Sugar Developer Guide 7.6

Sugar Developer Guide 7.6

The Sugar Developer Guide is designed for developers who are new to Sugar, or to CRM and Web-based applications. This guide introduces you to some basic CRM concepts and helps you get familiar with the Sugar system. It describes how to configure and customize the Sugar platform for a broad range of tasks applicable to companies, government agencies and other organizations that have a need to manage business relationships with people.

Readers are expected to have basic programming and software development knowledge, be familiar with the PHP programming language and the SQL database language.

Last Modified: 09/26/2015 04:14pm

Introduction

Overview

The Sugar Developer Guide for 7.6.x.

Background

Sugar was originally written on the LAMP stack (Linux, Apache, MySQL and PHP). Since version 1.0, the Sugar development team has added support for every operating system (including Windows, Unix, and Mac OSX) on which the PHP programming language runs, for the Microsoft IIS web server, and for the Microsoft SQL Server , IBMDB2, and Oracle databases. Designed as the most modern web-based CRM platform available today, Sugar has quickly become the business application standard for companies around the world. See the Supported Platforms page for detailed information on supported software versions and recommended stacks.

Sugar 7.x is available in four editions: Professional, Corporate, Enterprise, and Ultimate, which are sold under a commercial subscription agreement. All four editions are developed by the same team using a common source tree. Sugar customers using the commercial editions of Sugar 7.x have access to Sugar Support, Training, and Professional Services offerings. While contributions are occasionally accepted from the Sugar Community, not all contributions are included because SugarCRM maintains high standards for code quality.

From the very beginning of the SugarCRM Open Source project in 2004, the SugarCRM development team designed the application source code to be examined and modified by developers. The Sugar application framework has a very 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 one of the core files in the distribution; you should always check for an upgrade-safe way to make your changes. Educating developers on how to make upgrade-safe customizations is one of the key goals of this Developer Guide.

6 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:

1. Do not remove or alter any SugarCRM or Sugar copyright, trademark or proprietary notices that appear in the Sugar products.
2. 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).
3. Do not modify, remove or disable any portion of SugarCRM's "Critical Control Software."
4. Do not combine or use 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).
5. 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.
6. Do not use the Sugar products to develop or enhance SugarCRM's "Sugar Community Edition" product or any software code made to work with that open source product.

Development Tools

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

Developer Mode

Developer Mode will allow for Sugar to recompile some cached files when the page is reloaded:

- Rebuilds Handlebar files (.hbt)
- Rebuilds Smarty files (.tpl)
- The Sidecar JavaScript library references the full JavaScript files located in ./sidecar/ rather than the concatenated and minified cached version.

You can turn on Developer Mode by navigating to:

```
Admin > System Settings > Advanced
```

Note: This setting should remain off unless developing as 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 On-Demand.

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
```

Composer

When building applications, some developers prefer to use composer to manage their external dependencies and make them more intuitive. More information on composer can be found in the Composer section.

Last Modified: 05/10/2016 03:47pm

Composer

Overview

SugarCRM has exposed the usage of Composer to the public since Sugar version 7.5. As Composer is the defacto standard today for managing (primarily) PHP dependencies, this new enhancement to the SugarCRM platform will make customizations based on external libraries more intuitive.

Target Audience

The target audiences for this document are:

- Sugar instance administrators
- Sugar developers

Prerequisites

The Composer package needs to be available on your system if you wish to make any changes to the dependencies. Please consult Composer's Getting Started guide for more details on installation and usage.

Composer is a command line tool, so CLI access on the system is also required, along with the proper file system permissions to alter files in the Sugar instance directory.

Note: Before customizing the Composer configuration, make sure you have proper knowledge on how Composer works and the details explained in this document.

Restrictions

Customizations to the composer.json file are restricted by a certain set of rules to ensure continuity in our product.

The following root elements are owned by SugarCRM and should never be changed:

- name
- description
- type
- license
- homepage
- support
- autoload (*)
- minimum-stability (*)
- config (*)

(*) These configuration parameters are being considered to be more relaxed in the near future. As for now they are on the do-not-alter list.

The following root elements are restricted:

- require
- repositories

Restrictions which apply to "require" and "repositories" root elements:

- Adding new packages
- Adding repositories which do not conflict with SugarCRM owned packages
- Removing packages which are not owned by SugarCRM
- Removing repositories which are not owned by SugarCRM

All other root elements are currently available to be added/changed at the developer's discretion. Note: The list of restriction can change at any given time, opening up certain configuration keys or making them more restrictive. These changes will be communicated through release notes and this document will be updated as a reference reflecting the latest state.

Please file a ticket through support if the current restrictions block you from using the Composer integration in our product. Also review our Q&A section of this document for any additional clarifications.

Finding Packages

The packages which are available to Composer are published through Packagist. This will be the primary place to look for third party libraries. Note that Composer is not restricted to add packages which are published on Packagist. You can refer directly to both public and private repositories without the need of having your package(s) publish on Packagist. See the Repositories section in Composer's

documentation for more details.

Adding New Package

The following steps are involved to add a new package:

1. Backup composer.json and composer.lock.
2. Alter the composer.json "require" section.
3. Update dependencies.
4. Clear cache.

Adding Package to composer.json

This is an example to add the "monolog/monolog" package to composer.json. We simply add the package name and the required version constraint in the "require" section. More information on the version constraints can be found here and details on the "require" section can be found here.

```
{
   "name": "sugarcrm/sugarcrm",
   ...
   "require": {
       "monolog/monolog": "~1.11",
       "ruflin/elastica": "v1.2.1.0",
       "php": ">=5.3.0"
   },
   "require-dev": {
       "phpunit/phpunit": "4.1.4"
   },
   ...
}
```

Updating Dependencies

When done editing composer.json, we want to update our dependencies. The easiest way is to do so by running "composer update --no-dev" from the directory where composer.json lives. Composer will figure out automatically which packages need to be fetched and updated.

All dependencies are stored in the "vendor" folder and the composer.lock file is updated with the exact installed versions and repository information. The "--no-dev" switch skips the packages in the "require-dev" section, which are only

needed for development. This is the recommended way for production systems.

When following the above example, you will see the Monolog package has a dependency of its own, being the "psr/log" package. Composer automatically installed this dependency for you.

Clear Cache

The following important step is currently not yet automated. The cache used by Sugar's autoloader needs to be cleared to be sure the newly installed dependencies are available. To do this simply remove the following two files:

- cache/file_map.php
- cache/class_map.php

When developing new code in Sugar, as an alternative, you can also enable "developer mode" (see Admin > System Settings). When enabled, the autoloader will refresh itself automatically.

Removing Packages

Exactly the same procedure can be followed to remove packages instead of adding packages. The following steps are involved:

1. Backup composer.json and composer.lock.
2. Alter the composer.json "require" section.
3. Update dependencies.
4. Clear cache.

Note: Remember the restrictions of the "require" section. Only remove packages which you are controlling yourself.

Adding and Removing Repositories

To configure additional repositories in composer.json, the exact same procedure can be followed to add or remove entries in the "repositories" section. Here is an example:

```
{
   "name" : "sugarcrm/sugarcrm",
   ...
```

```
   "repositories" : [
       {
           "type" : "git",
           "url" : "https://github.com/yours/Monolog"
       }
       {
           "type" : "git",
           "url" : "https://github.com/sugarcrm/Elastica"
       }
   ]
}
```

Autoloader

SugarCRM has its own autoloader which is fully PSR-0 and PSR-4 compliant and is integrated 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 avoid endless file stats, SugarCRM'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 code base uses the autoloader to determine whether a file is available rather than performing expensive file_exists calls.

A second file maintains a flat list of class name to file mapping which is stored in ./cache/class_map.php. When we make any changes to the file system we need to clear both files before testing new code.

Optimization

When updating dependencies through Composer, it is advised that the production system 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.

Developer Mode

When "developer mode" is enabled, the autoloader will bypass any persistent setting from both file_map or class_map. Adding new files or updating dependencies will be directly picked up by the autoloader without performing a "Quick Repair & Rebuild".

Handling Upgrades

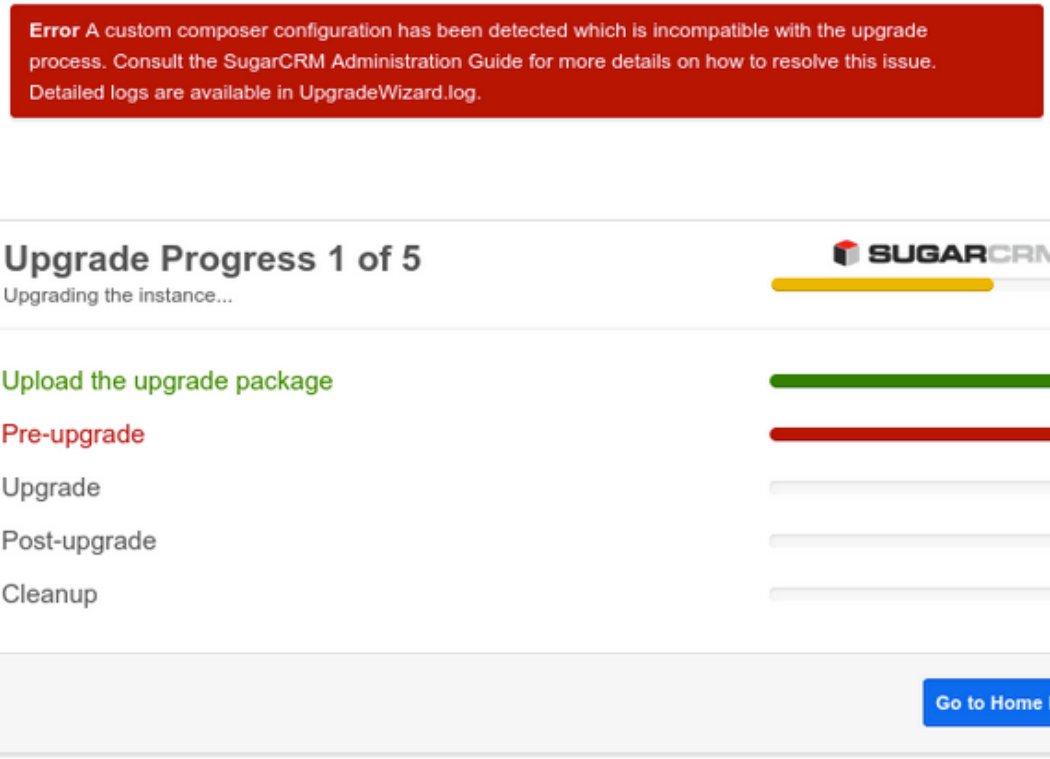
Once you take control of the Composer configuration, there may be complications during upgrades. The upgrade logic will determine whether or not the Composer configuration has been customized and whether a custom config is compatible for the upgrade.

When a custom configuration is not compatible (missing or conflicting dependencies), the upgrade process will generate a proposal. The administrator who is responsible for reviewing the proposal will need to make the necessary changes to the Composer configuration prior to running the upgrade again.

Upgrade Failure Notification

When an incompatible Composer configurations is detected, a notification will be shown depending on which type of upgrader you are using.

Web Upgrader



CLI Upgrader

```
***************         Step "healthcheck" OK - 0 seconds
***************         Step "unpack" OK - 8 seconds
ERROR: A custom composer configuration has been detected which is
incompatible with the upgrade process. Consult the SugarCRM
Administration Guide for more details on how to resolve this issue.
Detailed logs are available in UpgradeWizard.log.
***************         Step "pre" FAILED! - 5 seconds
```

The detailed logging is available in the upgrade wizard log file as reported in the above notifications. We will use the content to determine the course of action to solve the upgrade issues. When using the Web Upgrader the UpgradeWizard.log file is located in the root directory of your SugarCRM instance. For the CLI Upgrader, the full path will be reported where you can find it.

Example Failure

Search for "CheckComposerConfig" in the upgrade log. The red highlights indicate the detected problems.

```
Starting script CheckComposerConfig
Using sugarcrm/composer.json as composer.json source
Using sugarcrm/composer.lock as composer.lock source
```

```
Using
cache/upgrades/temp/SugarPro-Upgrade-7.5.0.x-to-7.6.0.0/composer.json
as composer.json target
Hash 3a5b0634383693d27c7c6054a69839fc does not match release hash for
7.5.0.0
Custom composer configuration detected
Found valid package ruflin/elastica with version constraint v1.2.1.0
Package onelogin/php-saml with version constraint 2.1.0.1 is missing
Skipping platform package php
Found valid repository https://github.com/sugarcrm/Elastica with type
git
Repository https://github.com/sugarcrm/php-saml of type git is missing
Missing configuration key 'minimum-stability'
Generating proposal file
cache/upgrades/temp/SugarPro-Upgrade-7.5.0.x-to-7.6.0.0/composer.json.
proposal
Saving file
cache/upgrades/temp/SugarPro-Upgrade-7.5.0.x-to-7.6.0.0/composer.json.
proposal to disk
ERROR: A custom composer configuration has been detected which is
incompatible with the upgrade process.
Finished script CheckComposerConfig
```

Process

The following actions are required:

1. Backup the current composer,json and composer.lock file.
2. Alter composer.json to match the requirements.
3. Run "composer update -o --no-dev" to deploy the new dependencies.
4. Clear cache (cache/file_map.php and cache/class_map.php).
5. Login and test the instance.
6. Perform upgrade again.

What is Wrong?

From the above log output we can learn that:

- A package "onelogin/php-saml" is missing
- A repository definition for php-saml is missing
- The configuration key "minimum-stability" is missing

In this case, for Sugar 7.6.0.0 these definitions were added and are owned by SugarCRM. In case one of these settings were already present before those were required by SugarCRM, a similar notice will be thrown stating any incompatible issues.

The Proposal

The upgrade logic provides us with a proposal on how to update the composer.json file to fix the missing dependencies in our custom configuration. The location of this proposal file can be seen above in the highlighted blue section.

Note: Do not blindly copy the proposal file over your existing composer.json file without verifying and understanding what the issue is. There can be an incompatibility between a dependency of one of your own explicitly defined packages and requirements from the upgrade process.

Resolving the Issues

As described in the chapter above we can make the change in our composer.json to satisfy all requirements. Based on the above example we make the following changes:

```
{
"name": "sugarcrm/sugarcrm",
...
},
   "require": {
       "psr/log": "1.0",
       "ruflin/elastica": "v1.2.1.0",
       "php": ">=5.3.0",
       "onelogin/php-saml": "2.1.0.1"
   },
   "require-dev": {
       "phpunit/phpunit": "4.1.4"
   },
   "repositories": [
       {
           "type": "git",
           "url": "https://github.com/sugarcrm/Elastica"
       },
       {
           "url": "https://github.com/sugarcrm/php-saml",
           "type": "git"
       }
```

```
    ],
    "minimum-stability": "stable"
}
```

From here, we follow the procedure to run "composer update", clear the autoloader cache, and test drive our SugarCRM instance. When approved, you can try to upgrade again.

Retry Upgrade

When all conflicts are properly resolved, the upgrade process will successfully finish. The log output will look like this on success:

```
Starting script CheckComposerConfig
Using sugarcrm/composer.json as composer.json source
Using sugarcrm/composer.lock as composer.lock source
Using
cache/upgrades/temp/SugarPro-Upgrade-7.5.0.x-to-7.6.0.0/composer.json
as composer.json target
Hash 79626e71bad09bf1c09585c89a28875e does not match release hash for
7.5.0.0
Custom composer configuration detected
Found valid package ruflin/elastica with version constraint v1.2.1.0
Found valid package onelogin/php-saml with version constraint 2.1.0.1
Skipping platform package php
Found valid repository https://github.com/sugarcrm/Elastica with type
git
Found valid repository https://github.com/sugarcrm/php-saml with type
git
Custom composer configuration is valid for upgrade
Finished script CheckComposerConfig
…
Starting script 8_ComposerConfig
Restoring custom composer file 'composer.json.valid' to
'composer.json'
Restoring custom composer file 'composer.lock.valid' to
'composer.lock'
Finished script 8_ComposerConfig
```

Stock Composer Configuration

As a reference, the output in the log file when no custom Composer configuration is detected. The upgrader will automatically continue in this case without throwing any notices as the upgrade archive contains the necessary code base and an

updated composer.json and composer.lock file.

```
Starting script CheckComposerConfig
Using sugarcrm/composer.json as composer.json source
Using sugarcrm/composer.lock as composer.lock source
Using
cache/upgrades/temp/SugarPro-Upgrade-7.5.0.x-to-7.6.0.0/composer.json
as composer.json target
Skipping merge, stock composer settings detected
Finished script CheckComposerConfig
```

Q&A

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 ship composer.json and composer.lock if the installer doesn't rely on it ?

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

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 case of a custom configuration, the upgrade process will report any issues which needs to be resolved before the upgrade can proceed. This is the responsibility of the system administrator.

Why are there no wildcards in the version constraints ? I thought composer.lock was designed to keep track of exact version numbers ?

The lock file is indeed designed to lock the dependencies to a specific version. 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 ?

If the package was added by SugarCRM then the answer is no. Sugar will always configure exact version numbers for all of its dependencies. Changing such version numbers results in an unsupported platform. The version constraints for all other packages can be changed as these settings are not owned by SugarCRM.

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 auto loading 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 working on a solution making this possible in the future.

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

Not at the moment, this feature is being investigated.

The Composer integration in Sugar is blocking a specific use case for me. What now ?

We continuously strive to make our platform better each day and facility both end users and developers to focus on what is important to them by delivering a state-of-the-art environment. Do not hesitate to reach out to support if there are use cases which have been overlooked or if there too many constraints in current implementation making it a useful feature.

Last Modified: 02/08/2017 01:25pm

UI Model

UI Model Overview.

Last Modified: 09/26/2015 04:14pm

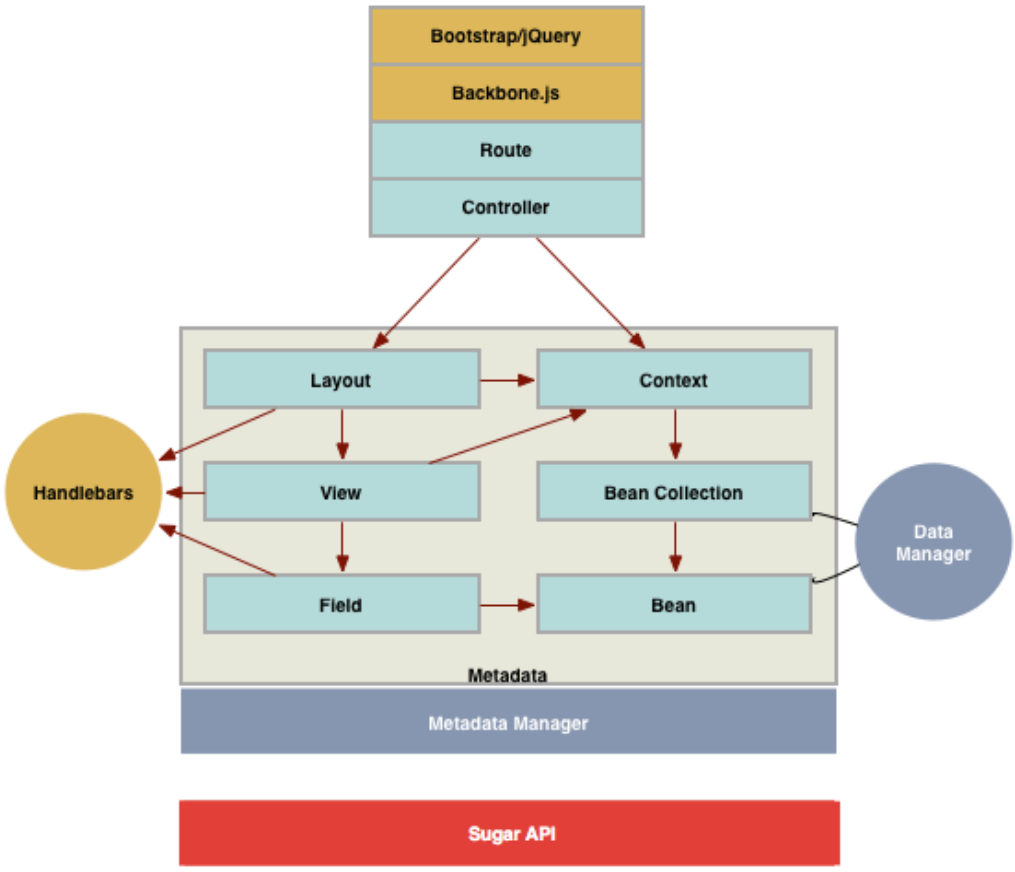
Sidecar

Overview

Sidecar is a client-side application.

Sidecar

Sidecar is a new client-side platform that moves processing to the client side for page rendering. It contains a complete MVC framework based on the Backbone.js library. By creating what is known as a Single Page Web App, server load is drastically reduced and the clients performance is increased. To explain this further, the application no longer sends down HTML content but rather pure JSON data. The JSON data, returned by the v10 API, defines the applications modules, records, and ACLs allowing the UI processing to happen on the client side and is significantly less in terms of data transfer.



Components

Sidecar is built atop several components

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 which serves as a canvas for one or more views and/or other layouts. All pages will have at least one master layout, and that master layout can contain multiple nested

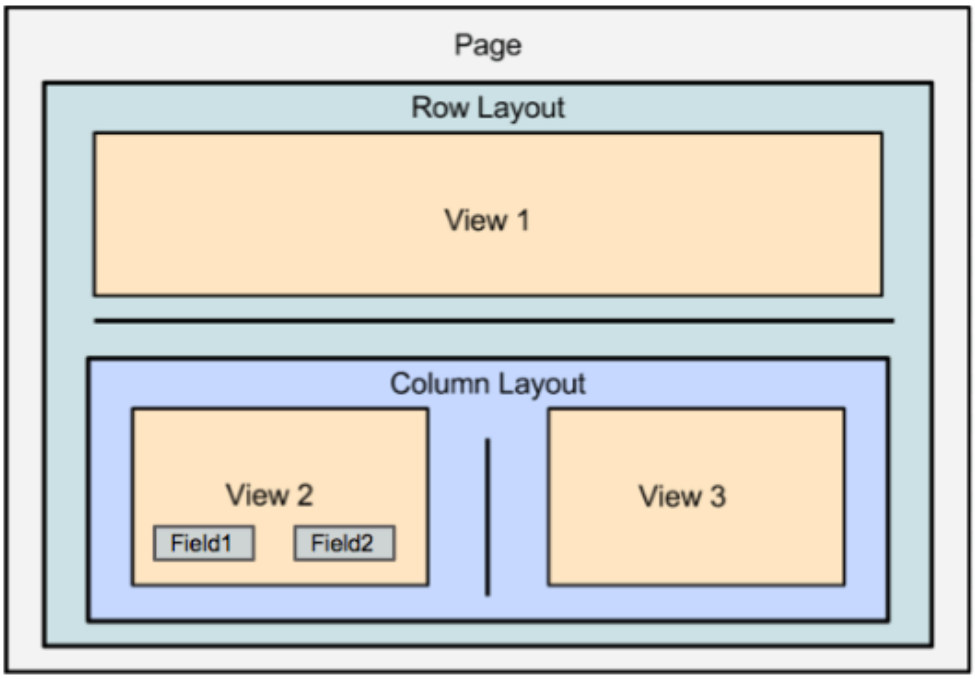
layouts.

Layouts

Layouts are components which render the overall page. They are used to define the rows, columns, and fluid layouts that the content is displayed to the user in.

Example layouts include:

- Rows
- Columns
- Bootstrap fluid layouts, and
- Drawers / Drop downs



More information on layouts can be found in the Layouts section.

Views

Views are components which 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 including:

- Graphs or other data visualizations
- External data views such as Twitter, LinkedIn, or other web service

integrations
- The global header

More information on views can be found in the Views section.

Fields

Fields render widgets for individual values pulled from the models and handle formatting (or stripping the formatting of) field values. Just like Layouts and Views, Fields also extend Backbone Views. More information on fields can be found in the

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 on this context

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

Last Modified: 03/31/2016 11:20pm

Clients

Overview

Clients are the various platforms that use Sidecar to render the user interface. Each platform type will have a specific path for its components.

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 client types are listed below in the following sections.

base

The base client is the Sugar application that you use to access your data. The framework specific views, layouts, and fields for this application will be found in:

- ./clients/base/
- ./custom/clients/base/
- ./modules//clients/base/
- ./custom/modules//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 will be found in:

- ./clients/mobile/
- ./custom/clients/mobile/
- ./modules//clients/mobile/
- ./custom/modules//clients/mobile/

portal

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

- ./clients/portal/
- ./custom/clients/portal/
- ./modules//clients/portal/
- ./custom/modules//clients/portal/

Last Modified: 11/18/2015 01:08am

Layouts

Layouts Overview.

Last Modified: 11/18/2015 01:08am

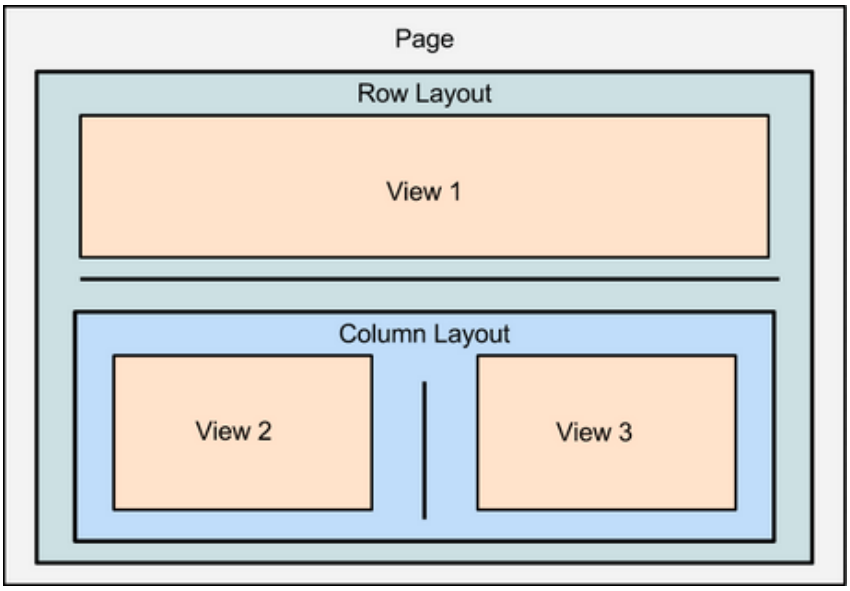
Introduction

Overview

A layout is a component which wraps and places multiple views or nested layouts on a page.

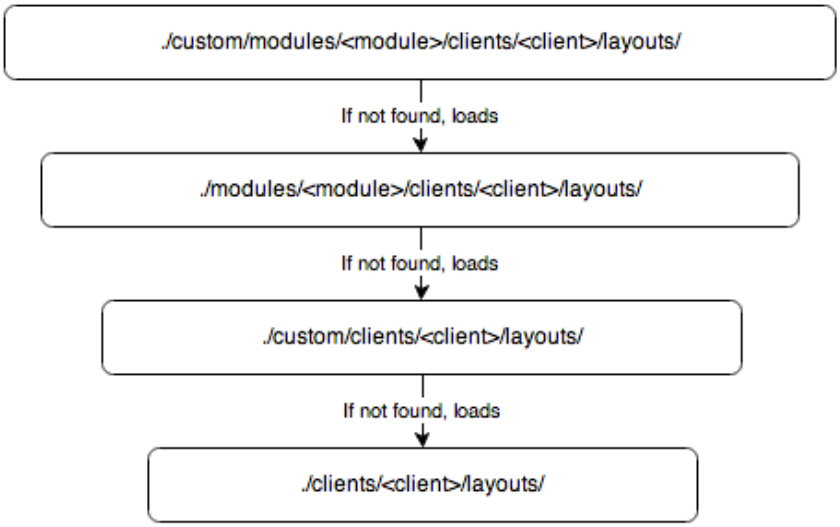
Layouts

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 popups. 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". More information on the various client types can be found in the Clients section.

Sidecar Layout Routing

Sidecar uses routing to determine where to direct the user. The default layout routes are:

To route the user to the list layout for a module, use:

```
http://{site url}/#<module>/
```

To route the user to the record layout for a specific record, use:

```
http://{site url}/#<module>/f82d09cb-48cd-a1fb-beae-521cf39247b5
```

To route the user to a custom layout for the module, use:

```
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 will outline the various files that render this view.

JavaScript

The 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. This should be overriden
by any custom layout types.
     * @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 4 views: massupdate, massaddtolist, recordlist, and 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,
);
```

Last Modified: 09/26/2015 04:14pm

Examples

Layout examples.

Last Modified: 11/18/2015 01:08am

Creating Layouts

Overview

How to create a custom layout component.

Creating a Layout

This example will create a custom layout named "my-layout". As layouts will define the various components to load on the page, this layout 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 once 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, you will need to navigate to Admin > Repair > Quick Repair and Rebuild.

Navigating to the Layout

To see your newly created layout and view, you will navigate to:

```
http://{site url}/#<module>/layout/my-layout
```

Last Modified: 09/26/2015 04:14pm

Overriding Layouts

Overview

How to override a stock layout component.

Overriding a Layout and Extending a 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 layout's override.

Overriding the Layout

The first step is to copy ./clients/base/layouts/record/record.php to ./custom/clients/base/layouts/record/record.php. Once copied, we will need to 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, your 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(
                'components' => array(
                    array(
                        'layout' => array(
                            'components' => array(
                                array(
                                    'view' => 'my-record',
                                    'primary' => true,
                                ),
                                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_VIEW',
                                            ),
                                            array(
                                                'name' => 'list',
                                                'icon' =>
'icon-table',
                                                'label' =>
'LBL_LISTVIEW',
                                            ),
                                            array(
                                                'name' =>
'activitystream',
                                                'icon' =>
'icon-th-list',
                                                'label' =>
'LBL_ACTIVITY_STREAM',
                                            ),
                                        ),
                                        'components' => array(
                                            array(
                                                'layout' => 'filter',
                                                'targetEl' =>
'.filter',
                                                'position' =>
'prepend'
                                            ),
                                            array(
```

```
                                        'view' =>
'filter-actions',
                                        "targetEl" =>
'.filter-options'
                                    ),
                                    array(
                                        'view' =>
'filter-rows',
                                        "targetEl" =>
'.filter-options'
                                    ),
                                    array(
                                        'layout' =>
'activitystream',
                                        'context' =>
                                        array(
                                            'module' =>
'Activities',
                                        ),
                                    ),
                                    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,
);
```

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, you will need to navigate to Admin > Repair > Quick Repair and Rebuild.

Last Modified: 01/15/2016 09:34pm

Views

Views Overview.

Last Modified: 09/26/2015 04:14pm

Introduction

Overview

An overview of how view components work.

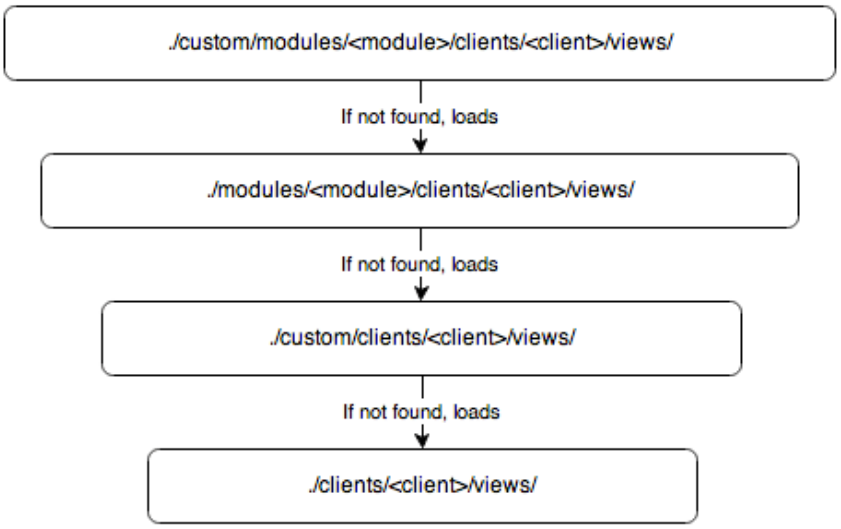
Views

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 (.hbt).

Load Order Hierarchy Diagram

The view components are loaded in the following manner:



Note: The Sugar application client type is "base". More information on the various client types can be found in the Clients section.

Components

Controller

The views 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.

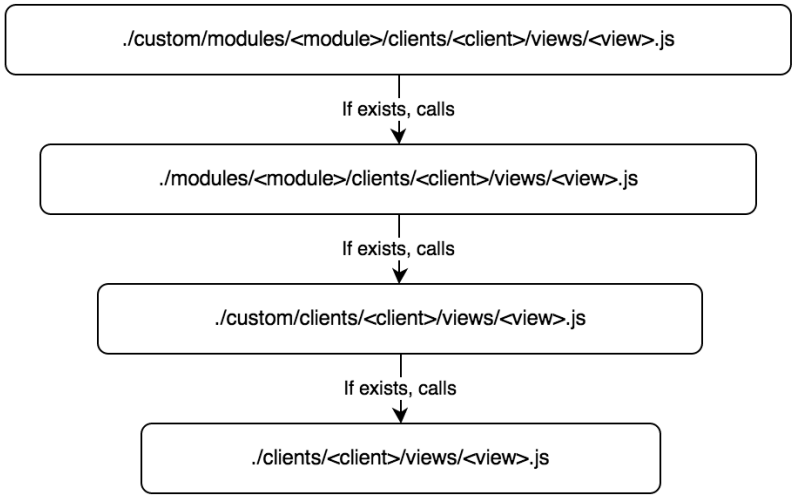
Extending Views

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:



Basic View Example

A simple view to start looking at is the 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

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

Last Modified: 09/26/2015 04:14pm

Filters

Overview

Filters are a way to predefine searches on views that render a list of records. This functionality can be used on 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 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.

Operator	Label / Key	Field Types	Description
$contains	is any of	multienum	Matches anything

	LBL_OPERATOR_CONTAINS		that contains the value.
$not_contains	is not any of LBL_OPERATOR_NOT_CONTAINS	multienum	Matches anything that does not contain the specified value.
$in	is any of LBL_OPERATOR_CONTAINS	enum and int	Finds anything where field matches one of the values as specified as an array.
$not_in	is not any of LBL_OPERATOR_NOT_CONTAINS	enum	Finds anything where the field does not match any of the values in the specified array of values.
$equals	is LBL_OPERATOR_IS	varchar, name, email, text, textarea, currency, int, double, float, decimal, date, bool, relate, teamset, phone, radioenum, and parent	Performs an exact match on that field.
$starts	starts with LBL_OPERATOR_STARTS_WITH	varchar, name, email, text, textarea, datetime, datetimecombo, and phone	Matches on anything that starts with the value.
$not_equals	is not LBL_OPERATOR_IS_NOT	currency, int, double, float, decimal, relate, teamset, and radioenum	Matches on non-matching values.
$gt	after LBL_OPERATOR_AFTER	currency, int, double, float, decimal, and date	Matches when the field is greater than the value.
$lt	before LBL_OPERATOR_BEFORE	currency, int, double, float, decimal, and date	Matches when the field is less than the value.
$gte	after LBL_OPERATOR_AFTER	currency, int,	Matches when the

		double, float, decimal, datetime, and datetimecombo	field is greater than or equal to the value
$lte	before LBL_OPERATOR_BEFORE	currency, int, double, float, decimal, datetime, and datetimecombo	Matches when the field is less than or equal to the value.
$between	is between LBL_OPERATOR_BETWEEN	currency, int, double, float, and decimal	Matches when a numerical value is between two other numerical values.
yesterday	yesterday LBL_OPERATOR_YESTERDAY	date, datetime, and datetimecombo	Matches dates on yesterday relative to the current date.
today	today LBL_OPERATOR_TODAY	date, datetime, and datetimecombo	Matches dates in the current date.
tomorrow	tomorrow LBL_OPERATOR_TOMORROW	date, datetime, and datetimecombo	Matches dates on tomorrow relative to the current date.
last_7_days	last 7 days LBL_OPERATOR_LAST_7_DAYS	date, datetime, and datetimecombo	Matches date in the last 7 days relative to the current date.
next_7_days	next 7 days LBL_OPERATOR_NEXT_7_DAYS	date, datetime, and datetimecombo	Matches dates in the next 7 days relative to the current date.
last_30_days	last 30 days LBL_OPERATOR_LAST_30_DAYS	date, datetime, and datetimecombo	Matches dates in the last 30 days relative to the current date.
next_30_days	next 30 days LBL_OPERATOR_NEXT_30_DAYS	date, datetime, and datetimecombo	Matches dates in the next 30 days relative to the current date.
last_month	last month LBL_OPERATOR_LAST_MONTH	date, datetime, and datetimecombo	Matches dates in the previous month relative to the current month.
this_month	this month LBL_OPERATOR_THIS_MONTH	date, datetime, and datetimecombo	Matches dates in the current month.
next_month	next month	date, datetime, and	Matches dates in

	LBL_OPERATOR_NEXT_MONTH	datetimecombo	the next month relative to the current month.
last_year	last year LBL_OPERATOR_LAST_YEAR	date, datetime, and datetimecombo	Matches dates in the last year relative to the current year.
this_year	this year LBL_OPERATOR_THIS_YEAR	date, datetime, and datetimecombo	Matches dates in the current year.
next_year	next year LBL_OPERATOR_NEXT_YEAR	date, datetime, and datetimecombo	Matches dates in the next year relative to the current year.
$dateBetween	is between LBL_OPERATOR_BETWEEN	date, datetime, and datetimecombo	Matches dates between two given dates.

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 being 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.

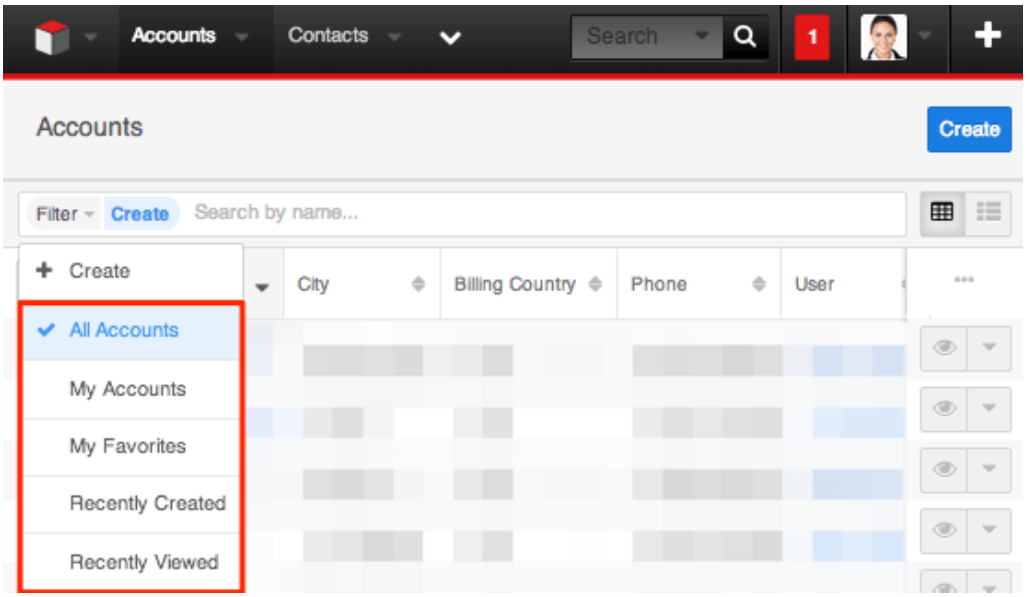
The example below defines a filter where records must be favorited items.

```
$filters = array(
    array(
        '$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/Other 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/filterAccountByTypeAndIndustry.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, we will 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 editable and is_template 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
definition
```

```
            'filter_populate' => array(
                'account_type' => array('Customer')
            ),

            //the dynamic filters to pass to the templates filter
definition
            //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 have a 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
    }
});
```

Last Modified: 07/22/2016 01:34pm

Examples

View Examples.

Last Modified: 09/26/2015 04:14pm

Adding Buttons to the Record View

Overview

How to create additional buttons on the record view and add events.

Adding Buttons to a View

For this example, 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.

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 change $module with the text string of your modules name.

For standard button types, the button definitions will contain the following properties:

Property	Description
Type	The widget type. Button types can be

	'button', 'rowaction', 'shareaction', or 'actiondropdown'.
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	The ACL action of the button. Values can be 'edit' or 'view'.

For this example we will be adding our custom button to the main dropdown. For actiondropdown types, there will be 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	Description
Type	The widget type. Button types can be 'button', 'rowaction', 'shareaction', or 'actiondropdown'. Most custom buttons are typed as 'rowaction'.
event	The event name of the button. The name format is 'button:button_name:click'.
name	The name of the button.
label	The label string key for the display text of the button.
acl_action	The ACL action of the button. Values can be 'edit' or 'view'.

Adding Custom Buttons

For our example, we will modify the accounts metadata to add our 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_street',
                    'billing_address_city' =>
'shipping_address_city',
                    'billing_address_state' =>
'shipping_address_state',
                    'billing_address_postalcode' =>
'shipping_address_postalcode',
                    'billing_address_country' =>
'shipping_address_country',
                  ),
                ),
              ),
            ),
            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 we will need to define the label for our 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, we will need to extend and override the record view controller.

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

```
({

    extendsFrom: 'RecordView',

    zipJSON: {},

    initialize: function (options) {
        app.view.invokeParent(this, {type: 'view', name: 'record',
method: 'initialize', args:[options]});

        //add listener for custom button
        this.context.on('button:validate_postal_code:click',
this.validate_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, you will need to navigate to Admin > Repair > Quick Repair and Rebuild.

Last Modified: 09/26/2015 04:14pm

Adding Field Validation to the Record View

Overview

How to add additional field validation to the record view.

Adding Validation to a 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 customer type is set to 'Customer' and also require the user to enter at least one email address.

Method 1: Extending a Modules Record and Create-Actions Controllers

You may choose to use is to override a specific modules record and create-actions controllers. This method requires a duplication of code due to the hierarchy design, however, it does allow for you to organize your code by module and to extend the core functionality. To accomplish this, we will override and extend the create-action view controller. This will handle the validation when a user is creating a new record. Once the controller has been properly extended, you can define your validation check and use the model.addValidationTask method to append your function to the save validation.

./custom/modules/Accounts/clients/base/views/create-actions/create-actions.js

```
({
    extendsFrom: 'CreateActionsView',

    initialize: function (options) {

        /*

         If you are on 7.2.0, there is a dependency bug that can cause
the screen to render an empty form.

         To correct this, uncomment the code below:

         if (!_.has(options.meta, "template"))
         {
            options.meta.template = 'record';
         }

         */

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

        //add validation tasks
        this.model.addValidationTask('check_account_type',
_.bind(this._doValidateCheckType, this));
        this.model.addValidationTask('check_email',
_.bind(this._doValidateEmail, 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, we will need to duplicate our validation logic for the record view. This will handle 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',

    /* if you are running 7.2.0, you will need to use RecordView */
    //extendsFrom: 'RecordView',

    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._doValidateEmail, 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, you will need to navigate to Admin > Repair > Quick Repair and Rebuild.

Method 2: Overriding a Modules Record and Create-Actions Layouts

Another method of defining your own custom validation is to override a modules record and create-actions layouts to append a new view with your logic. The benefits to this method are that you can use the single view to house the validation logic, however, this will mean that you will have to override the layout. When overriding a layout, you will need to verify that the layout hasn't changed when upgrading your instance. Once the layouts are overridden, you can define your validation check and use the model.addValidationTask method to append your function to the save validation.

First, we will need to create our custom view. For our accounts example, we will 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._doValidateEmail, 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, depending on your selected module, we will need to duplicate its create-actions layout to the modules custom folder to handle record creation. In our accounts example, we have an existing ./modules/Accounts/clients/base/layouts/create-actions/create-actions.php file so we will need to duplicate this file to ./custom/modules/Accounts/clients/base/layouts/create-actions/create-actions.php. After this has been completed, we will need to append our new custom view to the components. We will need to append:

```
array(
    'view' => 'validate-account',
),
```

As shown below:

./custom/modules/Accounts/clients/base/layouts/create-actions/create-actions.php

```
<?php
```

```
$viewdefs['Accounts']['base']['layout']['create-actions'] = array(
    'components' =>
        array(
            array(
                'layout' =>
                    array(
                        'components' =>
                            array(
                                array(
                                    'layout' =>
                                        array(
                                            'components' =>
                                                array(
                                                    array(
                                                        'view' =>
'create-actions',
                                                    ),
                                                    array(
                                                        'view' =>
'validate-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' =>
'preview',
                                                                ),
                                                            ),
                                                        'type' => 'simple',
                                                        'name' => 'preview-pane',
                                                        'span' => 8,
                                                    ),
                                            ),
                                        ),
                                    'type' => 'default',
                                    'name' => 'sidebar',
                                    'span' => 12,
                                ),
                        ),
                    ),
                'type' => 'simple',
                'name' => 'base',
                'span' => 12,
);
```

Lastly, depending on your selected module, we will need to duplicate its record layout to the modules custom folder to handle editing a record. In our accounts example, we do not have an existing ./modules/Accounts/clients/base/layouts/record/record.php file so we will need to

duplicate 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, we will need to modify:

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

To:

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

After this has been completed, we will need to append our new custom view to the components. We will need to append:

```
array(
    'view' => 'validate-account',
),
```

As 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_VIEW',
                                        ),
                                        array(
                                            'name' => 'list',
                                            'icon' =>
'icon-table',
                                            'label' =>
'LBL_LISTVIEW',
                                        ),
                                        array(
                                            'name' =>
'activitystream',
                                            'icon' =>
'icon-th-list',
                                            'label' =>
'LBL_ACTIVITY_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' =>
'activitystream',
                                            'context' =>
                                            array(
                                                'module' =>
'Activities',
                                            ),
                                        ),
                                        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, you will need to navigate to Admin > Repair > Quick Repair and Rebuild.

Error Messages

When throwing a validation error, Sugar has several stock messages you may choose to use. They are listed 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 your messages language key to app.error.errorName2Keys when initializing your extended view. The example below demonstrates this process expanding on the example above:

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

```
<?php

$app_strings['ERR_CUSTOM_MESSAGE'] = 'My custom error message.';
```

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

```
({

    extendsFrom: 'RecordView',

    initialize: function (options) {

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

        //add custom message key
        app.error.errorName2Keys['custom_message'] =
'ERR_CUSTOM_MESSAGE';

        //add validation
        this.model.addValidationTask('check_account_type',
_.bind(this._doValidateCheckType, 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'].custom_message = true;
        }

        callback(null, fields, errors);
    }

})
```

Last Modified: 09/22/2016 11:23am

Passing Data to Templates

Overview

How to create a custom view component that passes data to the handlebars template.

Creating a View and Passing Data

The view component will render the actual content we want to see 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();
    }

})
```

Next, you will add your handlebars template to render the data:

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

```
{{#with myData}}{{myProperty}}{{/with}}
```

Once the files are in place, you will need to add your view to a layout and then navigate to Admin > Repair > Quick Repair and Rebuild.

Last Modified: 09/26/2015 04:14pm

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
core buttons
if (!isset($viewdefs[$module]['base']['view']['record']['buttons']))
{
    require('clients/base/views/record/record.php');
    $viewdefs[$module]['base']['view']['record']['buttons'] =
$viewdefs['base']['view']['record']['buttons'];
    unset($viewdefs['base']);
}

foreach($viewdefs[$module]['base']['view']['record']['buttons'] as
$outerKey => $outerButton)
{
    if (
        isset($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'][$outer
Key]['buttons'] as $innerKey => $innerButton)
        {
            if (
                isset($innerButton['name'])
                && $innerButton['name'] == 'button_name'
            )
            {

unset($viewdefs[$module]['base']['view']['record']['buttons'][$outerKe
y]['buttons'][$innerKey]);
            }
        }
        */
```

```
        //appending buttons
        foreach ($addButtons as $addButton)
        {

$viewdefs[$module]['base']['view']['record']['buttons'][$outerKey]['bu
ttons'][]=$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
Subpanel';
$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.refresh_all_subpanels, this);
    },

    /**
     * Refreshes a specific subpanel given a link
     */
    refresh_specific_subpanel: function() {
        var linkName = 'contacts';
        var subpanelCollection =
this.model.getRelatedCollection(linkName);
        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.

Last Modified: 09/26/2015 04:14pm

Removing the Account Requirement on Opportunities

Overview

This will 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
- Contacts
- 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 View Metadata

Once you have updated your configuration, you may find the field is still required. This may be due to the views metadata having a the field's required attribute set to true. This will also 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.
  Note: If the path does not exist, you edit the Opportunities record view in Studio or you can duplicate ./modules/Opportunities/clients/base/views/record/record.php to ./custom/modules/Opportunities/clients/base/views/record/record.php.
- 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',
        ...
```

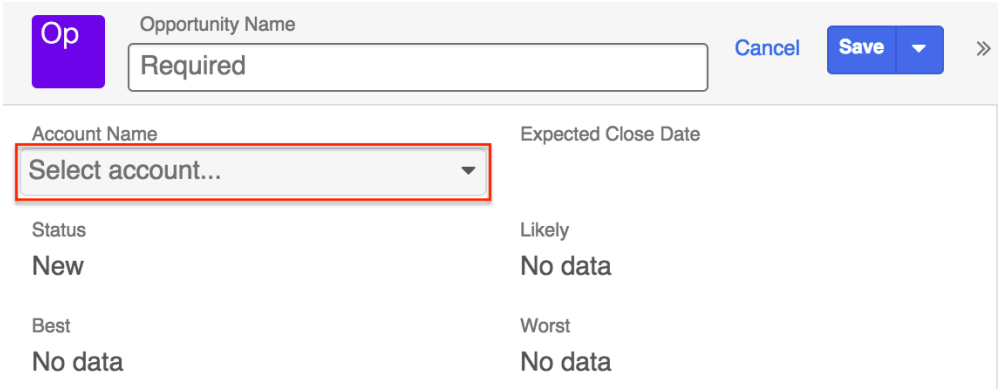
- These changes will only affect the record view for Opportunities. You may need to modify additional module layouts based on your use case. A list of other views you many need to create and/or modify for your module are shown below.

    - ./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.



Last Modified: 08/11/2016 10:48pm

Fields

Fields Overview.

Last Modified: 11/18/2015 01:08am

Introduction

Overview

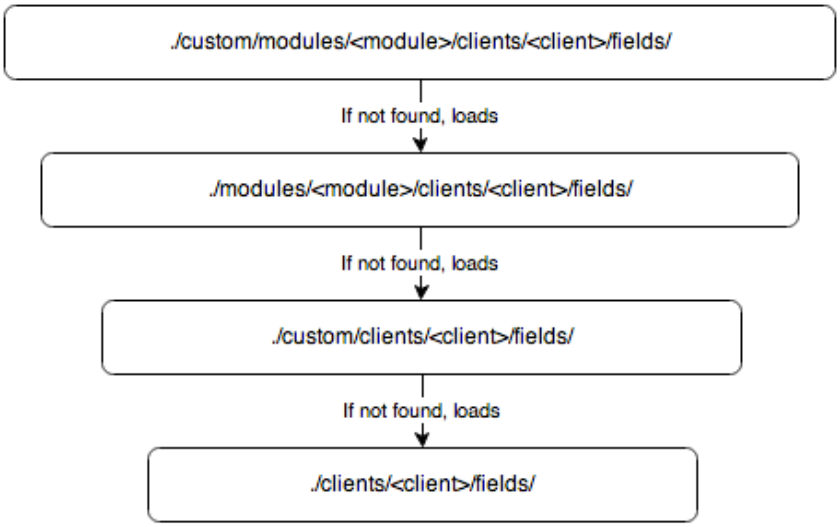
An overview of how field components work.

Fields

Fields are component plugins that render widgets for individual values pulled from models. Field components handle the rendering and formatting of field values and are made up of a controller JavaScript file (.js) and at least one Handlebars template (.hbt).

Hierarchy Diagram

The field components are loaded in the following manner:



Note: The Sugar application client type is "base". More information on the various client types can be found in the Clients section.

Field Example

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

Controller

The bool.js, shown below, overrides the base _render to disable the field. The format and unformat functions handle the manipulation of the fields 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>
```

Last Modified: 01/15/2016 09:05pm

Examples

Field Examples.

Last Modified: 11/18/2015 01:08am

Creating Custom Fields

Overview

How to create a custom field component.

Creating a Custom Field Type

This example will create a custom field type named "Highlightfield". This field will mimic the base field type that is used to render text fields. This custom field type will differ in that the displayed text for the field will be highlighted in a color chosen when the field is created in Studio. Custom field type names in Sugar must be capitalized with all other letters being lower case. In our example, HighLightField or highLightField would not work as they are camel case.

JavaScript Controller

The first thing we will need to do is create our controller file. Since we are starting from the ground up, we will want to extend the base field template. To accomplish this, we will 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. Our 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 contains methods for initialize, _render, format, and unformat. These are present for your ease of use and are not necessarily needed for this customization. The example being that you could choose to create an empty controller consisting of nothing other than ({}) and the field would render as expected given our specific example.

Handlebar Templates

We will now need to define our handlebar templates. The templates will nearly match the base template found in ./clients/base/fields/base/ with the minor difference being the additional attribute of style="background:{{def.backcolor}}; color:{{def.textcolor}}" to the detail and list templates. The first template we will

define is the Sidecar detail view. This template handles the display our our data for viewing on the RecordView.

./custom/clients/base/fields/Highlightfield/detail.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
indexes are defined in:

./custom/modules/DynamicFields/templates/Fields/TemplateHighlightfield
.php
}}

{{#if value}}
    <div class="ellipsis_inline" data-placement="bottom"
style="background:{{def.backcolor}}; color:{{def.textcolor}}">
        {{value}}
    </div>
{{/if}}
```

Next, we will defined our Sidecar edit view. This template handles the display of our field for editing.

./custom/clients/base/fields/Highlightfield/edit.hbs.

```
{{!
    We have not made any edits to this file that differ from stock,
however,
    we could add styling here just as we did for the detail and list
templates.
}}

<input type="text"
    name="{{name}}"
    value="{{value}}"
    {{#if def.len}}maxlength="{{def.len}}"{{/if}}
    {{#if def.placeholder}}placeholder="{{str def.placeholder
this.model.module}}"{{/if}}
    class="inherit-width">

<p class="help-block">
```

Finally, we will need to define our list view. This template handles the display of our field for viewing in lists.

./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
indexes are defined in:

./custom/modules/DynamicFields/templates/Fields/TemplateHighlightfield
.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>
```

Studio

To enable our field type for use in Studio, we will need to define the Studio field template. This will also allow us to map any additional properties we need for our templates. For our purposes, we will map the ext1 and ext2 fields from the fields_meta_data table to the backcolor and textcolor definitions. We will also specify the dbfield definition to be varchar so that the correct database field type is created.

./custom/modules/DynamicFields/templates/Fields/TemplateHighlightfield.php

```
<?php

if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

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
are 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
are required for the Highlightfield Field
     *                      the primary reason this function is here
is to set the dbType to 'varchar',
     *                      otherwise 'Highlightfield' would be used
by 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';

        //map our extension fields for colorizing the field
        $def['backcolor'] = !empty($this->backcolor) ?
$this->backcolor : $this->ext1;
        $def['textcolor'] = !empty($this->textcolor) ?
$this->textcolor : $this->ext2;

        return $def;
    }
}
```

Note: For your custom field type, you have access to the ext1, ext2, ext3, and ext4 database fields in the fields_meta_data table for additional property storage.

Next, we will need to set up the fields form controller. This controller will handle the fields form template in Admin > Studio > Fields and allow us to assign our color values to the Smarty template.

./custom/modules/DynamicFields/templates/Fields/Forms/Highlightfield.php

```
<?php

if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

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/Forms/Highli
ghtfield.tpl');
}

?>
```

Once our form controller is in place, we will create the Smarty template. The .tpl we create below will define the center content of the fields Studio edit view. As you may notice, 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="COLUMN_TITLE_DEFAULT_VALUE"}:</td>
    <td>
        {if $hideLevel < 5}
            <input type='text' name='default' id='default'
value='{$vardef.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="COLUMN_TITLE_MAX_SIZE"}:</td>
    <td>
        {if $hideLevel < 5}
            <input type='text' name='len' id='field_len'
value='{$vardef.len|default:25}'

onchange="forceRange(this,1,255);changeMaxLength(document.getElementBy
Id('default'),this.value);">
            <input type='hidden' id="orig_len" name='orig_len'
value='{$vardef.len}'>
            {if $action=="saveSugarField"}
                <input type='hidden' name='customTypeValidate'
id='customTypeValidate' value='{$vardef.len|default:25}'

                        onchange="if
(parseInt(document.getElementById('field_len').value) <
parseInt(document.getElementById('orig_len').value)) return
confirm(SUGAR.language.get('ModuleBuilder',
'LBL_CONFIRM_LOWER_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.maxLength);
                }
            </script>
        {/literal}
        {else}
            <input type='hidden' name='len'
value='{$vardef.len}'>{$vardef.len}
        {/if}
    </td>
</tr>
```

```
<tr>
    <td class='mbLBL'>{sugar_translate module="DynamicFields"
label="LBL_HIGHLIGHTFIELD_BACKCOLOR"}:</td>
    <td>
        {if $hideLevel < 5}
            {html_options name="ext1" id="ext1" selected=$BACKCOLOR
options=$highlightColors}
        {else}
            <input type='hidden' id='ext1' name='ext1'
value='{$BACKCOLOR}'>
            {$BACKCOLORNAME}
        {/if}
    </td>
</tr>
<tr>
    <td class='mbLBL'>{sugar_translate module="DynamicFields"
label="LBL_HIGHLIGHTFIELD_TEXTCOLOR"}:</td>
    <td>
        {if $hideLevel < 5}
            {html_options name="ext2" id="ext2" selected=$TEXTCOLOR
options=$highlightColors}
        {else}
            <input type='hidden' id='ext2' name='ext2'
value='{$TEXTCOLOR}'>
            {$TEXTCOLORNAME}
        {/if}
    </td>
</tr>

{include
file="modules/DynamicFields/templates/Fields/Forms/coreBottom.tpl"}
```

Once we have defined our Studio template, we will need to register our field as a valid field type. For our purposes, our field will inherit the base field type as it is a text field. In doing this, we are able 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
= '')
    {
        $GLOBALS['log']->debug("SugarFieldHighlightfield::save()
function called.");
        parent::save($bean, $params, $field, $properties, $prefix);
    }
}

?>
```

Language Definitions

For our field, we will need to define several language extensions to ensure everything is displayed correctly. We will first need to define our new field type in our field types list. This will allow an Administrator to select our 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';
```

Once we have our field type defined, we will need to add in the labels for our 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';
```

Next we will need to define our fields drop down lists. This list 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
contain';
$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'
);
```

Searching and Filtering

To enable your field for searching and filtering, we will need to define the filter operators and the Sugar widget. Our filter operators, defined in ./custom/clients/base/filters/operators/operators.php, will allow for our custom field be used for searching in Sidecar listviews.

./custom/clients/base/filters/operators/operators.php

```
<?php

require_once('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 our filters 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.

Lastly, we will need to define a 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
Point');

require_once('include/generic/SugarWidgets/SugarWidgetFieldvarchar.php
');

class SugarWidgetFieldHighlightfield extends SugarWidgetFieldVarchar
{
    function SugarWidgetFieldHighlightfield(&$layout_manager)
    {
        parent::SugarWidgetFieldVarchar($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 " .
$this->reporter->db->convert($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, you will need to navigate to Admin > Repair > Quick Repair and Rebuild. You may also need to clear your browser cache. Once completed, we will now be able to use our field in Studio.

Last Modified: 11/04/2015 08:30pm

Dashlets

Dashlets Overview.

Last Modified: 11/18/2015 01:08am

Introduction

Overview

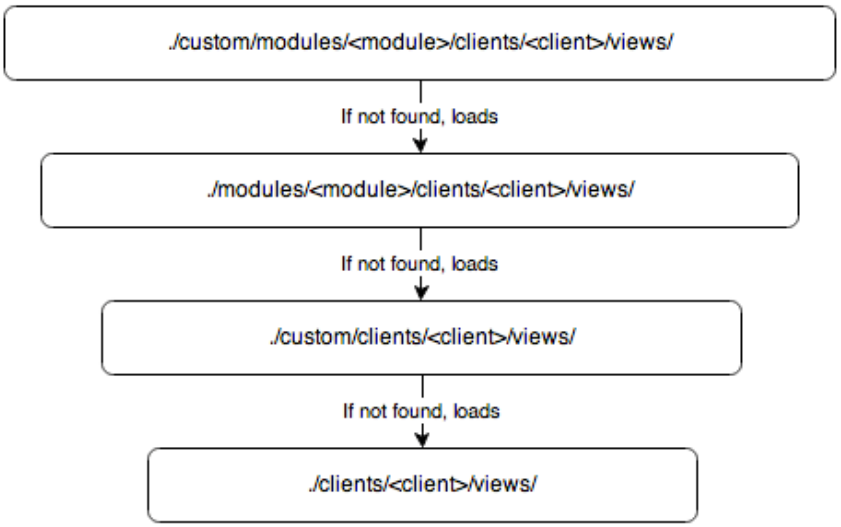
An overview of how dashlet view components work.

Dashlets

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 (.hbt).

Hierarchy Diagram

The dashlet view components are loaded in the following manner:



Note: The Sugar application client type is "base". More information on the various client types can be found in the Clients section.

Dashlet Views

The are 3 views when working with dashlets: Preview, Dashlet View, and Configuration View. More information on these views is outlined in the following sections.

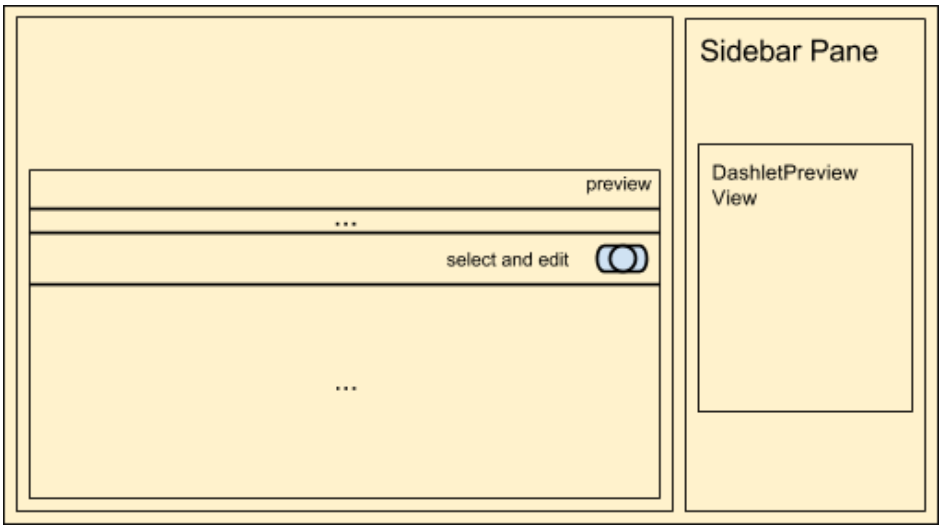
Preview

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',
),
```

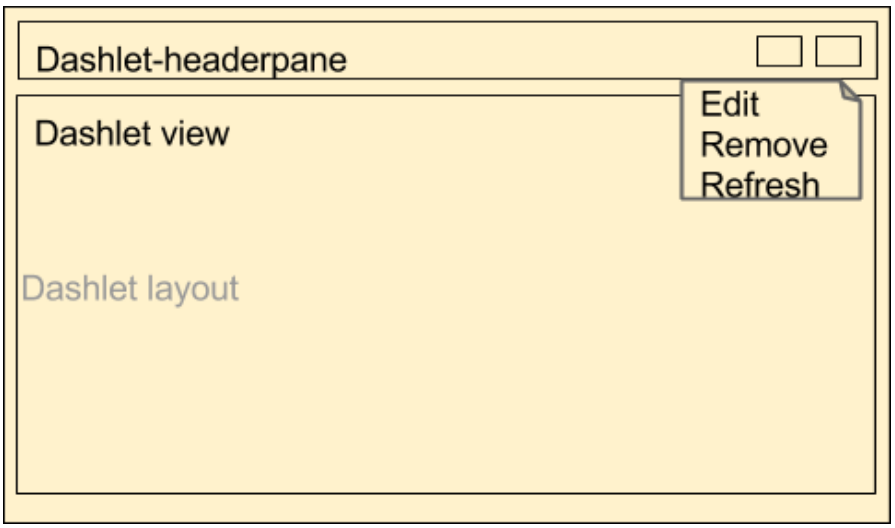
The values in the preview metadata can be retrieved using:

```
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.



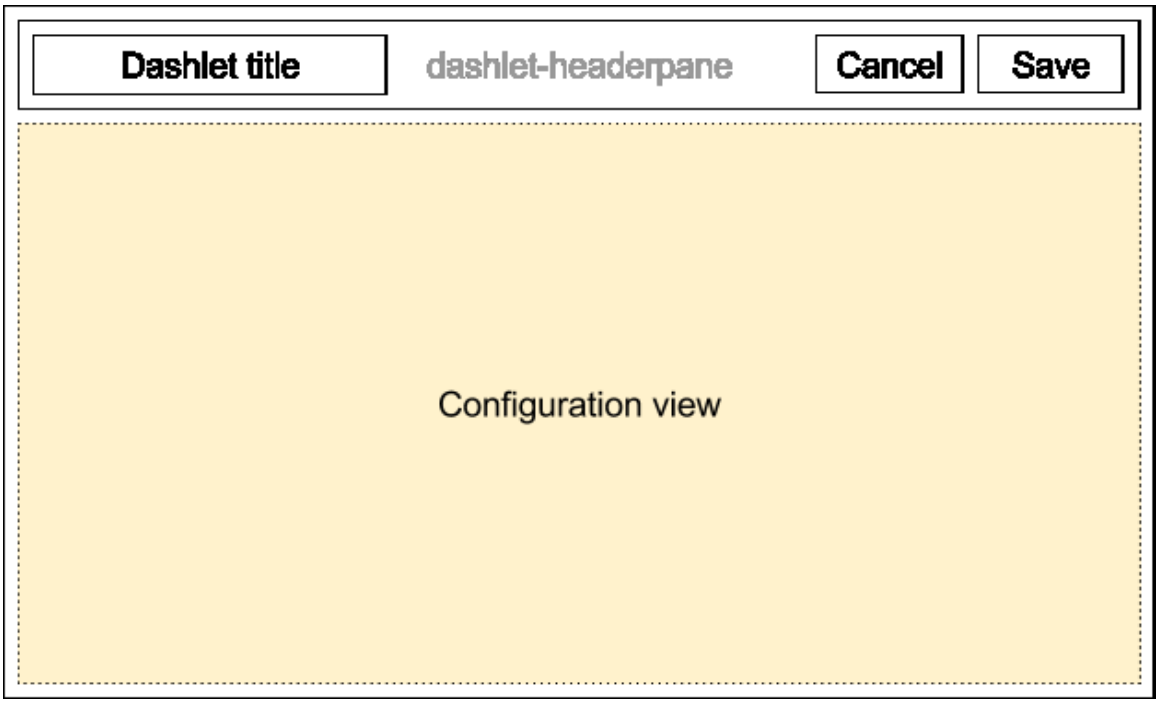
Configuration View

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

```
'config' => array(
    //label assigned for the default dashlet title
    'name' => 'LBL_DASHLET_NAME',
    'key1' => 'value1',
),
```

The values in the config metadata can be retrieved using:

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



Dashlet Example

The news view, located in ./clients/base/views/news/, is what handles the display for the news dashlet. The sections below will outline the various files that render this dashlet.

Metadata

The Dashlet view will contain the 'dashlets' metadata:

Parameters	Type	Description
name	String	Required. The name of the dashlet.

description	String	Optional. A description of the dashlet.
config	Object	Required. Pre-populated variables in the configuration view. Config variables in the metadata will be assigned to the custom model variables.
preview	Object	Required. Pre-populated variables in the preview.
filter	Object	Optional. Filter for display.

The news dashlets metadata is located in:

./clients/base/views/news/news.php

```
<?php

$viewdefs['base']['view']['news'] = array(
    'dashlets' => array(
        array(
            'name' => 'LBL_DASHLET_NEWS_NAME',
            'description' => 'LBL_DASHLET_NEWS_DESCRIPTION',
            'config' => array(
                'limit' => '3',
            ),
            'preview' => array(
                'limit' => '3',
            ),
            'filter' => array(
                'module' => array(
                    'Accounts',
                ),
                'view' => 'record'
            ),
        ),
    ),
    'config' => array(
        'fields' => array(
            array(
                'name' => 'limit',
                'label' => 'LBL_DASHLET_CONFIGURE_DISPLAY_ROWS',
                'type' => 'enum',
                'searchBarThreshold' => -1,
                'options' => array(
```

```
                    1 => 1,
                    2 => 2,
                    3 => 3,
                    4 => 4,
                    5 => 5,
                    6 => 6,
                    7 => 7,
                    8 => 8,
                ),
            ),
        ),
    ),
);
```

Controller

The news.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.

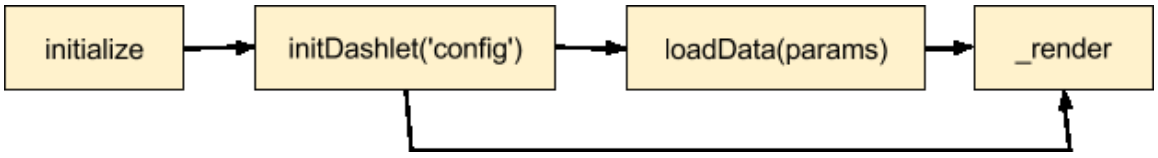
./clients/base/views/news/news.js

```
({

    plugins: ['Dashlet'],

    initDashlet: function() {
        if(this.meta.config) {
            var limit = this.settings.get("limit") || "5";
            this.settings.set("limit", limit);
        }
    },

    loadData: function (options) {
        var name, limit;

        if(_.isUndefined(this.model)){
            return;
        }

        var name = this.model.get("account_name") ||
this.model.get('name') || this.model.get('full_name'),
        limit = parseInt(this.settings.get('limit') || 5, 10);

        if (_.isEmpty(name)) {
```

```
            return;
        }

        $.ajax({
            url:
'https://ajax.googleapis.com/ajax/services/search/news?v=1.0&q=' +
name.toLowerCase() + '&rsz=' + limit,
            dataType: 'jsonp',
            success: function (data) {
                if (this.disposed) {
                    return;
                }

                _.extend(this, data);
                this.render();
            },
            context: this,
            complete: options ? options.complete : null
        });
    }

})
```

Workflow

- The following procedure will trigger to render the view area.



Retrieving Data

- Gather the data from the mainpane
    - this.model - RecordView
    - this.context.parent.get("model") - RecordView

    - this.context.parent.get("collection") - ListView
    - Gather the data from the user inputs

    - this.settings.get("custom_key")
    - Gather the data from the metadata

    - this.dashletConfig['metadata_key']
    - Gather the data from the module vardefs

    - app.metadata.getModule("ModuleName")

      Retrieve the 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()

Handlebar Template

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

./clients/base/views/news/news.hbs

```
{{#if responseData.results}}

{{#each responseData.results}}

    <div class="news-article">

        <h5><a
href="{{this.unescapedUrl}}">{{{this.titleNoFormatting}}}</a>

        </h5> ({{{this.publisher}}})

        <p>{{{this.content}}}</p>

    </div>

{{/each}}

{{else}}

    <div class="block-footer">{{str "LBL_NO_DATA_AVAILABLE"}}</div>

{{/if}}
```

Last Modified: 01/15/2016 09:05pm

Handlebars

Handlebars Overview.

Last Modified: 09/26/2015 04:14pm

Introduction

Overview

An overview of how Handlebars is used within Sugar.

Handlebars

The Handlebars library, located in ./sidecar/lib/handlebars/, is a JavaScript library that allows for Sugar to create semantic templates. It is used to help render content for layouts, views, and fields for Sidecar. Using Handlebars, you can make modifications to display of content such as adding HTML or CSS to your content. More information on the Handlebars library can be found 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 up your browsers console window and inspect the Handlebars.templates namespace.

Debugging Templates

When working with Handlebar templates, its sometimes 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 browsers console.

An example of using log in an hbs template is shown below:

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

Helpers

Handlebar helpers are a way of adding custom functionality to the templates. By default, Sidecar uses the helpers found in ./sidecar/src/view/hbs-helpers.js. Additional helpers used by the base client are found in ./include/javascript/sugar7/hbs-helpers.js. The unminified version of the base client helpers can be found in ./jssource/src_files/include/javascript/sugar7/hbs-helpers.js.

Creating Helpers

When working with Handlebar templates, you may find a need to create your own helper. To do this, you will need to do the following:

Create your Handlebars helper file. This file should be located within your ./custom/ directory. For our purposes, we will create two functions to convert a string to upper case or lower case:

./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);
```

Next, we will need to create a JSGrouping extension in ./custom/Extension/application/Ext/JSGroupings/. This file should be uniquely named for your customization. For our purposes, 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;
    }
}
```

Once the files are in place, you will need to navigate to Admin > Repair > Quick Repair and Rebuild and then to Admin > Repair > Rebuild JS Groupings. This will include your changes and you will be able to call your custom helpers in the hbs files by doing:

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

Last Modified: 09/26/2015 04:14pm

Subpanels

Subpanel Overview.

Last Modified: 11/18/2015 01:08am

Introduction

Overview

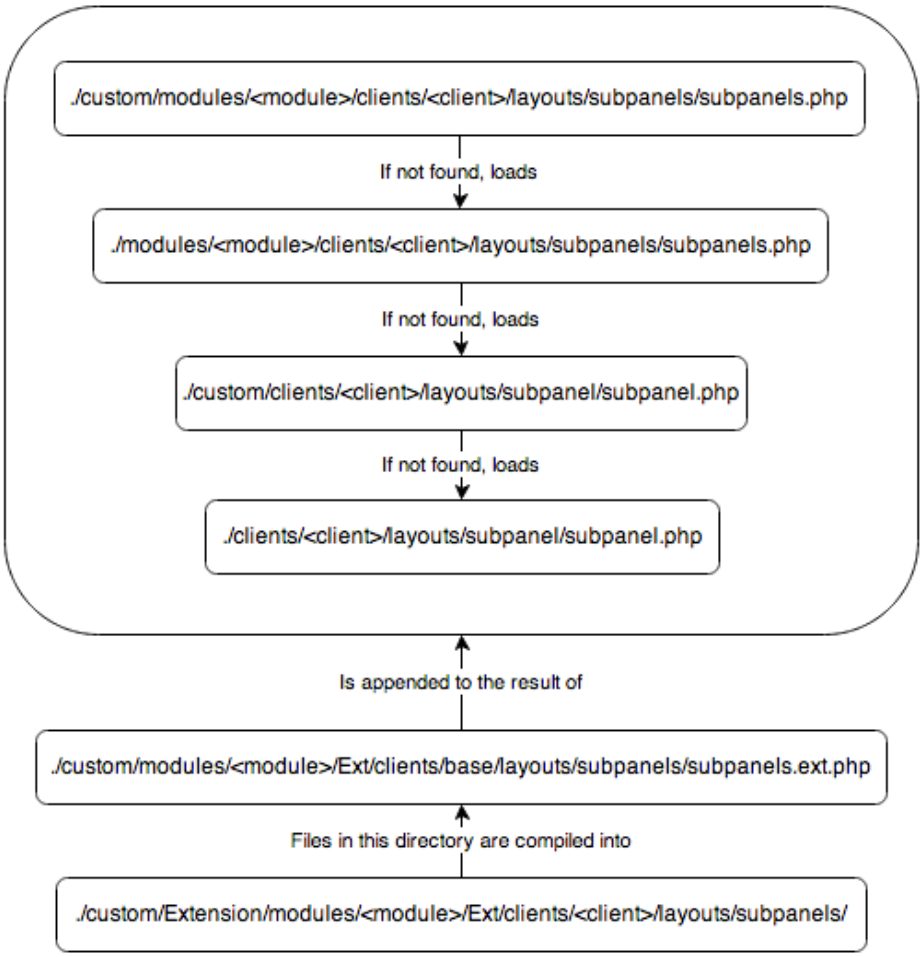
How the new subpanel framework works.

Sidecar Subpanel Layouts

Subpanel layouts have all been modified to work with Sidecar as simplified metadata. 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. You should note that 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 client type is "base". More information on the various client types can be found in the Clients section.

Subpanels and Subpanel Layouts

You will notice that Sugar contains both a subpanels (plural) layout and a subpanel (singular) layout. The subpanels layout contains the collection of subpanels while the subpanel layout renders the actual subpanel widget.

An example of a stock modules 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 3 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.

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

./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'][] =
array (
  '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',
```

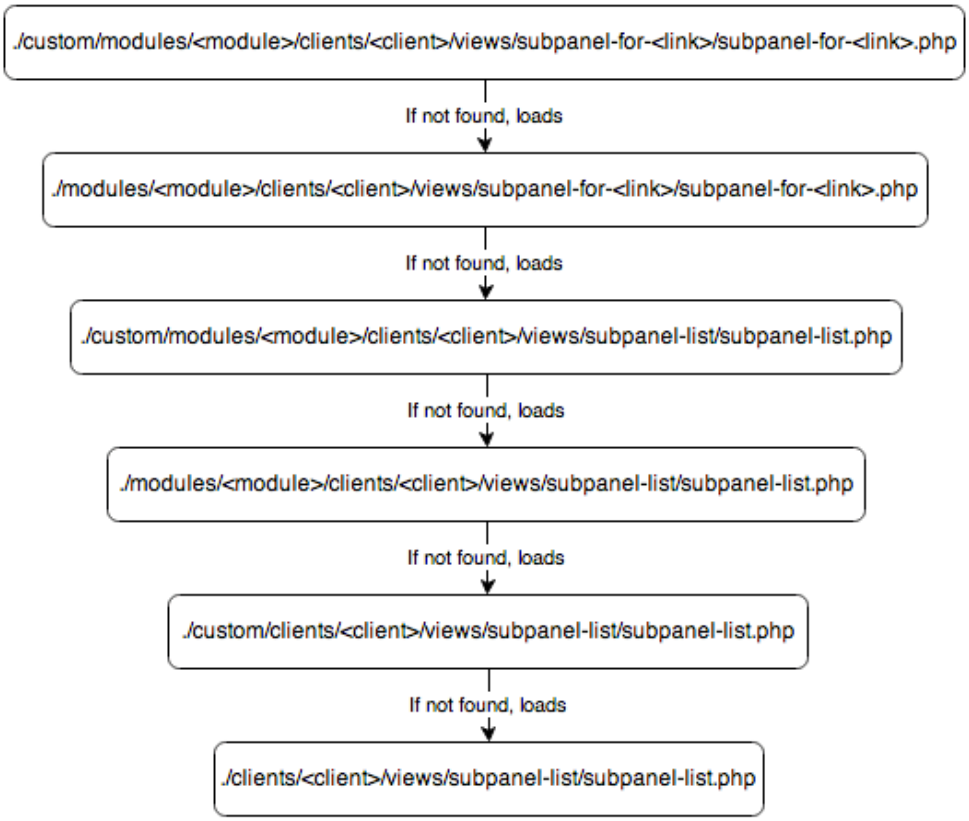
```
  'title_key' => 'LBL_BUGS_LEADS_1_FROM_LEADS_TITLE',
  'get_subpanel_data' => 'bugs_leads_1',
);
```

Fields Metadata

Sidecars 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 client type is "base". More information on the various client types can be found in the Clients section.

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, you can 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.

Last Modified: 09/26/2015 04:14pm

Metadata

Overview of the metadata framework.

Metadata is defined as information about data. In SugarCRM, 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.

Last Modified: 11/18/2015 01:08am

Application Metadata

Overview

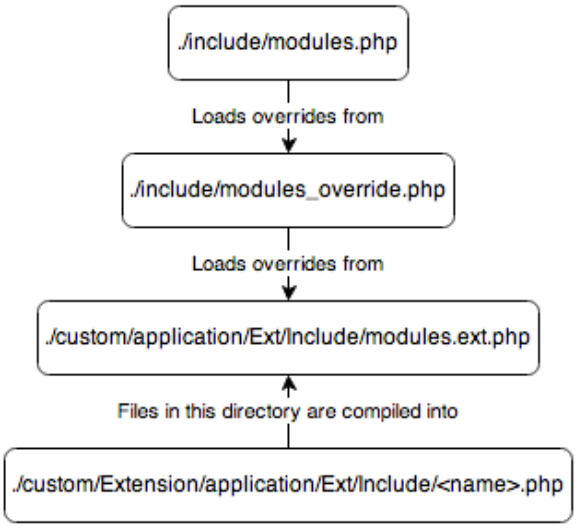
An overview of the application metadata framework.

Application Metadata Framework

The application metadata for stock modules are defined in ./include/modules.php. This file contains the definitions for 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.

Last Modified: 09/26/2015 04:14pm

Module Metadata

Overview

An overview of the Sidecar module metadata framework.

Module Metadata Framework

A modules 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 client type is "base". More information on the various client types can be found in the Clients section.

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

$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_STREET',
                        ),
                        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_STATE',
                        ),
                        array(
                            'name' => 'billing_address_postalcode',
                            'css_class' => 'address_zip',
                            'placeholder' =>
'LBL_BILLING_ADDRESS_POSTALCODE',
                        ),
                        array(
                            'name' => 'billing_address_country',
                            'css_class' => 'address_country',
                            'placeholder' =>
'LBL_BILLING_ADDRESS_COUNTRY',
                        ),
                    ),
                ),
                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_STREET',
                        ),
                        array(
                            'name' => 'shipping_address_city',
                            'css_class' => 'address_city',
                            'placeholder' =>
'LBL_SHIPPING_ADDRESS_CITY',
                        ),
                        array(
                            'name' => 'shipping_address_state',
                            'css_class' => 'address_state',
                            'placeholder' =>
'LBL_SHIPPING_ADDRESS_STATE',
                        ),
                        array(
                            'name' => 'shipping_address_postalcode',
                            'css_class' => 'address_zip',
                            'placeholder' =>
'LBL_SHIPPING_ADDRESS_POSTALCODE',
                        ),
                        array(
                            'name' => 'shipping_address_country',
                            'css_class' => 'address_country',
                            'placeholder' =>
'LBL_SHIPPING_ADDRESS_COUNTRY',
                        ),
                        array(
                            'name' => 'copy',
                            'label' => 'NTC_COPY_BILLING_ADDRESS',
                            'type' => 'copy',
                            'mapping' => array(
                                'billing_address_street' =>
'shipping_address_street',
                                'billing_address_city' =>
'shipping_address_city',
                                'billing_address_state' =>
'shipping_address_state',
                                'billing_address_postalcode' =>
'shipping_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 browsers console window by using the global App Identifier:

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

Last Modified: 09/26/2015 04:14pm

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 Sugar 7.5.x, you can add custom routes by including a JavaScript file in the system and registering a new route after the router has been started. The Sidecar router has a router:start event that can be used to trigger registration of custom routes for the application to utilize. The following example code shows how to use this event to register a new route in Sugar:

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

The above code would register the new route #test/<id> in the SugarCRM system, and if you navigated to that URL fragment, it would log the arguments to the browser console, and then attempt to load a custom view added to the system. To add a custom route to Sugar, you can use the JSGroupings.

Last Modified: 09/22/2016 10:50am

Legacy MVC

Legacy MVC Overview.

Last Modified: 11/18/2015 01:08am

Introduction

Overview

Introduction to 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 legacy MVC framework.

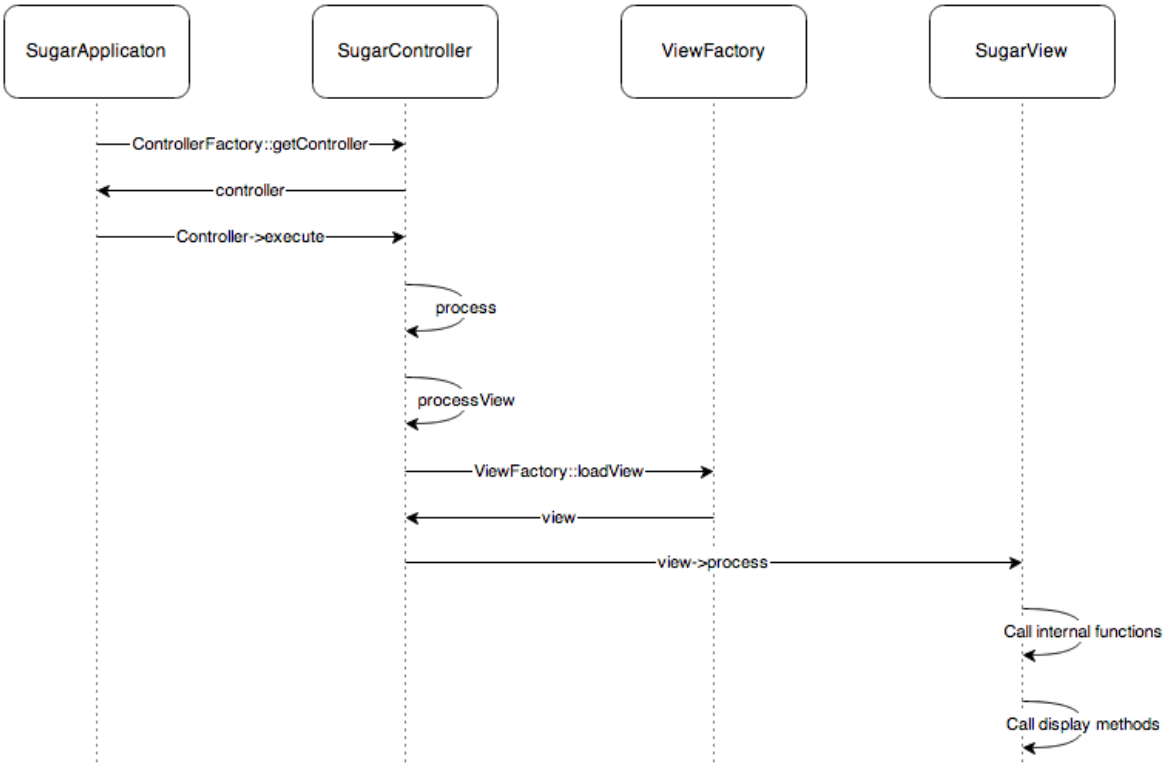
Model-View-Controller (MVC) Overview

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.



Last Modified: 09/26/2015 04:14pm

Model

Overview

The Sugar Model is represented by the SugarBean, and any subclass of the SugarBean. Many of the common Sugar modules also use the SugarObjects class.

Sugar Object Templates

Sugar Objects extend the concept of subclassing a step further and allows 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. If there were a Sugar Object for fields used across every module (e.g. id, deleted, or date_modified) you would be able to have your module inherit from both Basic Sugar Object and the Person Sugar Object.

For example, the Basic type has a field 'name' with length 10 and Company has a field 'name' with length 20. If you inherit from Basic first. then Company, your field

will be of length 20. Assuming you have defined the field 'name' in your module with length 60, then the module will always override any values provided by Sugar Objects.

There are six types of Sugar Object 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 modue
- Sale : Based on the Opportunities module

We can take this a step further and add assignable to the mix. An assignable module would be one that can be assigned to users. Although this is not used by every module, many modules allow assignment of records to users .SugarObject interfaces allow us to add "assignable" to modules in which we want to enable users to assign records.

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 you will notice that templates include many additional files, including a full metadata directory. Currently, 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 SugarObjects:

1) Your class needs to subclass the SugarObject class you wish to extend.

```
class MyClass extends Person
{
    function MyClass()
    {
        parent::Person();
    }
```

```
}
```

2) In your vardefs.php file, add the following to the end:

```
VardefManager::createVardef('Contacts', 'Contact', array('default',
'assignable', 'team_security', 'person'));
```

This tells the VardefManager to create a cache of the Contacts vardefs with the addition of all the default fields, assignable fields, team security fields (Sugar Professional and Enterprise only), and all fields from the person class.

Performance Considerations

VardefManager caches the generated vardefs into a single file that will be loaded at run time. If that file is not found, Sugar will load the vardefs.php file, located in your modules 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

Last Modified: 09/26/2015 04:14pm

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.

Last Modified: 09/26/2015 04:14pm

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 setup. 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 rebulding 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/DeleteRelationship.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".

Last Modified: 01/15/2016 09:05pm

Metadata

Legacy MVC Metadata Overview.

Last Modified: 11/18/2015 01:08am

Introduction

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.

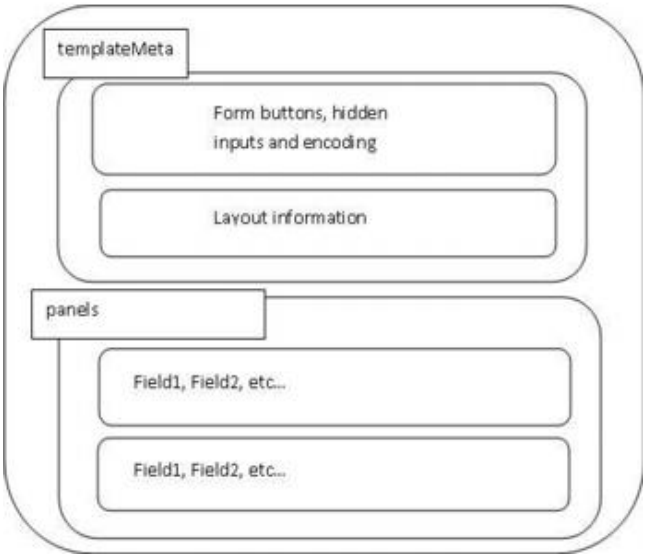
The 'layout' key points to another nested array which defines the fields to display in the basic and advanced search form tabs. Each individual field definition maps to a SugarField widget. See the SugarField widget section for an explanation about SugarField widgets and how they are rendered for the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

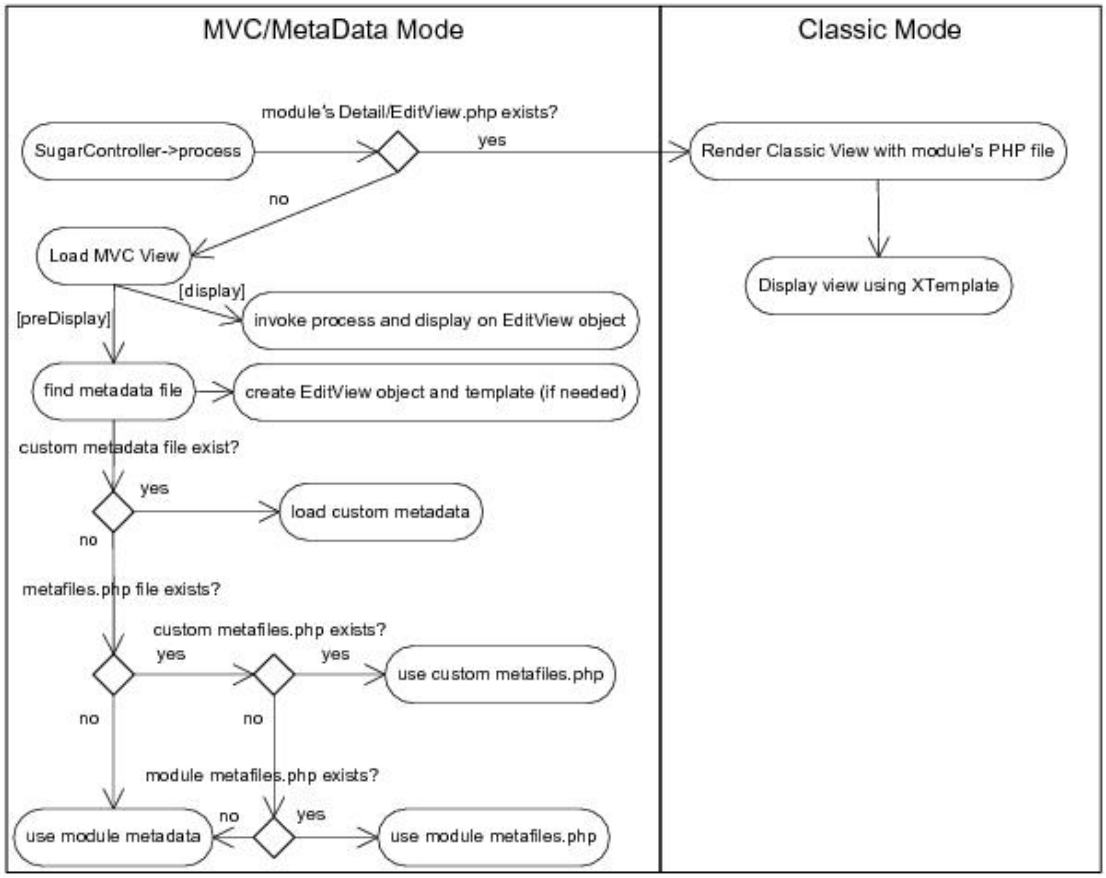
The processing flow for the search form using the metadata subpaneldefs.php file is similar to that of EdiView and DetailView.

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).

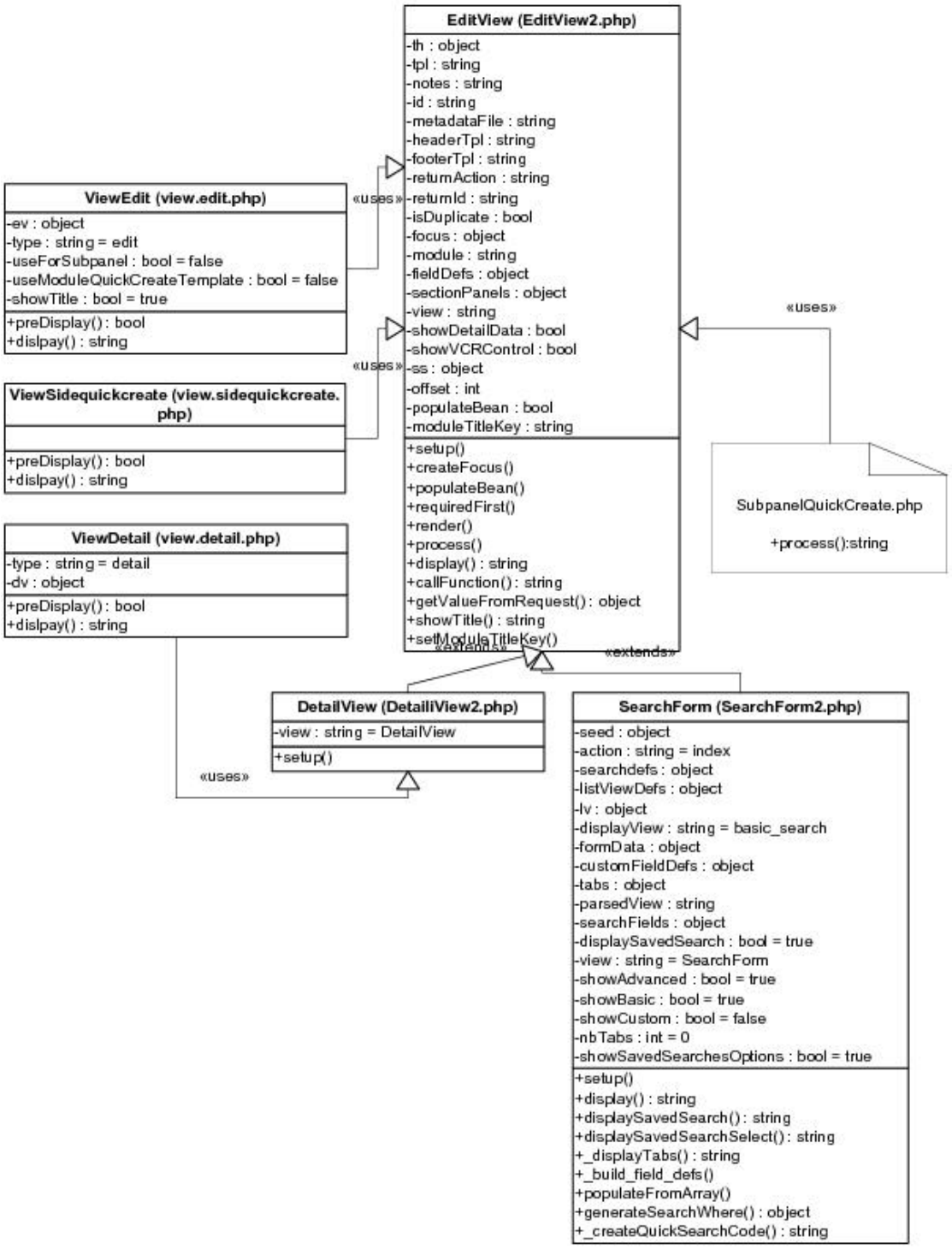
```
$metafiles['Accounts'] = array(
    'detailviewdefs' =>
'modules/Accounts/metadata/detailviewdefs.php',
    'editviewdefs' => 'modules/Accounts/metadata/editviewdefs.php',
    'ListViewdefs' => 'modules/Accounts/metadata/ListViewdefs.php',
    'searchdefs' => 'modules/Accounts/metadata/searchdefs.php',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

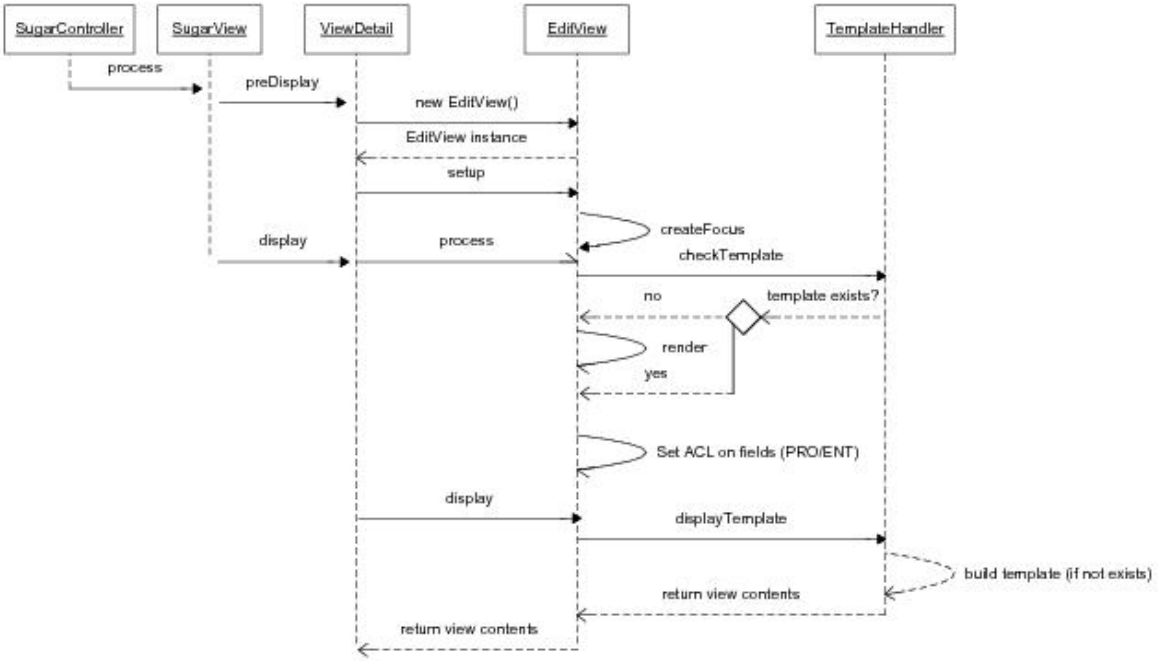
After the preDisplay method is called in the view code, the display method is called, resulting in a call to the EditView object's process method, as well as the EditView object's display method.

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 the Sugar Ultimate, Enterprise, Corporate, and Professional editions.

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.



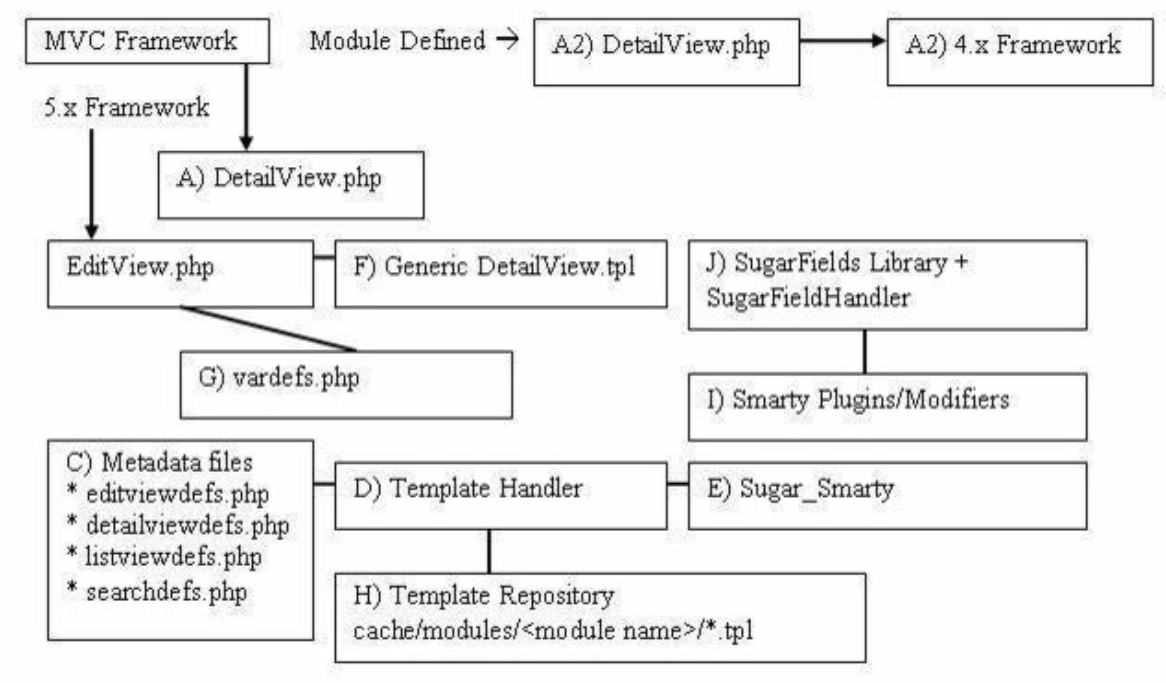
The following diagram highlights the EditView class's main responsibilities and their relationships with other classes in the system. We will use the example of a DetailView request although the sequence will be similar for other views that use the EditView class.



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/DetailView/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:

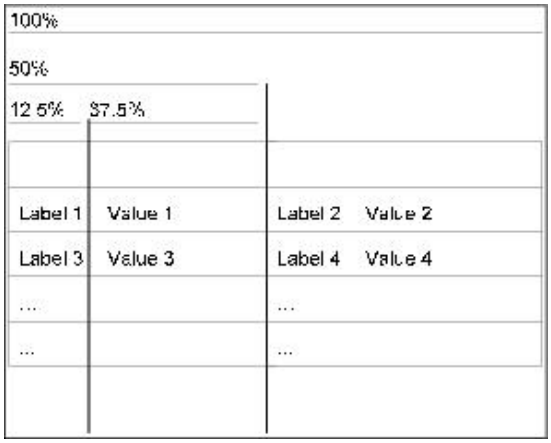
```
function process()
{
    //Format fields first
    if($this->formatFields)
    {
        $this->focus->format_all_fields();
    }
```

```
    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'
        )
    ),
),
```

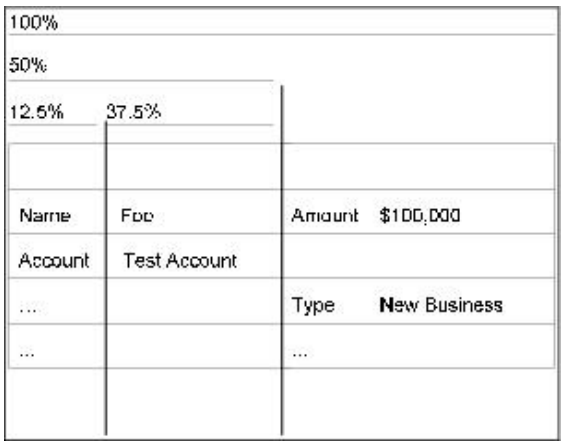
We can see that the labels and fields are mapped as a 1-to-3 ratio. The sum of the widths only equals a total of 80 (10 + 30 x 2) so the actual resulting values written to the Smarty template will be at a percentage ratio of 12.5-to-37.5. The resulting fields defined in the metadata file will be rendered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

Last Modified: 09/26/2015 04:14pm

SugarFields

SugarField Overview.

Last Modified: 11/18/2015 01:08am

Introduction

Overview

Overview of the MVC SugarField widgets.

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.

SugarField Widgets

SugarFields are the Objects that render the fields specified in the metadata (for example, your *viewdefs.php files). They can be found in include/SugarFields/Fields. In the directory, include/SugarFields/Fields/Base, you will see the files for the base templates for rendering a field for DetailView, EditView, ListView, and Search Forms, as well as the base class called SugarFieldBase.

File Structure

- ./include/SugarFields/Fields/
- ./include/SugarFields/Fields//DetailView.tpl
- ./modules//vardefs.php
- ./modules//metadata/defs.php

Implementation

This section describes the SugarFields widgets that are found in the ./include/SugarFields/Fields directory.

Inside this folder you will find a set of directories that encapsulate the rendering of a field type (for example, Boolean, Text, Enum, etc.). That is, there are user interface paradigms associated with a particular field type. For example, a Boolean field type as defined in a module's vardef.php file can be displayed with a checkbox indicating the boolean nature of the field value (on/off, yes/no, 0/1, etc.). Naturally there are some displays in which the rendered user interface components are very specific to the module's logic. In this example, it is likely that custom code was used in the metadata file definition. There are also SugarFields directories for grouped display values (e.g. Address, Datetime, Parent, and Relate).

Any custom code called by the metadata will be passed as unformatted data for numeric entries, and that custom code in the metadata will need individual handle formatting.

SugarFields widgets are rendered from the metadata framework whenever the MVC EditView, DetailView, or ListView actions are invoked for a particular module. Each of the SugarFields will be discussed briefly.

Most SugarFields will contain a set of Smarty files for abstract rendering of the field contents and supporting HTML. Some SugarFields will also contain a subclass of SugarFieldBase to override particular methods so as to control additional processing and routing of the corresponding Smarty file to use. The subclass naming convention is defined as SugarField[Sugar Field Type] where the first letter of the Sugar Field Type should be in uppercase. The contents should also be placed in a corresponding .php file. For example, the contents of the enum type SugarField (rendered as in HTML) is defined in ./include/SugarFields/Fields/Enum/SugarFieldEnum.php. In that file, you can see how the enum type will use one of six Smarty template files depending on the view (edit, detail or search) and whether or not the enum vardef definition has a 'function' attribute defined to invoke a PHP function to render the contents of the field.

Last Modified: 11/18/2015 01:38am

Widgets

Widget Overview.

Last Modified: 11/18/2015 12:07am

Address

Overview

Renders various fields that together represent an address value.

Address

By default SugarCRM renders address values in the United States format:

```
Street
City, State Zip
```

The Smarty template layout defined in DetailView.tpl reflects this. Should you wish to customize the layout, depending on the $current_language global variable, you may need to add new files [$current_language].DetailView.tpl or [$current_language].EditView.tpl to the Address directory that reflect the language locale's address formatting.

Within the metadata definition, the Address field can be rendered with the snippet:

```
array (
    'name' => 'billing_address_street',
    'hideLabel' => true,
    'type' => 'address',
    'displayParams' => array(
        'key' => 'billing',
        'rows' => 2,
        'cols' => 30,
        'maxlength' => 150
    )
```

),

name	The vardefs.php entry to key field off of. Though not 100% ideal, we use the street value
hideLabel	Boolean attribute to hide the label that is rendered by metadata framework. We hide the billing_address_street label because the Address field already comes with labels for the other fields (city, state).
type	This is the field type override. billing_address_street is defined as a varchar in the vardefs.php file, but since we are interested in rendering an address field, we are overriding this here
displayParams	key - This is the prefix for the address fields. The address field assumes there are [key]_address_street, [key]_address_state, [key]_address_city, [key]_address_postalcode fields so this helps to distinguish in modules where there are two or more addresses (e.g. 'billing' and 'shipping'). rows, cols, maxlength - This overrides the default HTML <textarea> attributes for the street field component.

Note also the presence of file ./include/SugarFields/Fields/Address/SugarFieldAddress.js. This file is responsible for handling the logic of copying address values (from billing to shipping, primary to alternative, etc.). The JavaScript code makes assumptions using the key value of the grouped fields.
To customize various address formats for different locales, you may provide a locale specific implementation in the folder ./include/SugarFields/Fields/Address. There is a default English implementation provided. Locale implementations are system wide specific (you cannot render an address format for one user with an English locale and another format for another with a Japanese locale). Sugar locale settings are system-wide and the SugarField implementation reflects this. To modify based on a user's locale preferences is possible, but will require some customization.

Last Modified: 03/09/2016 03:39pm

Base

Overview

The default parent field.

Base

The Base field is the default parent field. It simply renders the value as is for detail views, and an HTML text field for edit views. All SugarFields that have a corresponding PHP file extend from SugarFieldBase.php or from one of the other SugarFields.

Last Modified: 11/18/2015 12:07am

Bool

Overview

Renders a checkbox to reflect the state of the value.

Bool

The Bool field is responsible for rendering a checkbox to reflect the state of the value. All boolean fields are stored as integer values. The Bool field will render a disabled checkbox for detail views. If the field value is "1" then the checkbox field will be checked. There is no special parameter to pass into this field from the metadata definition. As with any of the fields you have the option to override the label key string.

For example, in the metadata definition, the Boolean field do_not_call can be specified as:

```
'do_not_call'
```

or

```
array (
    array(
        'name' => 'do_not_call',
        'label' => 'LBL_DO_NOT_CALL' // Overrides label as defined in
vardefs.php
    )
),
```

Last Modified: 09/26/2015 04:14pm

Currency

Overview

Renders a checkbox to reflect the state of the value.

Currency

The Currency field is responsible for rendering a field that is formatted according to the user's preferences for currencies. This field's handles will format the field differently if the field name contains the text '_usd', this is used internally to map to the amount_usdollar fields which will always display the currency in the user's preferred currency (or the system currency if the user does not have one selected). If the field name does not contain '_usd', the formatting code will attempt to find a field in the same module called 'currency_id' and use that to figure out what currency symbol to display next to the formatted number. In order for this to work on list views and sub-panels, you will need to add the 'currency_id' column as a 'query_only' field, for further reference please see the Opportunities list view definitions.

Within the metadata definition, the Currency field can be rendered with the snippet:

```
// Assumes that amount is defined as a currency field in vardefs.php
file
'amount',
```

or

```
array(
    'name' => 'amount',
```

```
    'displayParams' => array(
        'required' => true
    )
),
```

name	Standard name definition when metadata definition is defined as an array
displayParams	required (optional) - Marks the field as required and applies clients side validation to ensure value is present when save operation is invoked from edit view (Overrides the value set in vardefs.php). showFormats (optional) - Displays the user's date display preference as retrieved from the global $timedate variable's get_user_date_format() method.

Last Modified: 09/26/2015 04:14pm

Datetime

Overview

Renders an input text field along with an image to invoke the popup calendar picker.

Datetime

The Datetime field is responsible for rendering an input text field along with an image to invoke the popup calendar picker. This field differs from the Datetimecombo field in that there is no option to select the time values of a datetime database field.

Within the metadata definition, the Datetime field can be rendered with the snippet:

```
// Assumes that date_quote_exected_closed is defined as a datetime
field in vardefs.php file
'date_quote_expected_closed',
```

or

```
array(
    'name' => 'date_quote_expected_closed',
    'displayParams' => array(
        'required' => true,
        'showFormats' => true
    )
),
```

Last Modified: 09/26/2015 04:14pm

Datetimecombo

Overview

Renders a Datetime field with additional support to render dropdown lists for the hours and minutes.

Datetimecombo

The Datetimecombo field is similar to the Datetime field with additional support to render dropdown lists for the hours and minutes values, as well as a checkbox to enable/disable the entire field. The date portion (e.g. 12/25/2007) and time portion (e.g. 23:45) of the database fields are consolidated. Hence, the developer must take care to handle input from three HTML field values within the module class code. For example, in the vardefs.php definition for the date_start field in the Calls module:

```
'date_start' => array(
    'name' =>
    'date_start',
    'vname' => 'LBL_DATE',
    'type' => 'datetime',
    'required' => true,
    'comment' => 'Date in which call is schedule to (or did) start'
),
```

There is one database field, but when the Datetimecombo widget is rendered, it will produce three HTML fields for display- a text box for the date portion, and two dropdown lists for the hours and minutes values. The Datetimecombo widget will render the hours and menu dropdown portion in accordance to the user's $timedate preferences. An optional AM/PM meridiem drop down is also displayed should the user have selected a 12 hour base format (e.g. 11:00).

Within the metadata definition, the Datetimecombo field can be rendered with the snippet:

```
array(
    'name' =>
    'date_start',
    'type' => 'datetimecombo',
    'displayParams' => array(
        'required' => true,
        'updateCallback' => 'SugarWidgetScheduler.update_time();',
        'showFormats' => true,
        'showNoneCheckbox' => true
    ),

    'label'=>'LBL_DATE_TIME'
),
```

name	Standard name definition when metadata definition is defined as an array
type	metadata type override. By default, the field defaults to Datetime so we need to override it here in the definition.
displayParams	required (optional) - Marks the field as required and applies clients side validation to ensure value is present when save operation is invoked from edit view (Overrides the value set in vardefs.php). updateCallback (optional) - Defines custom JavaScript function to invoke when the values in the field are changed (date, hours or minutes). showFormats (optional) - Displays the user's date display preference as

	retrieved from the global $timedate variable's get_user_date_format() method. showNoneCheckbox (optional) - Displays a checkbox that when checked will disable all three field values
label (optional)	Standard metadata label override just to highlight this exhaustive example

Last Modified: 09/26/2015 04:14pm

Download

Overview

Renders the ability for a user to download a file.

Download

The File field renders a link that references the download.php file for the given displayParam['id'] value when in DetailView mode.

Within the metadata definition, the Download field can be rendered with the snippet:

```
array (
    'name' => 'filename',
    'displayParams' => array(
        'link' => 'filename',
        'id' => 'document_revision_id'
    )
),
```

name	Standard name definition when metadata definition is defined as an array
displayParams	required (optional) - Marks the field as required and applies clients side validation to ensure value is present

	when save operation is invoked from edit view (Overrides the value set in vardefs.php). id (required for detail view) - The field for which the id of the file will be opened via the download link. link (required for detail view) - The field for which the hyperlink value is displayed.

Last Modified: 09/26/2015 04:14pm

Enum

Overview

Renders an HTML select form element that allows for a single value to be chosen.

Enum

The size attribute of the <select> element is not defined so the element will render as a dropdown field in the EditView.

This field accepts the optional function override behavior that is defined at the vardefs.php file level. For example, in the Bugs module we have for the found_in_release field:

```
'found_in_release' => array(
    'name' => 'found_in_release',
    'type' => 'enum',
    'function' => 'getReleaseDropDown',
    'vname' => 'LBL_FOUND_IN_RELEASE',
    'reportable' => false,
    'merge_filter' => 'enabled',
    'comment' => 'The software or service release that manifested the
bug',
    'duplicate_merge' => 'disabled',
    'audited' => true,
),
```

The function override is not handled by the SugarFields library, but by the rendering code in ./include/EditView/EditView2.php.

Within the metadata definition, the Download field can be rendered with the snippet:

```
array (
    'name' => 'my_enum_field',
    'type' => 'enum',
    'displayParams' => array(
        'javascript' =>
        'onchange="alert(\'hello world!\')";'
    )
),
```

name	Standard name definition when metadata definition is defined as an array
displayParams	size (optional) – Controls the size of the field (affects SearchForm control on the browser only). Defaults to value of 6 for SearchForm. required (optional) - Marks the field as required and applies clients side validation to ensure value is present when save operation is invoked from edit view (Overrides the value set in vardefs.php). javascript (optional) - Custom JavaScript to embed within the <select> tag element (see above example for onchange event implementation).

Last Modified: 09/26/2015 04:14pm

File

Overview

Renders a file upload field.

File

The File field renders a file upload field in edit view, and a hyperlink to invoke download.php in detail view. Note that you will need to override the HTML form's enctype attribute to be "multipart/form-data" when using this field and handle the upload of the file contents in your code. This form enctype attribute should be set in the editviewdefs.php file. For example, for the Document's module we have the form override:

```
$viewdefs['Documents']['EditView'] = array(
    'templateMeta' => array(
        'form' => array(
            'enctype' => 'multipart/form-data', // <--- override the
enctype
        ),
    )
)
```

Within the metadata file, the File field can be rendered with the snippet:

```
array (
    'name' => 'filename',
    'displayParams' => array(
        'link' => 'filename',
        'id' => 'document_revision_id'
    ),
),
```

name	Standard name definition when metadata definition is defined as an array
displayParams	required (optional) - Marks the field as required and applies clients side validation to ensure value is present when save operation is invoked from edit view (Overrides the value set in vardefs.php). id (required for detailviewdefs.php) - The record id that download.php will use to retrieve file contents

	link (required for detailviewdefs.php) - The text to display between the <a> </> tags. size (optional) - Affects edit view only. Override to set the display length attribute of the input field. maxlength(optional) - Affects edit view only. Override to set the display maxlength attribute of the input field (defaults to 255).

Last Modified: 09/26/2015 04:14pm

Float

Overview

Renders a field that is formatted as a floating point number.

Float

The Float field is responsible for rendering a field that is formatted as a floating point number. The precision specified will be followed, or the user's default precision will take over.

Within the metadata definition, the Float field can be rendered with the snippet:

```
// Assumes that weight is defined as a float field in vardefs.php file
'weight',
```

or

```
array(
    'name' => 'weight',
    'displayParams' => array(
        'precision' => 1
    )
),
```

name	Standard name definition when metadata definition is defined as an array
displayParams	required (optional) - Marks the field as required and applies clients side validation to ensure value is present when save operation is invoked from edit view (Overrides the value set in vardefs.php). precision (optional) – Sets the displayed precision of the field, this specifies the number of digits after the decimal place that the field will attempt to format and display. Some databases may further limit the precision beyond what can be specified here.

Last Modified: 09/26/2015 04:14pm

Html

Overview

Renders read-only content.

Html

The Html field is a simple field that renders read-only content after the content is run through the from_html method of ./include/utils/db_utils.php to encode entity references to their HTML characters (i.e. ">" = ">"). The rendering of the Html field type should be handled by the custom field logic within SugarCRM.

name	Standard name definition when metadata definition is defined as an array.
displayParams	none

Last Modified: 11/18/2015 12:07am

Iframe

Overview

Renders an iFrame field.

iFrame

In the detail view, the iFrame field creates an HTML element where the src attribute points to the given URL value supplied in the edit view. Alternatively, the URL may be generated from the values of other fields, and the base URL in the vardefs. If this is the case, the field is not editable in the edit view.

name	Standard name definition when metadata definition is defined as an array.
displayParams	None

Last Modified: 11/18/2015 12:07am

Image

Overview

Renders an <img> tag where the src attribute points to the value of the field.

Image

Similar to the Html field, the Image field simply renders a <img> tag where the src attribute points to the value of the field.

Within the metadata file, the Image field can be rendered with the snippet:

```
array (
    // The value of this is assumed to be some URL to an image file
    'name' => 'my_image_value',
    'type' => 'image',
```

```
    'displayParams' => array(
        'width' => 100,
        'length' => 100,
        'link' => 'http://www.cnn.com',
        'border' => 0
    )
,),
```

name	Standard name definition when metadata definition is defined as an array
displayParams	link (optional) - Hyperlink for the image when clicked on. width (optional) - Width of image for display. height (optional) - Height of image for display. border (optional) - Border thickness of image for display

Last Modified: 09/26/2015 04:14pm

Int

Overview

Renders a field that is formatted as a whole number.

Int

The Int field is responsible for rendering a field that is formatted as a whole number. This will always be displayed as a whole number, and any digits after the decimal point will be truncated.

Within the metadata definition, the Int field can be rendered with the snippet:

```
// Assumes that quantity is defined as a int field in vardefs.php
```

```
file'quantity',
```

or

```
array(    'name' => 'quantity',    'displayParams' => array(
'required' => 1    )),
```

Last Modified: 09/26/2015 04:14pm

Link

Overview

Renders a hyperlink to a records detail view.

Link

The link field simply generates an <a> HTML tag with a hyperlink to the value of the field for detail views. For edit views, it provides the convenience of pre-filling the "http://" value. Alternatively, the URL may be generated from the values of other fields, and the base URL in the vardefs. If this is the case, the field is not editable in edit view.

Within the metadata file, the Link field can be rendered with the snippet:

```
array (
 // The value of this is assumed to be some URL to an image file
 'name' => 'my_image_value',
 'type' => 'link',
 'displayParams' => array(
 'title' => 'LBL_MY_TITLE'
 ),
),
```

name	Standard name definition when metadata definition is defined as an array
displayParams	required (optional) - Marks the field as required and applies clients side validation to ensure value is present

	when save operation is invoked from edit view (overrides the value set in vardefs.php). title (optional for detailviewdefs.php only) - The <a> tag's title attribute for browsers to display the link in their status area size (optional) - Affects edit view only. Override to set the display length attribute of the input field. maxlength(optional) - Affects edit view only. Override to set the display maxlength attribute of the input field (defaults to 255).

Last Modified: 03/17/2016 02:14pm

Multienum

Overview

Renders a bullet list of values for detail views.

Multienum

The Multienum fields renders a bullet list of values for detail views, and renders a <select> form element for edit views that allows multiple values to be chosen. Typically, the custom field handling in Sugar will map multienum types created through Studio, so you do not need to declare metadata code to specify the type override. Nevertheless, within the metadata file, the Multienum field can be rendered with the snippet:

```
array (
    'name' => 'my_multienum_field',
    'type' => 'multienum',
    'displayParams' => array(
        'javascript' => 'onchange="alert(\'hello world!\')";'
    )
),
```

name	Standard name definition when metadata definition is defined as an array
displayParams	size (optional) – Controls the size of the field (affects SearchForm control on browser only). Defaults to value of 6 for SearchForm. required (optional) - Marks the field as required and applies clients side validation to ensure value is present when save operation is invoked from edit view (Overrides the value set in vardefs.php). javascript (optional) - Custom JavaScript to embed within the <select> tag element (see above example for onchange event implementation).

Last Modified: 09/26/2015 04:14pm

Parent

Overview

Renders a dropdown for the parent module type.

Parent

The parent field combines a blend of a dropdown for the parent module type, and a text field with code to allow QuickSearch for quicksearch-enabled modules (see ./include/SugarFields/Fields/Parent/EditView.tpl file contents and JavaScript code for more information on enabling Quicksearch). There are also buttons to invoke popups and a button to clear the value. Because the parent field assumes proper relationships within the Sugar modules, it is not a field you can add through Studio or attempt to type override in the metadata files.

Last Modified: 11/18/2015 12:07am

Password

Overview

Renders an input text field for passwords.

Password

The password field simply renders an input text field with the type attribute set to "password" to hide user input values. It is available to edit views only.

name	Standard name definition when metadata definition is defined as an array
displayParams	required (optional) - Marks the field as required and applies clients side validation to ensure value is present when Save operation is invoked from edit view (Overrides the value set in vardefs.php). size (optional) - Override to set the display length attribute of the input field (defaults to 30).

Last Modified: 11/18/2015 12:07am

Phone

Overview

Renders a call to trigger VOIP applications.

Phone

The phone field simply invokes the callto:// URL references that could trigger Skype or other VOIP applications installed on the user's system. It is rendered for

detail views only.

Last Modified: 11/18/2015 01:08am

Radioenum

Overview

Renders a group of radio buttons.

Radioenum

The Radioenum field renders a group of radio buttons. Radioenum fields are similar to the enum field, but only one value can be selected from the group.

Last Modified: 11/18/2015 01:08am

Readonly

Overview

Directs edit view calls to use the SugarField's detail view display.

Readonly

The readonly field simply directs edit view calls to use the SugarField's detail view display. There are no Smarty .tpl files associated with this field.

Last Modified: 11/18/2015 01:08am

Relate

Overview

Combines a blend of a text field with code to allow quick search.

Relate

The Relate field combines a blend of a text field with code to allow quick search. The quicksearch code is generated for edit views (see ./include/TemplateHandler/TemplateHandler.php). There are also buttons to invoke popups and a button to clear the value. For detail views, the Relate field creates a hyperlink that will bring up a detail view request for the field's value.

Within the metadata file, the Relate field can be rendered with the snippet:

```
array (    array(        'name' => 'account_name',        'type' =>
'relate',        'displayParams' => array(            'required' =>
true        )    ),),
```

This will create a relate field that allows the user to input a value not found in the quicksearch list.

name	Standard name definition when metadata definition is defined as an array
displayParams	required (optional) - Marks the field as required and applies clients side validation to ensure value is present when Save operation is invoked from edit view (Overrides the value set in vardefs.php). readOnly (optional for editviewdefs.php file only) - Makes the text field input area readonly so that you have to just use the popup selection. popupData - This field is generated for you by default. See include/SugarFields/Fields/SugarFieldRelate.php for more information. You should not need to override this setting. hideButtons (optional for editviewdefs.php SearchForm.php and popupdefs.php) – Hides the Select and Clear buttons normally displayed next to the editable Relate field.

Last Modified: 01/15/2016 09:30pm

Text

Overview

Renders an HTML textarea form element for edit views.

Text

The Text field renders a HTML form element for edit views and displays the field value with newline characters converted to HTML elements in detail views.

Name	Standard name definition when metadata definition is defined as an array
displayParams	required (optional) - Marks the field as required and applies clients side validation to ensure value is present when Save operation is invoked from edit view (Overrides the value set in vardefs.php). maxlength (optional for editviewdefs.php file only) - Sets the maximum length of character input allowed in the field. rows (optional for editviewdefs.php file only) - Sets the number of rows in the field. cols (optional for editviewdefs.php file only) - Sets the number of cols in the field.

Last Modified: 11/18/2015 01:08am

Username

Overview

A helper field that assumes a salutation, first_name and last_name.

Username

The Username field is a helper field that assumes a salutation, first_name and last_name field exists for the vardefs of the module. It displays the three fields in the format:

```
{salutation} {first_name} {last_name}
```

Last Modified: 09/26/2015 04:14pm

Examples

Legacy MVC metadata examples.

Last Modified: 11/18/2015 01:08am

Adding QuickSearch to a custom field

Overview

How to add QuickSearch to a custom field.

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

Adding QuickSearch to a Custom Field

1. Include the default configs from ./include/QuickSearchDefaults.php. Most of the time, you can use the predefined configurations and scripts.

```
require_once('include/QuickSearchDefaults.php');
 $qsd = new QuickSearchDefaults();
```

2. Then, set up the config for the input box you wish to have SQS. Account, Team, and User configs are available. The following configures SQS for account search on an input box with an id of 'parent_name', and user and team in a similar fashion with defaults.

```
$sqs_objects = array(
     'parent_name' => $qsd->getQSParent(),
     'assigned_user_name' => $qsd->getQSUser(),
     'team_name' => $qsd->getQSTeam()
 );
```

Notes on structure of config - replace the default parameters if they are different for the page.

- method : Unless you make a new method on the JSON server, keep this as 'query'.
- populate_list : This defines the id's of the fields to be populated after a selection is made.

  QuickSearch will map the first item of field_list to the first item of populate_list. ie. field_list[0] = populate_list[0], field_list[1] = populate_list[1].... until the end of populate list.

- limit : reduce from 30 if the query is a large hit, but never have this less than 12.

- conditions : options are like_custom, contains, or default of starts with

  if using 'like_custom' also define 'begin'/'end' for strings to prepend or append to the user input

  disable: set this to true to disable SQS (optional, useful for disabling SQS on parent types in calls for example)

- post_onblur_function : this is an optional function to be called after the user has made a selection. It will be passed in an array with the items in field_list as keys and their corresponding values for the selection as values.

```
$sqs_objects = array (
    'account_name' => $qsd->getQSParent(),
    // the method on to use on the JSON server
    'method' => 'query',
    'modules' => array ('Accounts'), // modules to use
    'field_list' => array ('name', 'id'), // columns to select
```

```
            // id's of the html tags to populate with the columns.
            'populate_list' => array ('account_name', 'account_id'),
            'conditions' => // where clause, this code is for any
        account names that have A WORD beginning with ...
                array (array
        ('name'=>'name','op'=>'like_custom','end'=>'%','value'=>''),
                array ('name' => 'name', 'op' => 'like_custom',
        'begin' => '% ', 'end' => '%', 'value' => '')),
            'group' => 'or', // grouping of the where conditions
            'order' => 'name', // ordering
            'limit' => '30', // number of records to pull
            'no_match_text' => $app_strings['ERR_SQS_NO_MATCH'], //
        text for no matching results
        ),
```

3. Include the necessary javascript files if sugar_grp1.js is not already loaded on the page.

```
    $quicksearch_js = '<script type="text/javascript" src="' .
getJSPath('include/javascript/sugar_grp1.js') . '"></script>';
```

4. Assign your config array to sqs_objects (important!)

```
    require_once('include/JSON.php');
$json = new JSON();
$quicksearch_js .= '<script type="text/javascript"
language="javascript">
sqs_objects = ' . $json->encode($sqs_objects) . '</script>';
```

5. Add validation so that if there is no matching id for what the user has entered in an SQS field, an alert shows. This is for fields such as assigned users where the form must submit a valid ID.

```
    $javascript = new javascript();
$javascript->setFormName('EditView');
$javascript->setSugarBean($this->bean);
$javascript->addToValidateBinaryDependency('account_name', 'alpha',
app_strings['ERR_SQS_NO_MATCH_FIELD'] . $mod_strings['LBL_MEMBER_OF'],
'false', '', 'parent_id');
echo $javascript->getScript();
```

6. Add id tags and class name to the input box. Note that the input box must have class="sqsEnabled"!

```
    <input class="sqsEnabled" id="account_name" name='account_name'
size='30' type='text' value="{ACCOUNT_NAME}">
<input id='account_id' name='account_id' type="hidden"
value='{ACCOUNT_ID}'>
```
Having trouble? Take a look at the file ./module/Contacts/BusinessCard.php.

Last Modified: 09/26/2015 04:14pm

Creating a Custom Sugar Field

Overview

Creates a new SugarField for rendering a YouTube video.

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

Creating a Custom Sugar Field

In this example, we will use a custom text field in the Contacts module and then override the detail view of the custom field in the metadata file to link to our YouTube video.

The process is as follows:

1. Create a new TextField under Accounts named 'youtube_c'. You can do this by navigating to Admin > Studio > Accounts > Fields > Add Field.
2. Add this custom text field to both the edit view and detail view layouts. Save and deploy the updated layouts.
3. Create the directory ./custom/include/SugarFields/Fields/YouTube/. The name of the directory (YouTube) corresponds to the name of the field type you are creating.
4. In the ./include/SugarFields/Fields/YouTube/ directory, create the file DetailView.tpl. For the detail view we will use the "embed" tag to display the video. In order to do this, you need to add the following to the template file:

./include/SugarFields/Fields/YouTube/DetailView.tpl

```
{if !empty({{sugarvar key='value' string=true}})}

    <object width="425" height="350">
        <param name="movie" value="http://www.youtube.com/v/{{sugarvar
key='value'}}></param>
        <param name="wmode" value="transparent"></param>
        <embed src="http://www.youtube.com/v/{{sugarvar key='value'}}"
type="application/x-shockwave-flash" wmode="transparent" width="425"
height="350"></embed>
    </object>

{/if}
```

You will notice that we use the "{{" and "}}" double brackets around our variables. This implies that that section should be evaluated when we are creating the cache file. Remember that Smarty is used to generate the cached templates so we need the double brackets to distinguish between the stage for generating the template file and the stage for processing the runtime view.
Also note that will use the default edit view implementation that is provided by the base sugar field. This will give us a text field where people can input the YouTube video ID, so you do not need to create EditView.tpl. Also, we do not need to provide a PHP file to handle the SugarField processing since the defaults will suffice.

5. Now go to ./custom/modules/Accounts/metadata/detailview.php and add a type override to your YouTube field and save. In this example, the custom field is named "youtube_c".

```
array (
    'name' => 'youtube_c',
    'type' => 'YouTube',
```

```
    'label' => 'LBL_YOUTUBE',
),
```

6. Navigate to Admin > Repair > Quick Repair and Rebuild.
7. Your custom field is now ready to be displayed. You can now find the ID value of a YouTube video to insert into the edit view, and render the video in the detail view. An example you can try is "KDK1TmjmZsw".

Note: It is important to set your system to Developer Mode or run a Quick Repair and Rebuild to rebuild the cache directory.

Last Modified: 09/26/2015 04:14pm

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

```
$viewdefs['Quotes'] = array (
  'DetailView' =>
  array (
    'templateMeta' =>
    array (
      'form' =>
      array (
        'closeFormBeforeCustomButtons' => true,
        'buttons' =>
        array (
          0 => 'EDIT',
          1 => 'SHARE',
          2 => 'DUPLICATE',
          3 => 'DELETE',
          4 =>
          array (
            'customCode' => '<form action="index.php" method="POST"
name="Quote2Opp" id="form">
                    <input type="hidden" name="module" value="Quotes">
                    <input type="hidden" name="record"
value="{$fields.id.value}">
                    <input type="hidden" name="user_id"
value="{$current_user->id}">
                    <input type="hidden" name="team_id"
value="{$fields.team_id.value}">
                    <input type="hidden" name="user_name"
value="{$current_user->user_name}">
                    <input type="hidden" name="action"
value="QuoteToOpportunity">
                    <input type="hidden" name="opportunity_subject"
value="{$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="{$fields.total.value}">
                    <input type="hidden" name="valid_until"
value="{$fields.date_quote_expected_closed.value}">
                    <input type="hidden" name="currency_id"
value="{$fields.currency_id.value}">
                    <input id="create_opp_from_quote_button"
title="{$APP.LBL_QUOTE_TO_OPPORTUNITY_TITLE}"
                        class="button" type="submit"
name="opp_to_quote_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.

Last Modified: 09/26/2015 04:14pm

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 deactivatated 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 soley 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="JavaScript Button" class="button" type="button"
name="JavaScriptButton" value="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="customFormField" 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="SubmitStockFormButton" value="Submit Stock Form Button"
onclick="var _form = document.getElementById(\'formDetailView\');
_form.customFormField.value = \'CustomValue\'; _form.action.value =
\'CustomAction\'; SUGAR.ajaxUI.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"
name="CustomForm" id="form"><input type="hidden"
name="customFormField" name="customFormField"
value="CustomValue"><input id="SubmitCustomFormButton" title="Submit
Custom Form Button" class="button" type="submit"
name="SubmitCustomFormButton" value="Submit Custom Form
Button"></form>',

          ),

        ),

      ),
```

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',
                ),
            ),
```

Last Modified: 09/26/2015 04:14pm

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="Custom Button">'
    );

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

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

Last Modified: 09/26/2015 04:14pm

Modifying Layouts to Display Additional Columns

Overview

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

On-Demand

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:

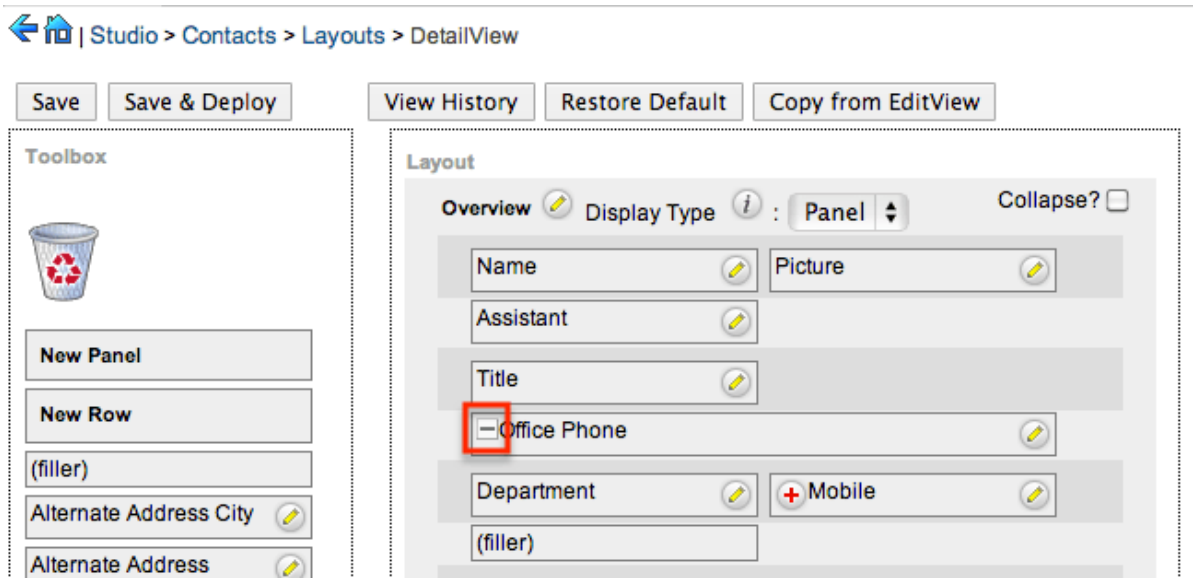
```
'maxColumns' => '3',
```

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 On-Demand 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.

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:

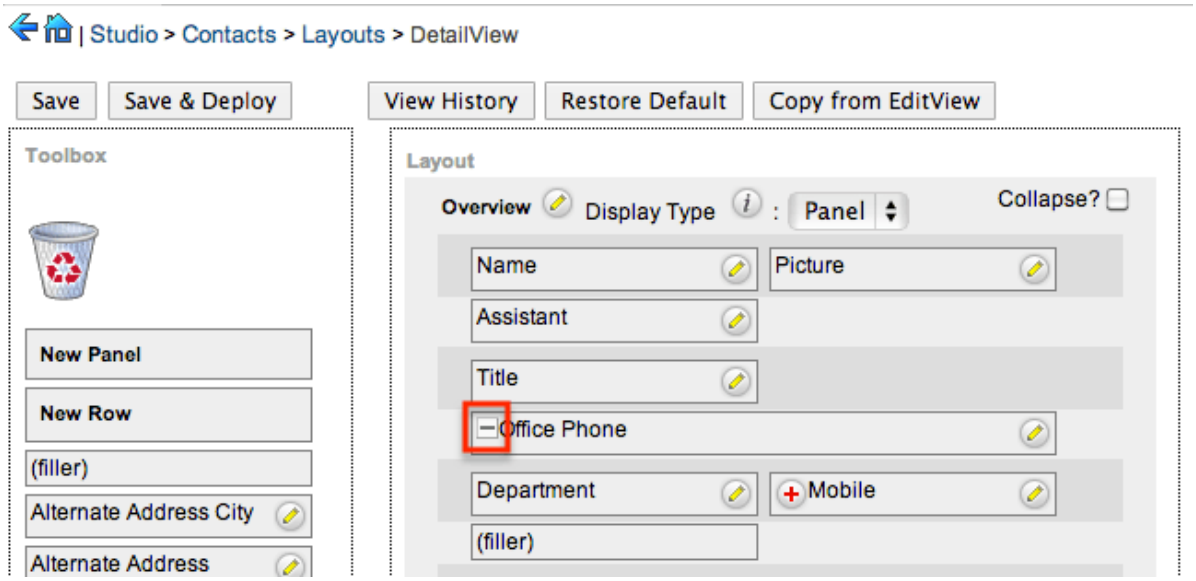
```
'maxColumns' => '3',
```

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!

Last Modified: 09/26/2015 04:14pm

Examples

Provides an overview of example MVC customizations

Last Modified: 11/18/2015 01:08am

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. Please note that when creating or moving files you will need to rebuild the file map. More information on rebulding the file map can be found in the SugarAutoLoader . You should also 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 $this->tabs to show as the
current tab
     *
     * @return string html
     */
    function _displayTabs($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']) ||
```

```
$_REQUEST['search_form'] != 'false')
        {
            if(isset($_REQUEST['searchFormTab']) &&
$_REQUEST['searchFormTab'] == '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->module . '/metadata/searchdefs.php')) {
            require_once('include/SearchForm/SearchForm.php');
            $this->searchForm = new SearchForm($this->module,
$this->seed);

            }  else {

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

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

            $this->searchForm = $this->getSearchForm2($this->seed,
$this->module, $this->action);
            $this->searchForm->setup($searchMetaData['searchdefs'],
$searchMetaData['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';
    }
}

?>
```

Last Modified: 09/26/2015 04:14pm

Data Framework

Data Framework Overview.

Last Modified: 11/18/2015 12:07am

Vardefs

Vardef Overview.

Last Modified: 11/18/2015 12:07am

Introduction

Overview

Vardefs (Variable Definitions) are used to provide the Sugar application with information about SugarBeans.

Vardefs

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.

Vardef files create an array called "$dictionary", which contains several entries including tables, fields, indices, and relationships.

Dictionary Array

- table : The name of the database table (usually, the name of the module) for this bean that contains the fields.
- audited : Defines whether the module has field level auditing enabled.
- fields : A list of fields and their attributes (see below).
- indices : A list of indexes that should be created in the database (see below).
- optimistic_locking : Set to true if this module should obey optimistic locking. Optimistic locking uses the modified date to ensure that the bean you are working on has not been modified by anybody else when you try and save to prevent loss of data.
- unified_search : Set to true if this module is available to be searched through the Global Search, defaults to false. This has no effect if none of the fields in the fields array have the 'unified_search' attribute set to true.
- unified_search_default_enabled : Set to true if this module should be searched by default for new users through the Global Search. This setting defaults to true and has no effect if 'unified_search' is not set to true.

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
    - name : A non-db 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 detailview
    - varchar : A variable-sized string
- table : The database table the field comes from. This is only used when needing to joining 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 (for commercial editions).
- default : The default value for this field.
- massupdate : Set to false if you do not want this field to show up in the mass update panel at the bottom of a module list view. Some field types are restricted from mass update regardless of this setting.
- rname : (for relate-type fields only) The field from the related variable that has the text.
- id_name : (for relate-type fields only) 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. This 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 only) An array containing the fields that are concatenated.
- db_concat_fields : (for concatenated values only) An array containing the fields to concatenate in the database.
- unified_search : Set to true if this field should be searched through 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 only) Set to true if this field should support range searches in the Basic and Advanced Search layouts of the module.
- 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 for which the field's visibility should be removed from (ex: 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 name of the index. This must be unique in most databases.
- type : Thy type of index (primary, unique, index).
- fields : The fields to index. This is an ordered array.
- source : Set to 'non-db' if you are creating an index for added application functionality such as duplication checking on imports.

The following example is to create a primary index called 'userspk' on the 'id' column

```
array(
    'name' => 'userspk',
    'type' => 'primary',
    'fields'=> array('id')
),
```

Relationships Array

The relationships array is used to specify 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 ('one-to-many' or '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 reporting relationship between a contact and the person that they report to. The reports_to_id field is used to map to the id field in the contact of the person they report to. This is a one-to-many relationship in that each person is only allowed to report to one person. Each person is allowed to have an unlimited number of direct reports.

```
'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'
),
```

Extending Vardefs

More information about extending and overriding vardefs can be found in the extensions framework under Vardefs.

Last Modified: 01/25/2016 09:48am

Examples

Vardef Examples.

Last Modified: 11/18/2015 12:07am

Manually Creating Custom Fields

Overview

An overview of alternate ways to create custom fields without using studio.

Using ModuleInstaller

There are two ways to create a field using the ModuleInstaller class. 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 Creating an Installable Package that Creates New Fields section. Alternatively, you can programmatically add your custom fields by using the ModuleInstaller class with the install_custom_fields() method.

An example is below:

```
<?php

    $fields = array (
        //Text
        array(
            'name' => 'text_field_example',
            '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',
            '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',
            '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',
            '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',
    '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',
    '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',
    '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'
        ),
    );

    require_once('ModuleInstall/ModuleInstaller.php');
    $moduleInstaller = new ModuleInstaller();
    $moduleInstaller->install_custom_fields($fields);
```

You can add in the labels for your custom fields by creating a corresponding language extension file and running a Quick Repair and Rebuild:

./custom/Extension/modules/Accounts/Ext/Language/en_us.<name>.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';
```

Using the Vardef Extensions

You should try to avoid creating your own custom fields using the vardefs as there are several caveats:

- If your installation does not already contain custom fields, you must manually create the custom table. Otherwise the system will not recognize your fields custom vardef. This situation is outlined in the section below.

- You will have to run a Quick Repair and Rebuild, then execute the generated SQL after the vardef is installed.
- You have to correctly define the properties of a vardef. If you miss any, your field may not work properly.
- Your field name should end with an "_c" and should have the property 'source' set as 'custom_fields'. This is required as you should not modify core tables in

Sugar and it is not permitted on On-Demand.
- Your vardef should specify the exact indexes of the properties you want to set. The example being to use: $dictionary['<module singular>']['fields']['example_c']['name'] = 'myfield_c'; instead of $dictionary['<module singular>']['fields']['example_c'] = array(['name' => 'myfield_c');. This will help prevent the system from losing any properties when loading from the extension framework.

The initial problem with creating your own custom vardef is to get the system to recognize the vardef and generate the database field. The problem will be illustrated with the example below:

./custom/Extension/modules/<module>/Ext/Vardefs/<file>.php

```
<?php

$dictionary['<module singular>']['fields']['example_c']['name'] =
'example_c';
$dictionary['<module singular>']['fields']['example_c']['vname'] =
'LBL_EXAMPLE_C';
$dictionary['<module singular>']['fields']['example_c']['type'] =
'varchar';
$dictionary['<module singular>']['fields']['example_c']['enforced'] =
'';
$dictionary['<module singular>']['fields']['example_c']['dependency']
= '';
$dictionary['<module singular>']['fields']['example_c']['required'] =
false;
$dictionary['<module singular>']['fields']['example_c']['massupdate']
= '0';
$dictionary['<module singular>']['fields']['example_c']['default'] =
'';
$dictionary['<module singular>']['fields']['example_c']['no_default']
= false;
$dictionary['<module singular>']['fields']['example_c']['comments'] =
'Example Varchar Vardef';
$dictionary['<module singular>']['fields']['example_c']['help'] = '';
$dictionary['<module singular>']['fields']['example_c']['importable']
= 'true';
$dictionary['<module
singular>']['fields']['example_c']['duplicate_merge'] = 'disabled';
$dictionary['<module
singular>']['fields']['example_c']['duplicate_merge_dom_value'] = '0';
$dictionary['<module singular>']['fields']['example_c']['audited'] =
false;
$dictionary['<module singular>']['fields']['example_c']['reportable']
```

```
= true;
$dictionary['<module
singular>']['fields']['example_c']['unified_search'] = false;
$dictionary['<module
singular>']['fields']['example_c']['merge_filter'] = 'disabled';
$dictionary['<module singular>']['fields']['example_c']['calculated']
= false;
$dictionary['<module singular>']['fields']['example_c']['len'] =
'255';
$dictionary['<module singular>']['fields']['example_c']['size'] =
'20';
$dictionary['<module singular>']['fields']['example_c']['id'] =
'example_c';
$dictionary['<module
singular>']['fields']['example_c']['custom_module'] = '';
//required to create the field in the _cstm table
$dictionary['<module singular>']['fields']['example_c']['source'] =
'custom_fields';
```

Once we have the vardef in place, we will need to consider if the custom field is for a module that doesn't currently have any other custom fields. If there are not any existing custom fields, we will need to create a corresponding record in fields_meta_data that will trigger the comparison process.

```
INSERT INTO fields_meta_data (id, name, vname, comments,
custom_module, type, len, required, deleted, audited, massupdate,
duplicate_merge, reportable, importable) VALUES ('<module>example_c',
'example_c', 'LBL_EXAMPLE_C', 'Example Varchar Vardef', '<module>',
'varchar', 255, 0, 0, 0, 0, 0, 1, 'true');
```

Finally, we can now navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions. After the repair, you will notice a section at the bottom stating that there are differences between the database and vardefs. You will need to execute the scripts generated to create your custom field:

Missing <module>_cstm Table:

```
/*Checking Custom Fields for module : <module>*/

CREATE TABLE <module>_cstm (id_c char(36)  NOT NULL  , PRIMARY KEY
(id_c)) CHARACTER SET utf8 COLLATE utf8_general_ci;
```

Missing Columns:

```
/*MISSING IN DATABASE - example_c -  ROW*/
```

```
ALTER TABLE <module>_cstm add COLUMN example_c varchar(255)  NULL ;
```

Last Modified: 01/15/2016 09:20pm

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 Custom 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:

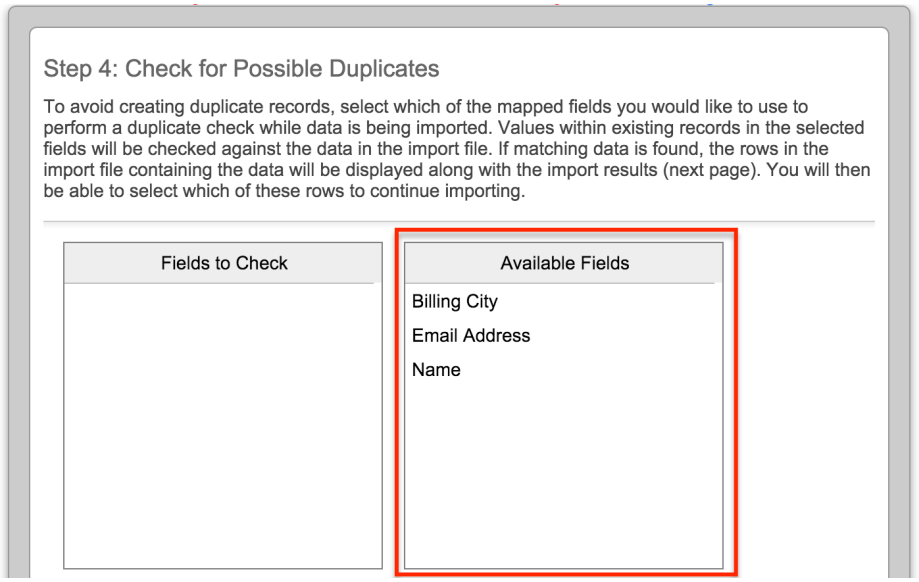
```
<?php

$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.

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 the following file from the root directory of your Sugar installation on the web server:

`./custom/Extension/modules/Contacts/Ext/Vardefs/custom_import_index.php`

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.

Last Modified: 07/14/2016 09:02am

Relationships

Relationship Overview.

Relationship Overview.

Last Modified: 11/18/2015 01:08am

Introduction

Overview

Provides a technical overview of relationships.

Relationships

Relationships are the basis for linking information within the system. This section will address the various aspects of relationships.

Definitions

Relationships are intially defined in the modules vardefs file under the relationships array. For your reference you can find them using the vardef path as follows:

```
./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). An example of this is the relationship between Accounts and Cases where there are 3 tables: accounts, accounts_cases, and cases. You will find that the joining table, accounts_cases, will contain the fields needed in order to establish the relationship link. The fields on the accounts_cases table are listed below:

Fields	Description
id	A unique identifier for the relationship row. This field is not typically used.
account_id	The ID for the related account record. This field will be named differently based on the relationship.
case_id	The ID for the related case record. This field will be named differently 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
```

Alternatively, a list of all the relationships can also be found in the relationships table:

```
SELECT * FROM relationships WHERE deleted = 0;
```

Last Modified: 01/15/2016 09:05pm

Custom Relationships

Overview

Provides a technical overview of custom relationships.

Custom Relationships

Relationships are intially defined in the modules vardefs file under the relationships array. For your 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 is as follows:

- Defining the Relationship MetaData.
- Defining the Relationship in the TableDictionary.

Defining the Relationship MetaData

The defintions 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 relationships structure. Possible values are: 'one-to-many' and '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 defininging 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 left table.
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 right table
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	

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 relationships 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');

?>
```

Considerations

If you have created your relationship through studio, the files above will be automatically created for you. If you are manually creating the files, you will need to 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.

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

Last Modified: 01/15/2016 09:05pm

Language Framework

An overview of languages, labels, and lists. Sugar as an application platform is internationalized and localizable. Data is stored and presented in the UTF8 codepage allowing for all character sets to be used. Sugar provides a language pack framework allowing developers to build support for any language to be used in the display of user interface labels. Every language pack has its own set of display strings which is the basis of language localization. You can add edit or modify these languages using this guide.

Last Modified: 02/02/2016 05:14pm

Language Keys

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
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
Slovak (Slovenčina)	sk_SK
Spanish (Español)	es_ES
Spanish (Latin America) (Español	es_LA

Language	Language Key
(Latinoamérica))	
Swedish (Svenska)	sv_SE
Turkish (Türkçe)	tr_TR
Ukrainian (Українська)	uk_UA

Change Log

The following table documents historical changes to Sugar's available languages.

Version	Change
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 English (UK) language pack.
6.4.0	Added Catalan language pack.
6.4.0	Added Portuguese Brazilian 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.

Version	Change
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.

Last Modified: 10/22/2016 11:34am

Application Labels and Lists

Overview

How to work with the application language strings.

Language Keys

As Sugar is fully internationalized and localizable, each language is defined by a unique language key. The language keys will prefix any files dealing with languages. The default language for the application is English (en_us). Any language label keys that are not found for a language will default to the English version. For more information regarding language keys, please refer to the Language Keys section.

Application Labels

$app_list_strings / $app_strings

The $app_list_strings contains the various drop down lists for the system while $app_strings contains the system application labels. The initial set of definitions

can be found in the following directory:

```
./include/language/<language key>.lang.php
```

As you begin working within the system and deploying modules and lists through Studio, any changes to these labels will be reflected in the corresponding custom directory:

```
./custom/include/language/<language key>.lang.php
```

Customizing Application Labels and Lists

If you are developing a customization and want to be able to create or edit existing label/list values, you will need to work within the extension application directory. To do this you will create a file as follows:

./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_strings['LBL_KEY'] = 'My Display Label';
    $app_list_strings['LIST_NAME']['Key_Value'] = 'My Display Value';
```

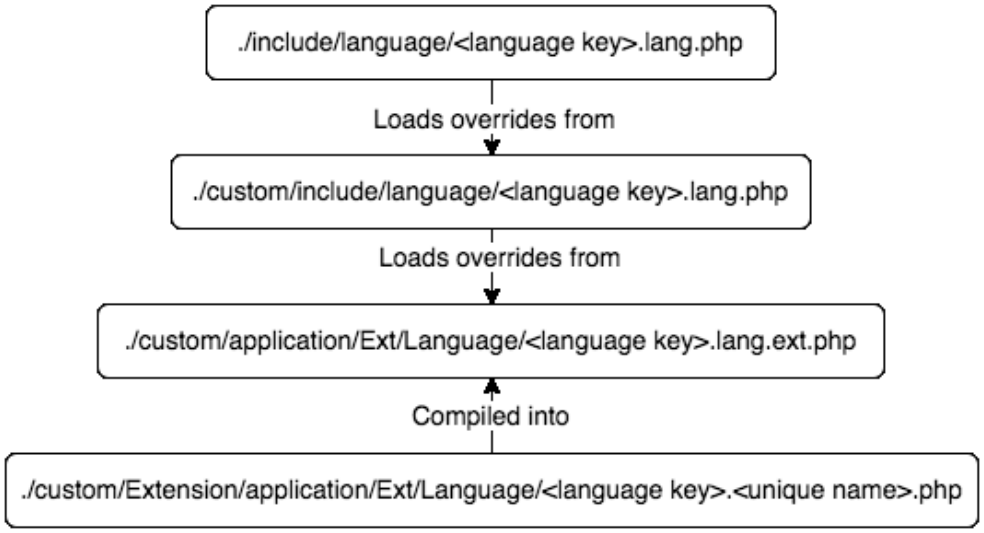
Once the file is created with your adjustments, you will then navigate to Admin > Repair > Quick Rebuild & Repair. This will compile all of the Extension files from:

```
./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'. In using this function, the label will be retrieved for the current users language. This function 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 users 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 list. The first is to use the 'translate' function found in 'include/utils.php'. In using this function, the label will be retrieved for the

current users 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 users preferred language, the system will default to "en_us" and pull the English (US) version of the label for display.

Accessing Application Strings with Sidecar

All language pack strings are accessible within the Sidecar framework.

$app_strings

To access the $app_strings in sidecar, you can use app.lang.getAppString:

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

To access the $app_strings in your browsers console, you can use SUGAR.App.lang.getAppString:

```
SUGAR.App.lang.getAppString('LBL_MODULE');
```

More information can be found in the app.lang.getAppString language section.

$app_list_strings

To access the $app_strings in sidecar, you can use app.lang.getAppListStrings:

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

To access the $app_strings in your browsers console, you can use SUGAR.App.lang.getAppListStrings:

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

More information can be found in the app.lang.getAppListStrings language section. section.

Last Modified: 03/08/2016 04:11pm

Module Labels

Overview

How to work with the module language strings.

Language Keys

As Sugar is fully internationalized and localizable, each language is defined by a unique language key. The language keys will prefix any files dealing with languages. An example of a language key is 'en_us' which is used for English (US). For more information regarding please refer to the Language Keys section.

Module Labels

$mod_strings

The module language strings are stored in $mod_strings. This section will outline how the $mod_strings are compiled. All modules, whether out-of-box or custom will have an initial set of language files in the following directory:

```
./modules/<module>/language/<language key>.lang.php
```

As you begin working within the system and modifying labels through Studio, any changes to these labels will be reflected in the corresponding modules custom directory:

```
./custom/modules/<module>/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 you will create a file as follows:

```
./custom/Extension/modules/<module>/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

    $mod_strings['LBL_KEY'] = 'My Display Label';
```

Once the file is created with your adjustments, you will then 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
```

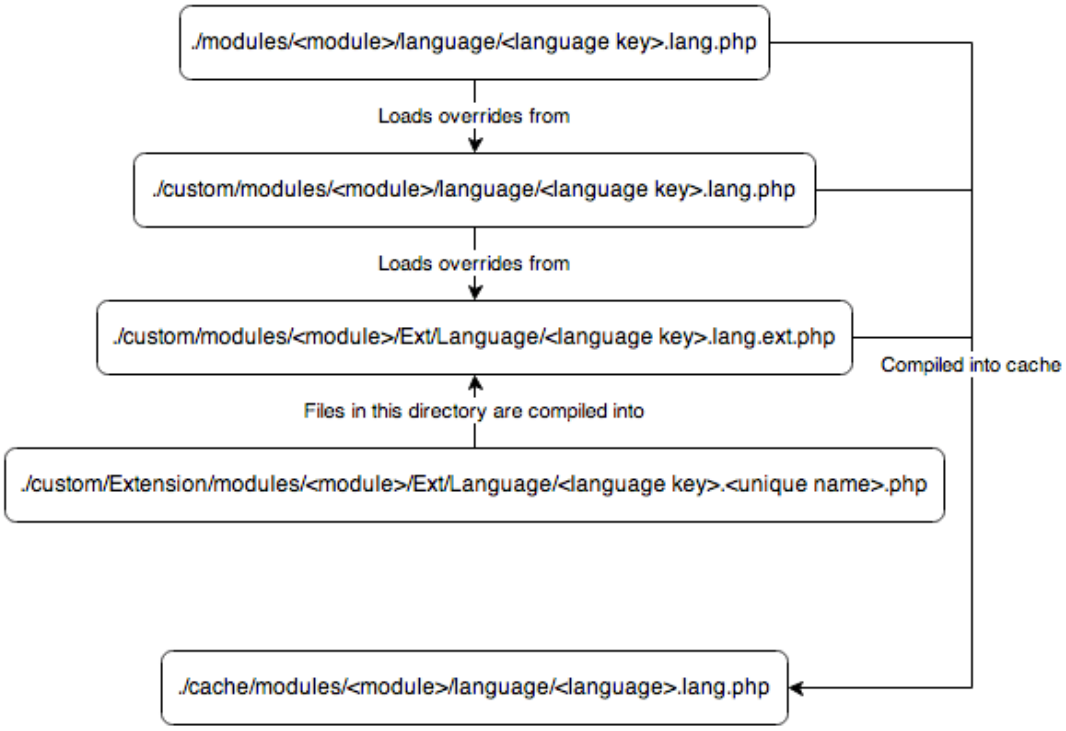
Label Cache

The file locations discussed above will all be compiled into the cache directory.

./cache/modules/<module>/language/<language key>.lang.php

The cached results of these files are what make up each modules '$mod_strings' defintion.

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'. In using this function, the label will be retrieved for the current users language. This function 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 also 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 with Sidecar

All language pack strings are accessible within the Sidecar framework.

$mod_strings

To access the $mod_strings in sidecar, you can use app.lang.get:

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

To access the $mod_strings in your browsers console, you can use SUGAR.App.lang.get:

```
SUGAR.App.lang.get('LBL_NAME', 'Accounts');
```

More information can be found in the app.lang.get language section.

Last Modified: 09/26/2015 04:14pm

Managing Lists

Overview

Provides a technical overview of the various ways to manage lists.

Modifying Lists

There are 3 ways to modify lists within Sugar.

Studio

Lists can be managed in 2 ways within Studio. If you know the name of the list you would like to edit, you can navigate to the dropdown editor:

```
Admin / Studio / Dropdown Editor
```

Alternatively, you can also navigate to your module field using the list and edit it by going to:

```
Admin / Studio / <module> / Fields / <field>
```

Direct Modification

There are two ways to directly modify the language strings. The first way is to modify the custom language file.

```
./custom/include/language/<language key>.lang.php
```

If you are developing a customization to be distributed and 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 a file as follows:

```
./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, you will then 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
```

Dropdown Helper

You can use the Dropdown Helper to manage the lists at a code level. The example below 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
values
    $parameters['use_push'] = true;
    $count++;
}

$dropdownHelper->saveDropDown($parameters);
```

Last Modified: 02/03/2016 11:43am

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.

Example Manifest File

```
<?php

$manifest = array (
  'acceptable_sugar_versions' =>
  array (
    'regex_matches' =>
    array (
      0 => '7\\.5\\.*.*',
      1 => '7\\.6\\.*.*',
      2 => '7\\.7\\.*.*',
    ),
  ),
  'acceptable_sugar_flavors' =>
  array (
    0 => 'PRO',
```

```
    1 => 'CORP',
    2 => 'ENT',
    3 => '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' => '2016-02-03 04:39:12',
  '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.

Last Modified: 02/03/2016 10:03am

Sugar Logic

Sugar Logic Overview.

Last Modified: 11/18/2015 01:08am

Introduction

Overview

Sugar Logic, a new feature in Sugar Enterprise and Sugar Professional, is designed to allow 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 which build off 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 which can be called in a formula.
- Trigger : A Formula which evaluates to either true or false. Triggers are evaluated whenever a field in the equation is updated or when a record is retrieved/saved.
- Action : A method which modifies the current record or layout in some way.
- Dependency : A complete logical unit which 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. They are: number, string, boolean, and enum (lists). Functions may take in any of these type 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 in 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, as in whether a condition is met or not. For example, the function contains takes in as input two strings 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 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

```
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 the steps listed below 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), 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 is performed, triggers the associated Actions in a Dependency.

Actions

Actions are functions which modify a target in some way. 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.

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 in the main directory 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
```

Last Modified: 11/18/2015 11:52pm

Dependency Actions

Overview

The complete list of available actions.

Actions

Class	Action	Parameters	Comments
AssignToUserAction	AssignTo	value: String username,	

PanelVisibilityAction	SetPanelVisibility	target: id of panel to hide, value: Formula used to determine if the panel should be visible,	Does not work on tabbed layouts.
ReadOnlyAction	ReadOnly	target: Name of field to make read-only. value: Formula used to determine if the field should be editable.	Commonly used by calculated fields to prevent editing.
SetOptionsAction	SetOptions	target: Name of field to make modify keys: List of option keys for the dropdown. label: List of option labels for the dropdown.	Used by Dependent Dropdowns.
SetRequiredAction	SetRequired	target: Name of field to make required label: id of label element for this field. value: Formula used to determine if the field should be required.	
SetValueAction	SetValue	target: Name of field to modify. value: Formula to get the value for target field.	Used by Calculated fields.
VisibilityAction	SetVisibility	target: Name of field to hide value: Formula used to determine if the field should be visible.	Used by Dynamic fields.

Last Modified: 09/26/2015 04:14pm

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

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 in to this object. So with all this information we can go ahead and write the code for the function.

```
<?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
positive");
        }

        // 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 the admin panel.

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".

A simple action could be a "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 run time. We would do this by creating a php file in ./custom/include/Expressions/Actions/WarningAction.php as shown below:

```
<?php

require_once("include/Expressions/Actions/AbstractAction.php");

class WarningAction extends AbstractAction    {

    protected $messageExp = "";
```

```
    function SetZipCodeAction($params)
    {
        $this->messageExp = $params['message'];
    }

    /**
    * 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.AbstractAction,
        {
            //javascript exection code
            exec : function()
            {
            //assume the message is a formula
            var msg =
SUGAR.forms.evalVariableExpression(this.messageExp);
            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";
    }
}

?>
```

Last Modified: 09/26/2015 04:14pm

Using Sugar Logic Directly

How to use Sugar Logic.

Last Modified: 11/18/2015 01:08am

Accessing an External API with a Sugar Logic Action

Let us say we were building a new Action called "SetZipCodeAction" that uses the yahoo geocode API to get the zip code for a given street + city + state address.

Since the Yahoo Geocode 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.9LTMPqQnSQo4QsCnjF5wIvyYRSXPIzqlDbI.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={$state}";

    //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'],
$_REQUEST['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. We need to 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['target'];
        $this->streetExp = empty($params['street']) ? " " :
$params['street'];
        $this->cityExp = empty($params['city']) ? " " :
$params['city'];
```

```
        $this->stateExp = empty($params['state']) ? " " :
$params['state'];
    }

    static function getJavascriptClass()
    {
        return "'($this->target}', '{$this->streetExp}',
'{$this->cityExp}', '{$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. This will change in the future and creating custom dependencies and taking advantage of custom actions should become a simpler process requiring little to no custom code.

Last Modified: 01/29/2016 04:43pm

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 in the call subject
            $dep->addAction(ActionFactory::getNewAction('SetRequired',
array(
                '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.

Last Modified: 09/26/2015 04:14pm

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(),
    );
```

Last Modified: 09/26/2015 04:14pm

Using dependencies in Logic Hooks

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);
    }
```

```
}
```

Last Modified: 09/26/2015 04:14pm

Extension Framework

Extension Framework

The extension framework provides the capability to modify Sugar metadata, such as vardefs and layouts, in a safe way that supports installing, uninstalling, disabling, and enabling without interfering with other customizations.

How it works

Extensions are stored under ./custom/Extension/application/Ext/ for application extensions and ./custom/Extension/modules/<module>/Ext/ for module extensions. The files in 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/. The files contained in this directory are merged into ./custom/modules/Accounts/Ext/Vardefs/vardefs.ext.php any time a module is installed, uninstalled, enabled, disabled, and when a Quick Repair & Rebuild is run. For your reference, the ./ModuleInstall/extensions.php file contains all of the core extension mappings.

Extensions Properties

Each extension contains the following properties:

Internal Name	The internal system name of the extension.
Manifest Installdef	The index of the $installdef in the manifest file.
Extension Directory	The directory containing the extension files.
Extension File	The file where extension files are aggregated to.

The various extensions are listed in the topics below:

Last Modified: 09/26/2015 04:14pm

ActionFileMap

Overview

Maps actions to files. This is handy if you want to map a file to a view outside of ./custom/modules/<module>/views/view.<name>.php. This is only applicable to modules running in backward compatibility mode.

Properties

Internal Name	actionfilemap
Manifest Installdef	action_file_map
Extension Directory	./custom/Extension/modules/<module>/Ext/ActionFileMap/
Extension File	./custom/modules/<module>/Ext/ActionFileMap/action_file_map.ext.php

Example

The following example will create a new action called 'example':

./custom/Extension/modules/<module>/Ext/ActionFileMap/<file>.php

```
<?php

    $action_file_map['example'] = 'custom/example.php';
```

Next, create your action file:

./custom/example.php

```
<?php

    //Encoded as JSON for AJAX layouts
    echo '{"content":"Example View"}';
```

```
?>
```

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

Last Modified: 01/08/2017 04:38pm

ActionReMap

Overview

Maps new actions to already existing actions. This is only applicable to modules running in backward compatibility mode.

Properties

Internal Name	actionremap
Manifest Installdef	action_remap
Extension Directory	./custom/Extension/modules/<module>/Ext/ActionReMap/
Extension File	./custom/modules/<module>/Ext/ActionReMap/action_remap.ext.php

Example

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.

Last Modified: 01/08/2017 04:38pm

ActionViewMap

Overview

Maps additional actions for a module. Previously, all actions needed to be mapped in ./custom/modules/<module>/controller.php. This is only applicable to modules running in backward compatibility mode.

Properties

Internal Name	actionviewmap
Manifest Installdef	action_view_map
Extension Directory	./custom/Extension/modules/<module>/Ext/ActionViewMap/
Extension File	./custom/modules/<module>/Ext/ActionViewMap/action_view_map.ext.php

Example

The following example will map a new action:

./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';
    }

}

?>
```

Finally, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions mapping the view.

Last Modified: 09/26/2015 04:14pm

Administration

Overview

Adds new administrative panels to the administration section.

Properties

Internal Name	administration
Manifest Installdef	administration
Extension Directory	./custom/Extension/modules/Administration/Ext/Administration/
Extension File	./custom/modules/Administration/Ext/Administration/administration.ext.php

More information can be found in the Administration Links section.

Example

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/images
        'Administration',

        //Link name label
        'LBL_LINK_NAME',

        //Link description label
        'LBL_LINK_DESCRIPTION',

        //Link URL - For Sidecar modules

'javascript:parent.SUGAR.App.router.navigate("<module>/<path>",
{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'] = '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 in the Admin section.

Last Modified: 09/26/2015 04:14pm

Dependencies

Overview

Creates dependent actions for fields and forms that can leverage more complicated logic.

Properties

Internal Name	dependencies
Manifest Installdef	dependencies
Extension Directory	./custom/Extension/modules/<module>/Ext/Dependencies/
Extension File	./custom/modules/<module>/Ext/Dependencies/deps.ext.php

More information about dependency actions can be found in the Sugar Logic section.

Example

The following example will create a new required field dependency:

./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")',
//Formula
            ),
        ),
    ),
);
```

Once a Quick Repair and Rebuild is run, the dependency will be in effect.

Last Modified: 09/26/2015 04:14pm

EntryPointRegistry

Overview

Maps additional entrypoints to the system. As entrypoints will soon be deprecated, it is recommended for developers to move any custom logic to endpoints.

Properties

Internal Name	entry_point_registry.ext.php
Manifest Installdef	action_view_map
Extension Directory	./custom/Extension/application/Ext/EntryPointRegistry/

Extension File	./custom/application/Ext/EntryPointRegistry/entry_point_registry.ext.php

More information can be found in the Entry Points section.

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:

./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 rebuild the extensions. More information on custom entry points can be found in the Creating Custom Entry Points section.

Last Modified: 09/26/2015 04:14pm

Extensions

Overview

Allows for custom extensions within the framework. Custom extensions added will

be used along side the extensions found in ./ModuleInstaller/extensions.php.

Properties

Internal Name	extensions
Manifest Installdef	extensions
Extension Directory	./custom/Extension/application/Ext/Extensions/
Extension File	./custom/application/Ext/EntryPointRegistry/extensions.ext.php

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 : Optional. Determines how the framework will interpret the extension.
    - <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 such as 'Accounts'.
        - 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

Example

The following example will create a new extension called 'example'. This will create an extension directory available in ./custom/Extension/application/Ext/Example/.

./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 then you will be able to add your own extension files in: ./custom/Extension/application/Ext/Example/.

Last Modified: 09/26/2015 04:14pm

FileAccessControlMap

Overview

Restricts specific view actions from users of the system. This is only applicable to modules running in backward compatibility mode.

Properties

Internal Name	file_access
Manifest Installdef	file_access
Extension Directory	./custom/Extension/modules/<module>/Ext/FileAccessControlMap/
Extension File	./custom/modules/<module>/Ext/FileAccessControlMap/file_access_control_map.ext.php

Example

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.

Last Modified: 09/26/2015 04:14pm

GlobalLinks (Deprecated)

Overview

Manages the global links in Sugar 6.x. This extension has been deprecated as of 7.x. Information on manipulating the 7.x profile action menu can be found in the MegaMenu section.

Properties

Internal Name	links
Manifest Installdef	linkdefs
Extension Directory	./custom/Extension/application/Ext/GlobalLinks/
Extension File	./custom/application/Ext/GlobalLinks/links.ext.php

Last Modified: 09/26/2015 04:14pm

Include

Overview

Maps additional modules in the system. Normally used when module builder deploys a module.

Properties

Internal Name	modules
Manifest Installdef	beans
Extension Directory	./custom/Extension/application/Ext/Include/
Extension File	./custom/application/Ext/Include/modules.ext.php

Example

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.

Last Modified: 09/26/2015 04:14pm

JSGroupings

Overview

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

Properties

Internal Name	jsgroupings
Manifest Installdef	jsgroups
Extension Directory	./custom/Extension/application/Ext/JSGroupings/
Extension File	./custom/application/Ext/JSGroupings/jsgroups.ext.php

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 the <file>.js in order to make the element key unique.

Examples

Creating New JSGroupings

The following example will add a new JSGrouping file. To accomplish this, create a

JSGrouping extension in ./custom/Extension/application/Ext/JSGroupings/. This file should be uniquely named for your customization. For this example, create:

./custom/Extension/application/Ext/JSGroupings/newGrouping.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',
);
```

./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 create the JSGrouping extension. Once completed, navigate to Admin > Repair > Rebuild JS Grouping Files. This will create the grouping file shown below:

./cache/include/javascript/newGrouping.js

```
function one(){}/* End of File custom/file1.js */function two(){}/*
End 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. The example below will demonstrate how to add a the file ./custom/JavaScript/customFile.js to ./cache/include/javascript/sugar_grp7.min.js. This example can be adapted for any JavaScript file and grouping you choose:

The custom JavaScript file will throw an alert just to let us know it is being included:

./custom/JavaScript/customFile.js

```
alert("Hello World!");
```

Next, create a JSGrouping extension in ./custom/Extension/application/Ext/JSGroupings/. This file should be uniquely named for your customization. For this example, create:

./custom/Extension/application/Ext/JSGroupings/customGrouping.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/JavaScript/customFile.js'] =
'include/javascript/sugar_grp7.min.js';
        }
        break;
    }
}
```

Finally, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and append the JSGrouping extension. After clearing your browser's cache, you should now see an alert message when refreshing your Sugar instance.

Last Modified: 01/08/2017 01:29am

Language

Overview

The Language extension adds or overrides language strings. This extension is applicable to both the application and module framework. More detailed

information can be found in the Language Framework section.

Application Properties

Internal Name	language
Manifest Installdef	language
Extension Directory	./custom/Extension/application/Ext/Language/
Extension File	./custom/application/Ext/Language/<language>.lang.ext.php

$app_strings Example

The following example will create a new $app_strings label:

./custom/Extension/application/Ext/Language/<language>.<name>.php

```
<?php
  $app_strings['LBL_STRING_KEY'] = 'Label Value';
```

Navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions.

Module Properties

Internal Name	language
Manifest Installdef	language
Extension Directory	./custom/Extension/modules/<module>/Ext/Language/
Extension File	./custom/modules/<module>/Ext/Language/<language>.lang.ext.php

$mod_strings Example

The following example will create a new $mod_strings label:

./custom/Extension/modules/<module>/Ext/Language/<language>.<name>.php

```
<?php
  $mod_strings['LBL_STRING_KEY'] = 'Label Value';
```

Navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions.

Last Modified: 12/29/2015 05:00pm

Layoutdefs

Overview

Adds or overrides subpanel definitions. This is only applicable to modules running in backward compatibility mode.

Properties

Internal Name	layoutdefs
Manifest Installdef	layoutdefs
Extension Directory	./custom/Extension/modules/<module>/Ext/Layoutdefs/
Extension File	./custom/modules/<module>/Ext/Layoutdefs/layoutdefs.ext.php

Example

The following example will create a new subpanel for a 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',
            ),
        ),
    );
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions.

Please note that if you are attempting to override parts of an existing subpanel defintion, 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_buttons"] = array(
        array(
            'widget_class' => 'SubPanelTopButtonQuickCreate',
        ),
    );
```

Last Modified: 09/26/2015 04:14pm

LogicHooks

Overview

Adds actions to specific events, such as before saving a bean.

More detailed information can be found in the Logic Hooks section.

Application Hooks

Internal Name	logichooks
Manifest Installdef	hookdefs
Extension Directory	./custom/Extension/application/Ext/LogicHooks/
Extension File	./custom/application/Ext/LogicHooks/logichooks.ext.php

Application Hook Example

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',
        'my_hook_class',
        'example_method'
    );
```

Next, create your hook class:

./custom/application_hook.php

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class my_hook_class
    {
        function example_method($bean, $event, $arguments)
        {
            //logic
        }
```

```
    }

?>
```

Finally, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and the hook will be available.

Module Hooks

Internal Name	logichooks
Manifest Installdef	hookdefs
Extension Directory	./custom/Extension/modules/<module>/Ext/LogicHooks/
Extension File	./custom/modules/<module>/Ext/LogicHooks/logichooks.ext.php

Module Hook Example

The following example will create a new before_save logic hook that executes for a specific module:

./custom/Extension/modules/<module>/Ext/LogicHooks/<file>.php

```
<?php

    $hook_array['before_save'][] = Array(
        1,
        'Custom Logic',
        'custom/modules/<module>/<module>_hook.php',
        'my_hook_class',
        'example_method'
    );
```

Next, create your hook class:

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

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');
```

```
    class my_hook_class
    {
        function example_method($bean, $event, $arguments)
        {
            //logic
        }
    }

?>
```

Finally, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and the hook will be available.

Last Modified: 09/26/2015 04:14pm

Menus (Deprecated)

Overview

Manages module menus in Sugar 6.x. This extension has been deprecated as of 7.x. Information on manipulating the 7.x menu can be found in the MegaMenu section.

Properties

Internal Name	menus
Manifest Installdef	menu
Extension Directory	./custom/Extension/modules//Ext/Menus/
Extension File	./custom/modules//Ext/Menus/menu.ext.php

Last Modified: 09/26/2015 04:14pm

ScheduledTasks

Overview

Adds custom functions that can be used by scheduler jobs.

Properties

Internal Name	schedulers
Manifest Installdef	scheduledefs
Extension Directory	./custom/Extension/modules/Schedulers/Ext/ScheduledTasks/
Extension File	./custom/modules/Schedulers/Ext/ScheduledTasks/scheduledtasks.ext.php

More information can be found in the Schedulers section.

Example

In this example, a custom file will be added that contains our new job function:

./custom/Extension/modules/Schedulers/Ext/ScheduledTasks/<name>.php

```
<?php

array_push($job_strings, 'custom_job');

function custom_job()
{
    //logic here

    //return true for completed
    return true;
}
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and the function will now be available.

Last Modified: 09/26/2015 04:14pm

Sidecar

Overview

Installs metadata files to their appropriate directories.

Properties

<table>
<tr><td>Internal Name</td><td>sidecar</td></tr>
<tr><td>Manifest Installdef</td><td>sidecar</td></tr>
<tr><td>Installdef Parameters</td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>from</td><td>String</td><td>The basepath of the file to be installed. When adding the file to your module loadable package, its from path must have a path in the format containing clients/<client>/<type>/<subtype>/<file>.php. This determines the path the file will be installed to.</td></tr>
<tr><td>to_module</td><td>String</td><td>The key of the module the file is to be installed to.</td></tr>
</table>
</td></tr>
</table>

Extension Directory	./custom/Extension/modules/<module>/Ext/clients/<client>/<type>/<subtype>/<file>.php
Extension File	./custom/modules/<module>/Ext/clients/<client>/<type>/<subtype>/<subtype>.ext.php

Examples

Filesystem

When working directly with the filesystem, you can create a file in ./custom/Extension/modules//Ext/clients/<client>/<type>/<subtype>/ to append your metadata extensions. The example below will demonstrate 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>',
  ),
);
```

Navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions.

Module Loadable Package

When building a module loadable package, you can use the $installdefs['sidecar'] index to install the metadata file. The example below will demonstrate how to add a new subpanel to a specific module using a module loadable package:

./manifest.php

```
<?php
```

```
$manifest = array (
  'built_in_version' => '<built_in_version>',
  'acceptable_sugar_versions' => array (),
  'acceptable_sugar_flavors' => array (),
  'readme' => '',
  'key' => 'example',
  'author' => '',
  'description' => 'A module loadable package',
  'icon' => '',
  'is_uninstallable' => true,
  'name' => 'Module Loadable Package',
  'published_date' => '2014-05-28 21:34:08',
  'type' => 'module',
  'version' => 1401312849,
  'remove_tables' => 'prompt',
);

$installdefs = array (
  'id' => 'example',
  'sidecar' =>
  array (
    0 =>
    array (
      'from' =>
'<basepath>/Files/clients/base/layouts/subpanels/<file>.php',
      'to_module' => '<module>',
    ),
  ),
);
```

./Files/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>',
  ),
);
```

Last Modified: 09/26/2015 04:14pm

UserPage

Overview

Adds sections to the User Management DetailView.

Properties

Internal Name	userpage
Manifest Installdef	user_page
Extension Directory	./custom/Extension/modules/Users/Ext/UserPage/
Extension File	./custom/modules/Users/Ext/UserPage/userpage.ext.php

Example

In this example, a custom table will be added to the users DetailView:

./custom/Extension/modules/Users/Ext/UserPage/<name>.php

```
<?php

$HTML=<<<HTML

    <table cellpadding="0" cellspacing="0" width="100%" border="0"
class="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 the table will now be available under Admin > User Management > {User}.

Last Modified: 09/26/2015 04:14pm

Utils

Overview

Adds additional functions to the global utility function list.

Properties

Internal Name	utils
Manifest Installdef	utils
Extension Directory	./custom/Extension/application/Ext/Utils/
Extension File	./custom/application/Ext/Utils/custom_utils.ext.php

Alternatively, additional 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.

Example

The example below will add a new function to the global utils:

./custom/Extension/application/Ext/Utils/<file>.php

```
<?php

function utilFunction()    '
```

```
{
    //logic
}
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions.

Last Modified: 09/26/2015 04:14pm

Vardefs

Overview

Adds or overrides vardefs in the system.

Properties

Internal Name	vardefs
Manifest Installdef	vardefs
Extension Directory	./custom/Extension/modules/<module>/Ext/Vardefs/
Extension File	./custom/modules/<module>/Ext/Vardefs/vardefs.ext.php

More information on Vardefs can be found in Module Framework under the Vardefs section.

Examples

Overriding an existing Vardef

The most common use of the vardef extensions from a customization standpoint is to override the attributes of an existing vardef. To do this, you should avoid trying to redefine the entire vardef and only update the specific index you want to change.

An example vardef is shown below. Your vardef override should not look like this unless you are creating a new vardef:

```
$dictionary['<module>']['fields']['name'] => array (
    'name' => 'name',
    'vname' => 'LBL_NAME',
    'dbType' => 'varchar',
    'type' => 'name',
    'len' => '50',
    'comment' => 'Example Vardef',
    'required' => false,
    'unified_search' => true,
    'full_text_search' => array('boost' => 3),
);
```

To update this vardef to be required, you should override it by doing the following:

./custom/Extension/modules/<module>/Ext/Vardefs/<file>.php

```
$dictionary['<module>']['fields']['name']['required'] = false;
```

Navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions.

Creating Custom Fields

If your goal is to manually create a custom field on an instance, you should be using the ModuleInstaller 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 Creating an Installable Package that Creates New Fields section. An example of programmatically creating a field can be found in Module Vardefs under the Manually Creating Custom Fields section.

Last Modified: 01/15/2016 09:10pm

WirelessLayoutdefs

Overview

Adds additional subpanels to the wireless views. This is only applicable to modules running in backward compatibility mode.

Properties

Internal Name	wireless_subpanels
Manifest Installdef	wireless_subpanels
Extension Directory	./custom/Extension/modules/<module>/Ext/WirelessLayoutdefs/
Extension File	./custom/modules/<module>/Ext/WirelessLayoutdefs/wireless.subpaneldefs.ext.php

Example

The following example will add a new subpanel to a select module':

./custom/Extension/modules/<module>/Ext/WirelessLayoutdefs/<file>.php

```
<?php

$layout_defs['<module>']['subpanel_setup']['<subpanel module>'] =
array(
    '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.

Last Modified: 09/26/2015 04:14pm

WirelessModuleRegistry

Overview

Adds additional modules to the available modules for mobile.

Properties

Internal Name	wireless_modules
Manifest Installdef	wireless_modules
Extension Directory	./custom/Extension/application/Ext/WirelessModuleRegistry/
Extension File	./custom/application/Ext/WirelessModuleRegistry/wireless_module_registry.ext.php

Example

The example below 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 the module will now be available on the mobile layout.

Last Modified: 09/26/2015 04:14pm

Logic Hooks

Logic Hooks

The Logic Hook framework allows you to add actions to specific events that occur within the system.

Hook Definitions

The logic hook actions and trigger events are defined in the hook definition.

Definition Locations

Custom logic hook definitions can be located in the following paths:

Module Specific Paths	
./custom/Extension/modules/<module>/Ext/LogicHooks/<file>.php	Defines hook actions that will be executed for the specified events on a specific module using the extension framework. This path should be used when developing module specific hooks. More information can be found in the LogicHooks extension section.
./custom/modules/<module>/logic_hooks.php	Defines hook actions that will be executed for the specified events on a specific module. You should avoid using this path as it may be overwritten by module loadable plugins.
Application Paths	
./custom/Extension/application/Ext/LogicHooks/<file>.php	Defines hook actions that will be executed for the specified events on all modules using the extension framework. This path should be used when developing application hooks. More information can be found in the LogicHooks extension section.
./custom/modules/logic_hooks.php	Defines hook actions that will be executed for the specified events on all modules. You should avoid using this path as it may be overwritten by module loadable plugins.

Definition Properties

The logic hooks must have $hook_version and $hook_array defined in its definition before it can be used by the system.

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 of the action to execute. The hook array will be defined as follows:

Name	Type	Description
event_name	String	The name of the event that append the action to.
process_index	Integer	The order to execute the actions in.
description	String	A short description to identify the hook action.
file_path	String	The path to the logic hook file. The file should reside within the ./custom/ directory.
class_name	String	The name of the logic hook action class.
method_name	String	The name of the logic hook action method.

```
$hook_array['<event_name>'][] = array(
    <process_index>, //Integer
    '<description>', //String
    '<file_path>', //String
    '<class_name>', //String
    '<method_name>', //String
);
```

Example Definition

```
<?php

    $hook_version = 1;

    $hook_array = array();
    $hook_array['before_save'] = array();
    $hook_array['before_save'][] = array(
        1,
        'Hook description',
        'custom/modules/Accounts/customLogicHook.php',
        'className',
```

```
        'methodName'
    );
?>
```

Hook Action Method

The hook action class can be located anywhere you choose. For best practices, it is recommended to group the hooks with the module they are associated with within the ./custom/ directory. An example of a method class for the definition above is shownbelow:

```
<?php
    class className
    {
        function methodName($bean, $event, $arguments)
        {
            //logic
        }
    }
?>
```

The logic hook method signature may contain different arguments depending on the hook event.

Considerations

A few cautions around using logic hooks:

- As of PHP 5.3, objects are automatically passed by reference. When creating your logic hook signatures, it is important that you 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. You will need to use either the 'process_record' or 'after_retrieve' logic hooks.
- In order to compare new values with previous values to determine whether a change has happened, you can use fetched_row to check. An Example is below:

```
if ($bean->fetched_row['{field}'] != $bean->{field})
{
    //logic
}
```

- Make sure that the permissions on your logic_hooks.php file and the class file that it references are readable by the web server. If this is not done, 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.

Last Modified: 01/15/2016 09:10pm

Application Hooks

Logic Hooks that can be used to execute logic when working with the global application.

Last Modified: 09/26/2015 04:14pm

after_entry_point

Overview

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. This

Name	Type	Description
		parameter is normally empty.

Considerations

- This 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.
- 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
        }
    }

?>
```

Last Modified: 01/15/2016 09:10pm

after_load_user

Overview

Executes after the current user is set for the current request. Used to handle actions that are dependent on the current current user such as setting ACLs or configuring user-dependent parameters.

Definition

```
function after_load_user($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. This parameter is normally 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',

        //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_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($bean, $event, $arguments)
        {
            //logic
        }
    }

?>
```

Last Modified: 01/15/2016 09:15pm

after_session_start

Overview

Executes after the after_load_user logic hook has executed and the users visibility rules have been setup. The is also executed before the users session is started.

Definition

```
function after_session_start($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object.

Name	Type	Description
event	String	The current event.
arguments	Array	Additional information related to the event. This parameter is normally 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($bean, $event, $arguments)
        {
            //logic
        }
    }

?>
```

Last Modified: 01/15/2016 09:15pm

after_ui_footer

Overview

Executes after the footer has been invoked for modules in backward compatibility mode.

Definition

```
function after_ui_footer($event, $arguments){}
```

Arguments

Name	Type	Description
event	String	The current event.
arguments	Array	Additional information related to the event. This parameter is normally

Name	Type	Description
		empty.

Considerations

- This 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 are intending to write display logic to the screen and 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 to avoid issues. An example of this is shown below:

```
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 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"]
        }
    }

?>
```

Last Modified: 01/15/2016 09:10pm

after_ui_frame

Overview

Executes after the frame has been invoked and before the footer has been invoked for modules in backward compatibility mode.

Definition

```
function after_ui_frame($event, $arguments){}
```

Arguments

Name	Type	Description
event	String	The current event.
arguments	Array	Additional information related to the event. This parameter is normally 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
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
        }
    }

?>
```

Last Modified: 01/15/2016 09:10pm

handle_exception

Overview

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()
        }
    }

?>
```

Last Modified: 11/28/2016 11:43am

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. This parameter is normally 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"]
        }
    }

?>
```

Last Modified: 01/15/2016 09:10pm

Module Hooks

Logic Hooks that can be used to execute logic when working with modules.

Last Modified: 11/18/2015 01:08am

after_delete

Overview

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
Point');

   class after_delete_class
   {
      function after_delete_method($bean, $event, $arguments)
      {
         //logic
      }
   }
```

?>

Last Modified: 01/15/2016 09:20pm

after_relationship_add

Overview

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,
$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_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
Point');

   class after_relationship_add_class
   {
      function after_relationship_add_method($bean, $event,
$arguments)
      {
         //logic
      }
   }

?>
```

Last Modified: 01/15/2016 09:25pm

after_relationship_delete

Overview

Executes after a relationship has been deleted between two records.

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.

Name	Type	Description
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 delete 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_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,
$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_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
Point');

   class after_relationship_delete_class
   {
      function after_relationship_delete_method($bean, $event,
$arguments)
      {
         //logic
      }
   }

?>
```

Last Modified: 01/15/2016 09:25pm

after_restore

Overview

Executes after a record is undeleted.

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. This parameter is normally 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
Point');

   class after_restore_class
   {
      function after_restore_method($bean, $event, $arguments)
      {
         //logic
      }
   }

?>
```

Last Modified: 01/15/2016 09:20pm

after_retrieve

Overview

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.

Name	Type	Description
arguments.encode	Boolean	Whether or not to encode.

Considerations

- This 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
Point');

   class after_retrieve_class
   {
      function after_retrieve_method($bean, $event, $arguments)
      {
         //logic
      }
   }

?>
```

Last Modified: 01/15/2016 09:15pm

after_save

Overview

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. This parameter is normally empty.
arguments.isUpdate	Boolean	Whether or not the record is newly created or not. True is an existing record

Name	Type	Description
		being saved. False is a new record being created.
arguments.dataChanges	Array	A list of the changes in the auditable fields on the bean.

Considerations

- Calling save on the bean in this hook will cause an infinite loop if not handled correctly. (i.e: $bean->save())

Change Log

Version	Note
7.0.0RC1	Added dataChanges to the $arguments parameter.
7.0.0RC1	Added isUpdate to the $arguments parameter.
4.5.0c	Added after_save hook.

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
Point');

   class after_save_class
   {
      function after_save_method($bean, $event, $arguments)
      {
         //logic
      }
   }

?>
```

Last Modified: 01/15/2016 09:15pm

before_delete

Overview

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
Point');

   class before_delete_class
   {
      function before_delete_method($bean, $event, $arguments)
      {
         //logic
      }
   }

?>
```

Last Modified: 01/15/2016 09:15pm

before_relationship_add

Overview

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.

Name	Type	Description
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
6.4.5	Added before_relationship_add hook.

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
        }
    }

?>
```

Last Modified: 01/15/2016 09:15pm

before_relationship_delete

Overview

Executes before a relationship has been deleted between two records.

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

Name	Type	Description
		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
6.4.5	Added before_relationship_delete hook.

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,
$arguments)
        {
            //logic
        }
    }

?>
```

Last Modified: 03/28/2016 04:13pm

before_restore

Overview

Executes before a record is undeleted.

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. This parameter is normally 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
Point');

   class before_restore_class
   {
      function before_restore_method($bean, $event, $arguments)
      {
         //logic
      }
   }

?>
```

Last Modified: 01/15/2016 09:15pm

before_save

Overview

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 or not.

Name	Type	Description
		True is an existing record being saved. False is a new record being created.

Considerations

- With certain modules, like Cases and Bugs, the human-readable ID of the record (like the case_number field in the Case module), is not available within a before_save call. This is because this business logic has not been executed yet.
- Calling save on the bean in this hook will cause an infinite loop if not handled correctly. (i.e: $bean->save())

Change Log

Version	Note
7.0.0RC1	Added dataChanges to the $arguments parameter.

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.
      'logic_hooks_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
      }
   }

?>
```

Last Modified: 01/15/2016 09:15pm

process_record

Overview

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. This parameter is normally 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
Point');

   class process_record_class
   {
      function process_record_method($bean, $event, $arguments)
      {
         //logic
      }
   }

?>
```

Last Modified: 12/05/2016 03:31pm

User Hooks

User specific logic hooks.

Last Modified: 09/26/2015 04:14pm

after_login

Overview

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. This parameter is normally 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
        }
    }

?>
```

Last Modified: 01/15/2016 09:15pm

after_logout

Overview

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. This parameter is normally 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
        }
    }

?>
```

Last Modified: 01/15/2016 09:15pm

before_logout

Overview

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. This parameter is normally 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
        }
    }

?>
```

Last Modified: 01/15/2016 09:15pm

login_failed

Overview

Executes on a failed login attempt.

Definition

```
function login_failed($bean, $event, $arguments){}
```

Arguments

Name	Type	Description

Name	Type	Description
bean	Object	The bean object.
event	String	The current event.
arguments	Array	Additional information related to the event. This parameter is normally 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
        }
    }

?>
```

Last Modified: 01/15/2016 09:20pm

Job Queue Hooks

Logic Hooks that can be used to execute logic when working with the Job Queue module.

Last Modified: 11/18/2015 01:08am

job_failure

Overview

Executes when a jobs 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 related to the event. This parameter is normally 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
        }
    }

?>
```

Last Modified: 01/15/2016 09:20pm

job_failure_retry

Overview

Executes any time a job fails before its final failure. Use job_failure if you only want action on a final failure.

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. This parameter is normally 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
        }
    }

?>
```

Last Modified: 01/15/2016 09:20pm

SNIP Hooks

Logic Hooks that can be used to execute logic when working with SNIP.

Last Modified: 11/18/2015 01:38am

after_email_import

Overview

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. This parameter is normally 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
        }
    }

?>
```

Last Modified: 01/15/2016 09:25pm

before_email_import

Overview

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. This parameter is normally 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
        }
    }

?>
```

Last Modified: 01/15/2016 09:30pm

API Hooks

Logic Hooks that can be used to execute logic when working with the REST API and routing.

Last Modified: 11/18/2015 01:38am

after_routing

Overview

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
        }
    }

?>
```

Last Modified: 01/15/2016 09:30pm

before_api_call

Overview

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.
        '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_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
        }
    }

?>
```

Last Modified: 01/15/2016 09:30pm

before_filter

Overview

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.

Version	Note

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
        }
```

```
    }
?>
```

Last Modified: 01/15/2016 09:25pm

before_respond

Overview

Executes when the v10+ REST Service:

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
        }
    }

?>
```

Last Modified: 01/15/2016 09:25pm

before_routing

Overview

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
        }
    }

?>
```

Last Modified: 01/15/2016 09:25pm

Web Logic Hooks

Overview

How web logic hooks work.

Web Logic Hooks

Web Logic Hooks allow for administrators to post record and event information to a specified URL when certain sugar events take place. The administrative panel for web logic hooks is located at:

```
Admin > Web Logic Hooks
```

When a web logic hook is triggered, a record will be created in the Job Queue . It is important to note that the POST of the information to the external URL will happen when the cron is run and not when the actual event occurs.

Note: You must have the cron setup and running in on order to process the job queue for web logic hooks.

Arguments

Name	Description
URL	The URL to post JSON encoded information.
Module Name	The name of the module your web logic hook will be applicable to.
Trigger Event	The event to 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 can be used: • POST • GET • PUT • DELETE

Example

The file below is an example on 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
)
```

Last Modified: 01/15/2016 09:25pm

Logic Hook Release Notes

Overview

Lists changes to the logic hook libraries.

Release Notes

7.0.0RC1

- after_routing hook was added
- after_save arguments.isUpdate parameter was added
- after_save arguments.dataChanges parameter was added

- before_api_call hook was added
- before_filter hook was added
- before_respond hook was added
- before_routing hook was added
- before_save arguments.isUpdate parameter was added
- web logic hooks were added

6.6.0

- after_load_user hook was added

6.5.0RC1

- after_email_import hook was added
- before_email_import hook was added
- job_failure hook was added
- job_failure_retry hook was added

6.4.5

- before_relationship_add hook was added
- before_relationship_delete hook was added

6.4.4

- handle_exception hook was added

6.4.3

- after_entry_point hook was added

6.0.0RC1

- after_relationship_add hook was added
- after_relationship_delete hook was added

5.0.0a

- after_login hook was added
- after_logout hook was added
- after_ui_footer hook was added
- after_ui_frame hook was added
- before_logout hook was added
- login_failed hook was added
- process_record hook was added
- server_round_trip hook was added

4.5.0c

- after_save hook was added

Last Modified: 11/18/2015 01:08am

Examples

Logic Hook Examples.

Last Modified: 11/18/2015 01:08am

Comparing Bean Properties Between Logic Hooks

Overview

This example will demonstrate 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.

Example

./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
Point');

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
        }
    }
}

?>
```

Last Modified: 09/26/2015 04:14pm

Manipulating Logic Hook References

Overview

How to add and remove logic hook references with code.

Logic Hooks

Adding Logic Hooks

In order to add a logic hook reference to ./custom/modules/<module>/logic_hooks.php, you can use check_logic_hook_file:

```
//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("<module>", "before_save", $my_hook);
```

Removing Logic Hooks

Removing a logic hook reference 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("<module>", "before_save", $my_hook);
```

Last Modified: 09/26/2015 04:14pm

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.

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. An example of a looping hook is shown below.

./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
Point');

class Accounts_Hook
{
    function update_self($bean, $event, $arguments)
    {
        //generic condition
        if ($bean->type == 'Analyst')
        {
            //update
            $bean->industry = 'Banking';
            $bean->save();
        }
    }
}
```

In the example above, we have a generic condition that when met, will trigger the update of a field on the record. This will cause an infinite loop as calling the save() method will once again trigger the before_save hook and then the after_save hook again. The solution to this problem is to add a 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. An example of this is shown below:

./custom/modules/Accounts/Accounts_Hook.php

```
<?php

if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

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->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
Point');

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
Point');

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 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
Point');

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_update_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
Point');

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_update_c === false)
                {
                    //set the ignore update attribute
                    $parentAccount->ignore_update_c = true;

                    //perform any other account logic
                    $parentAccount->save();
                }
            }
        }
    }
}
```

Last Modified: 11/28/2016 11:51am

API

An overview of webservices, developer classes, and application concepts.

Last Modified: 09/26/2015 04:14pm

Application

The follow sections outline how to develop using the applications framework.

Last Modified: 09/26/2015 04:14pm

Administration Links

Overview

Managing administration links.

Administration Links

Administration links are the links located in the admin section of Sugar. As of 6.3.x, administration links can be created using the extension framework. The global links extension directory is located at ./ custom/Extension/modules/Administration/Ext/Administration/. PHP files found is this directory will be compiled into ./custom/modules/Administration/Ext/Administration/administration.ext.php after a Quick Repair and Rebuild.Additional information on this can be found in the extensions Administration section. 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 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/images
        'Administration',

        //Link name label
        'LBL_LINK_NAME',

        //Link description label
        'LBL_LINK_DESCRIPTION',

        //Link URL - For Sidecar modules

'javascript:parent.SUGAR.App.router.navigate("<module>/<path>",
{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'] = '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 in the Admin section.

Last Modified: 09/26/2015 04:14pm

Databases

All five Sugar editions 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 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.

Indexes

Indexes can be defined in the main or custom vardefs.php for 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.

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 create_guid() utility function for creating these GUIDs in the following format: aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee. The primary key 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).

Sugar allows a primary key to contain any unique string. This can be a different GUID algorithm, a key that has some meaning (such as bean type first, followed by info), an external key, and/or auto-incrementing numbers (converted to strings). Sugar chose GUIDs over auto-incrementing keys to allow for easier data synchronization across databases and avoid primary key collisions when one of the following occurs:

- Sugar Offline Client (part of Sugar Enterprise) syncs data with the main Sugar installation.
- Sugar SOAP APIs are used for data synchronization.
- Tools like Talend are used for data synchronization.

Offline Client uses GUIDs for primary keys for ease of implementation and simpler handling of data conflicts compared to other schemes. If a developer changes Sugar to use some other ID scheme and needs to accommodate data synchronization across data stores, IDs need to be partitioned ahead of time or a system similar to the Sugar implementation for Cases, Quotes, and Bugs created. For modules like these that have human-readable ID numbers (integers) that need to be synchronized across databases, Sugar implements a server ID that is globally unique and concatenates it with an incrementing Case, Quotes or Bug number. Attempting such a change to Sugar requires some careful planning and implementation.

If data synchronization is not an issue, the primary key format can be changed to some other unique string.

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 need to be stored as globally unique strings with no more than 36 characters. If multiple modules contain records with matching ids, you may experience undesired behaviors within the system.

To perform any of the following:

- Implement a new primary key method
- Import existing data with a different primary key format based on the existing GUID mechanism for new records

Make note of the following:

- 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 : IDs 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 need to run an algorithm on the data to make sure all of the IDs are unique. One simple algorithm is to MD5 the ids 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, as a result of case sensitivity.
- 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 some of the more recent versions. To protect against this, you will want to make sure the GUIDs are not padded with blanks in the DB.

Last Modified: 09/26/2015 04:14pm

Entry Points

Entry Points Overview.

Last Modified: 11/18/2015 01:08am

Introduction

Overview

Introduction to entry points

Introduction to Entry Points

Entry points into the Sugar application are used to ensure that proper security and authentication steps are applied consistently across the entire application. The current entry points to note are:

- ./cron.php - Used by the Windows Scheduler Service or the cron service on Linux and Unix for executing the Sugar Scheduler periodically
- ./index.php - Default entry point into the Sugar application
- ./install.php - Used for initial install of the Sugar application
- ./soap.php - Entry point for all v1 SOAP calls. This entry point is deprecated and should not be used for API development
- ./service/{api version}/soap.php - Entry point for v2 - v4_1 SOAP calls
- ./service/{api version}/rest.php - Entry point for for v2 - v4_1 REST calls

Additional stock entry points can be found in ./include/MVC/Controller/entry_point_registry.php.

Accessing Custom Entry Points

Available custom entry points can be accessed using a URL pattern as follows:

```
http://{sugar url}/#bwc/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. More information on creating custom entry points can be found here.

Last Modified: 01/25/2016 09:45am

Custom Entry Points

Overview

How to create custom entry points.

Creating a Custom Entry Point

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
```

Last Modified: 03/29/2016 05:15pm

File Caching

Overview

An Overview of how caching works within SugarCRM

What It Does

Much of the user interface is built dynamically using templates, metadata and language files. Sugar implements a file caching mechanism to improve 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.

Where is the Cache?

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 your 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.

```
Admin > System Settings > Advanced > Developer Mode
```

This is especially helpful when you are 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.

Last Modified: 09/26/2015 04:14pm

File 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, the file will be stored in this directory with a GUID name.

You should note that when uploading files to Sugar, there are three file size limits to configure. The first two limits are your PHP upload_max_filesize and post_max_size which are configured in your php.ini. The second limit is the sugar

configuration setting for upload_maxsize , which will restrict the upload limit from within Sugar. This setting can also be modified in the application via Admin > System Settings.The smallest of these three values will be honored when an oversized file has been uploaded.

You should note that the PHP-defined file size settings and the upload directory cannot be modified for instances hosted on On-Demand.

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 On-Demand.

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 emails content.

The SQL to fetch information about an emails attachment is shown below:

```
SELECT filename, file_mime_type FROM notes INNER JOIN emails ON
notes.parent_type = 'Emails' AND notes.parent_id = emails.id INNER
JOIN emails_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;
```

Document Attachments

Files uploaded to Sugar as documents will be moved to the upload directory with a GUID name matching the document revision id. The document revision id can be found in the document_revision_id field on the documents table. This field maps to the id field on the document_revisions table. The attributes of the file, such as filename and file_mime_type, are stored in the document revision record.

The SQL to fetch information about a documents attachment is shown below:

```
SELECT filename, file_mime_type, file_ext FROM document_revisions
INNER JOIN documents ON documents.id = document_revisions.document_id;
```

Knowledge Base Attachments

When working with the Knowledge Base, files and images attached to the form before the record is saved are uploaded and stored in the upload directory with the name <GUID><File Name> using the KbdocAttachments action found in ./modules/KBDocuments/KbdocAttachments.php. Once the record is saved, these files are copied and saved in the upload directory with a GUID name matching that of the document revision id.

The SQL to fetch information about a knowledge base attachment is shown below:

```
SELECT document_revisions.filename, document_revisions.file_mime_type,
document_revisions.file_ext FROM document_revisions INNER JOIN
kbdocument_revisions ON document_revisions.id =
kbdocument_revisions.document_revision_id INNER JOIN kbdocuments ON
kbdocument_revisions.kbdocument_id = kbdocuments.id;
```

Module Loadable Packages

Module Loader packages are stored in the system differently than other uploads. They are uploaded to the ./upload/upgrades/module/ directory with their original file name. 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 show 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 users 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.

Last Modified: 01/15/2016 09:20pm

Job Queue

Job Queue Overview.

Last Modified: 11/18/2015 01:08am

Introduction

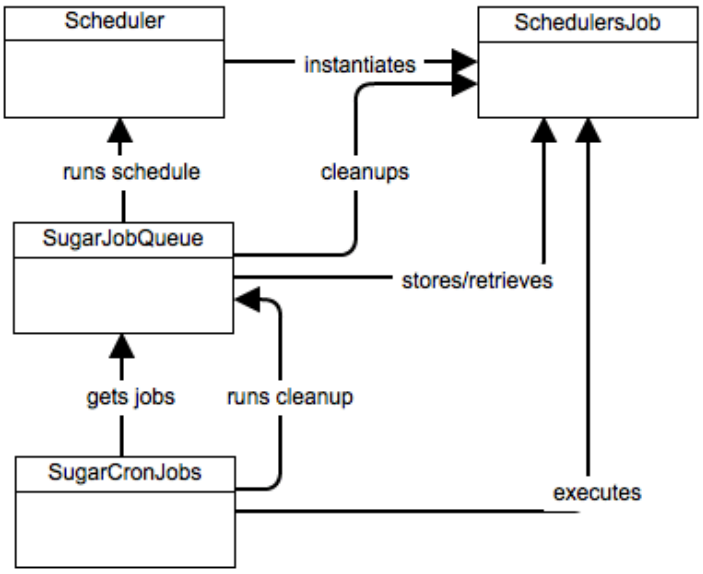
Overview

The Job Queue handles the executing of automated tasks within Sugar.

Components

The Job Queue consists of the followng 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 occuring job.
- SugarCronJobs - The cron process that uses SugarJobQueue to run jobs. It is run periodically and does not support parallel execution.



Stages

Schedule Stage

On the scheduling stage (checkPendingJobs in Scheduler class) we check if we have any schedules that are qualified to run at this time and do not have job instance already in the queue. If such schedules exist, each get created a job instance to run immediately.

Execution Stage

The SQL queue table is checked for any jobs in pending status. Any jobs found 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 running state longer than the defined timeout. Such jobs are failed (they may be requeued if their definition includes requeing on failure).

Last Modified: 01/08/2017 04:38pm

Schedulers

Schedulers overview.

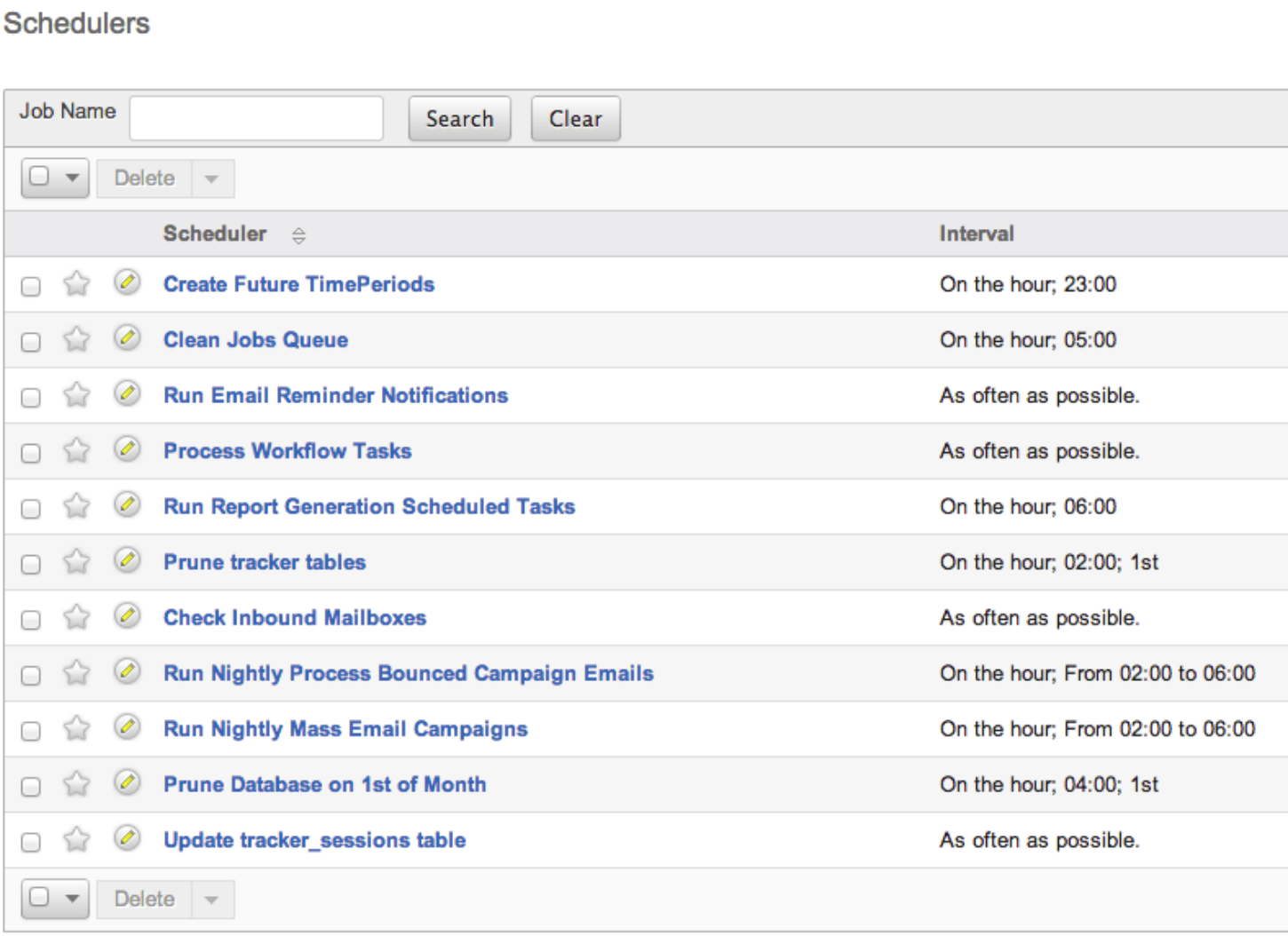
Last Modified: 11/18/2015 01:08am

Introduction

Scheduler

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 such as Process Workflow Tasks and Run Report Generation Scheduled Tasks.



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

- When defining schedulers, you should note that the scheduler will be run

after the specified time and that this execution may not be exact.

- If you are seeing the message "Job runs too frequently, throttled to protect the system." in your sugar log, the cron is being run too frequently. If you would prefer the cron to run more frequently, you can set the cron.min_cron_interval setting to 0 and disable throttling completely.

Last Modified: 09/26/2015 04:14pm

Custom Schedulers

Overview

How to create a custom scheduler for Sugar 6.3.0+

Scheduler Example

Defining the Job Label

The first step to create a custom scheduler is to create a label extension file. This will add the display text for our 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, the file will be nameden_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, we will define our custom jobs function using the extension framework. The file path of our file will be in the format of ./custom/Extension/modules/Schedulers/Ext/ScheduledTasks/<function_name>.php. For our example, the file will be named 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, we will need to 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.

Last Modified: 09/26/2015 04:14pm

Scheduler Jobs

Scheduler Jobs Overview.

Last Modified: 11/17/2015 11:07pm

Introduction

Scheduler Jobs

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

Last Modified: 09/26/2015 04:14pm

Logic Hooks

Overview

The scheduler jobs module has two additional logic hooks that can be used to monitor jobs.

Hooks

The additional hooks that can be used are.

- job_failure_retry : Executed when a job fails but will be retried
- job_failure : Executed when the fob fails for the final time

You can find more information on these hooks in the Job Queue Hooks section.

Last Modified: 11/18/2015 02:08am

Examples

Scheduler Job Examples.

Last Modified: 11/18/2015 12:07am

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

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
```

Last Modified: 09/26/2015 04:14pm

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');

    class Accounts_Save
```

```
    {
        function QueueJob(&$bean, $event, $arguments)
        {
            require_once('include/SugarQueue/SugarJobQueue.php');

            //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 ./c ustom/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.

Last Modified: 09/26/2015 04:14pm

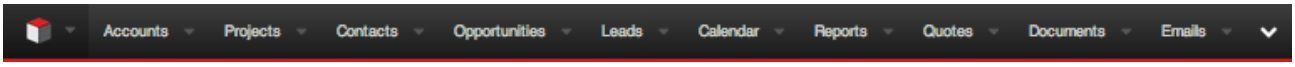
MegaMenu

Overview

Managing the MegaMenu links.

MegaMenu

The MegaMenu is the header navigation menu used to navigate the Sugar Application.



Link Properties

All links in the system will have the following properties that define their navigation, display, and visibility:

Name	Description
acl_action	The ACL action. This is used to verify the user has access to a specific action required for the link.
acl_module	The ACL module. This 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 can be found in the style guide located in Admin > Styleguide > Core Elements > Base CSS > Icons.
label	The label key that contains your links display text.
openwindow	Whether or not to open the link in a new window. Setting this to true will launch the link in a new window.
route	The route to direction the user. For sidecar modules this is "#<module>", however, modules in backward compatibility mode are routed as "#bwc/index.php?module=<module>". External links will need to have the full URL as well as openwindow set to true.
submenu	An array of sub-navigation links. The sub-navigation links will contain the same basic link properties.

Module Links

Module links are the top level links for each available module. When a top level link is clicked, the user is navigated to the selected modules "records" layout. These top level links are also expandable elements that contain sub-navigation links which are outlined in the following section.

Module Action Links

The modules action links are displayed to the user once a user clicks the down arrow of a module link. Module action extension files will be located in ./custom/Extension/modules/<module>/Ext/clients/base/menus/header/ and compiled into ./custom/modules/<module>/Ext/clients/base/menus/header/header.ext.php. By default, when you expand the sub-navigation links, you will see an element appear with the following:

- Create <module>
- List <module>
- View <module> Reports
- Import <module>

Adding Module Action Links

The example below will demonstrate how to add a module action link that points to the Leads module. To define your own module action link, you will need to create your own label extension for the links display label:

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

```
<?php

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

Next, you will need to create your 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 your extension files, you will need to 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, you will need to 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
=> $moduleAction)
    {
        //remove the link by label key
        if ($moduleAction['label'] == "<link label key>")
        {

unset($viewdefs['<module>']['base']['menu']['header'][$key]);
            //bug 74403 workaround
            $viewdefs['<module>']['base']['menu']['header'] =
array_values($viewdefs['<module>']['base']['menu']['header']);
        }
    }
}
```

Once you have created your extension files, you will need to navigate to Admin > Repair > Quick Repair and Rebuild. This will remove your menu action item to 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 users profile menu on the right side of the MegaMenu. Profile action extension files will be located in ./custom/Extension/application/Ext/clients/base/views/profileactions/ and compiled into ./custom/application/Ext/clients/base/views/profileactions/profileactions.ext.php.

Adding Profile Action Links

The example below will demonstrate how to add a profile action link to the Styleguide. To define your own profile action link, you will first need to create your own label extension for the links display label.

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

```
<?php

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

Next, you will need to create your 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 your extension files, you will need to 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, you will need to 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 =>
$profileAction)
    {
        //remove the link by label key
        if ($profileAction['label'] == "LNK_ABOUT")
        {
            unset($viewdefs['base']['view']['profileactions'][$key]);
            //bug 74403 workaround
            $viewdefs['base']['view']['profileactions'] =
array_values($viewdefs['base']['view']['profileactions']);
        }
    }
}
```

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

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

Last Modified: 12/29/2015 04:28pm

Module Builder

Module Builder Overview.

Last Modified: 11/18/2015 01:08am

Introduction

Overview

Overview of the Module Builder

Module Builder

The Module Builder functionality 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. In 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 applies 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 user names 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 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. 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.

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.

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.

Last Modified: 09/26/2015 04:14pm

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 in a Production Environment

Redeploying a package will remove all customizations related to your module in:

- ./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. Once the module is deployed, you should use Studio to perform any additional configurations to your module including but not limited to:

- adding a new field
- updating the properties of a field 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 environent. If you ever want to uninstall the module at a later date, you can do so under Admin > Module Loader.

Last Modified: 11/19/2015 10:08pm

Module Loader

Module Loader Overview.

Last Modified: 11/18/2015 01:08am

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 On-Demand have package scanner enabled by default. This setting is not configurable and all packages must pass the package scan for installation to the On-Demand environment.

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 editions of Sugar. Anyone who is hosting Sugar products can advantage of this feature, as well.

The specific Module Loader restrictions for the Sugar Open Cloud are documented in the Sugar Knowledge Base.

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 the On-Demand 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 On-Demand.

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

Blacklisted 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 blacklisted by default:
  - lua
  - pclzip
  - reflection
  - reflectionclass
  - reflectionexception
  - reflectionextension
  - reflectionfunction
  - reflectionfunctionabstract
  - reflectionmethod
  - reflectionobject
  - reflectionparameter
  - reflectionproperty
  - reflectionzendextension
  - reflector
  - splfileinfo
  - splfileobject
  - ziparchive

Blacklisted Function Calls

Files must not contain any of the following function calls that are considered suspicious by File Scan.

- Variable functions (i.e., $func(), as in $moduleName(), for example) are prohibited by default.

- Backticks (`) are prohibited by File Scan.
- The following PHP functions are blacklisted 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
- 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
- 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_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 blacklisted by default:
    - All variable functions (i.e., $func(), as in $moduleName(), for example) are prohibited by default.
    - SugarLogger::setLevel
    - SugarAutoLoader::put
    - SugarAutoLoader::unlink

Modifying File Scan

To disable File Scan, add the following configuration setting to config_override.php:

```
$sugar_config['moduleInstaller']['disableFileScan'] = false;
```

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'
);
```

To add additional function calls to the blacklist, 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'
);
```

To override the black list 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, it is necessary for system administrators to 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 any core Sugar source code file must not be overwritten. This is enabled by default when you enable Package Scan.

To disable Restricted Copy, use this configuration setting:

```
$sugar_config['moduleInstaller']['disableRestrictedCopy'] = true;
```

Module Loader Actions

The following module loader actions are enabled by default:

- pre_execute : Called before a package is installed
- install_copy : Copies files or directories
- install_images : Install images into the custom directory

- install_menus : Installs menus to a specific page or the entire Sugar application
- install_userpage : Adds a section to the User page
- install_dashlets : Installs dashlets into the Sugar application
- install_administration : Installs an administration section into the Admin page
- install_connectors : Installs Sugar Cloud Connectors
- install_vardefs : Modifies existing vardefs
- install_layoutdefs : Modifies existing layouts
- install_layoutfields : Adds custom fields
- install_relationships : Adds relationships
- install_languages : Installs language files
- install_logichooks : Installs a new logic hook
- post_execute : Called after 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.

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;
```

Last Modified: 12/07/2016 01:23pm

Module Loader Restriction Alternatives

Overview

This article provides workarounds for commonly used functions that are blacklisted by Sugar for the On-Demand environment.

Blacklisted 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 OnDemand use 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);
```

Last Modified: 11/17/2016 09:24am

SugarExchange Package Guidelines

Version 1.0

The Sugar platform is open and very flexible. Any system can be integrated with Sugar or any feature customized using a Sugar Module Loadable Package. This flexibility necessitates the establishment of guidelines so that customers of Sugar Exchange can expect that all Packages are of a consistently high quality and do not interfere with existing customizations nor prevent easy upgrades. These guidelines document some minimum best practices for the development of Sugar Packages, as well as provide guidelines for security, user interface, encapsulation, as well as performance. Taken as a whole, this guide provides the reader with minimum standards we expect from any Sugar Developer writing code for the Sugar platform. Compliance with these guidelines is a requirement for any package installed into Sugar OnDemand. Failure to follow these guidelines is grounds for having your package removed from Sugar OnDemand and de-listed from Sugar Exchange.

SugarCRM reserves the right to change these guidelines at any time. Questions or

feedback on these guidelines can be sent to developers@sugarcrm.com.

Best Practices

These best practice guidelines are formed from the years of collective experience in working with Sugar Packages that get used by Sugar customers every single 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 be run without dependency on a Sugar server.

We recommend running unit tests frequently during the development process. We suggest developers run tests prior to 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 to not 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 own 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. This becomes a powerful integration mechanism that doesn't require writing a line of Sugar code. Using Custom Modules will avoid 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 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 securely handled.

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 allows information to be stored in an encrypted form. These settings could then be changeable by the Administrator via the Administration panel or via some other end user input.

User Interface Guidelines

Sugar Packages contribute 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 within the Sugar application. It can be found on the Sugar Administration panel under the Styleguide link. There you can find all the information you need to leverage Sugar 7 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 used appropriately 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 behavior of core Sugar application files as a last resort.

Avoid customizations to core Sugar application files if at all possible

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 in order to get in front of potential conflicts with other packages and upgrades.

Use Package Scanner to ensure your Sugar Package is ready for Sugar OnDemand

Sugar Packages should be designed with Sugar OnDemand 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 OnDemand must pass the Package Scanner utility and must not adversely impact the SugarCRM infrastructure. This stipulation is outlined in the OnDemand MSA agreement under Exhibit A in sections 3 and 5. 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 blacklisted functions.

Performance Guidelines

Sugar is primarily a database application however with the introduction of Sugar 7, more and more business logic is executed within the user's web browser. It is best to avoid pre-optimization and use tools to properly identify the root causes of performance issues that users could encounter.

Sugar 7 has instrumentation built into it for New Relic (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 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 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 OnDemand because direct access to the underlying SQL database in order 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 sort of UI feedback within half a second. If you have transactions that will take longer than that it is best to use the

Introduction to the Manifest

Overview

Module loadable packages rely on a manifest.php file to install changes to an instance.

Manifest Definitions

Inside on the manifest.php file, there is a $manifest variable that defines the basic properties of the module loadable package. The properties of the manifest can be found in the table below:

Name	Type	Description

key

String|Integer

name

String

description

String

built_in_version

String

version

String|Integer

acceptable_sugar_versions

Array

acceptable_sugar_flavors

Array

author

String

readme

String

icon

String

is_uninstallable

Boolean

published_date

String

remove_tables

String

type

String

dependencies

Example

An example of a common manifest is shown below:

```
$manifest = array(
  'key' => 1397052912,
  'name' => 'Example Manifest',
  'description' => 'Example Description',
  'author' => 'SugarCRM',
  'version' => '1.0',
  'is_uninstallable' => true,
  'published_date' => '2014-04-09 14:15:12',
  'type' => 'module',
  'acceptable_sugar_versions' =>
  array(
    'exact_matches' => array(
        '7.2.0',
        '7.2.1',
        '7.6.0.0'
```

```
    ),
    //or
    'regex_matches' => array(
        '7\\.2\\.[0-1]$', //7.2.0 - 7.2.1
        '7\\.6\\.(.*?)\\.(.*?)' //any 7.6 release
    ),
  ),
  'acceptable_sugar_flavors' =>
  array(
    'PRO',
    'CORP',
    'ENT',
    'ULT'
  ),
  'readme' => '',
  'icon' => '',
  'remove_tables' => '',
);
```

Installdef Definitions

The following section outlines the parameters specified in the $installdefs array(contained in the manifest file (manifest.php). The $installdefs array(elements are used by the Module Loader to determine the actual installation steps that need to be taken to install the package.

Name	Type	Description

id

String

copy

Array

language

Array

vardefs

Array

layoutdefs

Array

layoutdeffields

Array

beans

Array

image_dir

String

relationships

Array

custom_fields

Array

logic_hooks

Array

pre_execute

Array

post_execute

Array

pre_uninstall

Array

post_uninstall

Array

Note: The following $installdef definitions are deprecated as of 7.0.0:

- connectors
- dashlets
- menu

Last Modified: 01/10/2017 10:16am

Examples

Module Loader Examples.

Last Modified: 11/17/2015 11:07pm

Creating an Installable Package for a Logic Hook

Overview

This is an overview of how to create a module loadable package that will install a logic hook.

This example will install a sample logic hook to the accounts module that will default the account name to 'My New Account Name ({time stamp})'. 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 .

Manifest Example

```
<?php

    $manifest = array(
        'acceptable_sugar_flavors' =>
array('CE','PRO','CORP','ENT','ULT'),
        'acceptable_sugar_versions' => array(
            'exact_matches' => array(),
            'regex_matches' => array('(.*?)\\.(.*?)\\.(.*?)$'),
        ),
        'author' => 'SugarCRM',
        'description' => 'Installs a sample logic hook',
        'icon' => '',
        'is_uninstallable' => true,
        'name' => 'Example Logic Hook Installer',
        'published_date' => '2012-07-06 2012 20:45:04',
        'type' => 'module',
```

```
        'version' => '1341607504',
    );

    $installdefs = array(
        'id' => 'package_1341607504',
        'copy' => array(
            0 => array(
                'from' =>
'<basepath>/Files/custom/modules/Accounts/accounts_save.php',
                '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',
            ),
        ),
    );

?>
```

Logic Hook Example

```
<?php

    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
Point');

    class Accounts_Save
    {
        function updateAccountName($bean, $event, $arguments)
        {
            $bean->name = "My New Account Name (" . time() . ")";
        }
    }

?>
```

Last Modified: 10/19/2015 09:25am

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 of Sugar is hosted in our On-Demand environment or by a third party. For more details on the $manifest or $installdef options, you can visit the Introduction to the Manifest .

Manifest Example

```
<?php

    $manifest = array(
        'acceptable_sugar_flavors' =>
array('CE','PRO','CORP','ENT','ULT'),
        'acceptable_sugar_versions' => array(
            'exact_matches' => array(),
            'regex_matches' => array('(.*?)\\.(.*?)\\.(.*?)$'),
        ),
        'author' => 'SugarCRM',
        'description' => 'Installs my files to the accounts module',
        'icon' => '',
        'is_uninstallable' => true,
        'name' => 'Example File Installer',
        'published_date' => '2012-11-01 2012 20:45:04',
        'type' => 'module',
        'version' => '1391608631',
    );

    $installdefs = array(
        'id' => 'package_1391608631',
        'copy' => array(
            0 => array(
                'from' =>
'<basepath>/Files/custom/modules/Accounts/myFile1.php',
                'to' => 'custom/modules/Accounts/myFile1.php',
            ),
```

```
            1 => array(
                'from' =>
'<basepath>/Files/custom/modules/Accounts/myFile2.php',
                'to' => 'custom/modules/Accounts/myFile2.php',
            ),
            2 => array(
                'from' =>
'<basepath>/Files/custom/modules/Accounts/myFile3.php',
                'to' => 'custom/modules/Accounts/myFile3.php',
            ),
        ),
    );

?>
```

Last Modified: 09/26/2015 04:14pm

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.

Manifest Example

<basepath>/manifest.php

```
<?php

$manifest = array(
    'acceptable_sugar_flavors' => array('CE', 'PRO', 'CORP', 'ENT',
'ULT'),
    'acceptable_sugar_versions' => array(
        'exact_matches' => array(),
```

```
        'regex_matches' => array(
            0 => '6\\.5\\.(.*?)',
            1 => '6\\.7\\.(.*?)',
            2 => '7\\.2\\.(.*?)',
            3 => '7\\.2\\.(.*?)\\.(.*?)',
            4 => '7\\.5\\.(.*?)\\.(.*?)',
            5 => '7\\.6\\.(.*?)\\.(.*?)'
        ),
    'author' => 'SugarCRM',
    'description' => 'Installs a sample set of custom fields to the
accounts module',
    'icon' => '',
    'is_uninstallable' => true,
    'name' => 'Example Custom Field Installer',
    'published_date' => '2015-05-11 20:45:04',
    'type' => 'module',
    'version' => '1391607505',
);

$installdefs = array(
    'id' => 'package_1341607504',
    'language' => array(
        array(
            'from' =>
'<basepath>/Files/Language/Accounts/en_us.lang.php',
            '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';
```

Last Modified: 09/26/2015 04:14pm

Teams

Team Framework Overview.

Last Modified: 11/18/2015 02:08am

Introduction

Overview

Overview of the team framework.

Teams

Teams provides the ability to assign a record to multiple teams that will limit access at the record level. This provides more flexibility in sharing data across functional groups and is only applicable to paid versions of sugar.

Database Tables

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 keeps track of which modules have records that are or were associated to a particular team set.

It is very important that you do not 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 conatins the unique combination of teams associated to the record. The team_id will contain the id of the promary team designated for a record.

Team Sets (team_set_id)

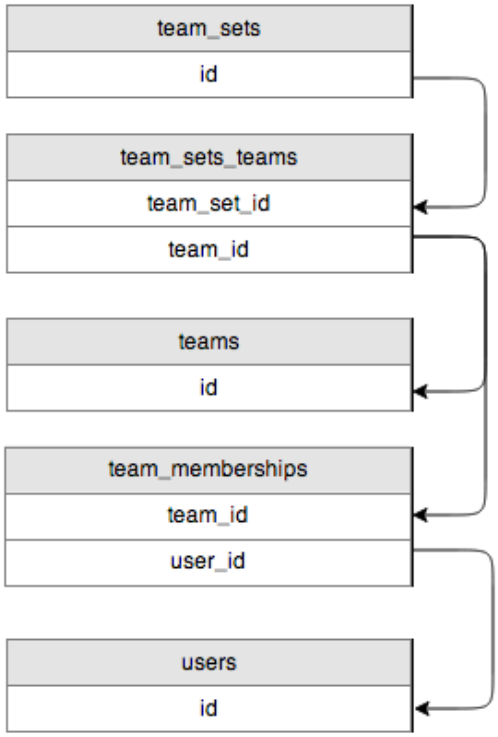
As mentioned above, Sugar implemented this feature not as a many-to-many relationship but as a one-to-many relationship. On each table that had a 'team_id' field we added a 'team_set_id' field. We have also added a 'team_sets' table which maintains the team_set_id, a 'team_sets_teams' table which relates 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, but when displaying the data, we show the team from team id in the list. When the user performs a mouse over 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 contains 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'.

Last Modified: 09/26/2015 04:14pm

Manipulating Teams Programatically

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 teams 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 teams 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 teams 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.

Example:

```
//before save function

function before_save_hook($bean, $event, $arguments)
{
    $bean->team_id = $team_id_1;
}
```

```
//after save function
function after_save_hook($bean, $event, $arguments)
{
    $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
    )
);
```

Last Modified: 09/26/2015 04:14pm

Classes

An overview of how to develop using the various classes.

Last Modified: 09/26/2015 04:14pm

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 Administraton 'saveSetting' function as shown below:

```
require_once('modules/Administration/Administration.php')

$administrationObj = new Administration();

//save the setting
$administrationObj->saveSetting("MyCategory", "MySetting",
'MySettingsValue');
```

Retrieving Settings

You can access the config settings by using the Administration 'retrieveSettings' function. You can filter the settings by category by passing in filter a parameter. If no value is passed to 'retrieveSettings', all settings will be returned. An example is show below:

```
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

When looking to store custom settings, the administration class will store the settings in the 'config' table. Alternatively, you can use the Configurator class located in ./modules/Configurator/Configurator.php to store the settings in the 'config_override.php' file.

Last Modified: 09/26/2015 04:14pm

BeanFactory

Overview

An overview of the BeanFactory class which is used to retrieve bean objects.

The BeanFactory Class

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, you can use the newBean() method. This method is typically used when creating a new record for a module or to call properties of the modules 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('disable_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 an 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);
```

Last Modified: 09/26/2015 04:14pm

Configurator

Configurator Overview.

Last Modified: 09/26/2015 04:14pm

Core Settings

Overview

Sugar configuration settings.

Settings Architecture

When you first install Sugar, all of the default settings are located in ./config.php. As you begin configuring the system, the modified settings are stored in ./config_override.php. Settings in ./config.php are overridden by the values in ./config_override.php.

Settings

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+
Editions	Professional, Corporate, Enterprise, Ultimate
Override Example	`$sugar_config['additional_js_config'] = array();`

additional_js_config.authStorage

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.
Type	String
Range of values	'cache' and 'cookie'
Versions	7.5.0.0+
Editions	Professional, Corporate, Enterprise, Ultimate
Default Value	cache

Override Example	`$sugar_config['additional_js_config']['authStore'] = 'cookie';`

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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	false
Override Example	`$sugar_config['admin_export_only'] = 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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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. Instances running on the On-Demand environment will have this setting enforced as false.
Type	Boolean
Range of values	true and false
Versions	5.5.1+

Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	false
On-Demand Value	false
Override Example	`$sugar_config['allow_sendmail_outbound'] = true;`

api

Description	API specific configurations.
Type	Integer
Versions	7.5.0.0+
Editions	Professional, Corporate, Enterprise, Ultimate
Override Example	`$sugar_config['api']=array();`

api.timeout

Description	The timeout in seconds when uploading files through the REST API.
Type	Integer : Seconds
Versions	7.5.0.0+
Editions	Professional, Corporate, Enterprise, Ultimate
Default Value	180
Override Example	`$sugar_config['api']['timeout'] = 240;`

aws

Description	The Amazon AWS configuration. Used for storing uploads on S3. The

	upload_wrapper_class setting must also be updated to SugarUploadS3 to enable this configuration. You can do this by placing $sugar_config['upload_wrapper_class'] = 'SugarUploadS3'; in your config_override.php.
Type	Array
Versions	6.7.0+
Editions	Professional, Corporate, Enterprise, Ultimate
Override Example	`$sugar_config['aws'] = array();`

aws.aws_key

Description	Amazon AWS public key.
Type	String : Key
Versions	6.7.0+
Editions	Professional, Corporate, Enterprise, Ultimate
Override Example	`$sugar_config['aws']['aws_key'] = 'key';`

aws.aws_secret

Description	Amazon AWS secret key.
Type	String : Key
Versions	6.7.0+
Editions	Professional, Corporate, Enterprise, Ultimate
Override Example	`$sugar_config['aws']['aws_secret'] = 'secret';`

aws.upload_bucket

Description	The Amazon S3 bucket name for uploads.
Type	String : S3 bucket name
Versions	6.7.0+
Editions	Professional, Corporate, Enterprise, Ultimate
Override Example	`$sugar_config['aws']['upload_bucket'] = 'bucket';`

cache_dir

Description	This is the directory SugarCRM will store all cached files. Can be relative to Sugar root directory.
Type	String : Directory
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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 : Seconds
Versions	6.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	300
Override Example	`$sugar_config['cache_expire_timeou`

	`t'] = 400;`

calendar

Description	An array that defines all of the various settings for the Calendar module.
Type	Array
Versions	6.4.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Override Example	`$sugar_config['calendar'] = array();`

calendar.day_timestep

Description	Sets the default day time step.
Type	Integer : Days
Range of values	15, 30 and 60
Versions	6.4.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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+

Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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 : Calendar Step Size
Range of values	15, 30, and 60
Versions	6.4.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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+
Editions	Enterprise, Ultimate
Default Value	10
Override Example	`$sugar_config['check_query_cost'] = 10;`

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+
Editions	Professional, Corporate, Enterprise, Ultimate
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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Override Example	`$sugar_config['cron'] = aray();`

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+
Editions	Professional, Corporate, Enterprise, Ultimate
Default Value	false
Override Example	`$sugar_config['cron']['enforce_runtime'] = 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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate

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 : Seconds
Versions	6.5.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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 : Seconds
Versions	6.5.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	30

Override Example	`$sugar_config['cron']['min_cron_interval'] = 30;`

custom_help_url

Description	Designate the URL used to redirect the user to help documentation.
Type	String : Website address
Versions	6.4.3+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Override Example	`$sugar_config['dbconfig'] = array();`

dbconfig.db_host_instance

Description	Defines the host instance for MSSQL connections.
Type	String

Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Override Example	`$sugar_config['dbconfig']['db_host_instance'] = 'SQLEXPRESS';`

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 the Ent and Ult editions of Sugar.
Type	String : Database Enigine
Range of values	'mysql', 'mssql', 'db2', and 'oracle'
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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 : Database User
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Override Example	`$sugar_config['dbconfig']['db_user_name'] = 'sql_user';`

default_currency_significant_digits

Description	Changes the number of significant digits in currency by default.
Type	Integer : Number of significant digits
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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 : Date format
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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 : Text character
Range of values	Any character
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	.
Override Example	`$sugar_config['default_decimal_sep`

	`erator'] = '.';`

default_email_client

Description	Sets the default email client that opens when users send emails.
Type	String : Email client
Range of values	'sugar', 'external'
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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 : Language key
Range of values	Any available language
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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 : Text character
Range of values	Any character
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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. Instances running on the On-Demand environment will have this setting enforced as 1528.
Type	Integer : Octal Value
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
On-Demand Value	1528
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. Instances running on the On-Demand environment will have this setting enforced as 432.
Type	Integer : Octal value
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
On-Demand Value	432
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. Instances running on the On-Demand environment will have this setting enforced as empty.
Type	String : Web group
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
On-Demand Value	empty
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. Instances running on the On-Demand environment will have this setting enforced as empty
Type	String : Web user
Range of values	Apache user
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
On-Demand Value	empty
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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	false
Override Example	`$sugar_config['default_user_is_admin'] = true;`

developerMode

Description	Rebuilds various cached files when a page is accessed. Can be set by an admin in Admin > System Settings. Instances running on the On-Demand environment will have this setting enforced as false.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	false
On-Demand Value	false
Override Example	`$sugar_config['developerMode'] = true;`

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 : Seconds
Versions	7.6.0.0+
Editions	Professional, Corporate, Enterprise, Ultimate
Default Value	604800
Override Example	`$sugar_config['diagnostic_file_max_lifetime'] = 604800;`

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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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+
Editions	Professional, Corporate, Enterprise, Ultimate
Override Example	`$sugar_config['disable_related_calc_fields'] = true;`

disable_unknown_platforms

Description	Controls whether or not unregistered platforms are allowed to by used when logging in using v10 REST API. Custom platforms can be registered to ./custom/client/platforms.php. Used to prevent excessive metadata generation when invliad platform types are specified in an API call.
Type	Boolean
Range of values	true and false
Versions	7.6.2.0+
Editions	Professional, Corporate, Enterprise, Ultimate
Default Value	false
Override Example	`$sugar_config['disable_unknown_platforms'] = true;`

disable_uw_upload

Description	Disables the upgrade wizard from being accessible through the Sugar admin interface. Instances running on the On-Demand environment will have this setting enforced as true.
Type	Boolean
Range of values	true and false
Versions	5.2.0.j+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	false
On-Demand Value	true
Override Example	`$sugar_config['disable_uw_upload'] = 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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	false
Override Example	`$sugar_config['disable_vcr'] = true;`

dump_slow_queries

Description	Logs slow queries to the sugar log file. Instances running on the On-Demand environment will have this setting enforced as false.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	false
On-Demand Value	false
Override Example	`$sugar_config['dump_slow_queries'] = true;`

email_address_separator

Description	Sets the character used to separate email addresses.
Type	String : Text character
Range of values	Any character
Versions	6.4.3+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	,
Override Example	`$sugar_config['email_address_separator'] = ',';`

email_default_client

Description	Sets the default email client for all users.
Type	String : String
Range of values	sugar, external
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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 : String
Range of values	'plain' and 'html'
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	html
Override Example	`$sugar_config['email_default_editor'] = 'plain';`

enable_inline_reports_edit

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+
Editions	Professional, Corporate, Enterprise,

	Ultimate
Default Value	false
Override Example	`$sugar_config['enable_inline_reports_edit'] = 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+
Editions	Professional, Corporate, Enterprise, Ultimate
Default Value	true
Override Example	`$sugar_config['enable_mobile_redirect'] = false;`

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. Instances running on the On-Demand environment will have this setting enforced as true.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	false

On-Demand Value	true
Override Example	`$sugar_config['external_cache_disabled'] = true;`

external_cache_disabled_apc

Description	Disables APC 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. Instances running on the On-Demand environment will have this setting enforced as true.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	false
On-Demand Value	true
Override Example	`$sugar_config['external_cache_disabled_apc'] = false;`

external_cache_disabled_memcache

Description	Disables Memcache caching in Sugar. Recommended setting is false. This setting is normally normall only enabled to determine if there is a conflict with PHP caching. Instances running on the On-Demand environment will have this setting enforced as true.
Type	Boolean
Range of values	true and false
Versions	5.2.0+

Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	false
On-Demand Value	true
Override Example	`$sugar_config['external_cache_disabled_memcache'] = 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. Instances running on the On-Demand environment will have this setting enforced as true.
Type	Boolean
Range of values	true and false
Versions	6.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	false
On-Demand Value	true
Override Example	`$sugar_config['external_cache_disabled_memcached'] = false;`

external_cache_disabled_mongo

Description	Disables Mongo caching in Sugar. Recommended setting is false. This setting is normally normall only enabled to determine if there is a conflict with PHP caching. Instances running on the On-Demand environment will have this setting enforced as true.

Type	Boolean
Range of values	true and false
Versions	6.0.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	false
On-Demand Value	true
Override Example	`$sugar_config['external_cache_disabled_mongo'] = true;`

external_cache_disabled_smash

Description	Disables Smash caching in 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	5.2.0c+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Override Example	`$sugar_config['external_cache_disabled_smash'] = 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. Instances running on the On-Demand environment will have this setting enforced as true.
Type	Boolean

Range of values	true and false
Versions	6.0.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	false
On-Demand Value	true
Override Example	`$sugar_config['external_cache_disabled_wincache'] = false;`

external_cache_disabled_zend

Description	Disables Zend 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. Instances running on the On-Demand environment will have this setting enforced as true.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	false
On-Demand Value	true
Override Example	`$sugar_config['external_cache_disabled_zend'] = false;`

forms

Description	An array defining form requirements.
Type	Array
Versions	5.2.0+
Editions	Community Edition, Professional,

	Corporate, Enterprise, Ultimate
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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	false
Override Example	`$sugar_config['forms']['requireFirst'] = true;`

freebusy_use_vcal_cache

Description	Forces the use of VCal Cache on FreeBusy requests.
Type	Boolean
Range of values	true and false
Versions	7.6.0.0+
Editions	Professional, Corporate, Enterprise, Ultimate
Default Value	false
Override Example	`$sugar_config['freebusy_use_vcal_cache'] = true;`

hide_admin_backup

Description	Removes the Backups option in the admin menu and also prevents direct access to the feature. Instances running on the On-Demand environment will have this setting enforced as true.
Type	Boolean
Range of values	true and false
Versions	6.5.1+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	false
On-Demand Value	true
Override Example	`$sugar_config['hide_admin_backup'] = true;`

hide_admin_licensing

Description	Hides the License settings subpanel in the administrative panel. Instances running on the On-Demand environment will have this setting enforced as false.
Type	Boolean
Range of values	true and false
Versions	6.5.0+
Editions	Professional, Corporate, Enterprise, Ultimate
Default Value	false
On-Demand 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. Instances running on the On-Demand environment will have this setting enforced as true.
Type	Boolean
Range of values	true and false
Versions	6.5.15+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	false
On-Demand Value	true
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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	false
Override Example	`$sugar_config['hide_subpanels_on_login'] = true;`

history_max_viewed

Description	The number of history items from the tracker to display for a user.
Type	Integer
Versions	5.2.0+
Editions	Professional, Corporate, Enterprise, Ultimate
Default Value	50
Override Example	`$GLOBALS['sugar_config']['history_max_viewed'] = 25;`

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. Instances running on the On-Demand environment will have this setting enforced as true.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	true
On-Demand Value	true
Override Example	`$sugar_config['installer_locked'] = false;`

jobs

Description	Job Queue configurations.
Type	Array
Versions	6.5.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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 : Days
Versions	6.5.1+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	21
Override Example	`$sugar_config['jobs']['hard_lifeti`

	`me'] = 21;`

jobs.max_retries

Description	Maximum number of failures for job. Default is 5.
Type	Integer : Number of failures
Versions	6.5.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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 : Seconds
Versions	6.5.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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 : Days
Versions	6.5.1+
Editions	Community Edition, Professional,

	Corporate, Enterprise, Ultimate
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 : Seconds
Versions	6.5.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	3600
Override Example	`$sugar_config['jobs']['timeout'] = 86400;`

list_max_entries_per_page

Description	Listview items per page.
Type	String : Records per page
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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 : Number of records
Versions	5.2.0+
Editions	Professional, Corporate, Enterprise, Ultimate
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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Override Example	`$sugar_config['logger'] = array();`

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 : Date format
Range of values	Pattern for date format
Versions	5.2.0+
Editions	Community Edition, Professional,

	Corporate, Enterprise, Ultimate
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'. Instances running on the On-Demand environment will have this setting enforced as .log.
Type	String : File extension
Range of values	Extension for the log
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	.log
On-Demand 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 : Number of logs
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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'. Instances running on the On-Demand environment will have this setting enforced as 10MB.
Type	String : Size
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	10MB
On-Demand Value	10MB
Override Example	`$sugar_config['logger']['file']['maxSize'] = '10MB';`

logger.file.name

Description	The name of the log file to be written to. Instances running on the On-Demand environment will have this setting enforced as sugarcrm.
Type	String : Filename
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	sugarcrm

On-Demand 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. Instances running on the On-Demand environment will have this setting enforced as empty.
Type	String : Suffix pattern
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	empty
On-Demand 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'. Instances running on the On-Demand environment will have this setting enforced as fatal.
Type	String : Logging level
Range of values	'debug', 'info', 'warn', 'deprecated', 'error', 'fatal', 'security', 'off'

Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	fatal
On-Demand Value	fatal
Override Example	`$sugar_config['logger']['level'] = 'fatal';`

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 browsers 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+
Editions	Professional, Corporate, Enterprise, Ultimate
Default Value	false
Override Example	`$sugar_config['logger']['write_to_server'] = true;`

logger_visible

Description	Determines whether the Logger Settings panel is visible to

	administrators in Admin > System Settings. Instances running on the On-Demand environment will have this setting enforced as false.
Type	Boolean
Range of values	true and false
Versions	6.7.0+
Editions	Professional, Corporate, Enterprise, Ultimate
Default Value	true
On-Demand Value	false
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. Instances running on the On-Demand environment will have this setting enforced as ..
Type	String : Directory Path
Versions	5.2.0+
Editions	Professional, Corporate, Enterprise, Ultimate
Default Value	.
On-Demand Value	.
Override Example	`$sugar_config['log_dir'] = '.';`

log_file

Description	Designates the file name where the instance's logs will be stored. Instances

	running on the On-Demand environment will have this setting enforced as sugarcrm.log.
Type	String : Name of the log file
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	sugarcrm.log
On-Demand Value	sugarcrm.log
Override Example	`$sugar_config['log_file'] = 'new_sugarcrm.log'`

log_memory_usage

Description	Logs the memory usage. Instances running on the On-Demand environment will have this setting enforced as false.
Type	Boolean
Range of values	true and false
Versions	5.5.0+
Editions	Professional, Corporate, Enterprise, Ultimate
Default Value	false
On-Demand Value	false
Override Example	`$sugar_config['log_memory_usage'] = true;`

maintenanceMode

Description	Allows the instance to be placed in a maintenance mode where users cannot access the instance.
Type	Boolean
Range of values	true and false

Versions	7.0.0+
Editions	Professional, Corporate, Enterprise, Ultimate
Default Value	false
Override Example	`$sugar_config['maintenanceMode'] = 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+
Editions	Professional, Corporate, Enterprise, Ultimate
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+
Editions	Professional, Corporate, Enterprise, Ultimate
Override Example	`$sugar_config['mass_actions'] =`

	`array();`

mass_actions.mass_link_chunk_size

Description	Number of records per chunk while performing mass linking updates.
Type	Integer
Versions	7.0.0+
Editions	Professional, Corporate, Enterprise, Ultimate
Default Value	20
Override Example	`$sugar_config['mass_actions']['mass_link_chunk_size'] = 20;`

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. Instances running on the On-Demand environment will have this setting enforced as 1.
Type	Integer : Seconds
Versions	6.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	null
On-Demand Value	1
Override Example	`$sugar_config['max_session_time'] = 1;`

moduleInstaller

Description	Array that defines restrictions on module installations via the Module Loader utility.
Type	Array
Versions	5.2.0.j+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Override Example	`$sugar_config['moduleInstaller'] = array();`

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 blacklisted 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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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. Instances running on the On-Demand environment will have this setting enforced as true.
Type	Boolean
Range of values	true and false
Versions	5.5.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	false
On-Demand Value	true
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. The moduleInstaller.validExt parameter allows you to define additional explicit extensions deemed safe to install on your instance of Sugar. Instances running on the On-Demand environment will have this setting enforced as array('xml').
Type	Array : Extensions
Range of values	File extensions to allow
Versions	6.0.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	array('png', 'gif', 'jpg', 'css', 'js', 'php', 'txt', 'html', 'htm', 'tpl', 'pdf', 'md5', 'xml',

	'hbs', 'less', 'wsdl')
On-Demand Value	array('xml')
Override Example	`$sugar_config['moduleInstaller']['validExt'] = array('swf', 'log');`

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+
Editions	Professional, Corporate, Enterprise, Ultimate
Default Value	false
Override Example	`$sugar_config['noPrivateTeamUpdate'] = true;`

oauth_token_expiry

Description	Sets whether OAuth tokens will expire.
Type	Boolean
Range of values	true and false
Versions	7.5.0.0+
Editions	Professional, Corporate, Enterprise, Ultimate
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 : Seconds
Versions	7.5.0.0+
Editions	Professional, Corporate, Enterprise, Ultimate
Default Value	86400
Override Example	`$sugar_config['oauth_token_life'] = '86400';`

oracle_enable_ci (Deprecated in future release)

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 - 7.7.2.0
Editions	Enterprise, Ultimate
Default Value	false
Override Example	`$sugar_config['oracle_enable_ci'] = true;`

passwordsetting

Description	Defines all of the password requirements for the instance.
Type	Array

Versions	5.5.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	1
Override Example	`$sugar_config['passwordsetting']['onenumber'] = '1';`

passwordsetting.systexpiretiontype

Description	Specifies the unit of measurement for passwordsetting.systexpiretiontime. 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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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 : Seconds
Versions	7.6.0.0+
Editions	Professional, Corporate, Enterprise, Ultimate
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+
Editions	Professional, Corporate, Enterprise, Ultimate
Override Example	`$sugar_config['perfProfile'] = array();`

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: $sugar_config['perfProfile']['TeamSecurity']['{module}']['teamset_prefetch'] = true;
Type	Boolean
Range of values	true and false
Versions	7.2.2.0+
Editions	Professional, Corporate, Enterprise, Ultimate
Default Value	false
Override Example	`$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: $sugar_config['perfProfile']['TeamSecurity']['{module}']['teamset_prefetch_max'] = true;
Type	Integer : Records
Versions	7.2.2.0+
Editions	Professional, Corporate, Enterprise, Ultimate
Default Value	empty
Override Example	`$sugar_config['perfProfile']['TeamSecurity']['default']['teamset_prefetch_max'] = 500;`

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+
Editions	Professional, Corporate, Enterprise, Ultimate
Default Value	false
Override Example	`$sugar_config['perfProfile']['Team Security']['default']['where_condition'] = true;`

pmse_settings_default.error_number_of_cycles

Description	Number of cycles before triggering an error.
Type	String : Cycles
Versions	7.6.0.0+
Editions	Enterprise, Ultimate
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.
Type	String : Seconds
Versions	7.6.0.0+
Editions	Enterprise, Ultimate
Default Value	40
Override Example	`$sugar_config['pmse_settings_default']['error_timeout'] = 45;`

pmse_settings_default.logger_level

Description	The defualt logger level
Type	String
Range of values	emergency, alert, error, warning, notice, info, debug
Versions	7.6.0.0+
Editions	Enterprise, Ultimate
Default Value	critical
Override Example	`$sugar_config['pmse_settings_default']['logger_level'] = 'emergency';`

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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	true
Override Example	`$sugar_config['require_accounts'] = false;`

roleBasedViews

Description	Enables role based views and dropdowns.
Type	Boolean
Range of values	true and false

Versions	7.6.0.0+
Editions	Enterprise, Ultimate
Default Value	false
Override Example	`$sugar_config['roleBasedViews'] = true;`

SAML_X509Cert

Description	The SAML Certificate Key.
Type	String : SAML Certificate Key
Versions	6.1.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Override Example	`$sugar_config['SAML_X509Cert'] = '-----BEGIN CERTIFICATE-----CERTIFICATE KEY-----END CERTIFICATE-----';`

search_engine

Description	Array that defines the search enginer behaviors.
Type	Array
Versions	7.2.0+
Editions	Professional, Corporate, Enterprise, Ultimate
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 : Number of records
Versions	7.2.0+
Editions	Professional, Corporate, Enterprise, Ultimate
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 : Number of records
Versions	7.2.0+
Editions	Professional, Corporate, Enterprise, Ultimate
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+
Editions	Professional, Corporate, Enterprise, Ultimate
Default Value	5000
Override Example	`$sugar_config['search_engine']['search_engine.max_bulk_threshold'] = 10000;`

search_engine.postpone_job_time

Description	Number of time to postpone a job by so that it's not executed twice during the same request.
Type	Integer
Versions	7.2.0+
Editions	Professional, Corporate, Enterprise, Ultimate
Default Value	60
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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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. Instances running on the On-Demand environment will have this setting enforced as empty.
Type	String
Range of values	Directory path
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	empty
On-Demand 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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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 : URL
Range of values	Current URL of Sugar
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Override Example	`$sugar_config['site_url'] = 'http://my.sugarinstance.com';`

slow_query_time_msec

Description	Slow query time threshold. Instances running on the On-Demand environment will have this setting enforced as 5000.
Type	String : Milliseconds
Versions	5.2.0+

Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	5000
On-Demand Value	5000
Override Example	`$sugar_config['slow_query_time_msec'] = '1000';`

stack_trace_errors

Description	Displays stack trace of errors.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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. Instances running on the On-Demand environment will have this setting enforced as 50.
Type	Integer : Number of files to keep
Range of values	Any integer

Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	50
On-Demand 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 : Version Number
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Override Example	`$sugar_config['sugar_version'] = '6.7.3';`

tmp_dir

Description	The directory path for temporary XML files used by charts and diagnostics.
Type	String : Directory path
Range of values	Filesystem path
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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 : Seconds
Versions	7.6.0.0+
Editions	Professional, Corporate, Enterprise, Ultimate
Default Value	86400
Override Example	`$sugar_config['tmp_file_max_lifetime'] = 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 : Number of records
Versions	6.0.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Override Example	`$sugar_config['tracker_max_display_length'] = 45;`

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 : Unique Identifier
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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.
Type	String

Range of values	Directory path
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	upload/
Override Example	`$sugar_config['upload_dir'] = 'upload/'`

upload_maxsize

Description	The maximum file size that users can upload into Sugar as attachments. When uploading files to Sugar, there are three file size limits to configure. The first two limits are your PHP upload_max_filesize and post_max_size which are configured in your php.ini. The second limit is the sugar configuration setting for upload_maxsize, which will restrict the upload limit from within Sugar. This setting can also be modified in the application via Admin > System Settings. The smallest of these three values will be honored when an oversized file has been uploaded.
Type	Integer : Filesize in bytes
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Override Example	`$sugar_config['upload_maxsize'] = 40000000;`

upload_wrapper_class

Description	The name of the class used as upload wrapper.

Type	String : Upload wrapper class
Versions	6.7.0+
Editions	Professional, Corporate, Enterprise, Ultimate
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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Override Example	`$sugar_config['use_sprites'] =`

	`false;`

vcal_time

Description	Used to determine the number of months in advance of the current date that the Free/Busy information for calls and meetings will be published. To turn Free/Busy publishing off, set this variable to '0'. The minimum is 1 month; the maximum is 12 months.
Type	String : Months
Range of values	'1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', and '12'
Versions	5.2.0.c+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	2
Override Example	`$sugar_config['vcal_time'] = '5';`

verify_client_ip

Description	Whether or not to verify the client IP. Setting this to false will disable the system checking to see if the user is accessing Sugar from the IP address of their last page load.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	true
Override Example	`$sugar_config['verify_client_ip'] = false;`

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 : Number of records to display
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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 : Records
Versions	5.2.0+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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 : Wall time in milliseconds
Versions	7.6.0.0+
Editions	Professional, Corporate, Enterprise, Ultimate
Override Example	`$sugar_config['xhprof_config']['filter_wt'] = 2;`

xhprof_config.flags

Description	The flags for xhprof profiler.
Type	String : xhprof flags

Versions	6.5.10+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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 : Function names
Versions	6.5.10+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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 : Directory path
Versions	6.5.10+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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 : Class
Versions	6.5.10+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
Default Value	SugarXHprof
Override Example	`$sugar_config['xhprof_config']['manager'] = 'CustomSugarXHprof';`

xhprof_config.mongodb_collection

Description	The name of the mongo db collections.
Type	String : Mongo collection name
Versions	7.6.0.0+
Editions	Professional, Corporate, Enterprise, Ultimate
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 : Database name
Versions	7.6.0.0+
Editions	Professional, Corporate, Enterprise, Ultimate
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 : Mongo db options
Versions	7.6.0.0+
Editions	Professional, Corporate, Enterprise, Ultimate
Default Value	results
Override Example	`$sugar_config['xhprof_config']['mongodb_options'] = array();`

xhprof_config.mongodb_uri

Description	The mongo server URL.
Type	String : URL
Versions	7.6.0.0+
Editions	Professional, Corporate, Enterprise,

	Ultimate
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 : Sample Rate (1/{value})
Versions	6.5.10+
Editions	Community Edition, Professional, Corporate, Enterprise, Ultimate
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 : Save Location
Range of values	'file' and 'mongodb'
Versions	7.6.0.0+
Editions	Professional, Corporate, Enterprise, Ultimate
Default Value	mongodb
Override Example	`$sugar_config['xhprof_config']['save_to'] = 'mongodb';`

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 : Save Location
Range of values	'file' and 'mongodb'
Versions	7.6.0.0+
Editions	Professional, Corporate, Enterprise, Ultimate
Default Value	mongodb
Override Example	`$sugar_config['xhprof_config']['save_to'] = 'mongodb';`

Last Modified: 12/05/2016 03:01pm

Using Configurator

Overview

The Configuator is a class used to manage the config settings found in 'config.php' and 'config_override.php'.

The Configurator Class

The Configurator class is located in 'modules/Configurator/Configurator.php'. Settings modified using this class are written to 'config_override.php' which is stored in the root directory of your SugarCRM installation.

Retrieving Settings

You can access the SugarCRM config settings by using the global variable '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. Alternatively, you can use the Administration class located in ./modules/Administration/Administration.php to store the settings in the 'config' table in the database.

Last Modified: 09/26/2015 04:14pm

DBManagerFactory

Overview

Provides an overview of the DBManagerFactory class that will allow you to generate the appropriate manager for the database you are using.

Instantiating a DB Object

To use the DB object you should use the global database object:

```
$GLOBALS['db']
```

If you should need to manually instantiate the DB object, you can use the DBManagerFactory class's getInstance() method. This method will create a reference to the DB object for the instance.

```
$db = DBManagerFactory::getInstance();
```

Querying The Database

Retrieving Results

To query the database looking for a result set, you will use the query() and fetchByAssoc() methods. The query() method will retrieve the results while the fetchByAssoc() method will allow for you to iterate through the results. An Example is below:

```
$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'];
}
```

Retrieving a Single Result

To retrieve a single result from the database, such as a specific record field, you can use the getOne() method.

```
$sql = "SELECT name FROM accounts WHERE id = '{$id}'";

$name = $GLOBALS['db']->getOne($sql);
```

Limiting Results

To limit the results of a query, you can add a limit to the sql string yourself or use the limitQuery() method. An example is below:

```
$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'];
}?
```

Generating SQL Queries

To have Sugar automatically generate SQL queries, you can use some methods from the bean class.

Select Queries

To create a select query you can use the create_new_list_query() method. An example is below:

```
$bean = BeanFactory::newBean($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. An example is below:

```
$bean = BeanFactory::newBean('Accounts');

$sql = "SELECT * FROM accounts WHERE deleted = 0";

$count_sql = $bean->create_list_count_query($sql);
```

Last Modified: 11/19/2015 12:22am

SugarApplication

Overview

An overview of the SugarApplication class used to help manage tasks for the controller and views.

This is only applicable to modules running in backward compatibility mode.

Displaying Error Messages

Sometimes error messages or notices need to be displayed to users, however, the executing logic cannot be exited. When this happens you can use the method appendErrorMessage() to display message to the user once the next view loads. An example is shown below:

```
SugarApplication::appendErrorMessage("{$message}");
```

Redirecting Users

To redirect a user to a new page, you can use the redirect() method. This method will handle exiting the code for you. An example is shown below:

```
$urlParameters = array(
  'module' => $module,
  'action' => $action,
);

$url = 'index.php?' . http_build_query($urlParameters);

SugarApplication::redirect($url);
```

Last Modified: 09/26/2015 04:14pm

SugarAutoLoader

SugarAutoLoader Overview.

Last Modified: 09/26/2015 04:14pm

Introduction

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 . The generated file map is located in the cache directory as ./cache/file_map.php. If this file is manually deleted, it will be automatically rebuilt.

Valid File Extensions

The autoloader will only map files with the following extensions:

- bmp
- css
- gif
- html
- jpg
- js
- less
- override
- php
- png
- tif
- tpl
- xml

*All other file extensions are ignored.

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/
- ./include/HTMLPurifier/
- ./include/nusoap/
- ./include/pclzip/
- ./include/phpmailer/
- ./include/reCaptcha/
- ./include/ytree/
- ./log4php/

- ./portal/
- ./tests/
- ./upload/
- ./Zend/

Initializing the AutoLoader

The init() function will Initialize the loader, register autoload() as an autoloader function, load the bean map, file map, and extension map. This function is called at the beginning of an entryPoint.

```
require_once('include/utils/autoloader.php');
SugarAutoloader::init();
```

Rebuilding the File Map

The autoloader rebuilds the map on Quick Repair and Rebuilds (Admin > Repair > Quick Repair and Rebuild), module installs, and can rebuilt at any time programmatically using the buildCache() function. Same other functions, such as write_array_to_file(), will automatically update the file map. If your code is writing custom files, it is strongly recommended to update the file map immediately. You should also note that if you're hosting your Sugar environment OnDemand, you are subject to the Module Loader Restrictions .

The buildCache Function

To programmatically rebuild the file file map, you can use the function as shown below:

```
SugarAutoloader::buildCache();
```

Last Modified: 09/26/2015 04:14pm

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/');
```

Last Modified: 09/26/2015 04:14pm

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/TimeDate.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
$file)
{
    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.

Last Modified: 09/26/2015 04:14pm

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);
```

Last Modified: 10/04/2016 02:02pm

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',
'editviewdefs');
```

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');
```

Last Modified: 09/26/2015 04:14pm

SugarBean

SugarBean Overview.

Last Modified: 09/26/2015 04:14pm

CRUD Handling

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.

Create & Read

Creating and Retrieving Records

The BeanFactory class is used for bean loading. The class should be used any time 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 records id by doing 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 as follows:

```
//Create bean
$bean = BeanFactory::newBean($module);

//Set the record id
$bean->id = '38c90c70-7788-13a2-668d-513e2b8df5e1';
$bean->new_with_id = true;

//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.

Identifying 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
Point');

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
}
```

Update

Updating a Bean

Updating a bean can be done 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('disable_row_level_security' => true));
```

Updating a Bean without Modifying 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 the 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('disable_row_level_security' => true));
```

Delete

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();

//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('disable_row_level_security' => true));
```

Last Modified: 08/08/2016 03:35pm

Fetching Relationships

Overview

The SugarBean class supports fetching related information from the database.

Fetching Related Records

To fetch related records, you will need to 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 a Parent Record

Fetching a parent record is very 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);
    }
}
```

Last Modified: 09/26/2015 04:14pm

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;

?>
```

Last Modified: 09/26/2015 04:14pm

SugarLogger

SugarLogger Overview.

Last Modified: 09/26/2015 04:14pm

Introduction

Overview

An overview of the SugarLogger.

The Sugar Logger

The Sugar Logger class, located in ./include/SugarLogger/SugarLogger.php, allows for developers and system administrators to log system events to a log file. The system then determines which events to write to the log based on the systems Log Level set in Admin > System Settings.

Log Levels

Sugar makes use of the following log levels:

- Debug
- Info
- Warn

- Deprecated
- Error
- Fatal
- Security
- Off

Each log level is weighted in a way that allows the logger to easily identify which levels to capture. Each level captures messages for all levels below it. The example being that when the log level is set to Fatal, the system will also log any Security level events that occur. If the Debug level were selected, the system would then log events for all levels. When you are not troubleshooting Sugar, the log level should be set to Fatal. Doing so will ensure your environment is not wasting unnecessary resources to write to to the Sugar log.

Logging Messages

When developing customizations, you may want to log custom messages to the Sugar log to help administrators identify issues. To go this, you can access the SugarLogger class by using the globally defined variable $GLOBALS['log']. Examples of the various ways to log a message are shown below:

Debugging Events

```
$GLOBALS['log']->debug('Debugging message');
```

Informational Events

```
$GLOBALS['log']->info('Informational message');
```

Warning Events

```
$GLOBALS['log']->warn('Warning message');
```

deprecated Events

```
$GLOBALS['log']->deprecated('Deprecated message');
```

Error Events

```
$GLOBALS['log']->error('Error message');
```

Fatal Events

```
$GLOBALS['log']->fatal('Fatal message');
```

Security Events

```
$GLOBALS['log']->security('Security message');
```

Debugging Messages

_ppl

When developing, it may be beneficial for a developer to use _ppl() to log a message to the Sugar log.

```
_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.

setLevel

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 $GLOBALS['log']->setLevel(). An example of this is shown below:

```
$GLOBALS['log']->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.

Log Rotation

The Sugar Logger will automatically rotate the logs when the 'maxSize' setting has been met or exceeded. When this happens, the Sugar log will be renamed with an integer. The example being that if the Sugar log was named "sugarcrm.log", it will then be renamed to "sugarcrm_1.log". The next log rotation after that would then create"sugarcrm_2.log". This will occur until the 'maxLogs' setting has been met. Once met, the log rollover will then start over.

Last Modified: 09/26/2015 04:14pm

Custom Loggers

Overview

How to integrate the logger with alternate logging systems.

Custom Loggers

Custom loggers 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 below code is an example of how to create a FirePHP logger.

You will first need to create your logger class in ./custom/include/SugarLogger/. For this example, we will create ./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'] ==
'FirePHP'
        )
        {
            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 constuctor 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';
```

Last Modified: 09/26/2015 04:14pm

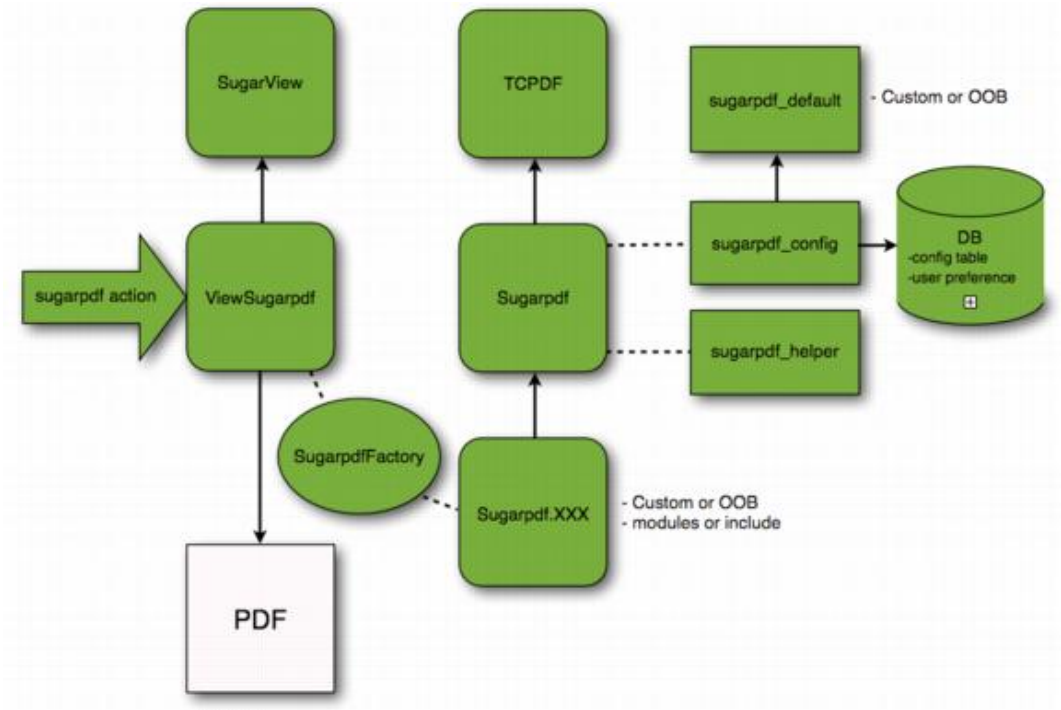
SugarPDF

Overview

An overview of the PDF framework.

SugarPDF Architecture

The PDF framework leverages the SugarCRM MVC architecture.



Key PDF Classes

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	Overriden 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	Overriden 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
predisply	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

Each extended class will define a specific PDF view that is accessible with the following URL parameters:

- module=<module>
- action=sugarpdf
- sugarpdf=<pdf view>

Withing 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 action>
```

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=sugarpdf&sugarpdf=pdfmanager&pdf_template_id=<template id>
```

SugarpdfFactory

The ViewSugarpd 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 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.

FontManager

The FontManagerclass, located in ./include/Sugarpdf/FontManager.php, is a stand-alone class that manages all the fonts for TCPDF.

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.

Last Modified: 01/15/2016 09:34pm

Generating PDFs

Overview

Overview of how to create a PDF document.

Generating a PDF

To generate a PDF in Sugar, you will need to create a PDF view that extends the Sugarpdf class. To do this you will need to 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->getScaleFactor(), 'cap' => 'butt', 'join' => 'miter', 'dash' =>
0, 'color' => array(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 name when downloaded by the user.

Once the file is in place, you can access it by navigating to:

```
http://{sugar
url}/index.php?module=<module>&action=sugarpdf&sugarpdf=<pdf action>
```

This will generate and download the PDF document as 'example.pdf'.

Last Modified: 09/26/2015 04:14pm

PDF Settings

Overview

An overview of how to manage PDF settings.

PDF Setting Framework

This section will outline how to configure the general PDF settings. There are the

settings that determine widths, heights, images and pdf metadata as well as 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_SETTING"]);

    //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

```
<?php

    $mod_strings['LBL_PDF_NEW_SETTING_TITLE'] = 'PDF New Setting';
    $mod_strings['LBL_PDF_NEW_SETTING_INFO'] = 'Display info for the
new PDF setting';
```

Once finished, navigate to Admin > Repair > Quick Repair and Rebuild. This will rebuild the language files for your display text.

Last Modified: 01/15/2016 09:34pm

Fonts

Overview

An overview of the FontManager.

Font Manager

The FontManager class is a stand-alone class that manages all the fonts for TCPDF.

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 Font loadable module 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.

Font Framework

The stock fonts for TCPDF are stored in the directory ./include/tcpdf/fonts/. If you would like to add additional fonts to the system, they should be added to the ./custom/include/tcpdf/fonts/ directory.

Font Pack

An additional set of fonts can be downloaded from SugarForge at the following link: http://www.sugarforge.org/projects/fontpack/.

This font pack will allow you to generate PDF documents in various languages. It extends the SugarCRM charsets to support most of the symbols in common use such as European, Cyrillic, Greek, Arabic and Hebrew.

Last Modified: 09/26/2015 04:14pm

SugarQuery

Overview

Provides an overview of the SugarQuery class that will allow you to generate the applicable SQL for the database backend you are using.

Instantiating a SugarQuery Object

To use SugarQuery, you will require the SugarQuery class and create a new object:

```
require_once('include/SugarQuery/SugarQuery.php');
$sugarQuery = new SugarQuery();
```

Building SQL Queries

The following sections will outline how to build a query using SugarQuery. All examples below ignore team security. To enable team security for your query, you can ignore passing in array('team_security' => false) as a second parameter to the from clause.

SELECT and FROM Clauses

The SELECT and FROM can be constructed by using the select() and from() methods. The example below will select the id and name fields from the accounts module.

```
//add fields to select
$sugarQuery->select(array('id', 'name'));

//fetch the bean of the module to query
$bean = BeanFactory::newBean('Accounts');

//add from table of the module we are querying
$sugarQuery->from($bean, array('team_security' => false));
```

SQL Result

The resulting SQL statement from the compileSql() function for the parameters above in MySQL would be:

```
SELECT accounts.id, accounts.name FROM accounts WHERE accounts.deleted
= 0
```

WHERE Clauses

To add a WHERE clause to the query you can use the where() method. The WHERE clause below will add records where the assigned_user_id is equal to 'seed_sally_id' and where the name field begins with the letter "I".

```
//add fields to select
$sugarQuery->select(array('id', 'name'));

//fetch the bean of the module to query
$bean = BeanFactory::newBean('Accounts');

//add from table of the module we are querying
$sugarQuery->from($bean, array('team_security' => false));

//add the where clause
$sugarQuery->where()

//where the assigned_user_id field is equal to 'seed_sally_id'
->equals('assigned_user_id', 'seed_sally_id')

//where the name field is starts with 'I'
->starts('name', 'I');
```

SQL Result

The resulting SQL statement from the compileSql() function for the parameters above in MySQL would be:

```
SELECT accounts.id, accounts.name FROM accounts WHERE accounts.deleted
= 0 AND accounts.assigned_user_id = 'seed_sally_id' AND accounts.name
LIKE 'I%'
```

WHERE Grouped Clauses

To add grouped where clauses, you can use the where() method with queryOr() or queryAnd().

Grouped Or

The WHERE clause below will add records where the assigned_user_id is equal to 'seed_sally_id' or where the name field begins with the letter "I".

```
//add fields to select
$sugarQuery->select(array('id', 'name'));

//fetch the bean of the module to query
$bean = BeanFactory::newBean('Accounts');

//add from table of the module we are querying
$sugarQuery->from($bean, array('team_security' => false));

//add the where clause
$sugarQuery->where()

//add grouped or
->queryOr()

//where the assigned_user_id field is equal to 'seed_sally_id'
->equals('assigned_user_id', 'seed_sally_id')

//where the name field is starts with 'I'
->starts('name', 'I');
```

SQL Result

The resulting SQL statement from the compileSql() function for the parameters above in MySQL would be:

```
SELECT accounts.id id, accounts.name name FROM accounts WHERE
accounts.deleted = 0 AND (accounts.assigned_user_id = 'seed_sally_id'
OR accounts.name LIKE 'I%')
```

Grouped And

The WHERE clause below will add records where the assigned_user_id is equal to

'seed_sally_id' and where the name field begins with the letter "I".

```
//add fields to select
$sugarQuery->select(array('id', 'name'));

//fetch the bean of the module to query
$bean = BeanFactory::newBean('Accounts');

//add from table of the module we are querying
$sugarQuery->from($bean, array('team_security' => false));

//add the where clause
$sugarQuery->where()

//add grouped and
->queryAnd()

//where the assigned_user_id field is equal to 'seed_sally_id'
->equals('assigned_user_id', 'seed_sally_id')

//where the name field is starts with 'I'
->starts('name', 'I');
```

SQL Result

The resulting SQL statement from the compileSql() function for the parameters above in MySQL would be:

```
SELECT accounts.id id, accounts.name name FROM accounts WHERE
accounts.deleted = 0 AND (accounts.assigned_user_id = 'seed_sally_id'
AND accounts.name LIKE 'I%')
```

JOIN Clauses

To add a relationship join you can use the join() method. The example below will join accounts to contacts.

```
//add fields to select
$sugarQuery->select(array('id', 'name'));

//fetch the bean of the module to query
$bean = BeanFactory::newBean('Accounts');
```

```
//add from table of the module we are querying
$sugarQuery->from($bean, array('team_security' => false));

//add join
$sugarQuery->join('contacts');
```

SQL Result

The resulting SQL statement from the compileSql() function for the parameters above in MySQL would be:

```
SELECT accounts.id, accounts.name FROM accounts INNER JOIN
accounts_contacts jt1_accounts_contacts ON (accounts.id =
jt1_accounts_contacts.account_id AND jt1_accounts_contacts.deleted =
'0') INNER JOIN contacts jt0_contacts ON (jt0_contacts.id =
jt1_accounts_contacts.contact_id AND jt0_contacts.deleted = 0) WHERE
accounts.deleted = 0
```

LIMIT and OFFSET Clauses

To add a limit and/or offset you can use the corresponding limit() and offset() methods.

```
//add fields to select
$sugarQuery->select(array('id', 'name'));

//fetch the bean of the module to query
$bean = BeanFactory::newBean('Accounts');

//add from table of the module we are querying
$sugarQuery->from($bean, array('team_security' => false));

//set the offset
$sugarQuery->offset(5);

//set the limit
$sugarQuery->limit(10);
```

SQL Result

The resulting SQL statement from the compileSql() function for the parameters above in MySQL would be:

```
SELECT accounts.id, accounts.name FROM accounts WHERE accounts.deleted
= 0 LIMIT 5,10
```

GROUP BY Clauses

To add a group by , you can use the corresponding groupBy() method.

```
//add fields to select
$sugarQuery->select(array('account_type', 'count(account_type)'));

//fetch the bean of the module to query
$bean = BeanFactory::newBean('Accounts');

//add from table of the module we are querying
$sugarQuery->from($bean, array('team_security' => false));

//add group by
$sugarQuery->groupBy('account_type');
```

SQL Result

The resulting SQL statement from the compileSql() function for the parameters above in MySQL would be:

```
SELECT accounts.account_type, count('account_type') FROM accounts
WHERE accounts.deleted = 0 GROUP BY accounts.account_type
```

UNIONS

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);
```

SQL Result

The resulting SQL statement from the compileSql() function for the parameters above in MySQL would be:

```
(SELECT accounts.id id, accounts.name name FROM accounts WHERE
accounts.deleted = 0 AND accounts.account_type IN ('Customer')) UNION
ALL (SELECT accounts.id id, accounts.name name FROM accounts WHERE
accounts.deleted = 0 AND accounts.account_type IN ('Investor')) LIMIT
0,5
```

Executing The SQL Query

Retrieving Results

To query the database for a result set, you will use the execute() method. The execute() method will retrieve the results in an array that you can iterate through. An example of fetching records from an account is below:

```
//fetch the bean of the module to query
```

```
$bean = BeanFactory::newBean('Accounts');

require_once('include/SugarQuery/SugarQuery.php');
$sugarQuery = new SugarQuery();

//add fields to select
$sugarQuery->select(array('id', 'name'));

//add from table of the module we are querying
$sugarQuery->from($bean, array('team_security' => false));

//add the where clause
$sugarQuery->where()
    //where the assigned_user_id field is equal to 'seed_sally_id'
    ->equals('assigned_user_id', 'seed_sally_id')
    //where the name field is starts with 'I'
    ->starts('name', 'I');

//fetch the result
$result = $sugarQuery->execute();
```

Result

The resulting SQL statement from the compileSql() function for the parameters above in MySQL would be:

```
SELECT accounts.id, accounts.name FROM accounts WHERE accounts.deleted
= 0 AND accounts.assigned_user_id = 'seed_sally_id' AND accounts.name
LIKE 'I%'
```

The execute() function will return an array of results that you can iterate through:

```
Array
(
    [0] => Array
    (
        [id] => f39593da-3f88-3059-4f18-524b4d23d07b
        [name] => International Art Inc
    )
)
```

Note: An empty resultset will return an empty array.

Last Modified: 08/12/2016 10:06am

TinyMCE

TinyMCE Overview.

Last Modified: 09/26/2015 04:14pm

Modifying the TinyMCE Editor

Overview

This article is a brief overview on how to modify the default settings for the TinyMCE Editor.

TinyMCE Editor

The default settings for TinyMCE can be configured 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

This is used when creating an email template.

email_compose

This is used when composing an email from the full form under the emails module.

email_compose_light

This is 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,strikethrough,separator,jus
tifyleft,justifycenter,justifyright,justifyfull,separator,forecolor,ba
ckcolor,separator,styleselect,formatselect,fontselect,fontsizeselect,"
,
        'buttonConfig2' =>
"cut,copy,paste,pastetext,pasteword,selectall,separator,search,replace
,separator,bullist,numlist,separator,outdent,indent,separator,ltr,rtl,
separator,undo,redo,separator,
link,unlink,anchor,image,separator,sub,sup,separator,charmap,visualaid
",
        'buttonConfig3' =>
"tablecontrols,separator,advhr,hr,removeformat,separator,insertdate,in
serttime,separator,preview"
    );

    //Standard email compose
    $buttonConfigs['email_compose'] = array(
        'buttonConfig' =>
"code,help,separator,bold,italic,underline,strikethrough,separator,bul
list,numlist,separator,justifyleft,justifycenter,justifyright,justifyf
ull,separator,forecolor,backcolor,separator,styleselect,formatselect,f
ontselect,fontsizeselect,",
        'buttonConfig2' => "", //empty
        'buttonConfig3' => "" //empty
    );

    //Quick email compose
```

```
    $buttonConfigs['email_compose_light'] = array(
        'buttonConfig' =>
"code,help,separator,bold,italic,underline,strikethrough,separator,bul
list,numlist,separator,justifyleft,justifycenter,justifyright,justifyf
ull,separator,forecolor,backcolor,separator,styleselect,formatselect,f
ontselect,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,searchreplace,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 here:
http://www.tinymce.com/wiki.php/Creating_a_plugin

Last Modified: 09/26/2015 04:14pm

UploadFile

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 files 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 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 as:

```
/Library/WebServer/htdocs/sugarcrm/upload/1d0fd9cc-02e5-f6cd-1426-51a5
09a63334
```

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);
```

Last Modified: 09/26/2015 04:14pm

Sidecar

An overview of how to develop using the various Sidecar layout and view widgets.

Last Modified: 09/26/2015 04:14pm

Alerts

Overview

The alert view widget.

Alerts

The alert view widget, located in ./clients/base/views/alert/, is used to display information to the user. It can be used to display loading messages, notices, or even a confirmation message to the user.

Methods

app.alert.show(id, options)

Displays an alert message to the user with the options provided.

Parameters

Name	Description
id	The id of the alert message.
options.level	The alert level. Possible values are: • info - displays the message in blue. • success - displays the message in green. • warning - displays the message in yellow. • error - displays the message in red. • process - displays the loading message. • confirmation - displays a confirmation dialog.
options.title	The alerts title. The default titles for each level are listed below: • info - "Notice". • success - "Success".

	• warning - "Warning!". • error - "Error". • process - "Loading...". • confirmation - "Warning".
options.messages	The message to display to the user. It is important to note that 'process' alerts do not display messages.
options.autoClose	Whether or not to auto-close the alert popup.
options.onClose	Callback handler of action Confirm for confirmation alerts
options.onCancel	Callback handler of action Cancel for confirmation alerts.
options.onLinkClick	Callback handler for click actions on a link inside of the 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
'Loading...'
});
```

app.alert.dismiss(id)

Dismisses an alert message from view.

Parameters

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

Example

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

app.alert.dismissAll

Dismisses all alert messages from view.

Example

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

Testing in Console

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

```
App.alert.show('message-id', {
    level: 'success',
    messages: 'Successful!',
```

```
    autoClose: false
});
```

Last Modified: 03/14/2016 07:17pm

Drawers

Overview

The drawer layout widget.

Drawers

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 previous content the user was viewing.

Methods

app.drawer.open(layoutDef, onClose)

Displays new content window over the current view.

Parameters

Name	Description
layoutDef.layout	The id of the layout to load.
layoutDef.context	Optional. Additional data you would like to pass to the drawer. Data passed in can be retrieved from the view by using: `this.context.get('<data key>');` You should 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	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)

Dismisses the top most drawer.

Parameters

Name	Description
callbackOptions	Optional. 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)

Destroys all drawers at once. By default, whenever the application is routed to another page, reset() is called.

Parameters

Name	Description
triggerBefore	Optional. Determines whether to triggerBefore. Defaults to false.

Example

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

Last Modified: 01/15/2016 09:30pm

Language

Overview

The helper library for labels and lists.

Language

The language library, located in ./sidecar/src/core/language.js, is used to manage the users display language as well as fetch labels and lists.

Methods

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

Fetches a string for a given key. Searches the module strings first and falls back to the app strings. If the label is a template, it will be compiled and executed with the given 'context'.

Parameters

Name	Description
key	Key of the string to retrieve.
module	Optional. Module the label belongs to.
context	Optional. Template context.

Example

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

app.lang.getAppString(key)

Retrieves an application string for a given key.

Parameters

Name	Description
key	Key of the string to retrieve.

Example

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

app.lang.getAppListStrings(key)

Retrieves an application list string or object.

Parameters

Name	Description
key	Key of the list to retrieve.

Example

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

app.lang.getModuleSingular(moduleKey)

Retrieves an application list string or object.

Parameters

Name	Description
moduleKey	Module key of the singular module label to retrieve.

Example

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

app.lang.getLanguage()

Retrieves the current users language key.

Example

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

app.lang.updateLanguage(languageKey)

Updates the current users language key.

Parameters

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

Example

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

Last Modified: 01/15/2016 09:30pm

Web Services

Web Services Overview.

Last Modified: 09/26/2015 04:14pm

API Versioning

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 preexisting 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 API specific to your release. A quick reference of this can be found below:

Release	REST Version	REST URL	SOAP Version	SOAP URL
7.7.x	v10	http://{site url}/rest/v10/	v4.1	http://{site url}/service/v4_1/soap.php
7.6.x	v10	http://{site url}/rest/v10/	v4.1	http://{site url}/service/v4_1/soap.php
7.5.x	v10	http://{site url}/rest/v10/	v4.1	http://{site url}/service/v4_1/soap.php
6.7.x	v4.1	http://{site url}/service/v4_1/rest.php	v4.1	http://{site url}/service/v4_1/soap.php
6.5.x	v4.1	http://{site url}/service/v4_1/rest.php	v4.1	http://{site url}/service/v4_1/soap.php

Note: As of 7.x, SOAP support is no longer offered with the new APIs. The legacy

APIs are still accessible in the product, however, any existing integrations should be updated to use the latest REST endpoints.

Last Modified: 07/27/2016 04:55pm

REST

Overview

v10+ REST service 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. This v10 API is separate from the v2-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.

How Do I Access The REST Service?

The v10 REST service can be found at:

```
http://{site url}/rest/v10/
```

Help documentation for the various v10 endpoints can be found by navigating to:

```
http://{site url}/rest/v10/help
```

Note: 'site url' is the URL of your Sugar instance.

You can find out more about versioning in the API Versioning section.

How Do I Login?

To authenticate you will need to use the oauth2/token endpoint. This endpoint will return an access_token that you will use to authenticate for the various other endpoints. This token must be passed to the other endpoints as the oauth-token

header or the system will not authenticate the request. It is also important to note that if you are building an integration, you should create new OAuth Key in Admin > OAuth Keys. This will prevent the integration from conflicting with existing sessions that might log users out of the system. An example of authenticating can be found in the /oauth2/token POST example.

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"]}}
```

Last Modified: 03/28/2016 04:18pm

Extending Web Services

Overview

How to add your own custom endpoints to the REST API.

Custom Endpoints

As of Sugar 7.0.0, you can easily define your own endpoint. All custom global endpoints will be located in the following directory:

`./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.

An example is shown below:

./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
documentation
                '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;
    }

}

?>
```

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	String	The request type 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 path of:
		
		`http://{site url}/rest/v10/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 in the following to your selected method as the $args

		parameter when calling http://{site url}/rest/v10/MyEndpoint/GetExample/MyData
		
		`stdClass Object`
		`(`
		`[__sugar_url] => v10/MyEndpoint/GetExample/mydata`
		`[data] => 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.

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 $endpointMethod($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
endpoint.
        </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.

Redefining Existing Endpoints

With endpoints, you may have a need to extend an existing endpoint to meet your needs. When doing this it in 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.

./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();
    }
}
```

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/v10/ping will result in a response of "ping <timestamp>".

Last Modified: 01/15/2016 09:30pm

Examples

Examples of API Calls.

Last Modified: 09/26/2015 04:14pm

v10

Examples using the v10 REST API

Last Modified: 05/16/2016 10:55am

/<module>/export/:record_list_id GET

Overview

Examples demonstrating how to export a list of records using the v10 REST API.

Examples

PHP

```
<?php

$base_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

/**
 * Generic function to make cURL request.
 * @param $url - The URL route to use.
 * @param string $oauthtoken - The oauth token.
 * @param string $type - GET, POST, PUT, DELETE. Defaults to GET.
 * @param array $arguments - Endpoint arguments.
 * @param array $encodeData - Whether or not to JSON encode the data.
 * @param array $returnHeaders - Whether or not to return the headers.
 * @return mixed
 */
function call(
    $url,
    $oauthtoken='',
    $type='GET',
    $arguments=array(),
    $encodeData=true,
    $returnHeaders=false
)
{
    $type = strtoupper($type);

    if ($type == 'GET')
    {
        $url .= "?" . http_build_query($arguments);
    }

    $curl_request = curl_init($url);

    if ($type == 'POST')
```

```
    {
        curl_setopt($curl_request, CURLOPT_POST, 1);
    }
    elseif ($type == 'PUT')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "PUT");
    }
    elseif ($type == 'DELETE')
    {
        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, $returnHeaders);
    curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

    if (!empty($oauthtoken))
    {
        $token = array("oauth-token: {$oauthtoken}","Content-Type:
application/json");
        curl_setopt($curl_request, CURLOPT_HTTPHEADER, $token);
    }

    if (!empty($arguments) && $type !== 'GET')
    {
        if ($encodeData)
        {
            //encode the arguments as JSON
            $arguments = json_encode($arguments);
        }
        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $arguments);
    }

    $result = curl_exec($curl_request);

    if ($returnHeaders)
    {
        //set headers from response
        list($headers, $content) = explode("\r\n\r\n", $result ,2);
        foreach (explode("\r\n",$headers) as $header)
        {
```

```
            header($header);
        }

        //return the nonheader data
        return trim($content);
    }

    curl_close($curl_request);

    //decode the response from JSON
    $response = json_decode($result);

    return $response;
}

//Login - POST /oauth2/token

$url = $base_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id/secret you created in Admin > OAuth Keys
    "client_id" => "<CustomID>",
    "client_secret" => "<CustomSecret>",
    "username" => $username,
    "password" => $password,
    "platform" => "base"
);

//Identify records to export - GET /<module>/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" => 500,
    "offset" => 0,
    "fields" => "id",
    "order_by" => "",
    "favorites" => false,
    "my_items" => false,
);

$url = $base_url . "/Accounts/filter";

$filter_response = call($url, $oauth2_token_response->access_token,
'GET', $filter_arguments);

//store ids of records to export
$export_ids = array();
foreach ($filter_response->records as $record)
{
    $export_ids[] = $record->id;
}

//Create a record list - POST /<module>/record_list

$url = $base_url . "/Accounts/record_list";

$record_list_arguments = array(
    "records" => $export_ids,
);

$record_list_response = call($url,
$oauth2_token_response->access_token, 'POST', $record_list_arguments);

//Export Records - GET /<module>/export/:record_list_id

$url = $base_url . "/Accounts/export/" . $record_list_response->id;

$export_response = call($url, $oauth2_token_response->access_token,
'GET', array(), true, true);
```

```
echo $export_response;

?>
```

Results

Request

This endpoint does not accept any request arguments.

Response


```
{
   "next_offset":2,
   "records":[
      {
         "id":"d8f05e67-dee3-553d-0040-5342e88f2fd1",
         "name":"Avery Software Co",
         "date_modified":"2014-04-07T13:58:50-04:00",
         "description":"",
         "_acl":{
            "fields":{

            }
         },
         "_module":"Accounts"
      },
      {
         "id":"282c529d-f808-a046-e901-5342e8a4aa2a",
         "name":"AtoZ Co Ltd",
         "date_modified":"2014-04-07T13:58:50-04:00",
         "description":"",
         "_acl":{
            "fields":{

            }
         },
         "_module":"Accounts"
      }
   ]
}
```


Last Modified: 05/16/2016 10:47am

/<module>/filter GET

Overview

Examples demonstrating how to filter records using the v10 REST API.

Examples

PHP

```
<?php

$base_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

/**
 * Generic function to make cURL request.
 * @param $url - The URL route to use.
 * @param string $oauthtoken - The oauth token.
 * @param string $type - GET, POST, PUT, DELETE. Defaults to GET.
 * @param array $arguments - Endpoint arguments.
 * @param array $encodeData - Whether or not to JSON encode the data.
 * @param array $returnHeaders - Whether or not to return the headers.
 * @return mixed
 */
function call(
    $url,
    $oauthtoken='',
    $type='GET',
    $arguments=array(),
    $encodeData=true,
    $returnHeaders=false
)
{
    $type = strtoupper($type);

    if ($type == 'GET')
    {
        $url .= "?" . http_build_query($arguments);
    }
```

```
    $curl_request = curl_init($url);

    if ($type == 'POST')
    {
        curl_setopt($curl_request, CURLOPT_POST, 1);
    }
    elseif ($type == 'PUT')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "PUT");
    }
    elseif ($type == 'DELETE')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "DELETE");
    }

    curl_setopt($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
    curl_setopt($curl_request, CURLOPT_HEADER, $returnHeaders);
    curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

    if (!empty($oauthtoken))
    {
        $token = array("oauth-token: {$oauthtoken}","Content-Type:
application/json");
        curl_setopt($curl_request, CURLOPT_HTTPHEADER, $token);
    }

    if (!empty($arguments) && $type !== 'GET')
    {
        if ($encodeData)
        {
            //encode the arguments as JSON
            $arguments = json_encode($arguments);
        }
        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $arguments);
    }

    $result = curl_exec($curl_request);

    if ($returnHeaders)
    {
```

```
        //set headers from response
        list($headers, $content) = explode("\r\n\r\n", $result ,2);
        foreach (explode("\r\n",$headers) as $header)
        {
            header($header);
        }

        //return the nonheader data
        return trim($content);
    }

    curl_close($curl_request);

    //decode the response from JSON
    $response = json_decode($result);

    return $response;
}

//Login - POST /oauth2/token

$url = $base_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id/secret you created in Admin > OAuth Keys
    "client_id" => "<CustomID>",
    "client_secret" => "<CustomSecret>",
    "username" => $username,
    "password" => $password,
    "platform" => "base"
);

$oauth2_token_response = call($url, '', 'POST',
$oauth2_token_arguments);

//Filter - /<module>/filter GET

$filter_arguments = array(
    "filter" => array(
        array(
            '$or' => array(
                array(
                    //name starts with 'a'
```

```
                    "name" => array(
                        '$starts'=>"a",
                    )
                ),
                array(
                    //or name is equal to 'My Account'
                    "name" => 'My Account'
                )
            ),
        ),
    ),
    "max_num" => 2,
    "offset" => 0,
    "fields" => "name,description",
    "order_by" => "name:DESC",
    "favorites" => false,
    "my_items" => false,
);

$url = $base_url . "/Accounts/filter";

$filter_response = call($url, $oauth2_token_response->access_token,
'GET', $filter_arguments);

echo "<pre>";
print_r($filter_response);
echo "</pre>";

?>
```

Results

Request

```
http://{site_url}/rest/v10/Accounts/filter?filter%5B0%5D%5B%24or%5D%5B
0%5D%5Bname%5D%5B%24starts%5D=a&filter%5B0%5D%5B%24or%5D%5B1%5D%5Bname
%5D=My+Account&max_num=2&offset=0&fields=name%2Cdescription&order_by=n
ame%3ADESC&favorites=0&my_items=0
```

Note: GET request arguments are passed in the form of a query string.

Response

```
{
   "next_offset":2,
   "records":[
      {
         "id":"d8f05e67-dee3-553d-0040-5342e88f2fd1",
         "name":"Avery Software Co",
         "date_modified":"2014-04-07T13:58:50-04:00",
         "description":"",
         "_acl":{
            "fields":{

            }
         },
         "_module":"Accounts"
      },
      {
         "id":"282c529d-f808-a046-e901-5342e8a4aa2a",
         "name":"AtoZ Co Ltd",
         "date_modified":"2014-04-07T13:58:50-04:00",
         "description":"",
         "_acl":{
            "fields":{

            }
         },
         "_module":"Accounts"
      }
   ]
}
```


Last Modified: 05/16/2016 10:48am

/<module>/filter POST

Overview

Examples demonstrating how to filter records with the v10 REST API.

Examples

PHP

```php
<?php

$base_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

/**
 * Generic function to make cURL request.
 * @param $url - The URL route to use.
 * @param string $oauthtoken - The oauth token.
 * @param string $type - GET, POST, PUT, DELETE. Defaults to GET.
 * @param array $arguments - Endpoint arguments.
 * @param array $encodeData - Whether or not to JSON encode the data.
 * @param array $returnHeaders - Whether or not to return the headers.
 * @return mixed
 */
function call(
    $url,
    $oauthtoken='',
    $type='GET',
    $arguments=array(),
    $encodeData=true,
    $returnHeaders=false
)
{
    $type = strtoupper($type);

    if ($type == 'GET')
    {
        $url .= "?" . http_build_query($arguments);
    }

    $curl_request = curl_init($url);

    if ($type == 'POST')
    {
        curl_setopt($curl_request, CURLOPT_POST, 1);
    }
    elseif ($type == 'PUT')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "PUT");
    }
    elseif ($type == 'DELETE')
    {
```

```
        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, $returnHeaders);
    curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

    if (!empty($oauthtoken))
    {
        $token = array("oauth-token: {$oauthtoken}","Content-Type:
application/json");
        curl_setopt($curl_request, CURLOPT_HTTPHEADER, $token);
    }

    if (!empty($arguments) && $type !== 'GET')
    {
        if ($encodeData)
        {
            //encode the arguments as JSON
            $arguments = json_encode($arguments);
        }
        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $arguments);
    }

    $result = curl_exec($curl_request);

    if ($returnHeaders)
    {
        //set headers from response
        list($headers, $content) = explode("\r\n\r\n", $result ,2);
        foreach (explode("\r\n",$headers) as $header)
        {
            header($header);
        }

        //return the nonheader data
        return trim($content);
    }

    curl_close($curl_request);
```

```
    //decode the response from JSON
    $response = json_decode($result);

    return $response;
}

//Login - POST /oauth2/token

$url = $base_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id/secret you created in Admin > OAuth Keys
    "client_id" => "<CustomID>",
    "client_secret" => "<CustomSecret>",
    "username" => $username,
    "password" => $password,
    "platform" => "base"
);

$oauth2_token_response = call($url, '', 'POST',
$oauth2_token_arguments);

//Filter - /<module>/filter POST

$filter_arguments = array(
    "filter" => array(
        array(
            '$or' => array(
                array(
                    //name starts with 'a'
                    "name" => array(
                        '$starts'=>"a",
                    )
                ),
                array(
                    //or name is equal to 'My Account'
                    "name" => 'My Account'
                )
            ),
        ),
    ),
    "max_num" => 2,
    "offset" => 0,
```

```
    "fields" => "name,description",
    "order_by" => "name:DESC",
    "favorites" => false,
    "my_items" => false,
);

$url = $base_url . "/Accounts/filter";

$filter_response = call($url, $oauth2_token_response->access_token,
'POST', $filter_arguments);

echo "<pre>";
print_r($filter_response);
echo "</pre>";

?>
```

Results

Request

```
{
   "filter":[
      {
         "$or":[
            {
               "name":{
                  "$starts":"a"
               }
            },
            {
               "name":"My Account"
            }
         ]
      }
   ],
   "max_num":2,
   "offset":0,
   "fields":"name,description",
   "order_by":"name:DESC",
   "favorites":false,
   "my_items":false
}
```

Response


```
{
   "next_offset":2,
   "records":[
      {
         "id":"d8f05e67-dee3-553d-0040-5342e88f2fd1",
         "name":"Avery Software Co",
         "date_modified":"2014-04-07T13:58:50-04:00",
         "description":"",
         "_acl":{
            "fields":{

            }
         },
         "_module":"Accounts"
      },
      {
         "id":"282c529d-f808-a046-e901-5342e8a4aa2a",
         "name":"AtoZ Co Ltd",
         "date_modified":"2014-04-07T13:58:50-04:00",
         "description":"",
         "_acl":{
            "fields":{

            }
         },
         "_module":"Accounts"
      }
   ]
}
```


Last Modified: 05/16/2016 10:48am

/<module>/ POST

Overview

Examples demonstrating how to create a record using the v10 REST API.

Examples

PHP

```
<?php

$base_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

/**
 * Generic function to make cURL request.
 * @param $url - The URL route to use.
 * @param string $oauthtoken - The oauth token.
 * @param string $type - GET, POST, PUT, DELETE. Defaults to GET.
 * @param array $arguments - Endpoint arguments.
 * @param array $encodeData - Whether or not to JSON encode the data.
 * @param array $returnHeaders - Whether or not to return the headers.
 * @return mixed
 */
function call(
    $url,
    $oauthtoken='',
    $type='GET',
    $arguments=array(),
    $encodeData=true,
    $returnHeaders=false
)
{
    $type = strtoupper($type);

    if ($type == 'GET')
    {
        $url .= "?" . http_build_query($arguments);
    }

    $curl_request = curl_init($url);

    if ($type == 'POST')
    {
        curl_setopt($curl_request, CURLOPT_POST, 1);
    }
    elseif ($type == 'PUT')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "PUT");
    }
    elseif ($type == 'DELETE')
```

```
    {
        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, $returnHeaders);
    curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

    if (!empty($oauthtoken))
    {
        $token = array("oauth-token: {$oauthtoken}","Content-Type:
application/json");
        curl_setopt($curl_request, CURLOPT_HTTPHEADER, $token);
    }

    if (!empty($arguments) && $type !== 'GET')
    {
        if ($encodeData)
        {
            //encode the arguments as JSON
            $arguments = json_encode($arguments);
        }
        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $arguments);
    }

    $result = curl_exec($curl_request);

    if ($returnHeaders)
    {
        //set headers from response
        list($headers, $content) = explode("\r\n\r\n", $result ,2);
        foreach (explode("\r\n",$headers) as $header)
        {
            header($header);
        }

        //return the nonheader data
        return trim($content);
    }

    curl_close($curl_request);
```

```
    //decode the response from JSON
    $response = json_decode($result);

    return $response;
}

//Login - POST /oauth2/token

$url = $base_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id/secret you created in Admin > OAuth Keys
    "client_id" => "<CustomID>",
    "client_secret" => "<CustomSecret>",
    "username" => $username,
    "password" => $password,
    "platform" => "base"
);

$oauth2_token_response = call($url, '', 'POST',
$oauth2_token_arguments);

//Create record - POST /<module>/

$url = $base_url . "/Accounts";

$record_arguments = array(
    "name" => "My Account",
    "description" => "Description of my account",
);

$record_response = call($url, $oauth2_token_response->access_token,
'POST', $record_arguments);

echo "<pre>";
print_r($record_response);
echo "</pre>";

?>
```

Results

Request


```
{
   "name":"My Account",
   "description":"Description of my account"
}
```


Response


```
{
   "my_favorite":false,
   "following":true,
   "id":"33fa8a36-0ca3-4325-7949-534599e62380",
   "name":"My Account",
   "date_entered":"2014-04-09T15:03:55-04:00",
   "date_modified":"2014-04-09T15:03:55-04:00",
   "modified_user_id":"1",
   "modified_by_name":"Administrator",
   "created_by":"1",
   "created_by_name":"Administrator",
   "doc_owner":"",
   "user_favorites":"",
   "description":"Description of my account",
   "deleted":false,
   "assigned_user_id":"",
   "assigned_user_name":"",
   "team_count":"",
   "team_name":[
      {
         "id":1,
         "name":"Global",
         "name_2":"",
         "primary":true
      }
   ],
   "email":[

   ],
   "email1":"",
   "email2":"",
   "invalid_email":"",
   "email_opt_out":"",
   "email_addresses_non_primary":"",
   "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":"",
   "campaign_id":"",
   "campaign_name":"",
   "_acl":{
      "fields":{

      }
   },
   "_module":"Accounts"
}
```


Last Modified: 05/16/2016 10:48am

/<module>/:record DELETE

Overview

Examples demonstrating how to delete a record using the v10 REST API.

Examples

PHP

```
<?php

$base_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

/**
 * Generic function to make cURL request.
 * @param $url - The URL route to use.
 * @param string $oauthtoken - The oauth token.
 * @param string $type - GET, POST, PUT, DELETE. Defaults to GET.
 * @param array $arguments - Endpoint arguments.
 * @param array $encodeData - Whether or not to JSON encode the data.
 * @param array $returnHeaders - Whether or not to return the headers.
 * @return mixed
 */
function call(
    $url,
    $oauthtoken='',
    $type='GET',
    $arguments=array(),
    $encodeData=true,
    $returnHeaders=false
)
{
    $type = strtoupper($type);

    if ($type == 'GET')
    {
        $url .= "?" . http_build_query($arguments);
    }
```

```
    $curl_request = curl_init($url);

    if ($type == 'POST')
    {
        curl_setopt($curl_request, CURLOPT_POST, 1);
    }
    elseif ($type == 'PUT')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "PUT");
    }
    elseif ($type == 'DELETE')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "DELETE");
    }

    curl_setopt($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
    curl_setopt($curl_request, CURLOPT_HEADER, $returnHeaders);
    curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

    if (!empty($oauthtoken))
    {
        $token = array("oauth-token: {$oauthtoken}","Content-Type:
application/json");
        curl_setopt($curl_request, CURLOPT_HTTPHEADER, $token);
    }

    if (!empty($arguments) && $type !== 'GET')
    {
        if ($encodeData)
        {
            //encode the arguments as JSON
            $arguments = json_encode($arguments);
        }
        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $arguments);
    }

    $result = curl_exec($curl_request);

    if ($returnHeaders)
    {
```

```
        //set headers from response
        list($headers, $content) = explode("\r\n\r\n", $result ,2);
        foreach (explode("\r\n",$headers) as $header)
        {
            header($header);
        }

        //return the nonheader data
        return trim($content);
    }

    curl_close($curl_request);

    //decode the response from JSON
    $response = json_decode($result);

    return $response;
}

//Login - POST /oauth2/token

$url = $base_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id/secret you created in Admin > OAuth Keys
    "client_id" => "<CustomID>",
    "client_secret" => "<CustomSecret>",
    "username" => $username,
    "password" => $password,
    "platform" => "base"
);

$oauth2_token_response = call($url, '', 'POST',
$oauth2_token_arguments);

//Delete record - DELETE /<module>/:record

$url = $base_url . "/Accounts/e8f7eb6d-2647-7e57-df30-5342e83c622d";

$record_response = call($url, $oauth2_token_response->access_token,
'DELETE');

echo "<pre>";
```

```
print_r($record_response);
echo "</pre>";

?>
```

Results

Request

```
This endpoint does not accept any request arguments.
```

Response

```
{
  "id":"e8f7eb6d-2647-7e57-df30-5342e83c622d"
}
```

Last Modified: 05/16/2016 10:49am

/<module>/:record/favorite PUT

Overview

Examples demonstrating how to favorite a record using the v10 REST API.

Examples

PHP

```
<?php

$base_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

/**
 * Generic function to make cURL request.
 * @param $url - The URL route to use.
 * @param string $oauthtoken - The oauth token.
```

```
 * @param string $type - GET, POST, PUT, DELETE. Defaults to GET.
 * @param array $arguments - Endpoint arguments.
 * @param array $encodeData - Whether or not to JSON encode the data.
 * @param array $returnHeaders - Whether or not to return the headers.
 * @return mixed
 */
function call(
    $url,
    $oauthtoken='',
    $type='GET',
    $arguments=array(),
    $encodeData=true,
    $returnHeaders=false
)
{
    $type = strtoupper($type);

    if ($type == 'GET')
    {
        $url .= "?" . http_build_query($arguments);
    }

    $curl_request = curl_init($url);

    if ($type == 'POST')
    {
        curl_setopt($curl_request, CURLOPT_POST, 1);
    }
    elseif ($type == 'PUT')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "PUT");
    }
    elseif ($type == 'DELETE')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "DELETE");
    }

    curl_setopt($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
    curl_setopt($curl_request, CURLOPT_HEADER, $returnHeaders);
    curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);
```

```
    if (!empty($oauthtoken))
    {
        $token = array("oauth-token: {$oauthtoken}","Content-Type:
application/json");
        curl_setopt($curl_request, CURLOPT_HTTPHEADER, $token);
    }

    if (!empty($arguments) && $type !== 'GET')
    {
        if ($encodeData)
        {
            //encode the arguments as JSON
            $arguments = json_encode($arguments);
        }
        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $arguments);
    }

    $result = curl_exec($curl_request);

    if ($returnHeaders)
    {
        //set headers from response
        list($headers, $content) = explode("\r\n\r\n", $result ,2);
        foreach (explode("\r\n",$headers) as $header)
        {
            header($header);
        }

        //return the nonheader data
        return trim($content);
    }

    curl_close($curl_request);

    //decode the response from JSON
    $response = json_decode($result);

    return $response;
}

//Login - POST /oauth2/token

$url = $base_url . "/oauth2/token";
```

```
$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id/secret you created in Admin > OAuth Keys
    "client_id" => "<CustomID>",
    "client_secret" => "<CustomSecret>",
    "username" => $username,
    "password" => $password,
    "platform" => "base"
);

$oauth2_token_response = call($url, '', 'POST',
$oauth2_token_arguments);

//Favorite record - PUT /<module>/:record/favorite

$url = $base_url .
"/Accounts/e5d4258b-c9b2-d7d6-bd9b-5342e8badbdd/favorite";

$favorite_response = call($url, $oauth2_token_response->access_token,
'PUT');

echo "<pre>";
print_r($favorite_response);
echo "</pre>";

?>
```

Results

Request

This endpoint does not accept any request arguments.

Response


```
{
  "my_favorite":true,
  "following":true,
  "id":"e5d4258b-c9b2-d7d6-bd9b-5342e8badbdd",
  "name":"DD Furniture 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":"",
"user_favorites":"",
"description":"",
"deleted":false,
"assigned_user_id":"seed_sarah_id",
"assigned_user_name":"Sarah Smith",
"team_count":"",
"team_name":[
   {
      "id":"West",
      "name":"West",
      "name_2":"",
      "primary":true
   }
],
"email":[
   {
      "email_address":"the.qa@example.com",
      "invalid_email":false,
      "opt_out":false,
      "primary_address":true,
      "reply_to_address":false
   },
   {
      "email_address":"support.vegan@example.org",
      "invalid_email":false,
      "opt_out":false,
      "primary_address":false,
      "reply_to_address":false
   }
],
"email1":"the.qa@example.com",
"email2":"support.vegan@example.org",
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"account_type":"Customer",
"industry":"Telecommunications",
```

```
  "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":"CA",
  "billing_address_postalcode":"77125",
  "billing_address_country":"USA",
  "rating":"",
  "phone_office":"(059) 483-7829",
  "phone_alternate":"",
  "website":"www.infokid.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":"CA",
  "shipping_address_postalcode":"77125",
  "shipping_address_country":"USA",
  "parent_id":"",
  "sic_code":"",
  "duns_num":"",
  "parent_name":"",
  "campaign_id":"",
  "campaign_name":"",
  "_acl":{
     "fields":{

     }
  },
  "_module":"Accounts"
}
```


Last Modified: 05/16/2016 10:49am

/<module>/:record/file/:field GET

Overview

Examples demonstrating how to fetch a note attachment using the v10 REST API.

Examples

PHP

```
<?php

$base_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

/**
 * Generic function to make cURL request.
 * @param $url - The URL route to use.
 * @param string $oauthtoken - The oauth token.
 * @param string $type - GET, POST, PUT, DELETE. Defaults to GET.
 * @param array $arguments - Endpoint arguments.
 * @param array $encodeData - Whether or not to JSON encode the data.
 * @param array $returnHeaders - Whether or not to return the headers.
 * @return mixed
 */
function call(
    $url,
    $oauthtoken='',
    $type='GET',
    $arguments=array(),
    $encodeData=true,
    $returnHeaders=false
)
{
    $type = strtoupper($type);

    if ($type == 'GET')
    {
        $url .= "?" . http_build_query($arguments);
    }

    $curl_request = curl_init($url);

    if ($type == 'POST')
```

```
    {
        curl_setopt($curl_request, CURLOPT_POST, 1);
    }
    elseif ($type == 'PUT')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "PUT");
    }
    elseif ($type == 'DELETE')
    {
        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, $returnHeaders);
    curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

    if (!empty($oauthtoken))
    {
        $token = array("oauth-token: {$oauthtoken}","Content-Type:
application/json");
        curl_setopt($curl_request, CURLOPT_HTTPHEADER, $token);
    }

    if (!empty($arguments) && $type !== 'GET')
    {
        if ($encodeData)
        {
            //encode the arguments as JSON
            $arguments = json_encode($arguments);
        }
        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $arguments);
    }

    $result = curl_exec($curl_request);

    if ($returnHeaders)
    {
        //set headers from response
        list($headers, $content) = explode("\r\n\r\n", $result ,2);
        foreach (explode("\r\n",$headers) as $header)
        {
```

```
            header($header);
        }

        //return the nonheader data
        return trim($content);
    }

    curl_close($curl_request);

    //decode the response from JSON
    $response = json_decode($result);

    return $response;
}

//Login - POST /oauth2/token

$url = $base_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id/secret you created in Admin > OAuth Keys
    "client_id" => "<CustomID>",
    "client_secret" => "<CustomSecret>",
    "username" => $username,
    "password" => $password,
    "platform" => "base"
);

$oauth2_token_response = call($url, '', 'POST',
$oauth2_token_arguments);

//Create note record - POST /<module>/:record

$url = $base_url . "/Notes";

$note_arguments = array(
    "name" => "My Note",
);

$note_response = call($url, $oauth2_token_response->access_token,
'POST', $note_arguments);

//Get File - GET /<module>/:id/file/:field
```

```
$url = $base_url .
"/Notes/bd490e66-2ea7-9349-19cf-535569400cca/file/filename";

$result = call($url, $oauth2_token_response->access_token, 'GET',
array(), true, true);

echo $result;

?>
```

Results

Request

```
http://{site_url}/rest/v10/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.
```

Last Modified: 05/16/2016 10:49am

/<module>/:record/file/:field POST

Overview

Examples demonstrating how to attach a file to a note using the v10 REST API.

Examples

PHP

```
<?php

$base_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

/**
 * Generic function to make cURL request.
 * @param $url - The URL route to use.
 * @param string $oauthtoken - The oauth token.
 * @param string $type - GET, POST, PUT, DELETE. Defaults to GET.
 * @param array $arguments - Endpoint arguments.
 * @param array $encodeData - Whether or not to JSON encode the data.
 * @param array $returnHeaders - Whether or not to return the headers.
 * @return mixed
 */
function call(
```

```
    $url,
    $oauthtoken='',
    $type='GET',
    $arguments=array(),
    $encodeData=true,
    $returnHeaders=false
)
{
    $type = strtoupper($type);

    if ($type == 'GET')
    {
        $url .= "?" . http_build_query($arguments);
    }

    $curl_request = curl_init($url);

    if ($type == 'POST')
    {
        curl_setopt($curl_request, CURLOPT_POST, 1);
    }
    elseif ($type == 'PUT')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "PUT");
    }
    elseif ($type == 'DELETE')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "DELETE");
    }

    curl_setopt($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
    curl_setopt($curl_request, CURLOPT_HEADER, $returnHeaders);
    curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

    if (!empty($oauthtoken))
    {
        $token = array("oauth-token: {$oauthtoken}");
        curl_setopt($curl_request, CURLOPT_HTTPHEADER, $token);
    }

    if (!empty($arguments) && $type !== 'GET')
```

```
    {
        if ($encodeData)
        {
            //encode the arguments as JSON
            $arguments = json_encode($arguments);
        }
        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $arguments);
    }

    $result = curl_exec($curl_request);

    if ($returnHeaders)
    {
        //set headers from response
        list($headers, $content) = explode("\r\n\r\n", $result ,2);
        foreach (explode("\r\n",$headers) as $header)
        {
            header($header);
        }

        //return the nonheader data
        return trim($content);
    }

    curl_close($curl_request);

    //decode the response from JSON
    $response = json_decode($result);

    return $response;
}

//Login - POST /oauth2/token

$url = $base_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id/secret you created in Admin > OAuth Keys
    "client_id" => "<CustomID>",
    "client_secret" => "<CustomSecret>",
    "username" => $username,
    "password" => $password,
    "platform" => "base"
```

```
);

$oauth2_token_response = call($url, '', 'POST',
$oauth2_token_arguments);

//Create note record - POST /<module>/:record

$url = $base_url . "/Notes";

$note_arguments = array(
    "name" => "My Note",
);

$note_response = call($url, $oauth2_token_response->access_token,
'POST', $note_arguments);

//Upload File - POST /<module>/:id/file/:field

$url = $base_url . "/Notes/{$note_response->id}/file/filename";;

$file_arguments = array(
    "format" => "sugar-html-json",
    "delete_if_fails" => true,
    "oauth_token" => $oauth2_token_response->access_token,
    'filename' => "@/path/to/file.txt",
);

$file_response = call($url, $oauth2_token_response->access_token,
'POST', $file_arguments, false);

echo "<pre>";
print_r($file_response);
echo "</pre>";

?>
```

Results

Request

```
//Raw Post - not json encoded
Array (
    [format] => sugar-html-json
    [delete_if_fails] => 1
```

```
    [oauth_token] => ad5302bc-9dd1-e88a-3a0a-53553f310038
    [filename] => @/path/to/file.txt
)
```

Response


```
{
  "filename":{
     "content-type":"text\/plain",
     "content-length":13,
     "name":"upload.txt",
     "uri":"http:\/\/<site
url>\/rest\/v10\/Notes\/7b49aebd-8734-9773-8ef1-53553fa369c7\/file\/fi
lename"
  },
  "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"
  }
}
```

Last Modified: 09/26/2016 11:18am

/<module>/:record GET

Overview

Examples demonstrating how to fetch a record using the v10 REST API.

Examples

PHP

```
<?php

$base_url = "http://{site_url}/rest/v10";
$username = "admin";
```

```
$password = "password";

/**
 * Generic function to make cURL request.
 * @param $url - The URL route to use.
 * @param string $oauthtoken - The oauth token.
 * @param string $type - GET, POST, PUT, DELETE. Defaults to GET.
 * @param array $arguments - Endpoint arguments.
 * @param array $encodeData - Whether or not to JSON encode the data.
 * @param array $returnHeaders - Whether or not to return the headers.
 * @return mixed
 */
function call(
    $url,
    $oauthtoken='',
    $type='GET',
    $arguments=array(),
    $encodeData=true,
    $returnHeaders=false
)
{
    $type = strtoupper($type);

    if ($type == 'GET')
    {
        $url .= "?" . http_build_query($arguments);
    }

    $curl_request = curl_init($url);

    if ($type == 'POST')
    {
        curl_setopt($curl_request, CURLOPT_POST, 1);
    }
    elseif ($type == 'PUT')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "PUT");
    }
    elseif ($type == 'DELETE')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "DELETE");
    }

    curl_setopt($curl_request, CURLOPT_HTTP_VERSION,
```

```
CURL_HTTP_VERSION_1_0);
    curl_setopt($curl_request, CURLOPT_HEADER, $returnHeaders);
    curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

    if (!empty($oauthtoken))
    {
        $token = array("oauth-token: {$oauthtoken}","Content-Type:
application/json");
        curl_setopt($curl_request, CURLOPT_HTTPHEADER, $token);
    }

    if (!empty($arguments) && $type !== 'GET')
    {
        if ($encodeData)
        {
            //encode the arguments as JSON
            $arguments = json_encode($arguments);
        }
        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $arguments);
    }

    $result = curl_exec($curl_request);

    if ($returnHeaders)
    {
        //set headers from response
        list($headers, $content) = explode("\r\n\r\n", $result ,2);
        foreach (explode("\r\n",$headers) as $header)
        {
            header($header);
        }

        //return the nonheader data
        return trim($content);
    }

    curl_close($curl_request);

    //decode the response from JSON
    $response = json_decode($result);

    return $response;
```

```
}

//Login - POST /oauth2/token

$url = $base_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id/secret you created in Admin > OAuth Keys
    "client_id" => "<CustomID>",
    "client_secret" => "<CustomSecret>",
    "username" => $username,
    "password" => $password,
    "platform" => "base"
);

$oauth2_token_response = call($url, '', 'POST',
$oauth2_token_arguments);

//Fetch record - GET /<module>/:record

$url = $base_url . "/Accounts/e8f7eb6d-2647-7e57-df30-5342e83c622d";

$record_response = call($url, $oauth2_token_response->access_token,
'GET');

echo "<pre>";
print_r($record_response);
echo "</pre>";

?>
```

Results

Request

This endpoint does not accept any request arguments.

Response


```
{
   "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":"",
"user_favorites":"",
"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 Modified: 05/16/2016 10:50am

/<module>/:record/link/:link GET

Overview

Examples demonstrating how to fetch related records using the v10 REST API.

Examples

PHP

```
<?php

$base_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

/**
 * Generic function to make cURL request.
 * @param $url - The URL route to use.
 * @param string $oauthtoken - The oauth token.
 * @param string $type - GET, POST, PUT, DELETE. Defaults to GET.
 * @param array $arguments - Endpoint arguments.
 * @param array $encodeData - Whether or not to JSON encode the data.
 * @param array $returnHeaders - Whether or not to return the headers.
 * @return mixed
 */
function call(
    $url,
    $oauthtoken='',
    $type='GET',
    $arguments=array(),
    $encodeData=true,
    $returnHeaders=false
)
{
    $type = strtoupper($type);

    if ($type == 'GET')
```

```
    {
        $url .= "?" . http_build_query($arguments);
    }

    $curl_request = curl_init($url);

    if ($type == 'POST')
    {
        curl_setopt($curl_request, CURLOPT_POST, 1);
    }
    elseif ($type == 'PUT')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "PUT");
    }
    elseif ($type == 'DELETE')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "DELETE");
    }

    curl_setopt($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
    curl_setopt($curl_request, CURLOPT_HEADER, $returnHeaders);
    curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

    if (!empty($oauthtoken))
    {
        $token = array("oauth-token: {$oauthtoken}","Content-Type:
application/json");
        curl_setopt($curl_request, CURLOPT_HTTPHEADER, $token);
    }

    if (!empty($arguments) && $type !== 'GET')
    {
        if ($encodeData)
        {
            //encode the arguments as JSON
            $arguments = json_encode($arguments);
        }
        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $arguments);
    }

    $result = curl_exec($curl_request);
```

```
    if ($returnHeaders)
    {
        //set headers from response
        list($headers, $content) = explode("\r\n\r\n", $result ,2);
        foreach (explode("\r\n",$headers) as $header)
        {
            header($header);
        }

        //return the nonheader data
        return trim($content);
    }

    curl_close($curl_request);

    //decode the response from JSON
    $response = json_decode($result);

    return $response;
}

//Login - POST /oauth2/token

$url = $base_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id/secret you created in Admin > OAuth Keys
    "client_id" => "<CustomID>",
    "client_secret" => "<CustomSecret>",
    "username" => $username,
    "password" => $password,
    "platform" => "base"
);

$oauth2_token_response = call($url, '', 'POST',
$oauth2_token_arguments);

//Fetch related records - GET /<module>/:record/link/:link

$url = $base_url .
"/Accounts/d8f05e67-dee3-553d-0040-5342e88f2fd1/link/contacts";
```

```
$link_response = call($url, $oauth2_token_response->access_token,
'GET');

echo "<pre>";
print_r($link_response);
echo "</pre>";

?>
```

Results

Request

```
http://{site_url}/rest/v10/Accounts/d8f05e67-dee3-553d-0040-5342e88f2f
d1/link/contacts
```

Note: GET request arguments are passed in the form of a query string.

Response


```
{
  "next_offset":-1,
  "records":[
     {
        "my_favorite":false,
        "following":false,
        "id":"d102d09b-55f5-b34c-4857-5342e8e37dac",
        "name":"Theresa Norcross",
        "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":"",
        "user_favorites":"",
        "description":"",
        "deleted":false,
        "assigned_user_id":"seed_sally_id",
        "assigned_user_name":"Sally Bronsen",
        "team_count":"",
        "team_name":[
           {
```

```
            "id":"1",
            "name":"Global",
            "name_2":"",
            "primary":false
        },
        {
            "id":"West",
            "name":"West",
            "name_2":"",
            "primary":true
        }
    ],
    "email":[
        {
            "email_address":"vegan98@example.org",
            "primary_address":true,
            "reply_to_address":false,
            "invalid_email":false,
            "opt_out":false
        },
        {
            "email_address":"the.kid.the@example.tv",
            "primary_address":false,
            "reply_to_address":false,
            "invalid_email":false,
            "opt_out":true
        }
    ],
    "email1":"vegan98@example.org",
    "email2":"",
    "invalid_email":false,
    "email_opt_out":false,
    "email_addresses_non_primary":"",
    "salutation":"",
    "first_name":"Theresa",
    "last_name":"Norcross",
    "full_name":"Theresa Norcross",
    "title":"VP Operations",
    "facebook":"",
    "twitter":"",
    "googleplus":"",
    "department":"",
    "do_not_call":false,
    "phone_home":"(357) 632-7168",
```

```
"phone_mobile":"(646) 634-4906",
"phone_work":"(777) 590-9962",
"phone_other":"",
"phone_fax":"",
"primary_address_street":"999 Baker Way",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Ohio",
"primary_address_state":"CA",
"primary_address_postalcode":"75926",
"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":"",
"account_name":"Avery Software Co",
"account_id":"d8f05e67-dee3-553d-0040-5342e88f2fd1",
"dnb_principal_id":"",
"opportunity_role_fields":"",
"opportunity_role_id":"",
"opportunity_role":"",
"reports_to_id":"",
"report_to_name":"",
"birthdate":"",
"portal_name":"TheresaNorcross68",
"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"
   },
   {
      "my_favorite":false,
      "following":false,
      "id":"493d334b-aec3-b94a-9a0d-5342e8692896",
      "name":"Jason Caplinger",
      "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":"",
      "user_favorites":"",
      "description":"",
      "deleted":false,
      "assigned_user_id":"seed_sally_id",
      "assigned_user_name":"Sally Bronsen",
      "team_count":"",
      "team_name":[
         {
            "id":"1",
            "name":"Global",
            "name_2":"",
            "primary":false
         },
         {
            "id":"West",
            "name":"West",
            "name_2":"",
            "primary":true
         }
```

```
],
"email":[
   {
      "email_address":"im.phone.kid@example.it",
      "primary_address":true,
      "reply_to_address":false,
      "invalid_email":false,
      "opt_out":false
   },
   {
      "email_address":"qa.dev.hr@example.tv",
      "primary_address":false,
      "reply_to_address":false,
      "invalid_email":false,
      "opt_out":true
   }
],
"email1":"im.phone.kid@example.it",
"email2":"",
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"",
"salutation":"",
"first_name":"Jason",
"last_name":"Caplinger",
"full_name":"Jason Caplinger",
"title":"Senior Product Manager",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(727) 624-2783",
"phone_mobile":"(589) 280-0981",
"phone_work":"(200) 407-8876",
"phone_other":"",
"phone_fax":"",
"primary_address_street":"9 IBM Path",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Santa Monica",
"primary_address_state":"CA",
"primary_address_postalcode":"46254",
"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":"Partner",
"account_name":"Avery Software Co",
"account_id":"d8f05e67-dee3-553d-0040-5342e88f2fd1",
"dnb_principal_id":"",
"opportunity_role_fields":"",
"opportunity_role_id":"",
"opportunity_role":"",
"reports_to_id":"",
"report_to_name":"",
"birthdate":"",
"portal_name":"JasonCaplinger182",
"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"
   },
   {
      "my_favorite":false,
      "following":false,
      "id":"44f694e5-bff2-2816-e42d-5342e8a62af0",
      "name":"Marie Demar",
      "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":"",
      "user_favorites":"",
      "description":"",
      "deleted":false,
      "assigned_user_id":"seed_sally_id",
      "assigned_user_name":"Sally Bronsen",
      "team_count":"",
      "team_name":[
         {
            "id":"1",
            "name":"Global",
            "name_2":"",
            "primary":false
         },
         {
            "id":"West",
            "name":"West",
            "name_2":"",
            "primary":true
         }
      ],
      "email":[
         {
            "email_address":"qa.dev.vegan@example.co.jp",
            "primary_address":true,
            "reply_to_address":false,
            "invalid_email":false,
            "opt_out":false
         },
         {
            "email_address":"kid25@example.de",
```

```
      "primary_address":false,
      "reply_to_address":false,
      "invalid_email":false,
      "opt_out":true
   }
],
"email1":"qa.dev.vegan@example.co.jp",
"email2":"",
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"",
"salutation":"",
"first_name":"Marie",
"last_name":"Demar",
"full_name":"Marie Demar",
"title":"VP Sales",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(073) 851-6296",
"phone_mobile":"(862) 842-4407",
"phone_work":"(596) 411-3006",
"phone_other":"",
"phone_fax":"",
"primary_address_street":"321 University Ave.",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Salt Lake City",
"primary_address_state":"CA",
"primary_address_postalcode":"10171",
"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":"Avery Software Co",
    "account_id":"d8f05e67-dee3-553d-0040-5342e88f2fd1",
    "dnb_principal_id":"",
    "opportunity_role_fields":"",
    "opportunity_role_id":"",
    "opportunity_role":"",
    "reports_to_id":"",
    "report_to_name":"",
    "birthdate":"",
    "portal_name":"MarieDemar62",
    "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"
},
{
    "my_favorite":false,
    "following":false,
    "id":"2f3a9654-c649-fd65-3f3c-5342e85e2ad8",
    "name":"Sherwood Mcshane",
    "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":"",
"user_favorites":"",
"description":"",
"deleted":false,
"assigned_user_id":"seed_sally_id",
"assigned_user_name":"Sally Bronsen",
"team_count":"",
"team_name":[
   {
      "id":"1",
      "name":"Global",
      "name_2":"",
      "primary":false
   },
   {
      "id":"West",
      "name":"West",
      "name_2":"",
      "primary":true
   }
],
"email":[
   {
      "email_address":"section.hr@example.us",
      "primary_address":true,
      "reply_to_address":false,
      "invalid_email":false,
      "opt_out":false
   },
   {
      "email_address":"dev26@example.biz",
      "primary_address":false,
      "reply_to_address":false,
      "invalid_email":false,
      "opt_out":true
   }
],
"email1":"section.hr@example.us",
"email2":"",
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"",
```

```
"salutation":"",
"first_name":"Sherwood",
"last_name":"Mcshane",
"full_name":"Sherwood Mcshane",
"title":"VP Operations",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(826) 121-9886",
"phone_mobile":"(373) 368-2401",
"phone_work":"(822) 814-0629",
"phone_other":"",
"phone_fax":"",
"primary_address_street":"67321 West Siam St.",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Alabama",
"primary_address_state":"CA",
"primary_address_postalcode":"19115",
"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":"",
"account_name":"Avery Software Co",
"account_id":"d8f05e67-dee3-553d-0040-5342e88f2fd1",
"dnb_principal_id":"",
"opportunity_role_fields":"",
"opportunity_role_id":"",
"opportunity_role":"",
"reports_to_id":"",
"report_to_name":"",
"birthdate":"",
"portal_name":"SherwoodMcshane164",
```

```
        "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 Modified: 08/25/2016 01:32pm

/<module>/:record PUT

Overview

Examples demonstrating how to update a record using the v10 REST API.

Examples

PHP

```
<?php
```

```
$base_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

/**
 * Generic function to make cURL request.
 * @param $url - The URL route to use.
 * @param string $oauthtoken - The oauth token.
 * @param string $type - GET, POST, PUT, DELETE. Defaults to GET.
 * @param array $arguments - Endpoint arguments.
 * @param array $encodeData - Whether or not to JSON encode the data.
 * @param array $returnHeaders - Whether or not to return the headers.
 * @return mixed
 */
function call(
    $url,
    $oauthtoken='',
    $type='GET',
    $arguments=array(),
    $encodeData=true,
    $returnHeaders=false
)
{
    $type = strtoupper($type);

    if ($type == 'GET')
    {
        $url .= "?" . http_build_query($arguments);
    }

    $curl_request = curl_init($url);

    if ($type == 'POST')
    {
        curl_setopt($curl_request, CURLOPT_POST, 1);
    }
    elseif ($type == 'PUT')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "PUT");
    }
    elseif ($type == 'DELETE')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "DELETE");
    }
```

```
    curl_setopt($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
    curl_setopt($curl_request, CURLOPT_HEADER, $returnHeaders);
    curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

    if (!empty($oauthtoken))
    {
        $token = array("oauth-token: {$oauthtoken}","Content-Type:
application/json");
        curl_setopt($curl_request, CURLOPT_HTTPHEADER, $token);
    }

    if (!empty($arguments) && $type !== 'GET')
    {
        if ($encodeData)
        {
            //encode the arguments as JSON
            $arguments = json_encode($arguments);
        }
        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $arguments);
    }

    $result = curl_exec($curl_request);

    if ($returnHeaders)
    {
        //set headers from response
        list($headers, $content) = explode("\r\n\r\n", $result ,2);
        foreach (explode("\r\n",$headers) as $header)
        {
            header($header);
        }

        //return the nonheader data
        return trim($content);
    }

    curl_close($curl_request);

    //decode the response from JSON
    $response = json_decode($result);
```

```
    return $response;
}

//Login - POST /oauth2/token

$url = $base_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id/secret you created in Admin > OAuth Keys
    "client_id" => "<CustomID>",
    "client_secret" => "<CustomSecret>",
    "username" => $username,
    "password" => $password,
    "platform" => "base"
);

$oauth2_token_response = call($url, '', 'POST',
$oauth2_token_arguments);

//Create record - PUT /<module>/:record

$url = $base_url . "/Accounts/dfef0784-38cc-077a-fda5-5342e8228fcf";

$record_arguments = array(
    "name" => "My Upated Account",
);

$record_response = call($url, $oauth2_token_response->access_token,
'PUT', $record_arguments);

echo "<pre>";
print_r($record_response);
echo "</pre>";

?>
```

Results

Request

```
{
   "name":"My Updated Account",
```

```
}
```

Response


```
{
   "my_favorite":false,
   "following":false,
   "id":"dfef0784-38cc-077a-fda5-5342e8228fcf",
   "name":"My Upated Account",
   "date_entered":"2014-04-07T13:58:50-04:00",
   "date_modified":"2014-04-09T15:19:04-04:00",
   "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_max_id",
   "assigned_user_name":"Max Jensen",
   "team_count":"",
   "team_name":[
      {
         "id":"East",
         "name":"East",
         "name_2":"",
         "primary":false
      },
      {
         "id":"West",
         "name":"West",
         "name_2":"",
         "primary":true
      }
   ],
   "email":[
      {
         "email_address":"the91@example.info",
         "invalid_email":false,
         "opt_out":false,
         "primary_address":true,
         "reply_to_address":false
      },
```

```
    {
       "email_address":"sales50@example.co.uk",
       "invalid_email":false,
       "opt_out":false,
       "primary_address":false,
       "reply_to_address":false
    }
],
"email1":"the91@example.info",
"email2":"sales50@example.co.uk",
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"account_type":"Customer",
"industry":"Apparel",
"annual_revenue":"",
"phone_fax":"",
"billing_address_street":"321 University Ave.",
"billing_address_street_2":"",
"billing_address_street_3":"",
"billing_address_street_4":"",
"billing_address_city":"Ohio",
"billing_address_state":"CA",
"billing_address_postalcode":"94089",
"billing_address_country":"USA",
"rating":"",
"phone_office":"(244) 255-8563",
"phone_alternate":"",
"website":"www.sectionvegan.tw",
"ownership":"",
"employees":"",
"ticker_symbol":"",
"shipping_address_street":"321 University Ave.",
"shipping_address_street_2":"",
"shipping_address_street_3":"",
"shipping_address_street_4":"",
"shipping_address_city":"Ohio",
"shipping_address_state":"CA",
"shipping_address_postalcode":"94089",
"shipping_address_country":"USA",
"parent_id":"",
```

```
   "sic_code":"",
   "duns_num":"",
   "parent_name":"",
   "campaign_id":"",
   "campaign_name":"",
   "_acl":{
      "fields":{

      }
   },
   "_module":"Accounts"
}
```

Last Modified: 05/16/2016 10:50am

/<module>/:record/subscribe POST

Overview

Examples demonstrating how to follow a record using the v10 REST API.

Examples

PHP

```
<?php

$base_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

/**
 * Generic function to make cURL request.
 * @param $url - The URL route to use.
 * @param string $oauthtoken - The oauth token.
 * @param string $type - GET, POST, PUT, DELETE. Defaults to GET.
 * @param array $arguments - Endpoint arguments.
 * @param array $encodeData - Whether or not to JSON encode the data.
 * @param array $returnHeaders - Whether or not to return the headers.
 * @return mixed
 */
```

```
function call(
    $url,
    $oauthtoken='',
    $type='GET',
    $arguments=array(),
    $encodeData=true,
    $returnHeaders=false
)
{
    $type = strtoupper($type);

    if ($type == 'GET')
    {
        $url .= "?" . http_build_query($arguments);
    }

    $curl_request = curl_init($url);

    if ($type == 'POST')
    {
        curl_setopt($curl_request, CURLOPT_POST, 1);
    }
    elseif ($type == 'PUT')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "PUT");
    }
    elseif ($type == 'DELETE')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "DELETE");
    }

    curl_setopt($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
    curl_setopt($curl_request, CURLOPT_HEADER, $returnHeaders);
    curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

    if (!empty($oauthtoken))
    {
        $token = array("oauth-token: {$oauthtoken}","Content-Type:
application/json");
        curl_setopt($curl_request, CURLOPT_HTTPHEADER, $token);
    }
```

```
    if (!empty($arguments) && $type !== 'GET')
    {
        if ($encodeData)
        {
            //encode the arguments as JSON
            $arguments = json_encode($arguments);
        }
        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $arguments);
    }

    $result = curl_exec($curl_request);

    if ($returnHeaders)
    {
        //set headers from response
        list($headers, $content) = explode("\r\n\r\n", $result ,2);
        foreach (explode("\r\n",$headers) as $header)
        {
            header($header);
        }

        //return the nonheader data
        return trim($content);
    }

    curl_close($curl_request);

    //decode the response from JSON
    $response = json_decode($result);

    return $response;
}

//Login - POST /oauth2/token

$url = $base_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id/secret you created in Admin > OAuth Keys
    "client_id" => "<CustomID>",
    "client_secret" => "<CustomSecret>",
    "username" => $username,
```

```
    "password" => $password,
    "platform" => "base"
);

$oauth2_token_response = call($url, '', 'POST',
$oauth2_token_arguments);

//Create note record - POST /<module>/:record

$url = $base_url . "/Notes";

$note_arguments = array(
    "name" => "My Note",
);

$note_response = call($url, $oauth2_token_response->access_token,
'POST', $note_arguments);

//Subscribe to record - POST /<module>/:record/subscribe

$url = $base_url .
"/Accounts/e1eefd87-50df-0f64-3c58-5342e73b36f0/subscribe";

$subscribe_response = call($url, $oauth2_token_response->access_token,
'POST');

echo "<pre>";
print_r($subscribe_response);
echo "</pre>";

?>
```

Results

Request

This endpoint does not accept any request arguments.

Response

```
"b468a2e7-0c52-ca4e-6adc-53567b42e022"
```

Last Modified: 05/16/2016 10:51am

/<module>/:record/unfavorite PUT

Overview

Examples demonstrating how to unfavorite a record using the v10 REST API.

Examples

PHP

```
<?php

$base_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

/**
 * Generic function to make cURL request.
 * @param $url - The URL route to use.
 * @param string $oauthtoken - The oauth token.
 * @param string $type - GET, POST, PUT, DELETE. Defaults to GET.
 * @param array $arguments - Endpoint arguments.
 * @param array $encodeData - Whether or not to JSON encode the data.
 * @param array $returnHeaders - Whether or not to return the headers.
 * @return mixed
 */
function call(
    $url,
    $oauthtoken='',
    $type='GET',
    $arguments=array(),
    $encodeData=true,
    $returnHeaders=false
)
{
    $type = strtoupper($type);

    if ($type == 'GET')
    {
        $url .= "?" . http_build_query($arguments);
    }
```

```
    $curl_request = curl_init($url);

    if ($type == 'POST')
    {
        curl_setopt($curl_request, CURLOPT_POST, 1);
    }
    elseif ($type == 'PUT')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "PUT");
    }
    elseif ($type == 'DELETE')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "DELETE");
    }

    curl_setopt($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
    curl_setopt($curl_request, CURLOPT_HEADER, $returnHeaders);
    curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

    if (!empty($oauthtoken))
    {
        $token = array("oauth-token: {$oauthtoken}","Content-Type:
application/json");
        curl_setopt($curl_request, CURLOPT_HTTPHEADER, $token);
    }

    if (!empty($arguments) && $type !== 'GET')
    {
        if ($encodeData)
        {
            //encode the arguments as JSON
            $arguments = json_encode($arguments);
        }
        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $arguments);
    }

    $result = curl_exec($curl_request);

    if ($returnHeaders)
    {
```

```
        //set headers from response
        list($headers, $content) = explode("\r\n\r\n", $result ,2);
        foreach (explode("\r\n",$headers) as $header)
        {
            header($header);
        }

        //return the nonheader data
        return trim($content);
    }

    curl_close($curl_request);

    //decode the response from JSON
    $response = json_decode($result);

    return $response;
}

//Login - POST /oauth2/token

$url = $base_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id/secret you created in Admin > OAuth Keys
    "client_id" => "<CustomID>",
    "client_secret" => "<CustomSecret>",
    "username" => $username,
    "password" => $password,
    "platform" => "base"
);

$oauth2_token_response = call($url, '', 'POST',
$oauth2_token_arguments);

//Unfavorite record - PUT /<module>/:record/unfavorite

$url = $base_url .
"/Accounts/e5d4258b-c9b2-d7d6-bd9b-5342e8badbdd/unfavorite";

$unfavorite_response = call($url,
$oauth2_token_response->access_token, 'PUT');
```

```
echo "<pre>";
print_r($unfavorite_response);
echo "</pre>";

?>
```

Results

Request

This endpoint does not accept any request arguments.

Response


```
{
  "my_favorite":false,
  "following":true,
  "id":"e5d4258b-c9b2-d7d6-bd9b-5342e8badbdd",
  "name":"DD Furniture 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":"",
  "user_favorites":"",
  "description":"",
  "deleted":false,
  "assigned_user_id":"seed_sarah_id",
  "assigned_user_name":"Sarah Smith",
  "team_count":"",
  "team_name":[
     {
        "id":"West",
        "name":"West",
        "name_2":"",
        "primary":true
     }
  ],
  "email":[
     {
        "email_address":"the.qa@example.com",
```

```
      "invalid_email":false,
      "opt_out":false,
      "primary_address":true,
      "reply_to_address":false
   },
   {
      "email_address":"support.vegan@example.org",
      "invalid_email":false,
      "opt_out":false,
      "primary_address":false,
      "reply_to_address":false
   }
],
"email1":"the.qa@example.com",
"email2":"support.vegan@example.org",
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"account_type":"Customer",
"industry":"Telecommunications",
"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":"CA",
"billing_address_postalcode":"77125",
"billing_address_country":"USA",
"rating":"",
"phone_office":"(059) 483-7829",
"phone_alternate":"",
"website":"www.infokid.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":"CA",
  "shipping_address_postalcode":"77125",
  "shipping_address_country":"USA",
  "parent_id":"",
  "sic_code":"",
  "duns_num":"",
  "parent_name":"",
  "campaign_id":"",
  "campaign_name":"",
  "_acl":{
     "fields":{

     }
  },
  "_module":"Accounts"
}
```

Last Modified: 05/16/2016 10:51am

/<module>/:record/unsubscribe DELETE

Overview

Examples demonstrating how to unfollow a record using the v10 REST API.

Examples

PHP

```
<?php

$base_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

/**
 * Generic function to make cURL request.
 * @param $url - The URL route to use.
 * @param string $oauthtoken - The oauth token.
 * @param string $type - GET, POST, PUT, DELETE. Defaults to GET.
```

```
 * @param array $arguments - Endpoint arguments.
 * @param array $encodeData - Whether or not to JSON encode the data.
 * @param array $returnHeaders - Whether or not to return the headers.
 * @return mixed
 */
function call(
    $url,
    $oauthtoken='',
    $type='GET',
    $arguments=array(),
    $encodeData=true,
    $returnHeaders=false
)
{
    $type = strtoupper($type);

    if ($type == 'GET')
    {
        $url .= "?" . http_build_query($arguments);
    }

    $curl_request = curl_init($url);

    if ($type == 'POST')
    {
        curl_setopt($curl_request, CURLOPT_POST, 1);
    }
    elseif ($type == 'PUT')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "PUT");
    }
    elseif ($type == 'DELETE')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "DELETE");
    }

    curl_setopt($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
    curl_setopt($curl_request, CURLOPT_HEADER, $returnHeaders);
    curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

    if (!empty($oauthtoken))
```

```
    {
        $token = array("oauth-token: {$oauthtoken}","Content-Type:
application/json");
        curl_setopt($curl_request, CURLOPT_HTTPHEADER, $token);
    }

    if (!empty($arguments) && $type !== 'GET')
    {
        if ($encodeData)
        {
            //encode the arguments as JSON
            $arguments = json_encode($arguments);
        }
        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $arguments);
    }

    $result = curl_exec($curl_request);

    if ($returnHeaders)
    {
        //set headers from response
        list($headers, $content) = explode("\r\n\r\n", $result ,2);
        foreach (explode("\r\n",$headers) as $header)
        {
            header($header);
        }

        //return the nonheader data
        return trim($content);
    }

    curl_close($curl_request);

    //decode the response from JSON
    $response = json_decode($result);

    return $response;
}

//Login - POST /oauth2/token

$url = $base_url . "/oauth2/token";

$oauth2_token_arguments = array(
```

```
    "grant_type" => "password",
    //client id/secret you created in Admin > OAuth Keys
    "client_id" => "<CustomID>",
    "client_secret" => "<CustomSecret>",
    "username" => $username,
    "password" => $password,
    "platform" => "base"
);

$oauth2_token_response = call($url, '', 'POST',
$oauth2_token_arguments);

//Create note record - POST /<module>/:record

$url = $base_url . "/Notes";

$note_arguments = array(
    "name" => "My Note",
);

$note_response = call($url, $oauth2_token_response->access_token,
'POST', $note_arguments);

//Subscribe to record - POST /<module>/:record/unsubscribe

$url = $base_url .
"/Accounts/e1eefd87-50df-0f64-3c58-5342e73b36f0/unsubscribe";

$unsubscribe_response = call($url,
$oauth2_token_response->access_token, 'DELETE');

echo "<pre>";
print_r($unsubscribe_response);
echo "</pre>";

?>
```

Results

Request

```
No arguments accepted.
```

Response

```
true
```

Last Modified: 05/16/2016 10:52am

/mostactiveusers POST

Overview

Examples demonstrating how to fetch the most active users for meetings, calls, inbound emails, and outbound emails using the v10 REST API.

Examples

PHP

```
<?php

$base_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

/**
 * Generic function to make cURL request.
 * @param $url - The URL route to use.
 * @param string $oauthtoken - The oauth token.
 * @param string $type - GET, POST, PUT, DELETE. Defaults to GET.
 * @param array $arguments - Endpoint arguments.
 * @param array $encodeData - Whether or not to JSON encode the data.
 * @param array $returnHeaders - Whether or not to return the headers.
 * @return mixed
 */
function call(
    $url,
    $oauthtoken='',
    $type='GET',
    $arguments=array(),
    $encodeData=true,
    $returnHeaders=false
)
{
```

```
    $type = strtoupper($type);

    if ($type == 'GET')
    {
        $url .= "?" . http_build_query($arguments);
    }

    $curl_request = curl_init($url);

    if ($type == 'POST')
    {
        curl_setopt($curl_request, CURLOPT_POST, 1);
    }
    elseif ($type == 'PUT')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "PUT");
    }
    elseif ($type == 'DELETE')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "DELETE");
    }

    curl_setopt($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
    curl_setopt($curl_request, CURLOPT_HEADER, $returnHeaders);
    curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

    if (!empty($oauthtoken))
    {
        $token = array("oauth-token: {$oauthtoken}","Content-Type:
application/json");
        curl_setopt($curl_request, CURLOPT_HTTPHEADER, $token);
    }

    if (!empty($arguments) && $type !== 'GET')
    {
        if ($encodeData)
        {
            //encode the arguments as JSON
            $arguments = json_encode($arguments);
        }
        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $arguments);
```

```
    }

    $result = curl_exec($curl_request);

    if ($returnHeaders)
    {
        //set headers from response
        list($headers, $content) = explode("\r\n\r\n", $result ,2);
        foreach (explode("\r\n",$headers) as $header)
        {
            header($header);
        }

        //return the nonheader data
        return trim($content);
    }

    curl_close($curl_request);

    //decode the response from JSON
    $response = json_decode($result);

    return $response;
}

//Login - POST /oauth2/token

$url = $base_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id/secret you created in Admin > OAuth Keys
    "client_id" => "<CustomID>",
    "client_secret" => "<CustomSecret>",
    "username" => $username,
    "password" => $password,
    "platform" => "base"
);

$oauth2_token_response = call($url, '', 'POST',
$oauth2_token_arguments);

//Fetch users - GET /mostactiveusers
```

```
$most_active_arguments = array(
    "days" => 30,
);

$url = $base_url . "/mostactiveusers";

$most_active_response = call($url,
$oauth2_token_response->access_token, 'GET', $most_active_arguments);

echo "<pre>";
print_r($most_active_response);
echo "</pre>";

?>
```

Results

Request

```
http://{site_url}/rest/v10/mostactiveusers?days=30
```

Note: GET endpoint arguments are passed in the form of a query string.

Response


```
{
   "meetings":{
      "user_id":"seed_sarah_id",
      "count":"18",
      "first_name":"Sarah",
      "last_name":"Smith"
   },
   "calls":{
      "user_id":"seed_will_id",
      "count":"8",
      "first_name":"Will",
      "last_name":"Westin"
   },
   "inbound_emails":{
      "user_id":"seed_will_id",
      "count":"19",
      "first_name":"Will",
```

```
      "last_name":"Westin"
   },
   "outbound_emails":{
      "user_id":"seed_sarah_id",
      "count":"23",
      "first_name":"Sarah",
      "last_name":"Smith"
   }
}
```

Last Modified: 05/16/2016 10:52am

/oauth2/logout POST

Overview

Examples demonstrating how to logout using the v10 REST API.

Examples

PHP

```
<?php

$base_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

/**
 * Generic function to make cURL request.
 * @param $url - The URL route to use.
 * @param string $oauthtoken - The oauth token.
 * @param string $type - GET, POST, PUT, DELETE. Defaults to GET.
 * @param array $arguments - Endpoint arguments.
 * @param array $encodeData - Whether or not to JSON encode the data.
 * @param array $returnHeaders - Whether or not to return the headers.
 * @return mixed
 */
function call(
    $url,
    $oauthtoken='',
```

```
    $type='GET',
    $arguments=array(),
    $encodeData=true,
    $returnHeaders=false
)
{
    $type = strtoupper($type);

    if ($type == 'GET')
    {
        $url .= "?" . http_build_query($arguments);
    }

    $curl_request = curl_init($url);

    if ($type == 'POST')
    {
        curl_setopt($curl_request, CURLOPT_POST, 1);
    }
    elseif ($type == 'PUT')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "PUT");
    }
    elseif ($type == 'DELETE')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "DELETE");
    }

    curl_setopt($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
    curl_setopt($curl_request, CURLOPT_HEADER, $returnHeaders);
    curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

    if (!empty($oauthtoken))
    {
        $token = array("oauth-token: {$oauthtoken}","Content-Type:
application/json");
        curl_setopt($curl_request, CURLOPT_HTTPHEADER, $token);
    }

    if (!empty($arguments) && $type !== 'GET')
    {
```

```
        if ($encodeData)
        {
            //encode the arguments as JSON
            $arguments = json_encode($arguments);
        }
        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $arguments);
    }

    $result = curl_exec($curl_request);

    if ($returnHeaders)
    {
        //set headers from response
        list($headers, $content) = explode("\r\n\r\n", $result ,2);
        foreach (explode("\r\n",$headers) as $header)
        {
            header($header);
        }

        //return the nonheader data
        return trim($content);
    }

    curl_close($curl_request);

    //decode the response from JSON
    $response = json_decode($result);

    return $response;
}

//Login - POST /oauth2/token

$url = $base_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id/secret you created in Admin > OAuth Keys
    "client_id" => "<CustomID>",
    "client_secret" => "<CustomSecret>",
    "username" => $username,
    "password" => $password,
    "platform" => "base"
);
```

```
$oauth2_token_response = call($url, '', 'POST',
$oauth2_token_arguments);

//Logout - POST /oauth2/logout

$url = $base_url . "/oauth2/logout";

$oauth2_logout_response = call($url,
$oauth2_token_response->access_token, 'POST');

echo "<pre>";
print_r($oauth2_logout_response);
echo "</pre>";

?>
```

Results

Request

```
This endpoint does not accept any request arguments.
```

Response

```
{
    "success":true
}
```

Last Modified: 05/16/2016 10:52am

/oauth2/token POST

Overview

Examples demonstrating how to log in and retrieve a session key with the v10 REST API.

Examples

PHP

```
<?php

$base_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

/**
 * Generic function to make cURL request.
 * @param $url - The URL route to use.
 * @param string $oauthtoken - The oauth token.
 * @param string $type - GET, POST, PUT, DELETE. Defaults to GET.
 * @param array $arguments - Endpoint arguments.
 * @param array $encodeData - Whether or not to JSON encode the data.
 * @param array $returnHeaders - Whether or not to return the headers.
 * @return mixed
 */
function call(
    $url,
    $oauthtoken='',
    $type='GET',
    $arguments=array(),
    $encodeData=true,
    $returnHeaders=false
)
{
    $type = strtoupper($type);

    if ($type == 'GET')
    {
        $url .= "?" . http_build_query($arguments);
    }

    $curl_request = curl_init($url);

    if ($type == 'POST')
    {
        curl_setopt($curl_request, CURLOPT_POST, 1);
    }
    elseif ($type == 'PUT')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "PUT");
```

```
    }
    elseif ($type == 'DELETE')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "DELETE");
    }

    curl_setopt($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
    curl_setopt($curl_request, CURLOPT_HEADER, $returnHeaders);
    curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

    if (!empty($oauthtoken))
    {
        $token = array("oauth-token: {$oauthtoken}","Content-Type:
application/json");
        curl_setopt($curl_request, CURLOPT_HTTPHEADER, $token);
    }

    if (!empty($arguments) && $type !== 'GET')
    {
        if ($encodeData)
        {
            //encode the arguments as JSON
            $arguments = json_encode($arguments);
        }
        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $arguments);
    }

    $result = curl_exec($curl_request);

    if ($returnHeaders)
    {
        //set headers from response
        list($headers, $content) = explode("\r\n\r\n", $result ,2);
        foreach (explode("\r\n",$headers) as $header)
        {
            header($header);
        }

        //return the nonheader data
        return trim($content);
    }
```

```
    curl_close($curl_request);

    //decode the response from JSON
    $response = json_decode($result);

    return $response;
}

//Login - POST /oauth2/token

$url = $base_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id/secret you created in Admin > OAuth Keys
    "client_id" => "<CustomID>",
    "client_secret" => "<CustomSecret>",
    "username" => $username,
    "password" => $password,
    "platform" => "base"
);

$oauth2_token_response = call($url, '', 'POST',
$oauth2_token_arguments);

echo "<pre>";
print_r($oauth2_token_response);
echo "</pre>";

?>
```

Results

Request

```
{
   "grant_type":"password",
   "client_id":"sugar",
   "client_secret":"",
   "username":"admin",
   "password":"password",
   "platform":"base"
}
```

Response

```
{
   "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"
}
```

Last Modified: 05/16/2016 10:52am

/ping GET

Overview

Examples demonstrating how use ping with the v10 REST API.

Examples

PHP

```
<?php

$base_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

/**
 * Generic function to make cURL request.
 * @param $url - The URL route to use.
 * @param string $oauthtoken - The oauth token.
 * @param string $type - GET, POST, PUT, DELETE. Defaults to GET.
 * @param array $arguments - Endpoint arguments.
 * @param array $encodeData - Whether or not to JSON encode the data.
 * @param array $returnHeaders - Whether or not to return the headers.
 * @return mixed
 */
```

```
function call(
    $url,
    $oauthtoken='',
    $type='GET',
    $arguments=array(),
    $encodeData=true,
    $returnHeaders=false
)
{
    $type = strtoupper($type);

    if ($type == 'GET')
    {
        $url .= "?" . http_build_query($arguments);
    }

    $curl_request = curl_init($url);

    if ($type == 'POST')
    {
        curl_setopt($curl_request, CURLOPT_POST, 1);
    }
    elseif ($type == 'PUT')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "PUT");
    }
    elseif ($type == 'DELETE')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "DELETE");
    }

    curl_setopt($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
    curl_setopt($curl_request, CURLOPT_HEADER, $returnHeaders);
    curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

    if (!empty($oauthtoken))
    {
        $token = array("oauth-token: {$oauthtoken}","Content-Type:
application/json");
        curl_setopt($curl_request, CURLOPT_HTTPHEADER, $token);
    }
```

```
    if (!empty($arguments) && $type !== 'GET')
    {
        if ($encodeData)
        {
            //encode the arguments as JSON
            $arguments = json_encode($arguments);
        }
        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $arguments);
    }

    $result = curl_exec($curl_request);

    if ($returnHeaders)
    {
        //set headers from response
        list($headers, $content) = explode("\r\n\r\n", $result ,2);
        foreach (explode("\r\n",$headers) as $header)
        {
            header($header);
        }

        //return the nonheader data
        return trim($content);
    }

    curl_close($curl_request);

    //decode the response from JSON
    $response = json_decode($result);

    return $response;
}

//Login - POST /oauth2/token

$url = $base_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id/secret you created in Admin > OAuth Keys
    "client_id" => "<CustomID>",
    "client_secret" => "<CustomSecret>",
    "username" => $username,
```

```
    "password" => $password,
    "platform" => "base"
);

$oauth2_token_response = call($url, '', 'POST',
$oauth2_token_arguments);

//Ping - GET /ping

$url = $base_url . "/ping";

$ping_response = call($url, $oauth2_token_response->access_token,
'GET');

echo "<pre>";
print_r($ping_response);
echo "</pre>";

?>
```

Results

Request

```
This endpoint does not accept any request arguments.
```

Response

```
"pong"
```

Last Modified: 05/16/2016 10:53am

/ping/whattimeisit GET

Overview

Examples demonstrating how to fetch the current users time with the v10 REST API.

Examples

PHP

```
<?php

$base_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

/**
 * Generic function to make cURL request.
 * @param $url - The URL route to use.
 * @param string $oauthtoken - The oauth token.
 * @param string $type - GET, POST, PUT, DELETE. Defaults to GET.
 * @param array $arguments - Endpoint arguments.
 * @param array $encodeData - Whether or not to JSON encode the data.
 * @param array $returnHeaders - Whether or not to return the headers.
 * @return mixed
 */
function call(
    $url,
    $oauthtoken='',
    $type='GET',
    $arguments=array(),
    $encodeData=true,
    $returnHeaders=false
)
{
    $type = strtoupper($type);

    if ($type == 'GET')
    {
        $url .= "?" . http_build_query($arguments);
    }

    $curl_request = curl_init($url);

    if ($type == 'POST')
    {
        curl_setopt($curl_request, CURLOPT_POST, 1);
    }
    elseif ($type == 'PUT')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "PUT");
```

```
    }
    elseif ($type == 'DELETE')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "DELETE");
    }

    curl_setopt($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
    curl_setopt($curl_request, CURLOPT_HEADER, $returnHeaders);
    curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

    if (!empty($oauthtoken))
    {
        $token = array("oauth-token: {$oauthtoken}","Content-Type:
application/json");
        curl_setopt($curl_request, CURLOPT_HTTPHEADER, $token);
    }

    if (!empty($arguments) && $type !== 'GET')
    {
        if ($encodeData)
        {
            //encode the arguments as JSON
            $arguments = json_encode($arguments);
        }
        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $arguments);
    }

    $result = curl_exec($curl_request);

    if ($returnHeaders)
    {
        //set headers from response
        list($headers, $content) = explode("\r\n\r\n", $result ,2);
        foreach (explode("\r\n",$headers) as $header)
        {
            header($header);
        }

        //return the nonheader data
        return trim($content);
    }
```

```
    curl_close($curl_request);

    //decode the response from JSON
    $response = json_decode($result);

    return $response;
}

//Login - POST /oauth2/token

$url = $base_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id/secret you created in Admin > OAuth Keys
    "client_id" => "<CustomID>",
    "client_secret" => "<CustomSecret>",
    "username" => $username,
    "password" => $password,
    "platform" => "base"
);

$oauth2_token_response = call($url, '', 'POST',
$oauth2_token_arguments);

//Ping time - GET /ping/whattimeisit

$url = $base_url . "/ping/whattimeisit";

$ping_whattimeisit_response = call($url,
$oauth2_token_response->access_token, 'GET');

echo "<pre>";
print_r($ping_whattimeisit_response);
echo "</pre>";

?>
```

Results

Request

This endpoint does not accept any request arguments.

Response

```
"2014-04-08T14:59:13-04:00"
```

Last Modified: 05/16/2016 10:53am

/recent GET

Overview

Examples demonstrating how to fetch recently viewed records using the v10 REST API.

Examples

PHP

```
<?php

$base_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

/**
 * Generic function to make cURL request.
 * @param $url - The URL route to use.
 * @param string $oauthtoken - The oauth token.
 * @param string $type - GET, POST, PUT, DELETE. Defaults to GET.
 * @param array $arguments - Endpoint arguments.
 * @param array $encodeData - Whether or not to JSON encode the data.
 * @param array $returnHeaders - Whether or not to return the headers.
 * @return mixed
 */
function call(
    $url,
    $oauthtoken='',
    $type='GET',
    $arguments=array(),
    $encodeData=true,
```

```
    $returnHeaders=false
)
{
    $type = strtoupper($type);

    if ($type == 'GET')
    {
        $url .= "?" . http_build_query($arguments);
    }

    $curl_request = curl_init($url);

    if ($type == 'POST')
    {
        curl_setopt($curl_request, CURLOPT_POST, 1);
    }
    elseif ($type == 'PUT')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "PUT");
    }
    elseif ($type == 'DELETE')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "DELETE");
    }

    curl_setopt($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
    curl_setopt($curl_request, CURLOPT_HEADER, $returnHeaders);
    curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

    if (!empty($oauthtoken))
    {
        $token = array("oauth-token: {$oauthtoken}","Content-Type:
application/json");
        curl_setopt($curl_request, CURLOPT_HTTPHEADER, $token);
    }

    if (!empty($arguments) && $type !== 'GET')
    {
        if ($encodeData)
        {
            //encode the arguments as JSON
```

```
            $arguments = json_encode($arguments);
        }
        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $arguments);
    }

    $result = curl_exec($curl_request);

    if ($returnHeaders)
    {
        //set headers from response
        list($headers, $content) = explode("\r\n\r\n", $result ,2);
        foreach (explode("\r\n",$headers) as $header)
        {
            header($header);
        }

        //return the nonheader data
        return trim($content);
    }

    curl_close($curl_request);

    //decode the response from JSON
    $response = json_decode($result);

    return $response;
}

//Login - POST /oauth2/token

$url = $base_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id/secret you created in Admin > OAuth Keys
    "client_id" => "<CustomID>",
    "client_secret" => "<CustomSecret>",
    "username" => $username,
    "password" => $password,
    "platform" => "base"
);

$oauth2_token_response = call($url, '', 'POST',
$oauth2_token_arguments);
```

```
//Recently viewed records - GET /recent

$recent_arguments = array(
    "module_list" => 'Accounts,Contacts'
);

$url = $base_url . "/recent";

$recent_response = call($url, $oauth2_token_response->access_token,
'GET', $recent_arguments);

echo "<pre>";
print_r($recent_response);
echo "</pre>";

?>
```

Results

Request

```
http://{site_url}/rest/v10/recent?module_list=Accounts%2CContacts
```

Note: GET request arguments are passed in the form of a query string.

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":"",
"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":"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":"",
      "user_favorites":"",
      "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"
         }
      ]
}
```


Last Modified: 05/16/2016 10:55am

/search GET

Overview

Examples demonstrating how to globally search records using the v10 REST API.

Examples

PHP

```php
<?php

$base_url = "http://{site url}/rest/v10";
$username = "admin";
$password = "password";

/**
 * Generic function to make cURL request.
 * @param $url - The URL route to use.
 * @param string $oauthtoken - The oauth token.
 * @param string $type - GET, POST, PUT, DELETE. Defaults to GET.
 * @param array $arguments - Endpoint arguments.
 * @param array $encodeData - Whether or not to JSON encode the data.
 * @param array $returnHeaders - Whether or not to return the headers.
 * @return mixed
 */
function call(
    $url,
    $oauthtoken='',
    $type='GET',
    $arguments=array(),
    $encodeData=true,
    $returnHeaders=false
)
{
    $type = strtoupper($type);

    if ($type == 'GET')
    {
        $url .= "?" . http_build_query($arguments);
    }

    $curl_request = curl_init($url);

    if ($type == 'POST')
    {
        curl_setopt($curl_request, CURLOPT_POST, 1);
    }
    elseif ($type == 'PUT')
    {
        curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "PUT");
    }
    elseif ($type == 'DELETE')
    {
```

```
        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, $returnHeaders);
    curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);

    if (!empty($oauthtoken))
    {
        $token = array("oauth-token: {$oauthtoken}","Content-Type:
application/json");
        curl_setopt($curl_request, CURLOPT_HTTPHEADER, $token);
    }

    if (!empty($arguments) && $type !== 'GET')
    {
        if ($encodeData)
        {
            //encode the arguments as JSON
            $arguments = json_encode($arguments);
        }
        curl_setopt($curl_request, CURLOPT_POSTFIELDS, $arguments);
    }

    $result = curl_exec($curl_request);

    if ($returnHeaders)
    {
        //set headers from response
        list($headers, $content) = explode("\r\n\r\n", $result ,2);
        foreach (explode("\r\n",$headers) as $header)
        {
            header($header);
        }

        //return the nonheader data
        return trim($content);
    }

    curl_close($curl_request);
```

```
    //decode the response from JSON
    $response = json_decode($result);

    return $response;
}

//Login - POST /oauth2/token

$url = $base_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    "client_id" => "sugar",
    "client_secret" => "",
    "username" => $username,
    "password" => $password,
    "platform" => "base"
);

$oauth2_token_response = call($url, '', 'POST',
$oauth2_token_arguments);

//Search Emails - GET /search

$search_arguments = array(
    "q" => "jsmith@sugar.crm",
    "max_num" => 2,
    "offset" => 0,
    "fields" => "",
    "order_by" => "",
    "favorites" => false,
    "my_items" => false,
);

$url = $base_url . "/search";

$search_response = call($url, $oauth2_token_response->access_token,
'GET', $search_arguments);

echo "<pre>";
print_r($search_response);
echo "</pre>";

?>
```

Results

Request

```
http://{site_url}/rest/v10/search?q=jsmith%40sugar.crm&max_num=2&offset=0&fields=&order_by=&favorites=0&my_items=0
```

Note: GET endpoint arguments are passed in the form of a query string.

Response


```
{
   "next_offset":-1,
   "records":[
      {
         "id":"ed5031be-4392-8de5-896e-533b11c814da",
         "name":"Tortoise Corp",
         "date_entered":"2014-04-01T19:19:23+00:00",
         "date_modified":"2014-04-04T19:12:52+00:00",
         "modified_user_id":"1",
         "modified_by_name":"Administrator",
         "created_by":"1",
         "created_by_name":"Administrator",
         "description":"",
         "deleted":false,
         "assigned_user_id":"seed_sarah_id",
         "assigned_user_name":"Sarah Smith",
         "team_count":"",
         "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
      }
   ],
   "facebook":"",
   "twitter":"",
   "googleplus":"",
   "account_type":"Customer",
   "industry":"Finance",
   "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":"Persistance",
   "billing_address_state":"CA",
   "billing_address_postalcode":"88782",
   "billing_address_country":"USA",
   "rating":"",
   "phone_office":"(169) 381-5287",
   "phone_alternate":"",
   "website":"www.kidvegan.tv",
   "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":"Persistance",
   "shipping_address_state":"CA",
   "shipping_address_postalcode":"88782",
   "shipping_address_country":"USA",
   "email":[
      {
         "email_address":"jsmith@sugar.crm",
         "invalid_email":false,
         "opt_out":false,
         "primary_address":true,
         "reply_to_address":false
```

```
            }
         ],
         "email1":"jsmith@sugar.crm",
         "parent_id":"",
         "sic_code":"",
         "duns_num":"",
         "parent_name":"",
         "email_opt_out":false,
         "invalid_email":false,
         "campaign_id":"",
         "campaign_name":"",
         "my_favorite":false,
         "_acl":{
            "fields":{

            }
         },
         "following":false,
         "_module":"Accounts",
         "_search":{
            "score":0.70710677,
            "highlighted":{
               "email1":{

"text":"\u003Cstrong\u003Ejsmith@sugar.crm\u003C\/strong\u003E",
                  "module":"Accounts",
                  "label":"LBL_EMAIL_ADDRESS"
               }
            }
         }
      }
   ]
}
```


Last Modified: 05/16/2016 10:55am

Legacy Web Services

Legacy REST/SOAP APIs.

Last Modified: 09/26/2015 04:14pm

Introduction

Introduction

The legacy V1 - V4_1 REST/SOAP APIs have been succeeded by the new v10 REST API. Using the legacy REST/SOAP APIs is still supported, however, you should be migrating any integrations toward using the v10 before the legacy APIs are fully deprecated.

Last Modified: 09/26/2015 04:14pm

REST

Overview

v2 - v4_1 legacy REST service documentation.

What is 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 API Versioning section.

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 Requests

If you are making a REST request via cURL from within the Sugar application, you can make use of the SugarHttpClient Class. This class will help automate your cURL requests for you.

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.

Last Modified: 01/15/2016 09:25pm

SOAP

Overview

v2 - v4_1 legacy SOAP service documentation.

What is 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=literal
```

URL Parameters

style

- rpc
- document

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.

Last Modified: 01/15/2016 09:25pm

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');
```

Last Modified: 09/26/2015 04:14pm

SOAP vs. REST

Overview

Information regarding the legacy SOAP and REST methods.

Methods

There are various methods that are only contained within the REST API. A list of examples are:

- 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

When using a method, you should first verify that it is available for your preferred API.

Technology

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. The calls are documented below in the 'Method Calls' section.

Last Modified: 09/26/2015 04:14pm

Method Calls

Web Service Method Calls.

Last Modified: 09/26/2015 04:14pm

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'.

PHP

```
$get_available_modules_parameters = array(
    //Session id
    "session" => $session_id,

    //Module filter. Possible values are 'default', 'mobile', 'all'.
    "filter" => 'all',
);
```

Last Modified: 09/26/2015 04:14pm

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
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.

PHP

```
$get_document_revision_parameters = array(
```

```
    //Session id
    "session" => $session_id,

    //The attachment details
    "i" => '723b7dcb-27b3-e53d-b348-50bd283f8e48',
);
```

Last Modified: 09/26/2015 04:14pm

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

Name	Type	Description
		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.

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,
);
```

Last Modified: 09/26/2015 04:14pm

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

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
);
```

Last Modified: 09/26/2015 04:14pm

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 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
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.

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,
);
```

Last Modified: 09/26/2015 04:14pm

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 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.

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,
);
```

Last Modified: 09/26/2015 04:14pm

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

Name	Type	Description
		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.

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,
);
```

Last Modified: 09/26/2015 04:14pm

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.

Name	Type	Description

Change Log

Version	Change

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'
    ),
);
```

Last Modified: 09/26/2015 04:14pm

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 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

Name	Type	Description
		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'.

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
deleted 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'
);
```

Last Modified: 09/26/2015 04:14pm

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

Name	Type	Description
		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.
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.

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(),
);
```

Last Modified: 09/26/2015 04:14pm

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.

Name	Type	Description
		Order is based on the module_names parameter.

Change Log

Version	Change

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'
    ),
);
```

Last Modified: 09/26/2015 04:14pm

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.

Name	Type	Description
result[][$view]	Array or String	The list of layout views requested or MD5 hashes. This is dependent on the 'md5' parameter.

Change Log

Version	Change
v3_1	Added acl_check parameter.
v3	Added get_module_layout method.

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,
```

```
);
```

Last Modified: 09/26/2015 04:14pm

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',

Name	Type	Description
		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.

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,
);
```

Last Modified: 09/26/2015 04:14pm

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.

Name	Type	Description
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.

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",
);
```

Last Modified: 09/26/2015 04:14pm

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.

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',
);
```

Last Modified: 09/26/2015 04:14pm

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 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	link_names_to_fields_array	For every related record

Name	Type	Description
_to_fields_array	| Array	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, 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.

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
records.
    'link_field_name' => 'opportunities',

    //The portion of the WHERE clause from the SQL statement used to
find 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,
);
```

Last Modified: 09/26/2015 04:14pm

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.

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()
);
```

Last Modified: 09/26/2015 04:14pm

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.

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',
```

```
);
```

Last Modified: 09/26/2015 04:14pm

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.

Name	Type	Description
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.

PHP

```
//this method does not have any parameters
$get_server_info_parameters = array();
```

Last Modified: 09/26/2015 04:14pm

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

PHP

```
$get_upcoming_activities_parameters = array(
    //Session id
    "session" => $session_id,
);
```

Last Modified: 09/26/2015 04:14pm

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

PHP

```
$get_user_id_parameters = array(
    //Session id
    "session" => $session_id,
);
```

Last Modified: 09/26/2015 04:14pm

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.

PHP

```
$get_user_team_id_parameters = array(
    //Session id
    "session" => $session_id,
);
```

Last Modified: 09/26/2015 04:14pm

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.

PHP

```
$job_queue_cycle_parameters = array(
    //Session id
    'session' => $session_id,

    //The ID of the calling application.
    'clientid' => 'MyAppID',
);
```

Last Modified: 09/26/2015 04:14pm

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
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.

PHP

```
$job_queue_next_parameters = array(
    //Session id
    'session' => $session_id,

    //The ID of the calling application.
    'clientid' => 'MyAppID',
);
```

Last Modified: 09/26/2015 04:14pm

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.

Name	Type	Description
result.message	String	This is only returned if a failure occurs.

Change Log

Version	Change
v4	Added job_queue_run method.

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',
);
```

Last Modified: 09/26/2015 04:14pm

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.

Name	Type	Description
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.user_currency_id	String	Default currency ID of the 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.

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
        ),
    ),
```

```
);
```

Last Modified: 09/26/2015 04:14pm

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.

PHP

```
$logout_parameters = array(
    //session id to expire
    "session" => $session_id,
);
```

Last Modified: 09/26/2015 04:14pm

oauth_access

Overview

Retrieves the OAuth access token.

Available APIs

- REST

Definition

oauth_access()

Parameters

Name	Type	Description

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.

PHP

```
$oauth_access_parameters = array(
    //Session id
    'session' => $session_id,
);
```

Last Modified: 09/26/2015 04:14pm

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}
```

PHP

```
$seamless_login_parameters = array(
    //Session id
    "session" => $session_id,
);
```

Last Modified: 01/15/2016 09:34pm

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

Name	Type	Description
		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.

Name	Type	Description

Change Log

Version	Change
v3_1	Added unified_search_only parameter.

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
listviewdefs.
    'select_fields' => array(
        'id',
        'name',
    ),
```

```
    //If the search is only search modules participating in the
unified search.
    'unified_search_only' => false,

    //If only records marked as favorites should be returned.
    'favorites' => false
);
```

Last Modified: 09/26/2015 04:14pm

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

Name	Type	Description
		identifying the targets used in the merge. The IDs used in this parameter come from the column 'prospect_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.

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_lists_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'
);
```

Last Modified: 09/26/2015 04:14pm

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

Name	Type	Description
		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.

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',
    ),
);
```

Last Modified: 09/26/2015 04:14pm

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.

Name	Type	Description
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_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.

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"
            ),
        ),
    )
,);
```

Last Modified: 09/26/2015 04:14pm

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.

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"
        ),
    ),
);
```

Last Modified: 09/26/2015 04:14pm

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
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

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),
    ),
);
```

Last Modified: 09/26/2015 04:14pm

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.

Name	Type	Description
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.
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.

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
records.
    '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,
);
```

Last Modified: 09/26/2015 04:14pm

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.

Name	Type	Description
		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.
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

Name	Type	Description
		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_lists parameter.
v2	Added delete_array parameter.
v2	Return type was changed from set_relationship_list_result to new_set_relationship_list_result.

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
```

```
    ),
);
```

Last Modified: 09/26/2015 04:14pm

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.
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.

Name	Type	Description
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.

Last Modified: 09/26/2015 04:14pm

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.

Name	Type	Description
result.message	String	The return message.

Change Log

Version	Change
v4	Added snip_update_contacts method.

Last Modified: 09/26/2015 04:14pm

Extending Legacy Web Services

Overview

The guide will demonstrate how to add your own custom methods to the REST and SOAP API or extend exisiting ones.

Extending the API

The following example will demonstrate how to extend the v4_1 API.

Defining the Entry Point Location

This is where you define the directory that will contain your new REST and SOAP entry points. We recommend a path formatted as follows:

```
./custom/service/{version}_custom/
```

The actual location of the entry points does not matter, however, using a path such as this will allow you to call your entry points as follows:

- http://{sugar_url}/custom/service/{version}_custom/rest.php
- http://{sugar_url}/custom/service/{version}_custom/soap.php

Define the SugarWebServiceImpl Class

The next step is to define a new 'SugarWebServiceImpl' class. Since we are using v4_1, we need to extend the file 'service/v4_1/SugarWebServiceImplv4_1.php' and add our new method. To do this, we will create the file:

./custom/service/v4_1_custom/SugarWebServiceImplv4_1_custom.php

```
<?php

    if(!defined('sugarEntry'))define('sugarEntry', true);

    require_once('service/v4_1/SugarWebServiceImplv4_1.php');

    class SugarWebServiceImplv4_1_custom extends
SugarWebServiceImplv4_1
    {
        /*
         * Returns the session id if authenticated
         *
         * @param string $session
         * @return string $session - false if invalid.
         *
         */
        function example_method($session)
        {
            $GLOBALS['log']->info('Begin:
SugarWebServiceImplv4_1_custom->example_method');
            $error = new SoapError();

            //authenticate
            if
(!self::$helperObject->checkSessionAndModuleAccess($session,
'invalid_session', '', '', '',  $error))
            {
                $GLOBALS['log']->info('End:
SugarWebServiceImplv4_1_custom->example_method.');
                return false;
```

```
            }

            return $session;
        }
    }

?>
```

Define the Registry Class

Next, we will define the registry class that will register our new function. This file will be located at:

./custom/service/v4_1_custom/registry.php

```
<?php

    require_once('service/v4_1/registry.php');

    class registry_v4_1_custom extends registry_v4_1
    {
        protected function registerFunction()
        {
            parent::registerFunction();
            $this->serviceClass->registerFunction('example_method',
array('session'=>'xsd:string'), array('return'=>'xsd:string'));
        }
    }

?>
```

Define the REST Entry Point

This REST entry point will be located at:

./custom/service/v4_1_custom/rest.php

```
<?php
```

```
    chdir('../../..');
    require_once('SugarWebServiceImplv4_1_custom.php');

    $webservice_path = 'service/core/SugarRestService.php';
    $webservice_class = 'SugarRestService';
    $webservice_impl_class = 'SugarWebServiceImplv4_1_custom';

    $registry_path = 'custom/service/v4_1_custom/registry.php';
    $registry_class = 'registry_v4_1_custom';

    $location = 'custom/service/v4_1_custom/rest.php';

    require_once('service/core/webservice.php');

?>
```

Define the SOAP Entry Point

This SOAP entry point will be located at:

./custom/service/v4_1_custom/soap.php

```
<?php

    chdir('../../..');
    require_once('SugarWebServiceImplv4_1_custom.php');

    $webservice_class = 'SugarSoapService2';
    $webservice_path = 'service/v2/SugarSoapService2.php';
    $webservice_impl_class = 'SugarWebServiceImplv4_1_custom';

    $registry_class = 'registry_v4_1_custom';
    $registry_path = 'custom/service/v4_1_custom/registry.php';

    $location = 'custom/service/v4_1_custom/soap.php';

    require_once('service/core/webservice.php');

?>
```

Last Modified: 11/19/2015 01:52am

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

Last Modified: 09/26/2015 04:14pm

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.

Last Modified: 09/26/2015 04:14pm

Examples

Examples of v4_1 Web Service API Calls.

Last Modified: 06/30/2016 04:36pm

REST

Examples of v4_1 REST API Calls.

Last Modified: 09/26/2015 04:14pm

PHP

PHP v4_1 REST Examples.

Last Modified: 09/26/2015 04:14pm

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($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
        curl_setopt($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;

    //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
below
            //array("name" => "id", "value" =>
```

```
"9b170af9-3080-e22b-fbc1-4fea74def88f"),
            array("name" => "document_name", "value" => "Example
Document"),
            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
)
```

Last Modified: 06/30/2016 03:54pm

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($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
        curl_setopt($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;

//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
id 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_note_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
)
```

Last Modified: 06/30/2016 03:54pm

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($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
        curl_setopt($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;

    //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
id 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
                )

        )

)
```

Last Modified: 09/26/2015 04:14pm

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($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
        curl_setopt($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;

//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
        )

)
```

Last Modified: 06/30/2016 03:55pm

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($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
        curl_setopt($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;

//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
record.
            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] =>
                )

        )

)
```

Last Modified: 06/30/2016 03:55pm

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($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
        curl_setopt($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;

?>
```

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($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
        curl_setopt($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 ----------------------------------
    $ldap_enc_key = substr(md5($ldap_enc_key), 0, 24);
    $login_parameters = array(
         "user_auth" => array(
              "user_name" => $username,
              "password" => bin2hex(mcrypt_cbc(MCRYPT_3DES,
$ldap_enc_key, $password, MCRYPT_ENCRYPT, '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;

?>
```

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
                )

        )

)
```

Last Modified: 06/30/2016 03:56pm

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($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
        curl_setopt($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;

    //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,
$url);

    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',
$createProductBundleParams, $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
)
```

Last Modified: 06/30/2016 03:56pm

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_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 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
empty 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
        (
        )

)
```

Last Modified: 06/30/2016 03:57pm

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($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
        curl_setopt($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;

//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
link 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_parameters, $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
            (
            )
```

)

Last Modified: 06/30/2016 03:57pm

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($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
        curl_setopt($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 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_parameters, $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] =>
stdClass Object
                                                        (
                                                            [id] =>
stdClass Object
                                                                (

[name] => id

[value] => 13066f13-d6ea-405c-0f95-4f7cb6fa3a08
                                                                )


[email_address] => stdClass Object
                                                                (

[name] => email_address

[value] => support52@example.org
                                                                )

                                                            [opt_out]
=> stdClass Object
                                                                (

[name] => opt_out
```

```
[value] => 0
                                                                    )


[primary_address] => stdClass Object
                                                                    (

[name] => primary_address

[value] =>
                                                                    )

                                                                )

                                                            )

                                                        [1] => stdClass Object
                                                            (
                                                                [link_value] =>
stdClass Object
                                                                (
                                                                    [id] =>
stdClass Object
                                                                    (

[name] => id

[value] => 13e3d111-b226-c363-4832-4f7cb699a3a0
                                                                    )


[email_address] => stdClass Object
                                                                    (

[name] => email_address

[value] => qa.section@example.it
                                                                    )

                                                                [opt_out]
=> stdClass Object
```

```
                                                                    (
[name] => opt_out
[value] => 1
                                                                    )

[primary_address] => stdClass Object
                                                                    (
[name] => primary_address
[value] =>
                                                                    )
                                                                )
                                                            )
                                                        )
                                                    )
                                                )
                                            )
                                        [1] => stdClass Object
                                            (
                                                [link_list] => Array
                                                    (
                                                        [0] => stdClass Object
                                                            (
                                                                [name] => email_addresses
                                                                [records] => Array
                                                                    (
                                                                        [0] => stdClass Object
                                                                            (
                                                                                [link_value] =>
stdClass Object
                                                                                (
```

```
                                                                [id] =>
stdClass Object
                                                                    (

[name] => id

[value] => 16aeaf8b-b31f-943d-3844-4f7cb6ac63f2
                                                                    )


[email_address] => stdClass Object
                                                                    (

[name] => email_address

[value] => vegan.sales.im@example.cn
                                                                    )

                                                                [opt_out]
=> stdClass Object
                                                                    (

[name] => opt_out

[value] => 0
                                                                    )


[primary_address] => stdClass Object
                                                                    (

[name] => primary_address

[value] =>
                                                                    )

                                                                )

                                                            )

                                                        [1] => stdClass Object
                                                            (
```

```
                                                    [link_value] =>
stdClass Object
                                                        (
                                                            [id] =>
stdClass Object
                                                                (
[name] => id
[value] => 173bacf5-5ea6-3906-d91d-4f7cb6c6e883
                                                                )

[email_address] => stdClass Object
                                                                (
[name] => email_address
[value] => kid.the.section@example.org
                                                                )

                                                            [opt_out]
=> stdClass Object
                                                                (
[name] => opt_out
[value] => 1
                                                                )

[primary_address] => stdClass Object
                                                                (
[name] => primary_address
[value] =>
                                                                )

                                                        )

                                                )
```

)

)

)

)

)

)

Last Modified: 06/30/2016 03:57pm

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_result->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_note_attachment_parameters, $url);

    //attachment contents
    echo "<pre>";
    print_r($get_note_attachment_result);
    echo "</pre>";

    $file_name =
$get_note_attachment_result->note_attachment->filename;
    //decode and get file contents
    $file_contents =
base64_decode($get_note_attachment_result->note_attachment->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
new 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] =>
1b63a8f9-ce67-6aad-b5a4-52054af18c47
                                                )

                                            [name] => stdClass Object
                                                (
                                                    [name] => name
                                                    [value] =>
Example.zip
                                                )

                                            [file_mime_type] =>
stdClass Object
                                                (
                                                    [name] =>
file_mime_type
                                                    [value] =>
application/zip
                                                )

                                            [filename] => stdClass
Object
                                                (
                                                    [name] => filename
                                                    [value] =>
Example2.zip
                                                )

                                            [description] => stdClass
Object
                                                (
                                                    [name] =>
description
                                                    [value] =>
```

```
                                        )

                                    [_empty_] => stdClass
Object
                                        (
                                            [name] =>
                                            [value] =>
                                        )

                                )

                            [1] => stdClass Object
                                (
                                    [id] => stdClass Object
                                        (
                                            [name] => id
                                            [value] =>
592f382d-b633-fd5f-803e-5205423a6d0b
                                        )

                                    [name] => stdClass Object
                                        (
                                            [name] => name
                                            [value] =>
Example2.zip
                                        )


                                    [file_mime_type] =>
stdClass Object
                                        (
                                            [name] =>
file_mime_type
                                            [value] =>
application/zip
                                        )


                                    [filename] => stdClass
Object
                                        (
                                            [name] => filename
                                            [value] =>
Example2.zip
```

```
                                            )

                                        [description] => stdClass
Object
                                            (
                                                [name] =>
description
                                                [value] =>
                                            )

                                        [_empty_] => stdClass
Object
                                            (
                                                [name] =>
                                                [value] =>
                                            )

                                    )

                                )

                            )

                    )

            )

)

//Attachment 1 Result
stdClass Object
(
    [note_attachment] => stdClass Object
        (
            [id] => 1b63a8f9-ce67-6aad-b5a4-52054af18c47
            [filename] => Example.zip
            [file] =>
UEsDBAoAAAAIAOujCEOkMbvsNa0AAMCFAgAXAAAARmlsZXMvaW
            [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] =>
AEUoaAAARmlslsZXMvaWZXujCEOkMbvsNa0AAMCFAgAXAAAARm
            [related_module_id] =>
5826bd75-527a-a736-edf5-5205421467bf
            [related_module_name] => Emails
        )

)
```

Last Modified: 06/30/2016 03:58pm

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($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
        curl_setopt($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 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
Object
                                (
                                    [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
Object
                                    (
                                        [name] => billing_address_country
                                        [value] => USA
                                    )

                            )

                    )

            )

        [relationship_list] => Array
            (
            )
```

)

Last Modified: 06/30/2016 03:58pm

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($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
        curl_setopt($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;

    //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_module_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
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' => false,
        );

        $get_entries_results[$module] = call('get_entries',
$get_entries_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-e7d6-5282a3b96953
                            [module_name] => Accounts
                            [name_value_list] => stdClass Object
                                (
                                    [name] => stdClass Object
                                        (
                                            [name] => name
                                            [value] => Underwater
Mining 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
Object
                                                        (

[link_value] => stdClass Object
                                                                (

[email_address] => stdClass Object

(

[name] => email_address

[value] => beans.the.qa@example.com
```

```
)

[opt_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] => 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
Object
                                                    (

[link_value] => stdClass Object
                                                                (

[email_address] => stdClass Object

(

[name] => email_address

[value] => section51@example.com

)


[opt_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] => 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-2622-5282a372f39b
                            [module_name] => Contacts
                            [name_value_list] => stdClass Object
                                (
                                    [name] => stdClass Object
                                        (
                                            [name] => name
                                            [value] => Errol Goldberg
                                        )

                                )

                        )

                    [1] => stdClass Object
                        (
                            [id] =>
```

```
2542dad2-85f3-5529-3a30-5282a3104e31
                            [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
Object
                                                        (

[link_value] => stdClass Object
                                                            (

[email_address] => stdClass Object

(

[name] => email_address

[value] => hr.phone@example.com
```

```
)

[opt_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] => the.info.phone@example.cn
```

```
)

[opt_out] => stdClass Object

(

[name] => opt_out

[value] => 1

)

[primary_address] => stdClass Object

(

[name] => primary_address

[value] =>

)
                                                                    )
                                                                )
                                                            )
                                                        )
                                                    )
                                                )
                                            )
                                        )
                                    )
                                [1] => stdClass Object
                                    (
                                        [link_list] => Array
                                            (
                                                [0] => stdClass Object
```

```
                                    (
                                        [name] => email_addresses
                                        [records] => Array
                                            (
                                                [0] => stdClass
Object
                                                    (

[link_value] => stdClass Object
                                                                (

[email_address] => stdClass Object

(

[name] => email_address

[value] => im.the.sales@example.name

)


[opt_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

)

[primary_address] => stdClass Object

(

[name] => primary_address

[value] =>

```
)
                                                                    )
                                                                )
                                                            )
                                                        )
                                                    )
                                                )
                                            )
                                        )
                                    )
                                )
                            )
                        )
                    )
                )
            )
        )
    [Leads] => stdClass Object
        (
            [entry_list] => Array
                (
                    [0] => stdClass Object
                        (
                            [id] =>
83abeeff-5e71-0f06-2e5f-5282a3f04f41
                            [module_name] => Leads
                            [name_value_list] => stdClass Object
                                (
                                    [name] => stdClass Object
                                        (
                                            [name] => name
                                            [value] => Caryn Wert
                                        )

                                )

                        )

                    [1] => stdClass Object
                        (
                            [id] =>
```

```
2a3b304b-5167-8432-d4e7-5282a300eabb
                            [module_name] => Leads
                            [name_value_list] => stdClass Object
                                (
                                    [name] => stdClass Object
                                        (
                                            [name] => name
                                            [value] => Marco
Castonguay
                                        )

                                )

                        )

                )

            [relationship_list] => Array
                (
                    [0] => stdClass Object
                        (
                            [link_list] => Array
                                (
                                    [0] => stdClass Object
                                        (
                                            [name] => email_addresses
                                            [records] => Array
                                                (
                                                    [0] => stdClass
Object
                                                        (

[link_value] => stdClass Object
                                                                (

[email_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
Object
                                                    (

[link_value] => stdClass Object
                                                                (

[email_address] => stdClass Object

(

[name] => email_address

[value] => im.the@example.com

)


[opt_out] => stdClass Object

(

[name] => opt_out

[value] => 0

)


[primary_address] => stdClass Object

(

[name] => primary_address

[value] =>

)
```

)

)

)

)

)

)

)

)

)

Last Modified: 09/26/2015 04:14pm

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($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
        curl_setopt($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 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
return 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
to 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_relationships_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
        (
        )

)
```

Last Modified: 06/30/2016 03:58pm

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($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
        curl_setopt($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;

//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
listviewdefs.
        '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_module_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
        (
        )

)
```

Last Modified: 06/30/2016 03:59pm

SOAP

Examples of v4_1 SOAP API Calls.

Last Modified: 09/26/2015 04:14pm

C#

C# SOAP v4_1 Examples.

Last Modified: 09/26/2015 04:14pm

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(UserAuth, "SoapTest", LoginList);

            //get session id
            SessionID = LoginResult.id;

            //create account --------------------------------

            NameValueCollection fieldListCollection = new
NameValueCollection();
            //to update a record, you will nee to pass in a record id
as commented below
            //fieldListCollection.Add("id",
"68c4781f-75d1-223a-5d8f-5058bc4e39ea");
            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_value();
                    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(PlainText);
            byte[] outputBuffer = md5.ComputeHash(inputBuffer);

            //Convert the byte[] to a hex-string
            StringBuilder builder = new
StringBuilder(outputBuffer.Length);
            for (int i = 0; i < outputBuffer.Length; i++)
            {
                builder.Append(outputBuffer[i].ToString("X2"));
            }

            return builder.ToString();
        }
    }
}
```

Result

```
68c4781f-75d1-223a-5d8f-5058bc4e39ea
```

Last Modified: 09/26/2015 04:14pm

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(UserAuth, "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(TextString);
            byte[] outputBuffer = md5.ComputeHash(inputBuffer);

            StringBuilder Builder = new
StringBuilder(outputBuffer.Length);
            for (int i = 0; i < outputBuffer.Length; i++)
            {
                Builder.Append(outputBuffer[i].ToString("X2"));
            }

            return Builder.ToString();
        }
    }
}
```

Result

`b2uv0vrjfiov41d03sk578ufq6`

Last Modified: 09/26/2015 04:14pm

PHP

PHP SOAP v4_1 Examples.

Last Modified: 09/26/2015 04:14pm

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 NuSOAP and the v4_1 SOAP API.

Example

```
<?php

    $url = "http://{site_url}/service/v4_1/soap.php?wsdl";
    $username = "admin";
    $password = "password";

    //require NuSOAP
    require_once("./nusoap/lib/nusoap.php");

    //retrieve WSDL
    $client = new nusoap_client($url, 'wsdl');

    //display errors
    $err = $client->getError();
```

```
    if ($err)
    {
        echo '<h2>Constructor error</h2><pre>' . $err . '</pre>';
        echo '<h2>Debug</h2><pre>' .
htmlspecialchars($client->getDebug(), ENT_QUOTES) . '</pre>';
        exit();
    }

    //login ------------------------------------------------
$login_parameters = array(
         'user_auth' => array(
              'user_name' => $username,
              'password' => md5($password),
              'version' => '1'
         ),
         'application_name' => 'SoapTest',
         'name_value_list' => array(
         ),
    );

    $login_result = $client->call('login', $login_parameters);

    /*
    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, you will nee to pass in a record id
as commented below
            //array("name" => "id", "value" =>
"9b170af9-3080-e22b-fbc1-4fea74def88f"),
            array("name" => "document_name", "value" => "Example
Document"),
            array("name" => "revision", "value" => "1"),
        ),
    );

    $set_entry_result = $client->call("set_entry",
$set_entry_parameters);

    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 =
$client->call("set_document_revision",
$set_document_revision_parameters);

    echo "<pre>";
    print_r($set_document_revision_result);
    echo "</pre>";

?>
```

Result

```
//set_entry result
Array
(
    [id] => 5ab53b9d-2f31-9b69-d92b-50abc2d0f6a2
)

//set_document_revision result
Array
(
    [id] => 906ad157-0aa0-c01e-2694-50abc2adcbf2
)
```

Last Modified: 06/30/2016 04:03pm

Creating Notes with Attachments

Overview

A PHP example demonstrating how to create a note using set_entry and an attachment with the set_note_attachment method using NuSOAP and the v4_1 SOAP API.

Example

```
<?php

    $url = "http://{site_url}/service/v4_1/soap.php?wsdl";
    $username = "admin";
    $password = "password";

    //require NuSOAP
    require_once("./nusoap/lib/nusoap.php");

    //retrieve WSDL
    $client = new nusoap_client($url, 'wsdl');

    //display errors
    $err = $client->getError();
    if ($err)
    {
        echo '<h2>Constructor error</h2><pre>' . $err . '</pre>';
        echo '<h2>Debug</h2><pre>' .
htmlspecialchars($client->getDebug(), ENT_QUOTES) . '</pre>';
        exit();
    }

    //login ----------------------------------------------------

    $login_parameters = array(
         'user_auth' => array(
              'user_name' => $username,
              'password' => md5($password),
              'version' => '1'
         ),
         'application_name' => 'SoapTest',
         'name_value_list' => array(
         ),
    );

    $login_result = $client->call('login', $login_parameters);

    /*
    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 id
as commented below
            //array("name" => "id", "value" =>
"9b170af9-3080-e22b-fbc1-4fea74def88f"),
            array("name" => "name", "value" => "Example Note"),
        ),
    );

    $set_entry_result = $client->call("set_entry",
$set_entry_parameters);

    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 = $client->call("set_note_attachment",
$set_note_attachment_parameters);

    echo "<pre>";
    print_r($set_note_attachment_result);
    echo "</pre>";

?>
```

Result

```
//set_entry_result
Array
(
    [id] => 59a33435-7c6a-2d97-e61b-50abcc5d2644
)

//set_note_attachment result
Array
(
    [id] => 59a33435-7c6a-2d97-e61b-50abcc5d2644
)
```

Last Modified: 06/30/2016 04:03pm

Creating or Updating a Record

Overview

A PHP example demonstrating how to create or update an Account with the set_entry method using NuSOAP and the v4_1 SOAP API.

Example

```
<?php

    $url = "http://{site_url}/service/v4_1/soap.php?wsdl";
    $username = "admin";
    $password = "password";

    //require NuSOAP
    require_once("./nusoap/lib/nusoap.php");

    //retrieve WSDL
    $client = new nusoap_client($url, 'wsdl');

    //display errors
    $err = $client->getError();
    if ($err)
    {
        echo '<h2>Constructor error</h2><pre>' . $err . '</pre>';
        echo '<h2>Debug</h2><pre>' .
htmlspecialchars($client->getDebug(), ENT_QUOTES) . '</pre>';
        exit();
    }

    //login ----------------------------------------------------

    $login_parameters = array(
         'user_auth' => array(
              'user_name' => $username,
              'password' => md5($password),
              'version' => '1'
         ),
         'application_name' => 'SoapTest',
         'name_value_list' => array(
         ),
    );
```

```
    $login_result = $client->call('login', $login_parameters);

    /*
    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
id as commented below
              //array("name" => "id", "value" =>
"9b170af9-3080-e22b-fbc1-4fea74def88f"),
              array("name" => "name", "value" => "Test Account"),
         ),
    );

    $set_entry_result = $client->call("set_entry",
$set_entry_parameters);

    echo "<pre>";
    print_r($set_entry_result);
    echo "</pre>";

?>
```

Result

```
Array
(
    [id] => 63c103dd-1f47-804c-1282-52af64b870d4
)
```

Last Modified: 06/30/2016 04:03pm

Creating or Updating Multiple Records

Overview

A PHP example demonstrating how to create or update multiple contacts with the set_entries method using NuSOAP and the v4_1 SOAP API.

Example

```
<?php

    $url = "http://{site_url}/service/v4_1/soap.php?wsdl";
    $username = "admin";
    $password = "password";

    //require NuSOAP
    require_once("./nusoap/lib/nusoap.php");

    //retrieve WSDL
    $client = new nusoap_client($url, 'wsdl');

    //display errors
    $err = $client->getError();
    if ($err)
    {
        echo '<h2>Constructor error</h2><pre>' . $err . '</pre>';
        echo '<h2>Debug</h2><pre>' .
htmlspecialchars($client->getDebug(), ENT_QUOTES) . '</pre>';
        exit();
```

```
    }

    //login ------------------------------------------------

    $login_parameters = array(
         'user_auth' => array(
              'user_name' => $username,
              'password' => md5($password),
              'version' => '1'
         ),
         'application_name' => 'SoapTest',
         'name_value_list' => array(
         ),
    );

    $login_result = $client->call('login', $login_parameters);

    /*
    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 = $client->call("set_entries",
$set_entries_parameters);

    echo "<pre>";
    print_r($set_entries_result);
    echo "</pre>";

?>
```

Result

```
Array
(
    [ids] => Array
        (
            [0] => 912e58c0-73e9-9cb6-c84e-4ff34d62620e
            [1] => 99d6ddfd-7d52-d45b-eba8-4ff34d684964
        )

)
```

Last Modified: 06/30/2016 04:03pm

Creating or Updating Teams

Overview

A PHP example demonstrating how to manipulate teams using NuSOAP and the v4_1 SOAP 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/soap.php?wsdl";
    $username = "admin";
    $password = "password";

    //require NuSOAP
    require_once("./nusoap/lib/nusoap.php");

    //retrieve WSDL
    $client = new nusoap_client($url, 'wsdl');

    //display errors
    $err = $client->getError();
    if ($err)
    {
        echo '<h2>Constructor error</h2><pre>' . $err . '</pre>';
        echo '<h2>Debug</h2><pre>' .
htmlspecialchars($client->getDebug(), ENT_QUOTES) . '</pre>';
        exit();
    }

    //login ----------------------------------------------------

    $login_parameters = array(
         'user_auth' => array(
              'user_name' => $username,
              'password' => md5($password),
              'version' => '1'
         ),
```

```
        'application_name' => 'SoapTest',
        'name_value_list' => array(
        ),
    );

    $login_result = $client->call('login', $login_parameters);

    /*
    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
record.
            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 = $client->call('set_entry',
$set_entry_parameters);

    echo "<pre>";
    print_r($set_entry_result);
    echo "</pre>";

?>
```

Result

```
Array
(
    [id] => 90130b4f-4d39-100b-98de-52ab7a638d0d
)
```

Last Modified: 06/30/2016 04:04pm

Logging In

Overview

A PHP example demonstrating how to log in and retrieve a session key using NuSOAP and the v4_1 SOAP API.

Standard Authentication Example

```
<?php

    $url = "http://{site_url}/service/v4_1/soap.php?wsdl";
    $username = "admin";
    $password = "password";

    //require NuSOAP
    require_once("./nusoap/lib/nusoap.php");

    //retrieve WSDL
    $client = new nusoap_client($url, 'wsdl');

    //display errors
    $err = $client->getError();
    if ($err)
    {
        echo '<h2>Constructor error</h2><pre>' . $err . '</pre>';
        echo '<h2>Debug</h2><pre>' .
htmlspecialchars($client->getDebug(), ENT_QUOTES) . '</pre>';
        exit();
    }

    //login ----------------------------------------------------

    $login_parameters = array(
         'user_auth' => array(
              'user_name' => $username,
              'password' => md5($password),
              'version' => '1'
         ),
         'application_name' => 'SoapTest',
         'name_value_list' => array(
         ),
    );

    $login_result = $client->call('login', $login_parameters);

    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/soap.php?wsdl";
    $username = "admin";
    $password = "password";
    $ldap_enc_key = 'LDAP_ENCRYPTION_KEY';

    //require NuSOAP
    require_once("./nusoap/lib/nusoap.php");

    //retrieve WSDL
    $client = new nusoap_client($url, 'wsdl');

    //display errors
    $err = $client->getError();
    if ($err)
    {
        echo '<h2>Constructor error</h2><pre>' . $err . '</pre>';
        echo '<h2>Debug</h2><pre>' .
htmlspecialchars($client->getDebug(), ENT_QUOTES) . '</pre>';
        exit();
    }

    //login ----------------------------------------------------
    $ldap_enc_key = substr(md5($ldap_enc_key), 0, 24);
    $login_parameters = array(
         'user_auth' => array(
              'user_name' => $username,
              'password' => bin2hex(mcrypt_cbc(MCRYPT_3DES,
$ldap_enc_key, $password, MCRYPT_ENCRYPT, 'password')),
              'version' => '1'
         ),
```

```
        'application_name' => 'SoapTest',
        'name_value_list' => array(
        ),
    );

    $login_result = $client->call('login', $login_parameters);

    echo '<pre>';
    print_r($login_result);
    echo '</pre>';

    //get session id
    $session_id =  $login_result['id'];

?>
```

Result

```
Array
(
    [id] => 9uukc92a6lb9v620reqv34mmg1
    [module_name] => Users
    [name_value_list] => Array
        (
            [0] => Array
                (
                    [name] => user_id
                    [value] => 1
                )

            [1] => Array
                (
                    [name] => user_name
                    [value] => admin
                )

            [2] => Array
                (
                    [name] => user_language
                    [value] => en_us
                )
```

```
[3] => Array
    (
        [name] => user_currency_id
        [value] => -99
    )

[4] => Array
    (
        [name] => user_is_admin
        [value] => 1
    )

[5] => Array
    (
        [name] => user_default_team_id
        [value] => 1
    )

[6] => Array
    (
        [name] => user_default_dateformat
        [value] => m/d/Y
    )

[7] => Array
    (
        [name] => user_default_timeformat
        [value] => h:ia
    )

[8] => Array
    (
        [name] => user_number_seperator
        [value] => ,
    )

[9] => Array
    (
        [name] => user_decimal_seperator
        [value] => .
    )
```

```
            [10] => Array
                (
                    [name] => mobile_max_list_entries
                    [value] => 10
                )

            [11] => Array
                (
                    [name] => mobile_max_subpanel_entries
                    [value] => 3
                )

            [12] => Array
                (
                    [name] => user_currency_name
                    [value] => US Dollars
                )

        )

)
```

Last Modified: 06/30/2016 04:04pm

Relating Quotes and Products

Overview

A PHP example demonstrating how to create and relate Products to Quotes using and the v4_1 SOAP API.

Example

```
<?php

    $url = "http://{site_url}/service/v4_1/soap.php?wsdl";
    $username = "admin";
```

```
    $password = "password";

    //require NuSOAP
    require_once("./nusoap/lib/nusoap.php");

    //retrieve WSDL
    $client = new nusoap_client($url, 'wsdl');

    //display errors
    $err = $client->getError();
    if ($err)
    {
        echo '<h2>Constructor error</h2><pre>' . $err . '</pre>';
        echo '<h2>Debug</h2><pre>' .
htmlspecialchars($client->getDebug(), ENT_QUOTES) . '</pre>';
        exit();
    }

    //login ----------------------------------------------------

    $login_parameters = array(
         'user_auth' => array(
              'user_name' => $username,
              'password' => md5($password),
              'version' => '1'
         ),
         'application_name' => 'SoapTest',
         'name_value_list' => array(
         ),
    );

    $login_result = $client->call("login", $login_parameters);

    /*
    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 = $client->call('set_entry',
$createQuoteParams);

    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 = $client->call('set_entry',
$createProductParams);

    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' => '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 = $client->call('set_entry',
$createProductBundleParams);

    echo "Create ProductBundles Result<br />";
```

```
    echo "<pre>";
    print_r($createProductBundleResult);
    echo "</pre>";

    //relate product to product-bundle
-------------------------------------

    $relationshipProductBundleProductsParams = array(
        'session' => $session_id,
        'module_name' => 'ProductBundles',
        'module_id' => $createProductBundleResult['id'],
        'link_field_name' => 'products',
        'related_ids' => array(
            $createProductResult['id']
        ),
        'name_value_list' => array(),
        'delete' => 0
    );

    // set the product bundles products relationship
    $relationshipProductBundleProductResult =
$client->call('set_relationship',
$relationshipProductBundleProductsParams);

    echo "Create ProductBundleProduct Relationship Result<br />";

    echo "<pre>";
    print_r($relationshipProductBundleProductResult);
    echo "</pre>";

    //relate product-bundle to quote
---------------------------------------

    $relationshipProductBundleQuoteParams = array(
        'session' => $session_id,
        'module_name' => 'Quotes',
        'module_id' => $createQuoteResult['id'],
        'link_field_name' => 'product_bundles',
        'related_ids' => array(
            $createProductBundleResult['id']
        ),
        'name_value_list' => array(),
```

```
        'delete' => 0
    );

    // set the product bundles quotes relationship
    $relationshipProductBundleQuoteResult =
$client->call('set_relationship',
$relationshipProductBundleQuoteParams);

    echo "Create ProductBundleQuote Relationship Result<br />";

    echo "<pre>";
    print_r($relationshipProductBundleQuoteResult);
    echo "</pre>";
```

Result

```
//Create Quote Result
Array
(
    [id] => ad88195a-9d36-20b6-beb1-517ea2b9278d
)

//Create Product Result
Array
(
    [id] => 6f920cba-0fed-7172-9dc0-517ea2adb1d1
)

//Create ProductBundles Result
Array
(
    [id] => 4f397baa-5522-fa0e-2bcf-517ea24a9546
)

//Create ProductBundleProduct Relationship Result
Array
(
    [created] => 1
    [failed] => 0
    [deleted] => 0
)
```

```
//Create ProductBundleQuote Relationship Result
Array
(
    [created] => 1
    [failed] => 0
    [deleted] => 0
)
```

Last Modified: 06/30/2016 04:10pm

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 NuSOAP and the v4_1 SOAP API.

This example will only retrieve the vardefs for the 'id' and 'name' fields.

Example

```
<?php

    $url = "http://{site_url}/service/v4_1/soap.php?wsdl";
    $username = "admin";
    $password = "password";

    //require NuSOAP
    require_once("./nusoap/lib/nusoap.php");

    //retrieve WSDL
    $client = new nusoap_client($url, 'wsdl');

    //display errors
    $err = $client->getError();
    if ($err)
    {
```

```
        echo '<h2>Constructor error</h2><pre>' . $err . '</pre>';
        echo '<h2>Debug</h2><pre>' .
htmlspecialchars($client->getDebug(), ENT_QUOTES) . '</pre>';
        exit();
    }

    //login ----------------------------------------------------

    $login_parameters = array(
         'user_auth' => array(
              'user_name' => $username,
              'password' => md5($password),
              'version' => '1'
         ),
         'application_name' => 'SoapTest',
         'name_value_list' => array(
         ),
    );

    $login_result = $client->call('login', $login_parameters);

    /*
    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 empty
array will return all fields.
```

```
        'fields' => array(
            'id',
            'name',
        ),
    );

    $get_module_fields_result = $client->call("get_module_fields",
$get_module_fields_parameters);

    echo "<pre>";
    print_r($get_module_fields_result);
    echo "</pre>";

?>
```

Result

```
Array
(
    [module_name] => Accounts
    [table_name] => accounts
    [module_fields] => Array
        (
            [0] => Array
                (
                    [name] => id
                    [type] => id
                    [group] =>
                    [label] => ID
                    [required] => 1
                    [options] => Array
                        (
                        )

                )

            [1] => Array
                (
                    [name] => name
                    [type] => name
                    [group] =>
```

```
                    [label] => Name:
                    [required] => 1
                    [options] => Array
                        (
                        )

                )

        )

    [link_fields] => Array
        (
        )

)
```

Last Modified: 06/30/2016 04:33pm

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 NuSOAP and the v4_1 SOAP API. This example will retrieve a list of leads.

Example

```
<?php

    $url = "http://{site_url}/service/v4_1/soap.php?wsdl";
    $username = "admin";
    $password = "password";

    //require NuSOAP
    require_once("./nusoap/lib/nusoap.php");

    //retrieve WSDL
```

```
    $client = new nusoap_client($url, 'wsdl');

    //display errors
    $err = $client->getError();
    if ($err)
    {
        echo '<h2>Constructor error</h2><pre>' . $err . '</pre>';
        echo '<h2>Debug</h2><pre>' .
htmlspecialchars($client->getDebug(), ENT_QUOTES) . '</pre>';
        exit();
    }

    //login ------------------------------------------------

    $login_parameters = array(
         'user_auth' => array(
              'user_name' => $username,
              'password' => md5($password),
              'version' => '1'
         ),
         'application_name' => 'SoapTest',
         'name_value_list' => array(
         ),
    );

    $login_result = $client->call('login', $login_parameters);

    /*
    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
link 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 = $client->call('get_entry_list',
$get_entry_list_parameters);
```

```
    echo '<pre>';
    print_r($get_entry_list_result);
    echo '</pre>';

?>
```

Result

```
Array
(
    [result_count] => 2
    [total_count] => 200
    [next_offset] => 2
    [entry_list] => Array
        (
            [0] => Array
                (
                    [id] => 18124607-69d1-b158-47ff-4f7cb69344f7
                    [module_name] => Leads
                    [name_value_list] => Array
                        (
                            [0] => Array
                                (
                                    [name] => id
                                    [value] =>
18124607-69d1-b158-47ff-4f7cb69344f7
                                )

                            [1] => Array
                                (
                                    [name] => name
                                    [value] => Bernie Worthey
                                )

                            [2] => Array
                                (
                                    [name] => title
                                    [value] => Senior Product Manager
                                )
```

```
                        )

                )

            [1] => Array
                (
                    [id] => 1cdfddc1-2759-b007-8713-4f7cb64c2e9c
                    [module_name] => Leads
                    [name_value_list] => Array
                        (
                            [0] => Array
                                (
                                    [name] => id
                                    [value] =>
1cdfddc1-2759-b007-8713-4f7cb64c2e9c
                                )

                            [1] => Array
                                (
                                    [name] => name
                                    [value] => Bobbie Kohlmeier
                                )

                            [2] => Array
                                (
                                    [name] => title
                                    [value] => Director Operations
                                )

                        )

                )

        )

    [relationship_list] => Array
        (
        )

)
```

Last Modified: 06/30/2016 04:33pm

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 NuSOAP and the v4_1 SOAP API.

This example will retrieve a list of contacts and their related email addresses.

Example

```
<?php

    $url = "http://{site_url}/service/v4_1/soap.php?wsdl";
    $username = "admin";
    $password = "password";

    //require NuSOAP
    require_once("./nusoap/lib/nusoap.php");

    //retrieve WSDL
    $client = new nusoap_client($url, 'wsdl');

    //display errors
    $err = $client->getError();
    if ($err)
    {
        echo '<h2>Constructor error</h2><pre>' . $err . '</pre>';
        echo '<h2>Debug</h2><pre>' .
htmlspecialchars($client->getDebug(), ENT_QUOTES) . '</pre>';
        exit();
    }

    //login ----------------------------------------------------

    $login_parameters = array(
         'user_auth' => array(
```

```
            'user_name' => $username,
            'password' => md5($password),
            'version' => '1'
        ),
        'application_name' => 'SoapTest',
        'name_value_list' => array(
        ),
);

$login_result = $client->call('login', $login_parameters);

/*
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 = $client->call("get_entry_list",
$get_entry_list_parameters);

    echo "<pre>";
    print_r($get_entry_list_result);
    echo "</pre>";

?>
```

Result

```
Array
(
```

```
    [result_count] => 2
    [total_count] => -1
    [next_offset] => 2
    [entry_list] => Array
        (
            [0] => Array
                (
                    [id] => 10613769-54e0-820d-de9b-5282a31b88cd
                    [module_name] => Contacts
                    [name_value_list] => Array
                        (
                            [0] => Array
                                (
                                    [name] => id
                                    [value] =>
10613769-54e0-820d-de9b-5282a31b88cd
                                )

                            [1] => Array
                                (
                                    [name] => first_name
                                    [value] => Son
                                )

                            [2] => Array
                                (
                                    [name] => last_name
                                    [value] => Roan
                                )

                        )

                )

            [1] => Array
                (
                    [id] => 129b5f1d-5f1f-c679-beab-5282a325a501
                    [module_name] => Contacts
                    [name_value_list] => Array
                        (
                            [0] => Array
                                (
```

```
                                    [name] => id
                                    [value] =>
129b5f1d-5f1f-c679-beab-5282a325a501
                                )

                            [1] => Array
                                (
                                    [name] => first_name
                                    [value] => Pasquale
                                )

                            [2] => Array
                                (
                                    [name] => last_name
                                    [value] => Gottlieb
                                )

                        )

                )

        )

    [relationship_list] => Array
        (
            [0] => Array
                (
                    [link_list] => Array
                        (
                            [0] => Array
                                (
                                    [name] => email_addresses
                                    [records] => Array
                                        (
                                            [0] => Array
                                                (
                                                    [link_value] =>
Array

                                                        (
                                                            [0] =>
Array

                                                                (
```

```
[name] => id

[value] => 11980e8f-9cb6-0019-ad4a-5282a3e705cf
                                                                    )

                                                                [1] =>
Array
                                                                    (

[name] => email_address

[value] => kid76@example.us
                                                                    )

                                                                [2] =>
Array
                                                                    (

[name] => opt_out

[value] => 0
                                                                    )

                                                                [3] =>
Array
                                                                    (

[name] => primary_address

[value] =>
                                                                    )

                                                            )

                                                        )

                                                    [1] => Array
                                                        (
                                                            [link_value] =>
Array
                                                                (
```

```
                                                                [0] =>
Array
                                                                    (
[name] => id
[value] => 11c82450-4664-b210-4b79-5282a339d730
                                                                    )
                                                                [1] =>
Array
                                                                    (
[name] => email_address
[value] => info.dev@example.name
                                                                    )
                                                                [2] =>
Array
                                                                    (
[name] => opt_out
[value] => 1
                                                                    )
                                                                [3] =>
Array
                                                                    (
[name] => primary_address
[value] =>
                                                                    )
                                                            )
                                                        )
                                                    )
```

```
                                )

                            )

                        )

                [1] => Array
                    (
                        [link_list] => Array
                            (
                                [0] => Array
                                    (
                                        [name] => email_addresses
                                        [records] => Array
                                            (
                                                [0] => Array
                                                    (
                                                        [link_value] =>
Array
                                                            (
                                                                [0] =>
Array
                                                                    (

[name] => id

[value] => 13bf6b4b-4feb-823f-7c54-5282a36a2f57
                                                                    )

                                                                [1] =>
Array
                                                                    (

[name] => email_address

[value] => phone.vegan@example.tv
                                                                    )

                                                                [2] =>
Array
                                                                    (
```

```
[name] => opt_out

[value] => 0
                                                                    )

                                                                [3] =>
Array
                                                                    (

[name] => primary_address

[value] =>
                                                                    )

                                                            )

                                                        )

                                                    [1] => Array
                                                        (
                                                            [link_value] =>
Array
                                                                (
                                                                    [0] =>
Array
                                                                    (

[name] => id

[value] => 13ee5142-17c1-76fd-3cc4-5282a32cf864
                                                                    )

                                                                    [1] =>
Array
                                                                    (

[name] => email_address

[value] => kid.hr.sugar@example.biz
                                                                    )

                                                                    [2] =>
```

```
Array
                                                                    (
[name] => opt_out
[value] => 1
                                                                    )
                                                                [3] =>
Array
                                                                    (
[name] => primary_address
[value] =>
                                                                    )
                                                                )
                                                            )
                                                        )
                                                    )
                                                )
                                            )
                                        )
                                    )
                                )
                            )
                        )
                    )
                )
            )
        )
)
```

Last Modified: 06/30/2016 04:34pm

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 NuSOAP and the v4_1 SOAP 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/soap.php?wsdl";
    $username = "admin";
    $password = "password";

    //require NuSOAP
    require_once("./nusoap/lib/nusoap.php");

    //retrieve WSDL
    $client = new nusoap_client($url, 'wsdl');

    //display errors
    $err = $client->getError();
    if ($err)
    {
        echo '<h2>Constructor error</h2><pre>' . $err . '</pre>';
        echo '<h2>Debug</h2><pre>' .
htmlspecialchars($client->getDebug(), ENT_QUOTES) . '</pre>';
        exit();
    }

    //login
----------------------------------------------------------------

    $login_parameters = array(
         'user_auth' => array(
              'user_name' => $username,
              'password' => md5($password),
              'version' => '1'
         ),
         'application_name' => 'SoapTest',
```

```
        'name_value_list' => array(
        ),
    );

    $login_result = $client->call('login', $login_parameters);

    /*
    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 = $client->call('get_entries',
$get_entries_parameters);

    echo '<pre>';
    print_r($get_entries_result);
    echo '</pre>';

?>
```

Result

```
Array
(
    [entry_list] => Array
        (
            [0] => Array
                (
                    [id] => 20328809-9d0a-56fc-0e7c-4f7cb6eb1c83
                    [module_name] => Accounts
                    [name_value_list] => Array
                        (
                            [0] => Array
                                (
                                    [name] => name
                                    [value] => Jungle Systems Inc
                                )

                            [1] => Array
                                (
                                    [name] => billing_address_state
                                    [value] => NY
                                )

                            [2] => Array
                                (
                                    [name] => billing_address_country
                                    [value] => USA
                                )
```

```
                    )

                )

            [1] => Array
                (
                    [id] => 328b22a6-d784-66d9-0295-4f7cb59e8cbb
                    [module_name] => Accounts
                    [name_value_list] => Array
                        (
                            [0] => Array
                                (
                                    [name] => name
                                    [value] => Riviera Hotels
                                )

                            [1] => Array
                                (
                                    [name] => billing_address_state
                                    [value] => CA
                                )

                            [2] => Array
                                (
                                    [name] => billing_address_country
                                    [value] => USA
                                )

                        )

                )

        )

    [relationship_list] => Array
        (
        )

)
```

Last Modified: 06/30/2016 04:34pm

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 NuSOAP and the v4_1 SOAP 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/soap.php?wsdl";
    $username = "admin";
    $password = "password";

    //require NuSOAP
    require_once("./nusoap/lib/nusoap.php");

    //retrieve WSDL
    $client = new nusoap_client($url, 'wsdl');

    //display errors
    $err = $client->getError();
    if ($err)
    {
        echo '<h2>Constructor error</h2><pre>' . $err . '</pre>';
        echo '<h2>Debug</h2><pre>' .
htmlspecialchars($client->getDebug(), ENT_QUOTES) . '</pre>';
        exit();
    }

    //login
----------------------------------------------------------------
```

```
$login_parameters = array(
     'user_auth' => array(
          'user_name' => $username,
          'password' => md5($password),
          'version' => '1'
     ),
     'application_name' => 'SoapTest',
     'name_value_list' => array(
     ),
);

$login_result = $client->call('login', $login_parameters);

/*
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 = $client->call('search_by_module',
```

```
$search_by_module_parameters);

    /*
    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
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' => false,
        );

        $get_entries_results[$module] = $client->call('get_entries',
$get_entries_parameters);
    }

    echo '<pre>';
    print_r($get_entries_results);
    echo '</pre>';
```

Result

```
Array
(
    [Accounts] => Array
        (
            [entry_list] => Array
                (
                    [0] => Array
```

```
                        (
                            [id] =>
1bb7ef28-64b9-cbd5-e7d6-5282a3b96953
                            [module_name] => Accounts
                            [name_value_list] => Array
                                (
                                    [0] => Array
                                        (
                                            [name] => name
                                            [value] => Underwater
Mining Inc.
                                        )

                                )

                        )

                    [1] => Array
                        (
                            [id] =>
efbc0c4e-cc72-f843-b6ed-5282a38dad7f
                            [module_name] => Accounts
                            [name_value_list] => Array
                                (
                                    [0] => Array
                                        (
                                            [name] => name
                                            [value] => 360 Vacations
                                        )

                                )

                        )

                )

            [relationship_list] => Array
                (
                    [0] => Array
                        (
                            [link_list] => Array
                                (
```

```
                                    [0] => Array
                                        (
                                            [name] => email_addresses
                                            [records] => Array
                                                (
                                                    [0] => Array
                                                        (

[link_value] => Array
                                                                        (

[0] => Array

(

[name] => email_address

[value] => beans.the.qa@example.com

)


[1] => Array

(

[name] => opt_out

[value] => 0

)


[2] => Array

(

[name] => primary_address

[value] =>

)
```

)

)

[1] => Array
(

[link_value] => Array
(

[0] => Array

(

[name] => email_address

[value] => section.sales@example.edu

)

[1] => Array

(

[name] => opt_out

[value] => 0

)

[2] => Array

(

[name] => primary_address

[value] =>

)

```
                                                                    )

                                                                )

                                                            )

                                                        )

                                                    )

                                                )

                                    [1] => Array
                                        (
                                            [link_list] => Array
                                                (
                                                    [0] => Array
                                                        (
                                                            [name] => email_addresses
                                                            [records] => Array
                                                                (
                                                                    [0] => Array
                                                                        (

[link_value] => Array
                                                                                                    (

[0] => Array

(

[name] => email_address

[value] => section51@example.com

)


[1] => Array

(
```

[name] => opt_out

[value] => 0

)

[2] => Array

(

[name] => primary_address

[value] =>

)

)

)

[1] => Array
(

[link_value] => Array

(

[0] => Array

(

[name] => email_address

[value] => kid.sugar.qa@example.co.uk

)

[1] => Array

(

```
[name] => opt_out

[value] => 0

)


[2] => Array

(

[name] => primary_address

[value] =>

)
                                                                    )
                                                              )
                                                         )
                                                    )
                                               )
                                          )
                                     )
                                )
                           )
                      )
                 )
            )
        [Contacts] => Array
            (
                [entry_list] => Array
                    (
                        [0] => Array
                            (
                                [id] =>
24330979-0e39-f9ec-2622-5282a372f39b
```

```
                            [module_name] => Contacts
                            [name_value_list] => Array
                                (
                                    [0] => Array
                                        (
                                            [name] => name
                                            [value] => Errol Goldberg
                                        )

                                )

                        )

                    [1] => Array
                        (
                            [id] =>
2542dad2-85f3-5529-3a30-5282a3104e31
                            [module_name] => Contacts
                            [name_value_list] => Array
                                (
                                    [0] => Array
                                        (
                                            [name] => name
                                            [value] => Luis Deegan
                                        )

                                )

                        )

                )

            [relationship_list] => Array
                (
                    [0] => Array
                        (
                            [link_list] => Array
                                (
                                    [0] => Array
                                        (
                                            [name] => email_addresses
                                            [records] => Array
```

(
[0] => Array
(
[link_value] => Array
(
[0] => Array
(
[name] => email_address
[value] => hr.phone@example.com
)
[1] => Array
(
[name] => opt_out
[value] => 0
)
[2] => Array
(
[name] => primary_address
[value] =>
)
)
)

```
                                                    [1] => Array
                                                        (
[link_value] => Array
                                                                (
[0] => Array
(
[name] => email_address
[value] => the.info.phone@example.cn
)

[1] => Array
(
[name] => opt_out
[value] => 1
)

[2] => Array
(
[name] => primary_address
[value] =>
)
                                                                )
                                                        )
```

```
                                                )

                                            )

                                        )

                                    )

                                [1] => Array
                                    (
                                        [link_list] => Array
                                            (
                                                [0] => Array
                                                    (
                                                        [name] => email_addresses
                                                        [records] => Array
                                                            (
                                                                [0] => Array
                                                                    (

[link_value] => Array
                                                                                (

[0] => Array

(

[name] => email_address

[value] => im.the.sales@example.name

)


[1] => Array

(

[name] => opt_out

[value] => 0
```

)

[2] => Array

(

[name] => primary_address

[value] =>

)

)

)

[1] => Array
(

[link_value] => Array

(

[0] => Array

(

[name] => email_address

[value] => the36@example.com

)

[1] => Array

(

[name] => opt_out

[value] => 1

```
)

[2] => Array

(

[name] => primary_address

[value] =>

)

                                                                    )

                                                                )

                                                            )

                                                        )

                                                    )

                                                )

                                            )

                                        )

                                    )

                                )

                            )

                        )

                    )

                )

            )

        )

    [Leads] => Array
        (
            [entry_list] => Array
                (
                    [0] => Array
                        (
                            [id] =>
83abeeff-5e71-0f06-2e5f-5282a3f04f41
                            [module_name] => Leads
                            [name_value_list] => Array
                                (
                                    [0] => Array
```

```
                                    (
                                        [name] => name
                                        [value] => Caryn Wert
                                    )

                                )

                            )

                        [1] => Array
                            (
                                [id] =>
2a3b304b-5167-8432-d4e7-5282a300eabb
                                [module_name] => Leads
                                [name_value_list] => Array
                                    (
                                        [0] => Array
                                            (
                                                [name] => name
                                                [value] => Marco
Castonguay
                                            )

                                    )

                            )

                    )

                [relationship_list] => Array
                    (
                        [0] => Array
                            (
                                [link_list] => Array
                                    (
                                        [0] => Array
                                            (
                                                [name] => email_addresses
                                                [records] => Array
                                                    (
                                                        [0] => Array
                                                            (
```

```
[link_value] => Array
                                                                (
[0] => Array
(
[name] => email_address
[value] => hr60@example.com
)

[1] => Array
(
[name] => opt_out
[value] => 0
)

[2] => Array
(
[name] => primary_address
[value] =>
)
                                                                )
                                                            )
                                                        )
```

```
                                        )

                                    )

                                )

                            [1] => Array
                                (
                                    [link_list] => Array
                                        (
                                            [0] => Array
                                                (
                                                    [name] => email_addresses
                                                    [records] => Array
                                                        (
                                                            [0] => Array
                                                                (

[link_value] => Array
                                                                                (

[0] => Array

(

[name] => email_address

[value] => im.the@example.com

)


[1] => Array

(

[name] => opt_out

[value] => 0

)
```

```
[2] => Array

(

[name] => primary_address

[value] =>

)
                                                                    )
                                                                )
                                                            )
                                                        )
                                                    )
                                                )
                                            )
                                        )
                                    )
                                )
                            )
                        )
                    )
                )
            )
        )
    )
)
```

Last Modified: 06/30/2016 04:35pm

Retrieving Related Records

Overview

A PHP example demonstrating how to retrieve a list of related records with the get_relationships method using NuSOAP and the v4_1 SOAP API.

This example will retrieve a list of leads related to a specific target list.

Example

```
<?php

    $url = "http://{site_url}/service/v4_1/soap.php?wsdl";
    $username = "admin";
    $password = "password";

    //require NuSOAP
    require_once("./nusoap/lib/nusoap.php");

    //retrieve WSDL
    $client = new nusoap_client($url, 'wsdl');

    //display errors
    $err = $client->getError();
    if ($err)
    {
        echo '<h2>Constructor error</h2><pre>' . $err . '</pre>';
        echo '<h2>Debug</h2><pre>' .
htmlspecialchars($client->getDebug(), ENT_QUOTES) . '</pre>';
        exit();
    }

    //login ----------------------------------------------------

    $login_parameters = array(
         'user_auth' => array(
              'user_name' => $username,
              'password' => md5($password),
              'version' => '1'
         ),
         'application_name' => 'SoapTest',
         'name_value_list' => array(
         ),
    );

    $login_result = $client->call('login', $login_parameters);
```

```
    /*
    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
return 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
to 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 = $client->call("get_relationships",
$get_relationships_parameters);

    echo "<pre>";
    print_r($get_relationships_result);
    echo "</pre>";

?>
```

Result

```
Array
(
    [entry_list] => Array
        (
            [0] => Array
                (
                    [id] => 117c26c0-11d4-7b8b-cb8f-4f7cb6823dd8
                    [module_name] => Leads
                    [name_value_list] => Array
                        (
                            [0] => Array
                                (
                                    [name] => id
                                    [value] =>
117c26c0-11d4-7b8b-cb8f-4f7cb6823dd8
                                )

                            [1] => Array
```

```
                                (
                                    [name] => first_name
                                    [value] => Diane
                                )

                            [2] => Array
                                (
                                    [name] => last_name
                                    [value] => Mckamey
                                )

                        )

                )

            [1] => Array
                (
                    [id] => 142faeef-1a19-b53a-b780-4f7cb600c553
                    [module_name] => Leads
                    [name_value_list] => Array
                        (
                            [0] => Array
                                (
                                    [name] => id
                                    [value] =>
142faeef-1a19-b53a-b780-4f7cb600c553
                                )

                            [1] => Array
                                (
                                    [name] => first_name
                                    [value] => Leonor
                                )

                            [2] => Array
                                (
                                    [name] => last_name
                                    [value] => Reser
                                )

                        )
```

```
                )

        )

    [relationship_list] => Array
        (
        )

)
```

Last Modified: 06/30/2016 04:36pm

Searching Records

Overview

A PHP example demonstrating how to search the accounts module with the search_by_module method using NuSOAP and the v4_1 SOAP 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/soap.php?wsdl";
    $username = "admin";
    $password = "password";

    //require NuSOAP
    require_once("./nusoap/lib/nusoap.php");

    //retrieve WSDL
    $client = new nusoap_client($url, 'wsdl');

    //display errors
    $err = $client->getError();
```

```
    if ($err)
    {
        echo '<h2>Constructor error</h2><pre>' . $err . '</pre>';
        echo '<h2>Debug</h2><pre>' .
htmlspecialchars($client->getDebug(), ENT_QUOTES) . '</pre>';
        exit();
    }

    //login ------------------------------------------------

    $login_parameters = array(
         'user_auth' => array(
              'user_name' => $username,
              'password' => md5($password),
              'version' => '1'
         ),
         'application_name' => 'SoapTest',
         'name_value_list' => array(
         ),
    );

    $login_result = $client->call('login', $login_parameters);

    /*
    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
listviewdefs.
        '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 = $client->call('search_by_module',
$search_by_module_parameters);

    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
        (
        )

)
```

Last Modified: 06/30/2016 04:36pm

Migration

An overview of migrating data.

Last Modified: 09/26/2015 04:14pm

Importing Records

Best practices when importing records.

Last Modified: 09/26/2015 04:14pm

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 a 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"
      }
   ]
}
```

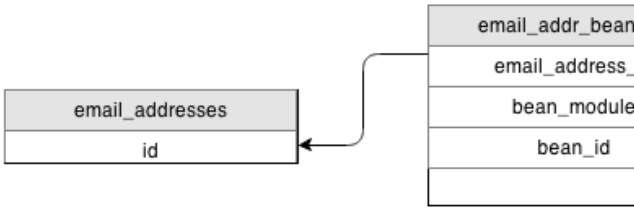
For more information on the /<module>/:record PUT endpoint, you can refer to your instance's help documentation found at:

`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 know and 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 duplicated in the system but can occasionally be missing critical data. Each bean record with an email address assigned to it, should in turn 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 cleanup 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;
```

Last Modified: 01/15/2016 09:30pm

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:

1. Have full permissions to modify the system files.
2. Identify your Sugar version. The version of your instance can be found by navigating to the About page in your SugarCRM instance from the global links menu.

3. Once you have identified your Sugar version, you will need to download the full installer package of your Sugar version from the Support Portal. If you are on 7.2.0 Pro, you must download the SugarPro-7.2.0.zip package or this 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/
    - ./uploads/

    - ./cache/images/ - This is optional and contains the embedded emails displayed in the UI.
    - 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.

Last Modified: 09/26/2015 04:14pm

Migrating From On-Demand to On-Site

Overview

Occasionally system administrators will have the need to deploy versions of their On-Demand instance to an On-Site system. Reasons for this type of deployment are:

- Testing locally developed modules
- Migrating from On-Demand to On-Site
    - Note: On-Demand only releases such as 6.7 are not supported for migrating to On-Site.

Prerequisites

- Before migrating Sugar to an On-Site environment, you will need to request a backup of your On-Demand 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.
   - Locate this group of lines and remove the entire set 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=301,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 On-Demand instance.

You should now have a working version of your On-Demand instance accessible from your local server.

Last Modified: 09/07/2016 02:48pm

Migrating From On-Site to On-Demand

Overview

The following article describes the process of migrating an On-Site deployment to the SugarCRM On-Demand environment.

Requirements

A few requirements must be met before an instance can be migrated to the SugarCRM On-Demand environment:

1. 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.
2. The instance must be updated to a supported version of Sugar. To find out if your current version is supported you can check our Supported Versions. Please refer to the appropriate Administrator/Upgrade Guide for your version of SugarCRM if you need to upgrade your instance (http://support.sugarcrm.com/02_Documentation)

How can I find the version of my SugarCRM instance?

The version of your instance can be found by navigating to the About page in your SugarCRM instance from the global links menu. Once identified you can check this version against our Supported Versions.

Migration

Once the above requirements have been met you are ready to migrate.

Support Portal

Inform SugarCRM Customer Support of your intention to migrate by opening a case through the support portal (http://www.sugarcrm.com/support/portal). They will provide you with an FTP site and a set of credentials so you can transfer your instance. SugarCRM Customer 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 SugarCRM 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 SugarCRM 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 Setup 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 your SugarCRM instance, archive the entire contents of the SugarCRM root directory using the archive utility of your choice (zip, tar, WinZip, WinRar, 7zip, etc.). The SugarCRM root directory is the directory that contains the files "config.php", "cron.php", "sugarcrm.log" and the folders "custom", "cache", "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
password] [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 SugarCRM 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.sql
```

Upload

Finally, upload the two files to the FTP site provided by the SugarCRM Customer Support team. The instance will be deployed to the SugarCRM On-Demand environment and a URL to the instance will be provided to you.

Last Modified: 01/15/2016 09:30pm

Integration

Overview

The following sections will outline best practices when integrating with Sugar.

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 post to an endpoint such as rest/v10/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.

An example of such a request is shown here:

```
curl -v -H "oauth-token: 4afd4aea-df99-7cec-4d94-560a97cda9f8" -H
"Content-Type: application/json" -X POST -d '{"name": "Example
Account"}'  http://<site_url>/rest/v10/Accounts
```

If this (or a similar) request is identified as the process delaying the system, remedy the situation using one of the approaches in the sections below.

Queuing Data in the Job Queue

One solution is to send the data to the Job Queue to be processed by the cron during one of its cycles. 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
"Content-Type: application/json" -X POST -d '{"assigned_user_id": "1",
"name":"Queue Create Account", "status":"queued", "data":"{\"name\":
\"Example Account\"}", "target":"function::CustomCreateAccountJob"}'
http://<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.

Last Modified: 09/29/2015 01:37pm

Backward Compatibility

Backward Compatibility Overview.

Last Modified: 09/26/2015 04:14pm

Introduction to Backward Compatibility Mode

Overview

As of Sugar 7, modules are built using the Sidecar framework as the MVC architecture is being deprecated. Modules that have not yet been moved over to the Sidecar framework are set in what Sugar refers to as "backward compatibility" (BWC) mode.

Currently, the Studio-enabled modules in backward compatibility are:

- Campaigns
- Contracts
- Documents
- Employees
- Knowledge Base
- Quotes
- Users

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 EditView and DetailView 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

Last Modified: 09/26/2015 04:14pm

Enabling Backward Compatibilty

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 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' => $moduleName,
        'icon' => 'icon-reorder',
    ),
    array(
        'route' =>
"#bwc/index.php?module=Import&action=Step1&import_module=$moduleName&r
eturn_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' => $moduleName,
        'icon' => 'icon-reorder',
    ),
    array(
        'route' =>
"#bwc/index.php?module=Import&action=Step1&import_module=$moduleName&r
eturn_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._dev_data.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._dev_data.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._dev_data.modules.<module key>.isBwcEnabled
   ```

Last Modified: 07/14/2016 10:53am

Converting Legacy Modules To Sidecar

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.

1. 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.
2. 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/
   ```
3. 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
   ```
4. 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.
5. 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.

Last Modified: 09/26/2015 04:14pm

Performance Tuning

An overview of performance settings and best practices.

Last Modified: 09/26/2015 04:14pm

Sugar Performance

Overview

As your company uses Sugar over time, the size of your database will naturally

grow and performance will eventually degrade without performing proper maintenance. The purpose of this article is to review some of the most common recommendations for increasing the performance in Sugar to help increase the efficiency amongst your users.

Note: This guide is intended for OnSite installations. Customers hosted on OnDemand 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 to keep 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

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';
```

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;
```

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 recomended 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.

Last Modified: 01/15/2016 09:47pm

PHP Profiling

Overview

Profile the performance of customizations and the overall application.

XHProf

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}/cache/xhprof';
// x where x is a number and 1/x requests are profiled. So to sample
all requests set it to 1
$sugar_config['xhprof_config']['sample_rate'] = 1;
// array of function names to ignore from the profile (pass into
xhprof_enable)
$sugar_config['xhprof_config']['ignored_functions'] = array();
// flags for xhprof
$sugar_config['xhprof_config']['flags'] = XHPROF_FLAGS_CPU +
XHPROF_FLAGS_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.

Last Modified: 01/08/2017 04:38pm

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 7 only.

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. On-demand customers who are experiencing performance-related issues should contact the Sugar Support team for assistance.
- 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

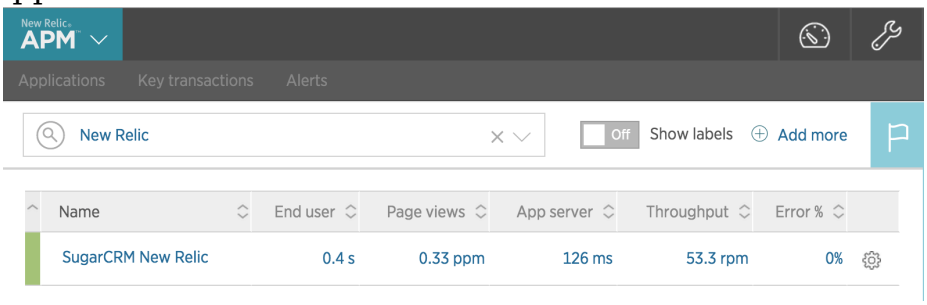
instance.

- 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

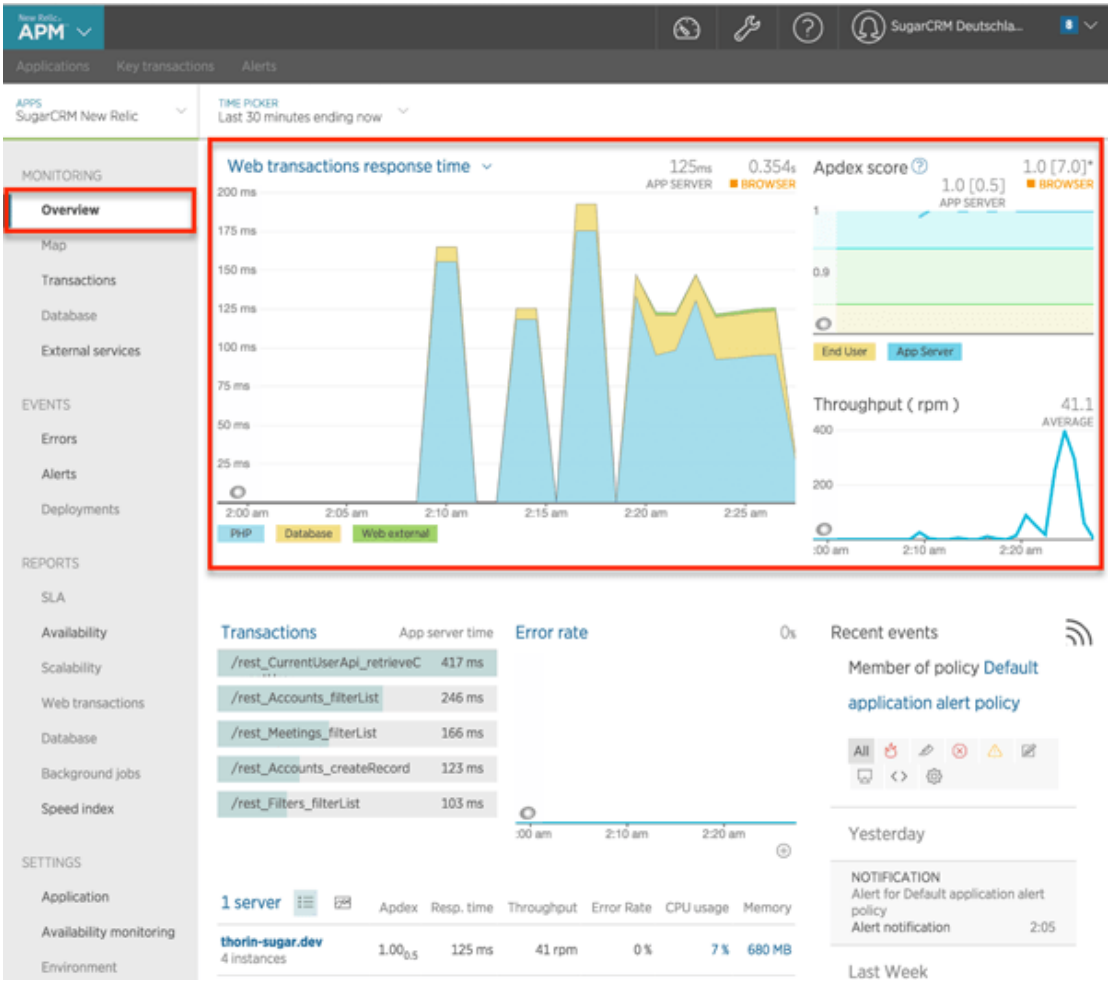
New Relic integrated with Sugar enables you to view the exact functions that cause unusual performance behavior in your instance, such as unexpected triggering of logic hooks, database queries that should be optimized, or customizations that are responding slower than expected.

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.

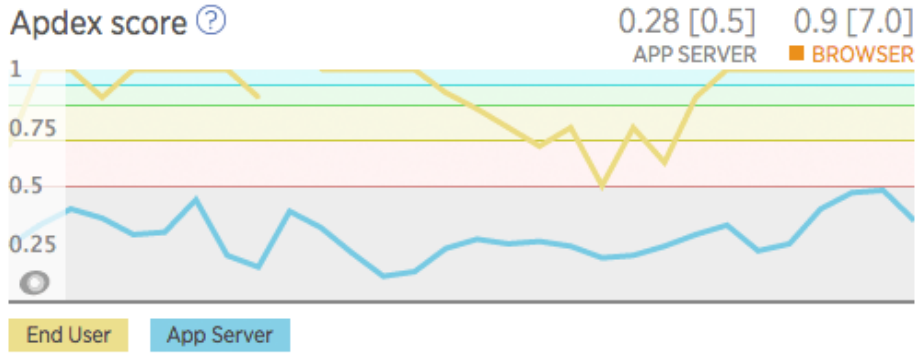


Overview Dashboard

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.

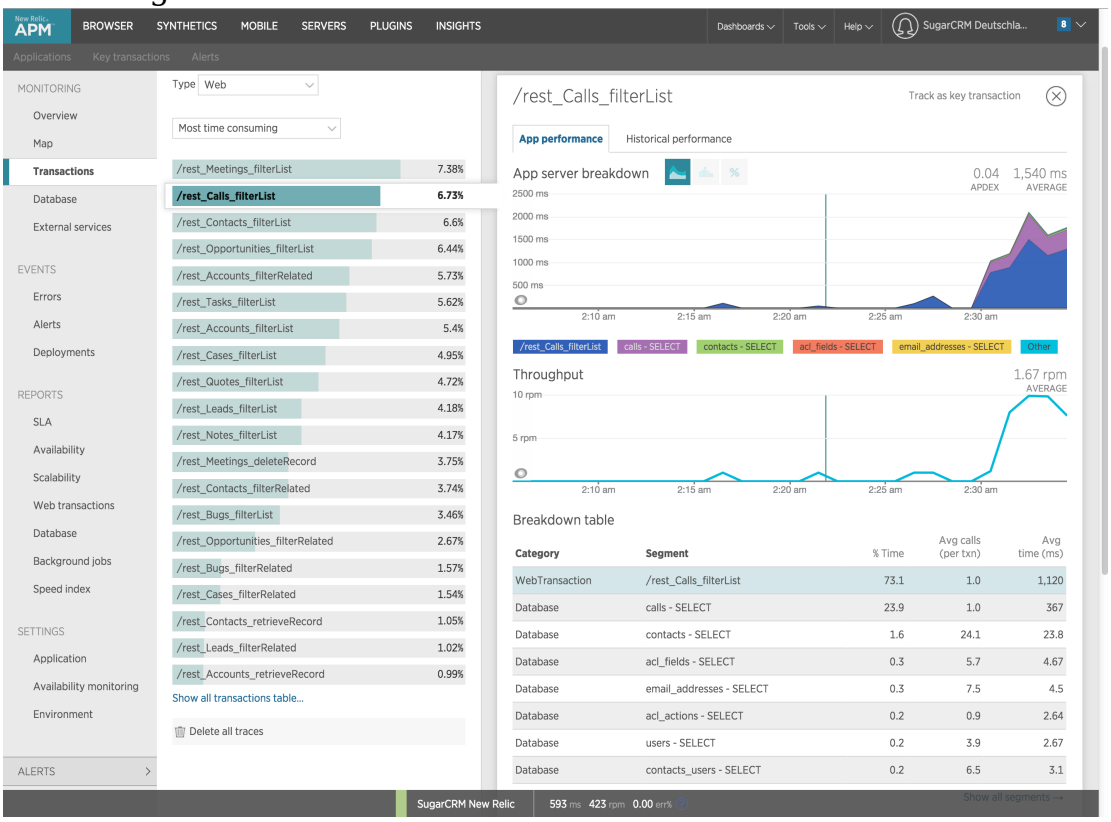


The default Apdex T-value for New Relic is 0.5 seconds but it can be configured to match your current environment and user expectations. For information on

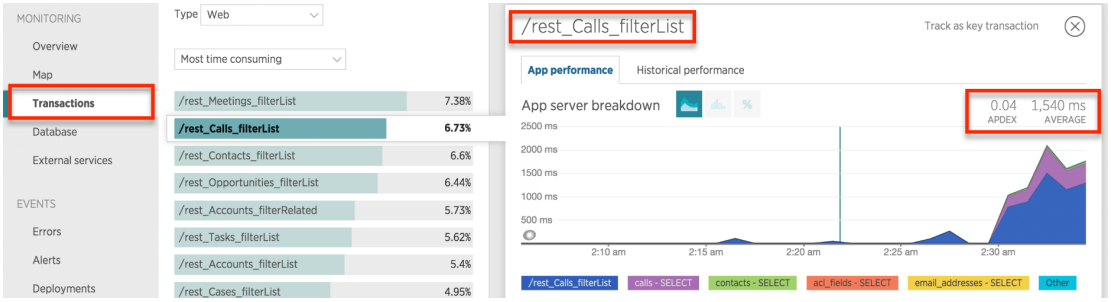
changing the Apdex T-value, please refer to the Change Your Apdex Settings article in the New Relic documentation.

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.

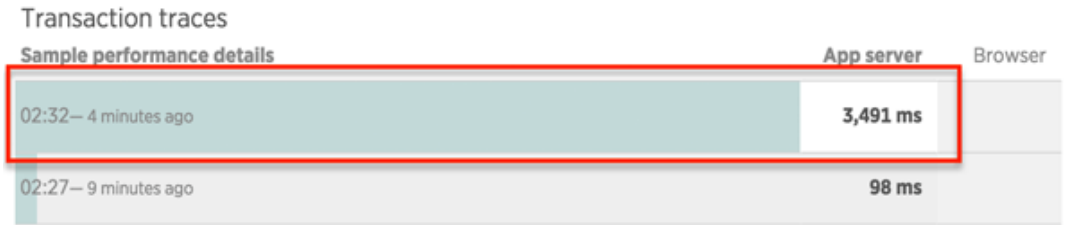


Refer to the breakdown table in the lower part of the screen to see which part of the call is taking the most time, with a representation of the performed actions on the database per segment.

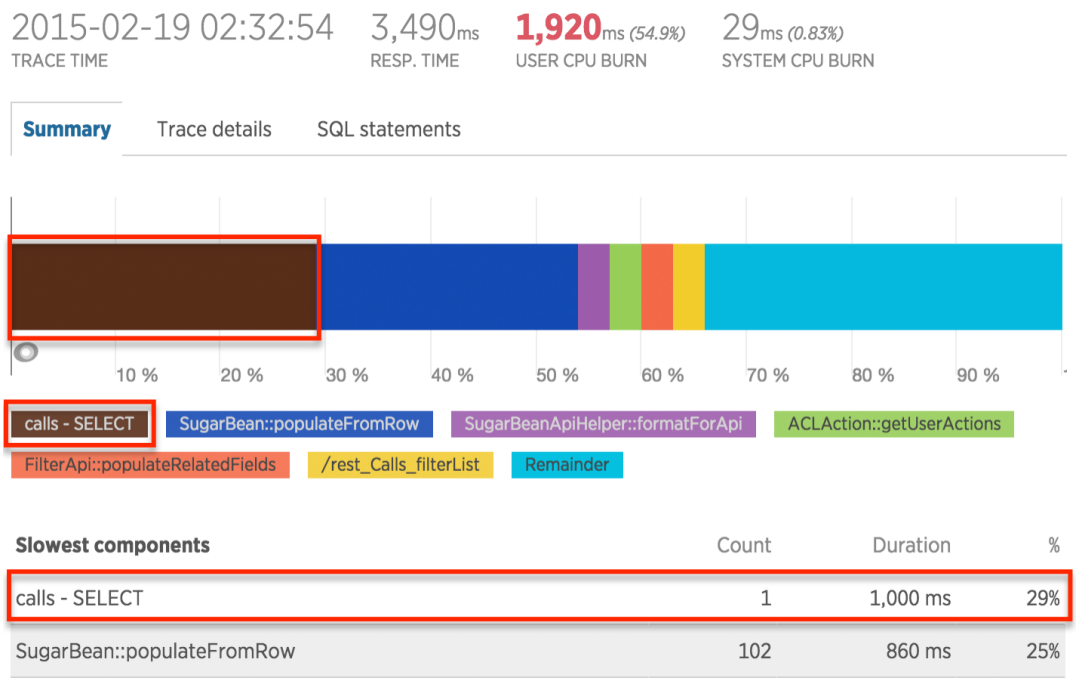
Category	Segment	% Time	(per txn)	time (ms)
WebTransaction	/rest_Calls_filterList	73.1	1.0	1,120
Database	calls - SELECT	23.9	1.0	367
Database	contacts - SELECT	1.6	24.1	23.8
Database	acl_fields - SELECT	0.3	5.7	4.67
Database	email_addresses - SELECT	0.3	7.5	4.5
Database	acl_actions - SELECT	0.2	0.9	2.64
Database	users - SELECT	0.2	3.9	2.67
Database	contacts_users - SELECT	0.2	6.5	3.1

Show all segments →

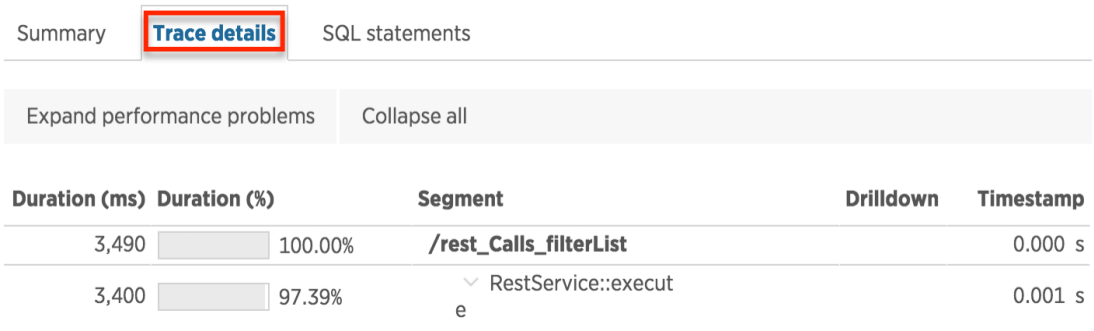
Below the breakdown table is a list of transaction traces. New Relic will automatically generate a transaction trace when a response time is in the frustrating zone of the Apdex or slower. Click on the transaction trace to learn what is having the negative impact on the performance for this activity.



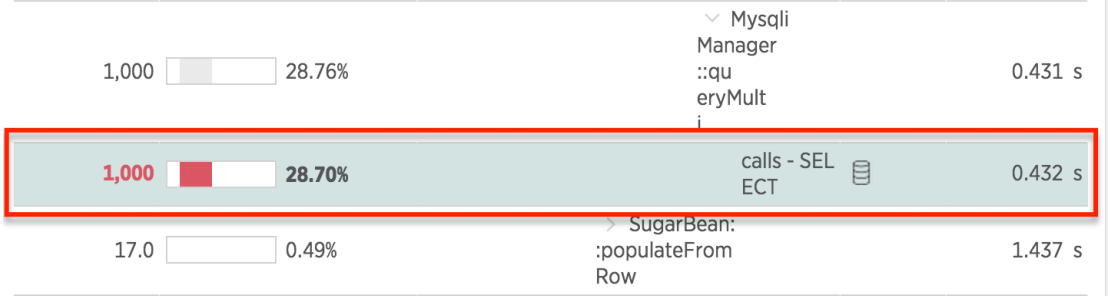
The transaction trace shows how long various components took to load. In this case, the SELECT query on the Calls table took a significant amount of time.



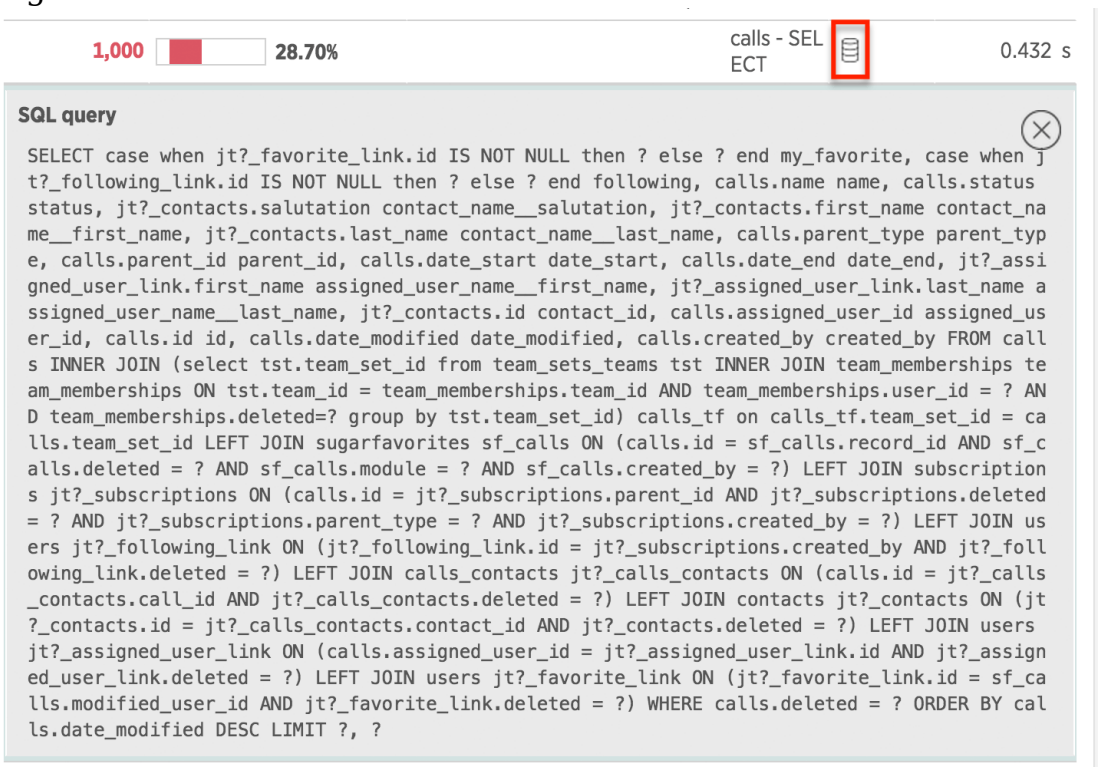
Click on the Trace Details tab above the chart to investigate further. The details page displays specific functions that are being called and how long they took to execute. By drilling down in the tree to the child functions, it is possible to find the root cause of the impaired performance.



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.

Last Modified: 09/27/2015 08:09pm