

    //create product ---------------------------------------------

    $createProductParams = array(
        'session' => $session_id,
        'module_name' => 'Products',
        'name_value_list' => array(
            array(
                'name' => 'name',
                'value' => 'Widget'
            ),
```

```
            array(
                'name' => 'quote_id',
                'value' => $createQuoteResult->id
            ),
            array(
                'name' => 'status',
                'value' => 'Quotes'
            )
        )
    );

    $createProductResult = call('set_entry', $createProductParams, $ur
l);

    echo "Create Product Result<br />";

    echo "<pre>";
    print_r($createProductResult);
    echo "</pre>";

    //create product-
bundle ----------------------------------------------

    $createProductBundleParams = array(
        "session"         => $session_id,
        "module_name"     => "ProductBundles",
        "name_value_list" => array(
            array(
                'name' => 'name',
                'value' => 'Rest SugarOnline Order'),
            array(
                'name' => 'bundle_stage',
                'value' => 'Draft'
            ),
            array(
                'name' => 'tax',
                'value' => '0.00'
            ),
            array(
                'name' => 'total',
                'value' => '0.00'
            ),
            array(
                'name' => 'subtotal',
                'value' => '0.00'
            ),
```

```
            array(
                'name' => 'shippint',
                'value' => '0.00'
            ),
            array(
                'name' => 'currency_id',
                'value' => '-99'
            ),
        )
    );

    $createProductBundleResult = call('set_entry', $createProductBundl
eParams, $url);

    echo "Create ProductBundles Result<br />";

    echo "<pre>";
    print_r($createProductBundleResult);
    echo "</pre>";

    //relate product to product-
bundle ----------------------------------------

    $relationshipProductBundleProductsParams = array(
        'sesssion' => $session_id,
        'module_name' => 'ProductBundles',
        'module_id' => $createProductBundleResult->id,
        'link_field_name' => 'products',
        'related_ids' => array(
            $createProductResult->id
        ),
    );

    // set the product bundles products relationship
    $relationshipProductBundleProductResult = call('set_relationship',
 $relationshipProductBundleProductsParams, $url);

    echo "Create ProductBundleProduct Relationship Result<br />";

    echo "<pre>";
    print_r($relationshipProductBundleProductResult);
    echo "</pre>";

    //relate product-
bundle to quote ---------------------------------------
```

```
    $relationshipProductBundleQuoteParams = array(
        'sesssion' => $session_id,
        'module_name' => 'Quotes',
        'module_id' => $createQuoteResult->id,
        'link_field_name' => 'product_bundles',
        'related_ids' => array(
            $createProductBundleResult->id
        ),
        'name_value_list' => array()
    );

    // set the product bundles quotes relationship
    $relationshipProductBundleQuoteResult = call('set_relationship', $
relationshipProductBundleQuoteParams, $url);

    echo "Create ProductBundleQuote Relationship Result<br />";

    echo "<pre>";
    print_r($relationshipProductBundleQuoteResult);
    echo "</pre>";
```

Result

```
//Create Quote Result
stdClass Object
(
    [id] => 2e0cd18b-21da-50f0-10f6-517e835a1e09
    [entry_list] => stdClass Object
        (
            [name] => stdClass Object
                (
                    [name] => name
                    [value] => Widget Quote
                )

            [team_count] => stdClass Object
                (
                    [name] => team_count
                    [value] =>
                )

            [team_name] => stdClass Object
                (
                    [name] => team_name
```

```
                [value] =>
            )

        [date_quote_expected_closed] => stdClass Object
            (
                [name] => date_quote_expected_closed
                [value] => 2013-05-06
            )

        [quote_stage] => stdClass Object
            (
                [name] => quote_stage
                [value] => Negotiation
            )

        [quote_num] => stdClass Object
            (
                [name] => quote_num
                [value] =>
            )

        [quote_type] => stdClass Object
            (
                [name] => quote_type
                [value] => Quotes
            )

        [subtotal] => stdClass Object
            (
                [name] => subtotal
                [value] => 1230.23
            )

        [subtotal_usdollar] => stdClass Object
            (
                [name] => subtotal_usdollar
                [value] => 1230.23
            )

    )

)

//Create Product Result
stdClass Object
(
```

```
    [id] => 6c40f344-a269-d4d0-9929-517e83884fb2
    [entry_list] => stdClass Object
        (
            [name] => stdClass Object
                (
                    [name] => name
                    [value] => Widget
                )

            [quote_id] => stdClass Object
                (
                    [name] => quote_id
                    [value] => 2e0cd18b-21da-50f0-10f6-517e835a1e09
                )

            [status] => stdClass Object
                (
                    [name] => status
                    [value] => Quotes
                )

        )

)

//Create ProductBundles Result
stdClass Object
(
    [id] => a8a4e449-7e72-dea5-9495-517e830a7353
    [entry_list] => stdClass Object
        (
            [name] => stdClass Object
                (
                    [name] => name
                    [value] => Rest SugarOnline Order
                )

            [bundle_stage] => stdClass Object
                (
                    [name] => bundle_stage
                    [value] => Draft
                )

            [tax] => stdClass Object
                (
                    [name] => tax
```

```
                    [value] => 0
                )

            [total] => stdClass Object
                (
                    [name] => total
                    [value] => 0
                )

            [subtotal] => stdClass Object
                (
                    [name] => subtotal
                    [value] => 0
                )

            [currency_id] => stdClass Object
                (
                    [name] => currency_id
                    [value] => -99
                )

        )

)

//Create ProductBundleProduct Relationship Result
stdClass Object
(
    [created] => 1
    [failed] => 0
    [deleted] => 0
)

//Create ProductBundleQuote Relationship Result
stdClass Object
(
    [created] => 1
    [failed] => 0
    [deleted] => 0
)
```

Retrieving a List of Fields From a Module

Overview

A PHP example demonstrating how to retrieve fields vardefs from the accounts module with the get_module_fields method using cURL and the v4_1 REST API.

This example will only retrieve the vardefs for the 'id' and 'name' fields.

Example

```
<?php

    $url = "http://{site_url}/service/v4_1/rest.php";
    $username = "admin";
    $password = "password";

    function call($method, $parameters, $url)
    {
        ob_start();
        $curl_request = curl_init();

        curl_setopt($curl_request, CURLOPT_URL, $url);
        curl_setopt($curl_request, CURLOPT_POST, 1);
        curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VER
SION_1_0);
        curl_setopt($curl_request, CURLOPT_HEADER, 1);
        curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
        curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
        curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

        $jsonEncodedData = json_encode($parameters);

        $post = array(
             "method" => $method,
             "input_type" => "JSON",
             "response_type" => "JSON",
             "rest_data" => $jsonEncodedData
        );

        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $post);
        $result = curl_exec($curl_request);
        curl_close($curl_request);

        $result = explode("\r\n\r\n", $result, 2);
```

```
        $response = json_decode($result[1]);
        ob_end_flush();

        return $response;
    }

    //login ----------------------------------------

    $login_parameters = array(
         "user_auth" => array(
              "user_name" => $username,
              "password" => md5($password),
              "version" => "1"
         ),
         "application_name" => "RestTest",
         "name_value_list" => array(),
    );

    $login_result = call("login", $login_parameters, $url);

    /*
    echo "<pre>";
    print_r($login_result);
    echo "</pre>";
    */

    //get session id
    $session_id = $login_result->id;

    //retrieve fields -------------------------------

    $get_module_fields_parameters = array(

         //session id
         'session' => $session_id,

         //The name of the module from which to retrieve records
         'module_name' => 'Accounts',

         //Optional. Returns vardefs for the specified fields. An empt
y array will return all fields.
         'fields' => array(
             'id',
             'name',
         ),
    );
```

```
    $get_module_fields_result = call("get_module_fields", $get_module_
fields_parameters, $url);

    echo "<pre>";
    print_r($get_module_fields_result);
    echo "</pre>";

?>
```

Result

```
stdClass Object
(
    [module_name] => Accounts
    [table_name] => accounts
    [module_fields] => stdClass Object
        (
            [id] => stdClass Object
                (
                    [name] => id
                    [type] => id
                    [group] =>
                    [id_name] =>
                    [label] => ID
                    [required] => 1
                    [options] => Array
                        (
                        )

                    [related_module] =>
                    [calculated] =>
                    [len] =>
                )

            [name] => stdClass Object
                (
                    [name] => name
                    [type] => name
                    [group] =>
                    [id_name] =>
                    [label] => Name:
                    [required] => 1
                    [options] => Array
```

```
                        (
                        )

                    [related_module] =>
                    [calculated] =>
                    [len] => 150
                )

        )

    [link_fields] => Array
        (
        )

)
```

Retrieving a List of Records

Overview

A PHP example demonstrating how to retrieve a list of records from a module with the get_entry_list method using cURL and the v4_1 REST API.

This example will retrieve a list of leads.

Example

```
<?php

    $url = "http://{site_url}/service/v4_1/rest.php";
    $username = "admin";
    $password = "password";

    //function to make cURL request
    function call($method, $parameters, $url)
    {
        ob_start();
        $curl_request = curl_init();

        curl_setopt($curl_request, CURLOPT_URL, $url);
```

```
        curl_setopt($curl_request, CURLOPT_POST, 1);
        curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VER
SION_1_0);
        curl_setopt($curl_request, CURLOPT_HEADER, 1);
        curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
        curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
        curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

        $jsonEncodedData = json_encode($parameters);

        $post = array(
             "method" => $method,
             "input_type" => "JSON",
             "response_type" => "JSON",
             "rest_data" => $jsonEncodedData
        );

        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $post);
        $result = curl_exec($curl_request);
        curl_close($curl_request);

        $result = explode("\r\n\r\n", $result, 2);
        $response = json_decode($result[1]);
        ob_end_flush();

        return $response;
    }

    //login ----------------------------------------
    $login_parameters = array(
         "user_auth" => array(
              "user_name" => $username,
              "password" => md5($password),
              "version" => "1"
         ),
         "application_name" => "RestTest",
         "name_value_list" => array(),
    );

    $login_result = call("login", $login_parameters, $url);

    /*
    echo "<pre>";
    print_r($login_result);
    echo "</pre>";
    */
```

```
    //get session id
    $session_id = $login_result->id;

    //get list of records --------------------------------

    $get_entry_list_parameters = array(

         //session id
         'session' => $session_id,

         //The name of the module from which to retrieve records
         'module_name' => 'Leads',

         //The SQL WHERE clause without the word "where".
         'query' => "",

         //The SQL ORDER BY clause without the phrase "order by".
         'order_by' => "",

         //The record offset from which to start.
         'offset' => '0',

         //Optional. A list of fields to include in the results.
         'select_fields' => array(
              'id',
              'name',
              'title',
         ),

         /*
         A list of link names and the fields to be returned for each l
ink name.
         Example: 'link_name_to_fields_array' => array(array('name' =>
 'email_addresses', 'value' => array('id', 'email_address', 'opt_out',
 'primary_address')))
         */
         'link_name_to_fields_array' => array(
         ),

         //The maximum number of results to return.
         'max_results' => '2',

         //To exclude deleted records
         'deleted' => '0',
```

```
        //If only records marked as favorites should be returned.
        'Favorites' => false,
    );

    $get_entry_list_result = call('get_entry_list', $get_entry_list_pa
rameters, $url);

    echo '<pre>';
    print_r($get_entry_list_result);
    echo '</pre>';

?>
```

Result

```
stdClass Object
(
    [result_count] => 2
    [total_count] => 200
    [next_offset] => 2
    [entry_list] => Array
        (
            [0] => stdClass Object
                (
                    [id] => 18124607-69d1-b158-47ff-4f7cb69344f7
                    [module_name] => Leads
                    [name_value_list] => stdClass Object
                        (
                            [id] => stdClass Object
                                (
                                    [name] => id
                                    [value] => 18124607-69d1-b158-47ff
-4f7cb69344f7
                                )

                            [name] => stdClass Object
                                (
                                    [name] => name
                                    [value] => Bernie Worthey
                                )

                            [title] => stdClass Object
                                (
                                    [name] => title
```

```
                                    [value] => Senior Product Manager
                                )

                            )

                    )

                [1] => stdClass Object
                    (
                        [id] => 1cdfddc1-2759-b007-8713-4f7cb64c2e9c
                        [module_name] => Leads
                        [name_value_list] => stdClass Object
                            (
                                [id] => stdClass Object
                                    (
                                        [name] => id
                                        [value] => 1cdfddc1-2759-b007-8713
-4f7cb64c2e9c
                                    )

                                [name] => stdClass Object
                                    (
                                        [name] => name
                                        [value] => Bobbie Kohlmeier
                                    )

                                [title] => stdClass Object
                                    (
                                        [name] => title
                                        [value] => Director Operations
                                    )

                            )

                    )

            )

        [relationship_list] => Array
            (
            )

)
```

Retrieving a List of Records With Related Info

Overview

A PHP example demonstrating how to retrieve a list of records with info from a related entity with the get_entry_list method using cURL and the v4_1 REST API.

This example will retrieve a list of contacts and their related email addresses.

Example

```
<?php

    $url = "http://{site_url}/service/v4_1/rest.php";
    $username = "admin";
    $password = "password";

    //function to make cURL request
    function call($method, $parameters, $url)
    {
        ob_start();
        $curl_request = curl_init();

        curl_setopt($curl_request, CURLOPT_URL, $url);
        curl_setopt($curl_request, CURLOPT_POST, 1);
        curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VER
SION_1_0);
        curl_setopt($curl_request, CURLOPT_HEADER, 1);
        curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
        curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
        curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

        $jsonEncodedData = json_encode($parameters);

        $post = array(
             "method" => $method,
             "input_type" => "JSON",
             "response_type" => "JSON",
             "rest_data" => $jsonEncodedData
        );

        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $post);
```

```
    $result = curl_exec($curl_request);
    curl_close($curl_request);

    $result = explode("\r\n\r\n", $result, 2);
    $response = json_decode($result[1]);
    ob_end_flush();

    return $response;
}

//login ------------------------------------------------
$login_parameters = array(
     "user_auth" => array(
          "user_name" => $username,
          "password" => md5($password),
          "version" => "1"
     ),
     "application_name" => "RestTest",
     "name_value_list" => array(),
);

$login_result = call("login", $login_parameters, $url);

/*
echo "<pre>";
print_r($login_result);
echo "</pre>";
*/

//get session id
$session_id = $login_result->id;

//retrieve records ----------------------------------
$get_entry_list_parameters = array(

     //session id
     'session' => $session_id,

     //The name of the module from which to retrieve records
     'module_name' => "Contacts",

     //The SQL WHERE clause without the word "where".
     'query' => "",

     //The SQL ORDER BY clause without the phrase "order by".
     'order_by' => "",
```

```
        //The record offset from which to start.
        'offset' => "0",

        //Optional. The list of fields to be returned in the results
        'select_fields' => array(
             'id',
             'first_name',
             'last_name',
        ),

        //A list of link names and the fields to be returned for each
 link name
        'link_name_to_fields_array' => array(
             array(
                  'name' => 'email_addresses',
                  'value' => array(
                       'id',
                       'email_address',
                       'opt_out',
                       'primary_address'
                  ),
             ),
        ),

        //The maximum number of results to return.
        'max_results' => '2',

        //To exclude deleted records
        'deleted' => 0,

        //If only records marked as favorites should be returned.
        'Favorites' => false,

    );

    $get_entry_list_result = call("get_entry_list", $get_entry_list_pa
rameters, $url);

    echo "<pre>";
    print_r($get_entry_list_result);
    echo "</pre>";

?>
```

Result

```
stdClass Object
(
    [result_count] => 2
    [total_count] => 200
    [next_offset] => 2
    [entry_list] => Array
        (
            [0] => stdClass Object
                (
                    [id] => 116d9bc6-4a24-b826-952e-4f7cb6b25ea7
                    [module_name] => Contacts
                    [name_value_list] => stdClass Object
                        (
                            [id] => stdClass Object
                                (
                                    [name] => id
                                    [value] => 116d9bc6-4a24-b826-952e
-4f7cb6b25ea7
                                )

                            [first_name] => stdClass Object
                                (
                                    [name] => first_name
                                    [value] => Lucinda
                                )

                            [last_name] => stdClass Object
                                (
                                    [name] => last_name
                                    [value] => Jacoby
                                )

                        )

                )

            [1] => stdClass Object
                (
                    [id] => 11c263ef-ff61-71ff-f090-4f7cb6fc9f68
                    [module_name] => Contacts
                    [name_value_list] => stdClass Object
                        (
                            [id] => stdClass Object
```

```
                                (
                                    [name] => id
                                    [value] => 11c263ef-ff61-71ff-
f090-4f7cb6fc9f68
                                )

                            [first_name] => stdClass Object
                                (
                                    [name] => first_name
                                    [value] => Ike
                                )

                            [last_name] => stdClass Object
                                (
                                    [name] => last_name
                                    [value] => Gassaway
                                )

                        )

                )

        )

    [relationship_list] => Array
        (
            [0] => stdClass Object
                (
                    [link_list] => Array
                        (
                            [0] => stdClass Object
                                (
                                    [name] => email_addresses
                                    [records] => Array
                                        (
                                            [0] => stdClass Object
                                                (
                                                    [link_value] => st
dClass Object
                                                        (
                                                            [id] => st
dClass Object
                                                                (
                                                                    [n
ame] => id
                                                                    [v
```

```
alue] => 13066f13-d6ea-405c-0f95-4f7cb6fa3a08
                                                                    )

                                                                [email_add
ress] => stdClass Object
                                                                    (
                                                                        [n
ame] => email_address
                                                                        [v
alue] => support52@example.org
                                                                    )

                                                                [opt_out]
=> stdClass Object
                                                                    (
                                                                        [n
ame] => opt_out
                                                                        [v
alue] => 0
                                                                    )

                                                                [primary_a
ddress] => stdClass Object
                                                                    (
                                                                        [n
ame] => primary_address
                                                                        [v
alue] =>
                                                                    )

                                                            )

                                                        )

                                                    [1] => stdClass Object
                                                        (
                                                            [link_value] => st
dClass Object
                                                                (
                                                                    [id] => st
dClass Object
                                                                    (
                                                                        [n
ame] => id
                                                                        [v
alue] => 13e3d111-b226-c363-4832-4f7cb699a3a0
```

```
                                                                    )

                                                                [email_add
ress] => stdClass Object
                                                                    (
                                                                        [n
ame] => email_address
                                                                        [v
alue] => qa.section@example.it
                                                                    )

                                                                [opt_out]
=> stdClass Object
                                                                    (
                                                                        [n
ame] => opt_out
                                                                        [v
alue] => 1
                                                                    )

                                                                [primary_a
ddress] => stdClass Object
                                                                    (
                                                                        [n
ame] => primary_address
                                                                        [v
alue] =>
                                                                    )

                                                            )

                                                    )

                                            )

                                    )

                            )

                    )

            [1] => stdClass Object
                (
                    [link_list] => Array
                        (
                            [0] => stdClass Object
```

```
                                (
                                    [name] => email_addresses
                                    [records] => Array
                                        (
                                            [0] => stdClass Object
                                                (
                                                    [link_value] => st
dClass Object
                                                        (
                                                            [id] => st
dClass Object
                                                                (
                                                                    [n
ame] => id
                                                                    [v
alue] => 16aeaf8b-b31f-943d-3844-4f7cb6ac63f2
                                                                )

                                                            [email_add
ress] => stdClass Object
                                                                (
                                                                    [n
ame] => email_address
                                                                    [v
alue] => vegan.sales.im@example.cn
                                                                )

                                                            [opt_out]
=> stdClass Object
                                                                (
                                                                    [n
ame] => opt_out
                                                                    [v
alue] => 0
                                                                )

                                                            [primary_a
ddress] => stdClass Object
                                                                (
                                                                    [n
ame] => primary_address
                                                                    [v
alue] =>
                                                                )

                                                        )
```

```
                )

            [1] => stdClass Object
                (
                    [link_value] => st
dClass Object
                        (
                            [id] => st
dClass Object
                                (
                                    [n
ame] => id
                                    [v
alue] => 173bacf5-5ea6-3906-d91d-4f7cb6c6e883
                                )

                            [email_add
ress] => stdClass Object
                                (
                                    [n
ame] => email_address
                                    [v
alue] => kid.the.section@example.org
                                )

                            [opt_out]
=> stdClass Object
                                (
                                    [n
ame] => opt_out
                                    [v
alue] => 1
                                )

                            [primary_a
ddress] => stdClass Object
                                (
                                    [n
ame] => primary_address
                                    [v
alue] =>
                                )

                        )
```

```
                                                    )

                                                )

                                            )

                                        )

                                    )

                                )

)
```

Retrieving Email Attachments

Overview

A PHP example demonstrating how to retrieve an email and its attachments from using the get_entry and get_note_attachment methods using cURL and the v4_1 REST API.

This example will retrieve a specified email record by its ID and write the attachments to the local filesystem.

Example

```
<?php

$url = "http://{site_url}/service/v4_1/rest.php";
$username = "admin";
$password = "password";

//function to make cURL request
function call($method, $parameters, $url)
{
    ob_start();
    $curl_request = curl_init();

    curl_setopt($curl_request, CURLOPT_URL, $url);
```

```
    curl_setopt($curl_request, CURLOPT_POST, 1);
    curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION
_1_0);
    curl_setopt($curl_request, CURLOPT_HEADER, 1);
    curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

    $jsonEncodedData = json_encode($parameters);

    $post = array(
        "method" => $method,
        "input_type" => "JSON",
        "response_type" => "JSON",
        "rest_data" => $jsonEncodedData
    );

    curl_setopt($curl_request, CURLOPT_POSTFIELDS, $post);
    $result = curl_exec($curl_request);
    curl_close($curl_request);

    $result = explode("\r\n\r\n", $result, 2);
    $response = json_decode($result[1]);
    ob_end_flush();

    return $response;
}

//login ------------------------------------------

$login_parameters = array(
    "user_auth" => array(
        "user_name" => $username,
        "password" => md5($password),
        "version" => "1"
    ),
    "application_name" => "RestTest",
    "name_value_list" => array(),
);

$login_result = call("login", $login_parameters, $url);

/*
echo "<pre>";
print_r($login_result);
echo "</pre>";
```

```
*/

//get session id
$session_id = $login_result->id;

//retrieve the email ------------------------------------------

// email id of an email with an attachment
$email_id   = '5826bd75-527a-a736-edf5-5205421467bf';

// use get_entry to get the email contents
$get_entry_parameters = array(
    'session' => $session_id,
    'module_name' => 'Emails',
    'id' => $email_id,
    'select_fields' => array(),
    'link_name_to_fields_array' => array(
        array(
            'name' => 'notes',
            'value' => array(
                'id',
                'name',
                'file_mime_type',
                'filename',
                'description',
            ),
        ),
    ),
    'track_view' => false
);

$get_entry_result = call('get_entry', $get_entry_parameters, $url);

//Email record contents
echo "<pre>";
print_r($get_entry_result);
echo "</pre>";

if (!isset($get_entry_result->entry_list[0]))
{
    echo "Email not found!";
    die();
}

if (!isset($get_entry_result->relationship_list) || count($get_entry_r
esult->relationship_list) == 0)
```

```
{
    echo "No attachments found!";
    die();
}

//retrieve any attachments ------------------------------------------
--

foreach ($get_entry_result->relationship_list[0][0]->records as $key =
> $attachmentInfo)
{
    $get_note_attachment_parameters = array(
        'session' => $session_id,
        'id'      => $attachmentInfo->id->value,
    );

    $get_note_attachment_result = call('get_note_attachment', $get_not
e_attachment_parameters, $url);

    //attachment contents
    echo "<pre>";
    print_r($get_note_attachment_result);
    echo "</pre>";

    $file_name = $get_note_attachment_result->note_attachment->filenam
e;
    //decode and get file contents
    $file_contents = base64_decode($get_note_attachment_result->note_a
ttachment->file);

    //write file
    file_put_contents($file_name, $file_contents);
}
```

Result

```
//Email Result
stdClass Object
(
    [entry_list] => Array
        (
            [0] => stdClass Object
                (
                    [id] => 5826bd75-527a-a736-edf5-5205421467bf
```

```
                    [module_name] => Emails
                    [name_value_list] => stdClass Object
                        (
                            [assigned_user_name] => stdClass Object
                                (
                                    [name] => assigned_user_name
                                    [value] => Administrator
                                )

                            [modified_by_name] => stdClass Object
                                (
                                    [name] => modified_by_name
                                    [value] => Administrator
                                )

                            [created_by_name] => stdClass Object
                                (
                                    [name] => created_by_name
                                    [value] => Administrator
                                )

                            [team_id] => stdClass Object
                                (
                                    [name] => team_id
                                    [value] => 1
                                )

                            [team_set_id] => stdClass Object
                                (
                                    [name] => team_set_id
                                    [value] => 1
                                )

                            [team_name] => stdClass Object
                                (
                                    [name] => team_name
                                    [value] => Global
                                )

                            [id] => stdClass Object
                                (
                                    [name] => id
                                    [value] => 5826bd75-527a-
a736-edf5-5205421467bf
                                )
```

```
[date_entered] => stdClass Object
    (
        [name] => date_entered
        [value] => 2013-08-09 19:28:00
    )

[date_modified] => stdClass Object
    (
        [name] => date_modified
        [value] => 2013-08-09 19:29:08
    )

[assigned_user_id] => stdClass Object
    (
        [name] => assigned_user_id
        [value] => 1
    )

[modified_user_id] => stdClass Object
    (
        [name] => modified_user_id
        [value] => 1
    )

[created_by] => stdClass Object
    (
        [name] => created_by
        [value] => 1
    )

[deleted] => stdClass Object
    (
        [name] => deleted
        [value] => 0
    )

[from_addr_name] => stdClass Object
    (
        [name] => from_addr_name
        [value] => SugarCRM
    )

[to_addrs_names] => stdClass Object
    (
        [name] => to_addrs_names
        [value] => email@address.com
```

```
    )

[description_html] => stdClass Object
    (
        [name] => description_html
        [value] =>


    )

[description] => stdClass Object
    (
        [name] => description
        [value] =>
    )

[date_sent] => stdClass Object
    (
        [name] => date_sent
        [value] => 2013-08-09 19:28:00
    )

[message_id] => stdClass Object
    (
        [name] => message_id
        [value] =>
    )

[message_uid] => stdClass Object
    (
        [name] => message_uid
        [value] =>
    )

[name] => stdClass Object
    (
        [name] => name
        [value] => Example
    )

[type] => stdClass Object
    (
        [name] => type
        [value] => out
    )
```

```
                    [status] => stdClass Object
                        (
                            [name] => status
                            [value] => read
                        )

                    [flagged] => stdClass Object
                        (
                            [name] => flagged
                            [value] => 0
                        )

                    [reply_to_status] => stdClass Object
                        (
                            [name] => reply_to_status
                            [value] => 0
                        )

                    [intent] => stdClass Object
                        (
                            [name] => intent
                            [value] => pick
                        )

                    [mailbox_id] => stdClass Object
                        (
                            [name] => mailbox_id
                            [value] =>
                        )

                    [parent_name] => stdClass Object
                        (
                            [name] => parent_name
                            [value] => Having trouble adding n
ew items
                        )

                    [parent_type] => stdClass Object
                        (
                            [name] => parent_type
                            [value] => Cases
                        )

                    [parent_id] => stdClass Object
                        (
                            [name] => parent_id
```

```
                                    [value] => 116d4d46-928c-d4af-
c22e-518ae4eb13fc
                                )
                            )
                        )
                )
            [relationship_list] => Array
                (
                    [0] => Array
                        (
                            [0] => stdClass Object
                                (
                                    [name] => notes
                                    [records] => Array
                                        (
                                            [0] => stdClass Object
                                                (
                                                    [id] => stdClass Object
                                                        (
                                                            [name] => id
                                                            [value] => 1b63a8f
9-ce67-6aad-b5a4-52054af18c47
                                                        )

                                                    [name] => stdClass Object
                                                        (
                                                            [name] => name
                                                            [value] => Example
.zip
                                                        )

                                                    [file_mime_type] => stdCla
ss Object
                                                        (
                                                            [name] => file_mim
e_type
                                                            [value] => applica
tion/zip
                                                        )

                                                    [filename] => stdClass Obj
ect
                                                        (
                                                            [name] => filename
                                                            [value] => Example
2.zip
```

```
                                        )

                                    [description] => stdClass
Object
                                        (
                                            [name] => descript
ion
                                            [value] =>
                                        )

                                    [_empty_] => stdClass Obje
ct
                                        (
                                            [name] =>
                                            [value] =>
                                        )

                                )

                            [1] => stdClass Object
                                (
                                    [id] => stdClass Object
                                        (
                                            [name] => id
                                            [value] => 592f382
d-b633-fd5f-803e-5205423a6d0b
                                        )

                                    [name] => stdClass Object
                                        (
                                            [name] => name
                                            [value] => Example
2.zip
                                        )

                                    [file_mime_type] => stdCla
ss Object
                                        (
                                            [name] => file_mim
e_type
                                            [value] => applica
tion/zip
                                        )

                                    [filename] => stdClass Obj
ect
```

```
                                                    (
                                                        [name] => filename
                                                        [value] => Example
2.zip
                                                    )

                                                [description] => stdClass
Object
                                                    (
                                                        [name] => descript
ion
                                                        [value] =>
                                                    )

                                                [_empty_] => stdClass Obje
ct
                                                    (
                                                        [name] =>
                                                        [value] =>
                                                    )

                                            )

                                    )

                            )

                )

        )

)

//Attachment 1 Result
stdClass Object
(
    [note_attachment] => stdClass Object
        (
            [id] => 1b63a8f9-ce67-6aad-b5a4-52054af18c47
            [filename] => Example.zip
            [file] => UEsDBAoAAAAIAOujCEOkMbvsNa0AAMCFAgAXAAAARmlsZXMv
aW
            [related_module_id] => 5826bd75-527a-
a736-edf5-5205421467bf
            [related_module_name] => Emails
        )
```

```
)

//Attachment 2 Result
stdClass Object
(
    [note_attachment] => stdClass Object
        (
            [id] => 592f382d-b633-fd5f-803e-5205423a6d0b
            [filename] => Example2.zip
            [file] => AEUoaAAARmlslsZXMvaWZXujCEOkMbvsNa0AAMCFAgAXAAAA
Rm
            [related_module_id] => 5826bd75-527a-
a736-edf5-5205421467bf
            [related_module_name] => Emails
        )

)
```

Related

Retrieving Multiple Records by ID

Overview

A PHP example demonstrating how to retrieve multiple records from the accounts module with the get_entries method using cURL and the v4_1 REST API.

This example will only retrieve 2 records and return the following fields: 'name', 'billing_address_state' and 'billing_address_country'.

Example

```
<?php

    $url = "http://{site_url}/service/v4_1/rest.php";
    $username = "admin";
    $password = "password";

    //function to make cURL request
```

```
    function call($method, $parameters, $url)
    {
        ob_start();
        $curl_request = curl_init();

        curl_setopt($curl_request, CURLOPT_URL, $url);
        curl_setopt($curl_request, CURLOPT_POST, 1);
        curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VER
SION_1_0);
        curl_setopt($curl_request, CURLOPT_HEADER, 1);
        curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
        curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
        curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

        $jsonEncodedData = json_encode($parameters);

        $post = array(
             "method" => $method,
             "input_type" => "JSON",
             "response_type" => "JSON",
             "rest_data" => $jsonEncodedData
        );

        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $post);
        $result = curl_exec($curl_request);
        curl_close($curl_request);

        $result = explode("\r\n\r\n", $result, 2);
        $response = json_decode($result[1]);
        ob_end_flush();

        return $response;
    }

    //login ------------------------------------------

    $login_parameters = array(
         "user_auth" => array(
              "user_name" => $username,
              "password" => md5($password),
              "version" => "1"
         ),
         "application_name" => "RestTest",
         "name_value_list" => array(),
    );
```

```
   $login_result = call("login", $login_parameters, $url);

   /*
   echo "<pre>";
   print_r($login_result);
   echo "</pre>";
   */

   //get session id
   $session_id = $login_result->id;

   //retrieve records ----------------------------------

   $get_entries_parameters = array(

        //session id
        'session' => $session_id,

        //The name of the module from which to retrieve records
        'module_name' => 'Accounts',

        //An array of SugarBean IDs
        'ids' => array(
            '20328809-9d0a-56fc-0e7c-4f7cb6eb1c83',
            '328b22a6-d784-66d9-0295-4f7cb59e8cbb',
        ),

        //Optional. The list of fields to be returned in the results
        'select_fields' => array(
           'name',
           'billing_address_state',
           'billing_address_country'
        ),

        //A list of link names and the fields to be returned for each
link name
        'link_name_to_fields_array' => array(
        ),
   );

   $get_entries_result = call("get_entries", $get_entries_parameters,
$url);

   echo "<pre>";
   print_r($get_entries_result);
   echo "</pre>";
```

```
?>
```

Result

```
stdClass Object
(
    [entry_list] => Array
        (
            [0] => stdClass Object
                (
                    [id] => 20328809-9d0a-56fc-0e7c-4f7cb6eb1c83
                    [module_name] => Accounts
                    [name_value_list] => stdClass Object
                        (
                            [name] => stdClass Object
                                (
                                    [name] => name
                                    [value] => Jungle Systems Inc
                                )

                            [billing_address_state] => stdClass Object
                                (
                                    [name] => billing_address_state
                                    [value] => NY
                                )

                            [billing_address_country] => stdClass Obje
ct
                                (
                                    [name] => billing_address_country
                                    [value] => USA
                                )

                        )

                )

            [1] => stdClass Object
                (
                    [id] => 328b22a6-d784-66d9-0295-4f7cb59e8cbb
                    [module_name] => Accounts
                    [name_value_list] => stdClass Object
                        (
```

```
                                [name] => stdClass Object
                                    (
                                        [name] => name
                                        [value] => Riviera Hotels
                                    )

                                [billing_address_state] => stdClass Object
                                    (
                                        [name] => billing_address_state
                                        [value] => CA
                                    )

                                [billing_address_country] => stdClass Obje
ct
                                    (
                                        [name] => billing_address_country
                                        [value] => USA
                                    )

                            )

                )

        )

    [relationship_list] => Array
        (
        )

)
```

Retrieving Records by Email Domain

Overview

A PHP example demonstrating how to retrieve email addresses based on an email domain with the search_by_module and get_entries methods using cURL and the v4_1 REST API.

When using the search_by_module method, the email address information is not

returned in the result. Due to this behavior, we will gather the record ids and pass them back to the get_entries method to fetch our related email addresses.

Example

```
<?php

    $url = "http://{site_url}/service/v4_1/rest.php";
    $username = "admin";
    $password = "password";

    //function to make cURL request
    function call($method, $parameters, $url)
    {
        ob_start();
        $curl_request = curl_init();

        curl_setopt($curl_request, CURLOPT_URL, $url);
        curl_setopt($curl_request, CURLOPT_POST, 1);
        curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VER
SION_1_0);
        curl_setopt($curl_request, CURLOPT_HEADER, 1);
        curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
        curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
        curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

        $jsonEncodedData = json_encode($parameters);

        $post = array(
            "method" => $method,
            "input_type" => "JSON",
            "response_type" => "JSON",
            "rest_data" => $jsonEncodedData
        );

        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $post);
        $result = curl_exec($curl_request);
        curl_close($curl_request);

        $result = explode("\r\n\r\n", $result, 2);
        $response = json_decode($result[1]);
        ob_end_flush();

        return $response;
    }
```

```
    //login ------------------------------------------

    $login_parameters = array(
        "user_auth" => array(
            "user_name" => $username,
            "password" => md5($password),
            "version" => "1"
        ),
        "application_name" => "RestTest",
        "name_value_list" => array(),
    );

    $login_result = call("login", $login_parameters, $url);

    /*
    echo "<pre>";
    print_r($login_result);
    echo "</pre>";
    */

    //get session id
    $session_id = $login_result->id;

    //search_by_module ------------------------------------------------
--

    $search_by_module_parameters = array(
        "session" => $session_id,
        'search_string' => '%@example.com',
        'modules' => array(
            'Accounts',
            'Contacts',
            'Leads',
        ),
        'offset' => 0,
        'max_results' => 1,
        'assigned_user_id' => '',
        'select_fields' => array('id'),
        'unified_search_only' => false,
        'favorites' => false
    );

    $search_by_module_results = call('search_by_module', $search_by_mo
dule_parameters, $url);
```

```
/*
echo '<pre>';
print_r($search_by_module_results);
echo '</pre>';
*/

$record_ids = array();
foreach ($search_by_module_results->entry_list as $results)
{
    $module = $results->name;

    foreach ($results->records as $records)
    {
        foreach($records as $record)
        {
            if ($record->name = 'id')
            {
                $record_ids[$module][] = $record->value;
                //skip any additional fields
                break;
            }
        }
    }

}

$get_entries_results = array();
$modules = array_keys($record_ids);

foreach($modules as $module)
{
    $get_entries_parameters = array(
        //session id
        'session' => $session_id,

        //The name of the module from which to retrieve records
        'module_name' => $module,

        //An array of record IDs
        'ids' => $record_ids[$module],

        //The list of fields to be returned in the results
        'select_fields' => array(
            'name',
        ),
```

```
            //A list of link names and the fields to be returned for e
ach link name
            'link_name_to_fields_array' => array(
                array(
                    'name' => 'email_addresses',
                    'value' => array(
                        'email_address',
                        'opt_out',
                        'primary_address'
                    ),
                ),
            ),

            //Flag the record as a recently viewed item
            'track_view' => false,
        );

        $get_entries_results[$module] = call('get_entries', $get_entri
es_parameters, $url);
    }

    echo '<pre>';
    print_r($get_entries_results);
    echo '</pre>';
```

Result

```
Array
(
    [Accounts] => stdClass Object
        (
            [entry_list] => Array
                (
                    [0] => stdClass Object
                        (
                            [id] => 1bb7ef28-64b9-cbd5-e7d6-5282a3b969
53
                            [module_name] => Accounts
                            [name_value_list] => stdClass Object
                                (
                                    [name] => stdClass Object
                                        (
                                            [name] => name
```

```
                                            [value] => Underwater Mini
ng Inc.
                                        )

                                    )

                            )

                        [1] => stdClass Object
                            (
                                [id] => efbc0c4e-
cc72-f843-b6ed-5282a38dad7f
                                [module_name] => Accounts
                                [name_value_list] => stdClass Object
                                    (
                                        [name] => stdClass Object
                                            (
                                                [name] => name
                                                [value] => 360 Vacations
                                            )

                                    )

                            )

                    )

            [relationship_list] => Array
                (
                    [0] => stdClass Object
                        (
                            [link_list] => Array
                                (
                                    [0] => stdClass Object
                                        (
                                            [name] => email_addresses
                                            [records] => Array
                                                (
                                                    [0] => stdClass Ob
ject
                                                        (
                                                            [link_valu
e] => stdClass Object
                                                                (
                                                                    [e
mail_address] => stdClass Object
```

```
  (
      [name] => email_address
      [value] => beans.the.qa@example.com
  )
                                                                  [o
pt_out] => stdClass Object
  (
      [name] => opt_out
      [value] => 0
  )
                                                                  [p
rimary_address] => stdClass Object
  (
      [name] => primary_address
      [value] =>
  )
                                                                      )
                                                                  )
                                                              [1] => stdClass Ob
ject
                                                                  (
                                                                      [link_valu
e] => stdClass Object
                                                                      (
                                                                          [e
mail_address] => stdClass Object
  (
```

```
      [name] => email_address
      [value] => section.sales@example.edu
  )
                                                                    [opt_out] => stdClass Object
  (
      [name] => opt_out
      [value] => 0
  )
                                                                    [primary_address] => stdClass Object
  (
      [name] => primary_address
      [value] =>
  )
                                                                        )
                                                                )
                                                            )
                                                        )
                                                    )
                                                )
                                            )
                                        )
                                    )
                                )
                            )
                        )
                    )
                [1] => stdClass Object
                    (
                        [link_list] => Array
                            (
                                [0] => stdClass Object
                                    (
```

```
                                                [name] => email_addresses
                                                [records] => Array
                                                    (
                                                        [0] => stdClass Ob
ject
                                                            (
                                                                [link_valu
e] => stdClass Object
                                                                    (
                                                                        [e
mail_address] => stdClass Object

  (

      [name] => email_address

      [value] => section51@example.com

  )

                                                                        [o
pt_out] => stdClass Object

  (

      [name] => opt_out

      [value] => 0

  )

                                                                        [p
rimary_address] => stdClass Object

  (

      [name] => primary_address

      [value] =>

  )

                                                                    )

                                                            )
```

```
                                                    [1] => stdClass Object
                                                        (
                                                            [link_value] => stdClass Object
                                                                (
                                                                    [email_address] => stdClass Object
  (
      [name] => email_address
      [value] => kid.sugar.qa@example.co.uk
  )
                                                                    [opt_out] => stdClass Object
  (
      [name] => opt_out
      [value] => 0
  )
                                                                    [primary_address] => stdClass Object
  (
      [name] => primary_address
      [value] =>
  )
                                                                )
                                                        )
                                                )
```

```
                                )

                        )

                )

        )

    [Contacts] => stdClass Object
        (
            [entry_list] => Array
                (
                    [0] => stdClass Object
                        (
                            [id] => 24330979-0e39-f9ec-2622-5282a372f3
9b
                            [module_name] => Contacts
                            [name_value_list] => stdClass Object
                                (
                                    [name] => stdClass Object
                                        (
                                            [name] => name
                                            [value] => Errol Goldberg
                                        )

                                )

                        )

                    [1] => stdClass Object
                        (
                            [id] => 2542dad2-85f3-5529-3a30-5282a3104e
31
                            [module_name] => Contacts
                            [name_value_list] => stdClass Object
                                (
                                    [name] => stdClass Object
                                        (
                                            [name] => name
                                            [value] => Luis Deegan
                                        )

                                )

                        )
```

```
                )

            [relationship_list] => Array
                (
                    [0] => stdClass Object
                        (
                            [link_list] => Array
                                (
                                    [0] => stdClass Object
                                        (
                                            [name] => email_addresses
                                            [records] => Array
                                                (
                                                    [0] => stdClass Ob
ject
                                                        (
                                                            [link_valu
e] => stdClass Object
                                                                (
                                                                    [e
mail_address] => stdClass Object

  (

      [name] => email_address

      [value] => hr.phone@example.com

  )

                                                                    [o
pt_out] => stdClass Object

  (

      [name] => opt_out

      [value] => 0

  )

                                                                    [p
rimary_address] => stdClass Object

  (
```

```
      [name] => primary_address
      [value] =>
  )
                                                                  )
                                                              )
                                                          [1] => stdClass Ob
ject
                                                              (
                                                                  [link_valu
e] => stdClass Object
                                                                  (
                                                                      [e
mail_address] => stdClass Object
  (
      [name] => email_address
      [value] => the.info.phone@example.cn
  )
                                                                      [o
pt_out] => stdClass Object
  (
      [name] => opt_out
      [value] => 1
  )
                                                                      [p
rimary_address] => stdClass Object
  (
      [name] => primary_address
```

```
      [value] =>

  )

                                                                  )

                                                              )

                                                          )

                                                  )

                                          )

                                  )

                      [1] => stdClass Object
                          (
                              [link_list] => Array
                                  (
                                      [0] => stdClass Object
                                          (
                                              [name] => email_addresses
                                              [records] => Array
                                                  (
                                                      [0] => stdClass Ob
ject
                                                          (
                                                              [link_valu
e] => stdClass Object
                                                                  (
                                                                      [e
mail_address] => stdClass Object

  (

      [name] => email_address

      [value] => im.the.sales@example.name

  )

                                                                  [o
pt_out] => stdClass Object

  (
```

```
      [name] => opt_out
      [value] => 0
  )
                                                                [primary_address] => stdClass Object
  (
      [name] => primary_address
      [value] =>
  )
                                                                    )
                                                            )
                                                        [1] => stdClass Object
                                                            (
                                                                [link_value] => stdClass Object
                                                                    (
                                                                        [email_address] => stdClass Object
  (
      [name] => email_address
      [value] => the36@example.com
  )
                                                                        [opt_out] => stdClass Object
  (
      [name] => opt_out
```

```
      [value] => 1

  )

                                                                  [p
rimary_address] => stdClass Object

  (

      [name] => primary_address

      [value] =>

  )

                                                                )

                                                            )

                                                        )

                                                    )

                                                )

                                            )

                                        )

                                    )

        )

    [Leads] => stdClass Object
        (
            [entry_list] => Array
                (
                    [0] => stdClass Object
                        (
                            [id] => 83abeeff-5e71-0f06-2e5f-5282a3f04f
41
                            [module_name] => Leads
                            [name_value_list] => stdClass Object
                                (
                                    [name] => stdClass Object
                                        (
                                            [name] => name
                                            [value] => Caryn Wert
```

```
                                        )

                                    )

                                )

                            [1] => stdClass Object
                                (
                                    [id] => 2a3b304b-5167-8432-d4e7-5282a300ea
bb
                                    [module_name] => Leads
                                    [name_value_list] => stdClass Object
                                        (
                                            [name] => stdClass Object
                                                (
                                                    [name] => name
                                                    [value] => Marco Castongua
y
                                                )

                                        )

                                )

                        )

                    [relationship_list] => Array
                        (
                            [0] => stdClass Object
                                (
                                    [link_list] => Array
                                        (
                                            [0] => stdClass Object
                                                (
                                                    [name] => email_addresses
                                                    [records] => Array
                                                        (
                                                            [0] => stdClass Ob
ject
                                                                (
                                                                    [link_valu
e] => stdClass Object
                                                                        (
                                                                            [e
mail_address] => stdClass Object
```

```
  (
      [name] => email_address
      [value] => hr60@example.com
  )
                                                                  [opt_out] => stdClass Object
  (
      [name] => opt_out
      [value] => 0
  )
                                                                  [primary_address] => stdClass Object
  (
      [name] => primary_address
      [value] =>
  )
                                                                      )
                                                                  )
                                                              )
                                                          )
                                                      )
                                                  )
                                              )
                                          )
                                      )
                                  )
                              )
                          )
                      [1] => stdClass Object
                          (
                              [link_list] => Array
                                  (
```

```
                                    [0] => stdClass Object
                                        (
                                            [name] => email_addresses
                                            [records] => Array
                                                (
                                                    [0] => stdClass Ob
ject
                                                        (
                                                            [link_valu
e] => stdClass Object
                                                                (
                                                                    [e
mail_address] => stdClass Object
                                                                  (
                                                                      [name] => email_address
                                                                      [value] => im.the@example.com
                                                                  )
                                                                    [o
pt_out] => stdClass Object
                                                                  (
                                                                      [name] => opt_out
                                                                      [value] => 0
                                                                  )
                                                                    [p
rimary_address] => stdClass Object
                                                                  (
                                                                      [name] => primary_address
                                                                      [value] =>
                                                                  )
                                                                )
```

```
                                                        )

                                                    )

                                                )

                                            )

                                        )

                                    )

                                )

        )

)
```

Retrieving Related Records

Overview

A PHP example demonstrating how to retrieve a list of related records with the get_relationships method using cURL and the v4_1 REST API.

This example will retrieve a list of leads related to a specific target list.

Example

```
<?php

    $url = "http://{site_url}/service/v4_1/rest.php";
    $username = "admin";
    $password = "password";

    //function to make cURL request
    function call($method, $parameters, $url)
    {
        ob_start();
        $curl_request = curl_init();

        curl_setopt($curl_request, CURLOPT_URL, $url);
        curl_setopt($curl_request, CURLOPT_POST, 1);
        curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VER
SION_1_0);
```

```
    curl_setopt($curl_request, CURLOPT_HEADER, 1);
    curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

    $jsonEncodedData = json_encode($parameters);

    $post = array(
         "method" => $method,
         "input_type" => "JSON",
         "response_type" => "JSON",
         "rest_data" => $jsonEncodedData
    );

    curl_setopt($curl_request, CURLOPT_POSTFIELDS, $post);
    $result = curl_exec($curl_request);
    curl_close($curl_request);

    $result = explode("\r\n\r\n", $result, 2);
    $response = json_decode($result[1]);
    ob_end_flush();

    return $response;
}

//login ------------------------------------------

$login_parameters = array(
     "user_auth" => array(
          "user_name" => $username,
          "password" => md5($password),
          "version" => "1"
     ),
     "application_name" => "RestTest",
     "name_value_list" => array(),
);

$login_result = call("login", $login_parameters, $url);

/*
echo "<pre>";
print_r($login_result);
echo "</pre>";
*/

//get session id
```

```
    $session_id = $login_result->id;

    //retrieve related list ------------------------------

    $get_relationships_parameters = array(

         'session'=>$session_id,

         //The name of the module from which to retrieve records.
         'module_name' => 'ProspectLists',

         //The ID of the specified module bean.
         'module_id' => '76d0e694-ef66-ddd5-9bdf-4febd3af44d5',

         //The relationship name of the linked field from which to ret
urn records.
         'link_field_name' => 'leads',

         //The portion of the WHERE clause from the SQL statement used
 to find the related items.
         'related_module_query' => '',

         //The related fields to be returned.
         'related_fields' => array(
            'id',
            'first_name',
            'last_name',
         ),

         //For every related bean returned, specify link field names t
o field information.
         'related_module_link_name_to_fields_array' => array(
         ),

         //To exclude deleted records
         'deleted'=> '0',

         //order by
         'order_by' => '',

         //offset
         'offset' => 0,

         //limit
         'limit' => 5,
    );
```

```
    $get_relationships_result = call("get_relationships", $get_relatio
nships_parameters, $url);

    echo "<pre>";
    print_r($get_relationships_result);
    echo "</pre>";

?>
```

Results

```
stdClass Object
(
    [entry_list] => Array
        (
            [0] => stdClass Object
                (
                    [id] => 117c26c0-11d4-7b8b-cb8f-4f7cb6823dd8
                    [module_name] => Leads
                    [name_value_list] => stdClass Object
                        (
                            [id] => stdClass Object
                                (
                                    [name] => id
                                    [value] => 117c26c0-11d4-7b8b-
cb8f-4f7cb6823dd8
                                )

                            [first_name] => stdClass Object
                                (
                                    [name] => first_name
                                    [value] => Diane
                                )

                            [last_name] => stdClass Object
                                (
                                    [name] => last_name
                                    [value] => Mckamey
                                )

                        )

                )
```

```
            [1] => stdClass Object
                (
                    [id] => 142faeef-1a19-b53a-b780-4f7cb600c553
                    [module_name] => Leads
                    [name_value_list] => stdClass Object
                        (
                            [id] => stdClass Object
                                (
                                    [name] => id
                                    [value] => 142faeef-1a19-b53a-
b780-4f7cb600c553
                                )

                            [first_name] => stdClass Object
                                (
                                    [name] => first_name
                                    [value] => Leonor
                                )

                            [last_name] => stdClass Object
                                (
                                    [name] => last_name
                                    [value] => Reser
                                )

                        )

                )

        )

    [relationship_list] => Array
        (
        )

)
```

Searching Records

Overview

A PHP example demonstrating how to search the accounts module with the search_by_module method using cURL and the v4_1 REST API.

This script will return two results, sorted by the id field, and return the value of the id, name, account_type, phone_office, and assigned_user_name fields.

Example

```
<?php

    $url = "http://{site_url}/service/v4_1/rest.php";
    $username = "admin";
    $password = "password";

    //function to make cURL request
    function call($method, $parameters, $url)
    {
        ob_start();
        $curl_request = curl_init();

        curl_setopt($curl_request, CURLOPT_URL, $url);
        curl_setopt($curl_request, CURLOPT_POST, 1);
        curl_setopt($curl_request, CURLOPT_HTTP_VERSION, CURL_HTTP_VER
SION_1_0);
        curl_setopt($curl_request, CURLOPT_HEADER, 1);
        curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
        curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
        curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

        $jsonEncodedData = json_encode($parameters);

        $post = array(
             "method" => $method,
             "input_type" => "JSON",
             "response_type" => "JSON",
             "rest_data" => $jsonEncodedData
        );

        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $post);
        $result = curl_exec($curl_request);
        curl_close($curl_request);

        $result = explode("\r\n\r\n", $result, 2);
```

```
    $response = json_decode($result[1]);
    ob_end_flush();

    return $response;
}

//login ----------------------------------------

$login_parameters = array(
     "user_auth" => array(
          "user_name" => $username,
          "password" => md5($password),
          "version" => "1"
     ),
     "application_name" => "RestTest",
     "name_value_list" => array(),
);

$login_result = call("login", $login_parameters, $url);

/*
echo "<pre>";
print_r($login_result);
echo "</pre>";
*/

//get session id
$session_id = $login_result->id;

//search ---------------------------------------

$search_by_module_parameters = array(
    //Session id
    "session" => $session_id,

    //The string to search for.
    'search_string' => 'Customer',

    //The list of modules to query.
    'modules' => array(
    'Accounts',
    ),

    //The record offset from which to start.
    'offset' => 0,
```

```
        //The maximum number of records to return.
        'max_results' => 2,

        //Filters records by the assigned user ID.
        //Leave this empty if no filter should be applied.
        'id' => '',

        //An array of fields to return.
        //If empty the default return fields will be from the active l
istviewdefs.
        'select_fields' => array(
            'id',
            'name',
            'account_type',
            'phone_office',
            'assigned_user_name',
        ),

        //If the search is to only search modules participating in the
 unified search.
        //Unified search is the SugarCRM Global Search alternative to
Full-Text Search.
        'unified_search_only' => false,

        //If only records marked as favorites should be returned.
        'favorites' => false
    );

    $search_by_module_result = call('search_by_module', $search_by_mod
ule_parameters, $url);

    echo '<pre>';
    print_r($search_by_module_result);
    echo '</pre>';

?>
```

Result

```
stdClass Object
(
    [result_count] => 2
    [total_count] => 200
    [next_offset] => 2
```

```
    [entry_list] => Array
        (
            [0] => stdClass Object
                (
                    [id] => 18124607-69d1-b158-47ff-4f7cb69344f7
                    [module_name] => Leads
                    [name_value_list] => stdClass Object
                        (
                            [id] => stdClass Object
                                (
                                    [name] => id
                                    [value] => 18124607-69d1-b158-47ff
-4f7cb69344f7
                                )

                            [name] => stdClass Object
                                (
                                    [name] => name
                                    [value] => Bernie Worthey
                                )

                            [title] => stdClass Object
                                (
                                    [name] => title
                                    [value] => Senior Product Manager
                                )

                        )

                )

            [1] => stdClass Object
                (
                    [id] => 1cdfddc1-2759-b007-8713-4f7cb64c2e9c
                    [module_name] => Leads
                    [name_value_list] => stdClass Object
                        (
                            [id] => stdClass Object
                                (
                                    [name] => id
                                    [value] => 1cdfddc1-2759-b007-8713
-4f7cb64c2e9c
                                )

                            [name] => stdClass Object
                                (
```

```
                                    [name] => name
                                    [value] => Bobbie Kohlmeier
                                )

                            [title] => stdClass Object
                                (
                                    [name] => title
                                    [value] => Director Operations
                                )

                        )

                )

        )

    [relationship_list] => Array
        (
        )

)
```

SOAP

Examples of v4.1 SOAP API calls.

C#

C# SOAP v4_1 Examples.

Creating or Updating a Record

Overview

A C# example demonstrating how to create or update an account with the set_entry method using SOAP and the v4 SOAP API.

Example

```
using System;
using System.Text;
using System.Security.Cryptography;
using System.Collections.Specialized;

namespace SugarSoap
{
    class Program
    {
        static void Main(string[] args)
        {
            //login ---------------------------------------

            string UserName = "admin";
            string Password = "password";
            string URL = "http://{site_url}/service/v4/soap.php";

            //SugarCRM is a web reference added that points to http://
{site_url}/service/v4/soap.php?wsdl
            SugarCRM.sugarsoap SugarClient = new SugarCRM.sugarsoap();
            SugarClient.Timeout = 900000;
            SugarClient.Url = URL;

            string SessionID = String.Empty;

            //Create authentication object
            SugarCRM.user_auth UserAuth = new SugarCRM.user_auth();

            //Populate credentials
            UserAuth.user_name = UserName;
            UserAuth.password = getMD5(Password);

            //Try to authenticate
            SugarCRM.name_value[] LoginList = new SugarCRM.name_value[
0];
            SugarCRM.entry_value LoginResult = SugarClient.login(UserA
uth, "SoapTest", LoginList);

            //get session id
            SessionID = LoginResult.id;

            //create account -------------------------------
```

```
            NameValueCollection fieldListCollection = new NameValueCol
lection();
            //to update a record, you will nee to pass in a record id
as commented below
            //fieldListCollection.Add("id", "68c4781f-75d1-223a-5d8f-5
058bc4e39ea");
            fieldListCollection.Add("name", "Test Account");

            //this is just a trick to avoid having to manually specify
 index values for name_value[]
            SugarCRM.name_value[] fieldList = new SugarCRM.name_value[
fieldListCollection.Count];

            int count = 0;
            foreach (string name in fieldListCollection)
            {
                foreach (string value in fieldListCollection.GetValues
(name))
                {
                    SugarCRM.name_value field = new SugarCRM.name_valu
e();
                    field.name = name; field.value = value;
                    fieldList[count] = field;
                }
                count++;
            }

            try
            {
                SugarCRM.new_set_entry_result result = SugarClient.set
_entry(SessionID, "Accounts", fieldList);
                string RecordID = result.id;

                //show record id to user
                Console.WriteLine(RecordID);
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
                Console.WriteLine(ex.Source);
            }

            //Pause Window
            Console.ReadLine();
        }
```

```
        static private string getMD5(string PlainText)
        {
            MD5 md5 = MD5.Create();
            byte[] inputBuffer = System.Text.Encoding.ASCII.GetBytes(P
lainText);
            byte[] outputBuffer = md5.ComputeHash(inputBuffer);

            //Convert the byte[] to a hex-string
            StringBuilder builder = new StringBuilder(outputBuffer.Len
gth);
            for (int i = 0; i < outputBuffer.Length; i++)
            {
                builder.Append(outputBuffer[i].ToString("X2"));
            }

            return Builder.ToString().ToLower(); //lowercase as of 7.9
.0.0
        }
    }
}
```

Result

```
68c4781f-75d1-223a-5d8f-5058bc4e39ea
```

Logging In

Overview

A C# example demonstrating how to log in and retrieve a session key using SOAP and the v4 SOAP API.

Example

```
using System;
using System.Text;
using System.Security.Cryptography;
```

```
namespace SugarSoap
{
    class Program
    {
        static void Main(string[] args)
        {
            string UserName = "admin";
            string Password = "password";
            string URL = "http://{site_url}/service/v4/soap.php";
            string SessionID = String.Empty;

            //SugarCRM is a web reference added that points to http://
{site_url}/service/v4/soap.php?wsdl
            SugarCRM.sugarsoap SugarClient = new SugarCRM.sugarsoap();
            SugarClient.Timeout = 900000;
            SugarClient.Url = URL;

            //Create authentication object
            SugarCRM.user_auth UserAuth = new SugarCRM.user_auth();

            //Populate credentials
            UserAuth.user_name = UserName;
            UserAuth.password = getMD5(Password);

            //Try to authenticate
            SugarCRM.name_value[] LoginList = new SugarCRM.name_value[
0];
            SugarCRM.entry_value LoginResult = SugarClient.login(UserA
uth, "SoapTest", LoginList);

            //get session id
            SessionID = LoginResult.id;

            if (SessionID != String.Empty)
            {
                //print session
                Console.WriteLine(SessionID);
            }

            //Pause Window
            Console.ReadLine();
        }

        static private string getMD5(string TextString)
        {
            MD5 md5 = MD5.Create();
```

```
            byte[] inputBuffer = System.Text.Encoding.ASCII.GetBytes(T
extString);
            byte[] outputBuffer = md5.ComputeHash(inputBuffer);

            StringBuilder Builder = new StringBuilder(outputBuffer.Len
gth);
            for (int i = 0; i < outputBuffer.Length; i++)
            {
                Builder.Append(outputBuffer[i].ToString("X2"));
            }

            return Builder.ToString().ToLower(); //lowercase as of 7.9
.0.0
        }
    }
}
```

Note: As of 7.9.0.0, the md5 of the password must be lowercase.

Result

```
b2uv0vrjfiov41d03sk578ufq6
```