
Sugar Developer Guide 7.8

Sugar Developer Guide 7.8	171
Introduction	171
Overview	171
Prerequisites	171
Understanding Sugar's Framework	171
Supported Platforms	172
Sugar Editions	172
Basic Development Rules for Sugar Products	172
Development Tools	173
Developer Mode	173
Diagnostic Tool	173
Composer	174
Development Methodology	174
Overview	174
Development Best Practices	174
Using .gitignore Files	174
Recommendations for Development Teams	175
Deploying Sugar Code	175
Packaging	176
Deployment	177
Managing Multiple Environments	177
Consistency	178
Testing	178
Sugar Test Tools	179
Sugar Unit Tests	179
Sugar Performance Tests	

DevOps	180
Co-Existing with Studio Customizations	180
Composer	181
Overview	182
Prerequisites	182
Restrictions	182
Finding Packages	183
Adding a New Package	183
Removing Packages	185
Adding and Removing Repositories	185
Autoloader	185
Integrations	185
Internals	186
Optimization	186
Developer Mode	186
Handling Upgrades	186
Upgrade Failure Notification	187
Web Upgrader	187
CLI Upgrader	187
Example Failure	188
Process	188
What's Wrong?	189
The Proposal	189
Resolving the Issues	189
Retry Upgrade	

.....	190
Stock Composer Configuration	191
Frequently Asked Questions	191
Delivery and Deployment Guide for Enterprises	193
Overview	193
Deploying Application Configuration and Metadata	193
config_override.php	194
database 'config' table	194
Use Case: Deployment of Reports	195
Use Case: Deployment of Dashboards	196
Use Case: Deployment of Roles	197
Use Case: Deployment of Teams	198
Use Case: Deployment of User Settings	199
User preferences	199
User table	200
Use Case: Deployment of custom fields	201
Use Case: Deployment of custom modules	201
Use Case: Deployment of custom Relationships	203
Use Case: Deployment of custom View or Layout metadata (Web)	203
Use Case: Deployment of custom View or Layout metadata (mobile)	204
Deploying Application Code and Integrations	204
Use Case: Deployment of Logic Hooks and Web Logic Hooks	205
Use Case: Deployment of custom API endpoints	205
Use Case: Deployment of custom Administration Panels	206
Use Case: Deployment of custom Jobs / Schedulers	207

Migration Guide	207
Purpose	208
Expected to Affect Most Developers	208
Support for PHP 5.6 only	208
Upgrade to Backbone v1.2.3	208
Expected to Affect Some Developers	209
Removal of deprecated or empty Sidecar controllers, classes, and methods	209
Sugar Dashlets	209
Sidecar Methods	209
Sidecar Controllers	210
Enforcement of CSRF Tokens	210
jQuery library upgrade	211
Underscore library upgrade in Sidecar framework	211
Changes in application footer Handlebars template	212
Changes to the Preview layout	212
The oauth_token URL parameter is no longer enabled by default	212
Expected to Affect Few Developers	213
Removal of external Job Queue framework classes added in Sugar 7.7	213
Removal of unused code in Advanced Workflow	214
Removal of unused SugarMVC Notifications controller	214
TinyMCE library upgrade for Sidecar	215
Timepicker library upgrade	215
Extraneous render in Record View	215
Removal of oracle_enable_ci setting	215
Deprecation of Tooltip and EllipsisInline Sidecar plug-ins	216
Sidecar Relate Field changes	

Sidecar Alert auto-close changes	216
User Interface	217
Overview	217
Clients	217
base	217
mobile	217
portal	218
Sidecar	218
Overview	218
Composition	219
Backbone.js	219
Components	220
Layouts	220
Views	220
Fields	221
Context	221
Events	221
Overview	221
Existing Backbone Event Catalog	222
Sidecar Events	222
Bean Events	223
Context Events	224
Utilizing Events	224
Routes	225
Overview	

Routes	225
Route Definitions	225
Route Patterns	226
Custom Routes	226
route()	226
Example	227
addRoutes()	227
Arguments	228
Example	228
Handlebars	229
Overview	229
Templates	229
Debugging Templates	230
Helpers	230
Creating Helpers	230
Layouts	232
Overview	232
Hierarchy Diagram	233
Sidecar Layout Routing	233
Layout Example	233
JavaScript	234
Layout Definition	234
Application	235
Creating Layouts	235
Overview	

.....	235
Creating the Layout	236
Creating a View	236
Navigating to the Layout	237
Overriding Layouts	237
Overview	237
Overriding the Layout	238
Extending the View	241
Views	242
Overview	242
Load Order Hierarchy Diagram	242
Components	243
Controller	243
Handlebar Template	243
Extending Views	243
Basic View Example	244
Controller	244
Attributes	245
Handlebar Template	245
Helpers	246
Metadata	246
Overview	246
View Metadata Framework	246
View Metadata	246
Removing the Account Requirement on Opportunities	251
Overview	

.....	251
Removing the Requirement in Configuration	252
Removing the Requirement in View Metadata	252
Application	253
Adding Buttons to the Record View	254
Overview	254
Steps To Complete	254
Defining the Metadata	254
Adding Custom Buttons	258
Defining the Button Label	266
Extending and Overriding the Controller	266
Adding Field Validation to the Record View	267
Overview	267
Error Messages	267
Custom Error Messages	268
Method 1: Extending the RecordView and CreateView Controllers	269
Method 2: Overriding the RecordView and CreateView Layouts	271
Passing Data to Templates	279
Overview	279
Steps to Complete	279
Refreshing Subpanels on the RecordView	280
Overview	280
Refreshing Subpanels	280
Adding the Button Metadata	280
Fields	284
Overview	

.....	284
Hierarchy Diagram	284
Field Example	284
Controller	284
Attributes	285
Handlebar Templates	285
Helpers	286
Creating Custom Field Types	286
Overview	286
Naming a Custom Field Type in Sugar	287
Creating the JavaScript Controller	287
Defining the Handlebar Templates	288
Detail View	288
Edit View	289
List View	289
Adding the Field Type to Studio	290
Creating the Form Controller	293
Creating the Smarty Template	294
Registering the Field Type	296
Creating Language Definitions	297
Adding the Custom Field to the Type List	297
Creating Labels for Studio	297
Defining Dropdown Controls	298
Enabling Search and Filtering	299
Defining the Filter Operators	299
Defining the Sugar Widget	

.....	299
Converting Address' Country Field to a Dropdown	301
Overview	301
Use Case	301
Prerequisites	301
Steps to Complete	302
Updating the Field Type for List View Filter	303
Application	304
Subpanels	305
Overview	305
Hierarchy Diagram	306
Subpanels and Subpanel Layouts	307
Adding Subpanel Layouts	308
Sidecar Layouts	308
Legacy MVC Subpanel Layouts	309
Fields Metadata	310
Hierarchy Diagram	311
Subpanel List Views	311
Dashlets	313
Overview	314
Hierarchy Diagram	314
Dashlet Views	314
Preview	314
Dashlet View	315
Configuration View	315
Dashlet Example	

Metadata	316
Controller	318
Workflow	323
Retrieving Data	323
Handlebar Template	323
Drawers	325
Overview	325
Methods	325
Parameters	325
Example	326
app.drawer.close(callbackOptions)	326
Parameters	326
Standard Example	326
Callback Example	326
app.drawer.load(options)	327
Parameters	327
Example	327
app.drawer.reset(triggerBefore)	327
Parameters	327
Example	327
Alerts	328
Overview	328
Methods	328
Parameters	328
Default Alert Values	

Alert Examples	329
Confirmation Alert	329
Process Alert	329
app.alert.dismiss(id)	330
Parameters	330
Example	330
app.alert.dismissAll	330
Example	330
Testing in Console	330
Language	330
Overview	331
Methods	331
Parameters	331
Example	331
app.lang.getAppString(key)	331
Parameters	331
Example	332
app.lang.getAppListStrings(key)	332
Parameters	332
Example	332
app.lang.getModuleSingular(moduleKey)	332
Parameters	332
Example	332
app.lang.getLanguage()	333
Example	

.....	333
app.lang.updateLanguage(languageKey)	333
Parameters	333
Example	333
Testing in Console	333
MegaMenu	333
Overview	333
Layout Components	334
Link Actions	334
Module Links	335
Module Action Links	335
Adding Module Action Links	335
Removing Module Action Links	336
Profile Action Links	337
Adding Profile Action Links	337
Removing Profile Action Links	338
Administration Links	339
Overview	339
Example	339
Legacy MVC	341
Overview	341
Model-View-Controller (MVC) Overview	341
SugarCRM MVC Implementation	341
View	342
Overview	342
What are Views?	

Implementation	342
Class Loading	343
Methods	344
Creating Views	344
Overriding Views	345
Display Options for Views	346
Implementation	346
Controller	347
Overview	347
Controllers	347
Upgrade-Safe Implementation	348
File Structure	348
Implementation	348
Mapping Actions to Files	349
Upgrade-Safe Implementation	350
File Structure	350
Implementation	350
Classic Support (Not Recommended)	350
File Structure	350
Controller Flow Overview	350
Metadata	351
Overview	351
Metadata Framework	351
Application Metadata	352
Module Metadata	

.....	352
SearchForm Metadata	353
DetailView and EditView Metadata	356
Examples	365
Hiding the Quotes Module PDF Buttons	365
Overview	365
Hiding the PDF Buttons	365
Manipulating Buttons on Legacy MVC Layouts	368
Overview	368
Metadata	368
Custom Layouts	368
Editing Layouts	369
Developer Mode	369
Quick Repair and Rebuild	369
Saving & Deploying the Layout in Studio	369
Adding Custom Buttons	370
JavaScript Actions	370
Example	370
Submitting the Stock View Form	371
Example	371
Submitting Custom Forms	373
Example	373
Removing Buttons	374
Manipulating Layouts Programmatically	376
Overview	376
The ParserFactory	

.....	376
Modifying Layouts to Display Additional Columns	376
Overview	376
Resolution	377
On-Site	378
Examples	380
Changing the ListView Default Sort Order	380
Overview	380
Customization Information	380
Extending the Search Form	380
Extending the List View	381
Extending the Sugar Controller	383
Data Framework	384
Modules	384
Overview	385
Module Definitions	385
Hierarchy Diagram	385
\$moduleList	385
\$beanList	385
\$beanFiles	386
\$modInvisList	386
\$modules_exempt_from_availability_check	386
\$report_include_modules	386
\$adminOnlyList	387
\$bwcModules	387

Models	387
Overview	387
SugarObject Templates	387
SugarObject Interfaces	388
File Structure	388
Implementation	388
Performance Considerations	389
Cache Files	389
SugarBean	389
Overview	389
CRUD Handling	390
Creating and Retrieving Records	390
Obtaining the Id of a Recently Saved Bean	390
Saving a Bean with a Specific ID	390
Distinguishing New from Existing Records	391
Retrieving a Bean by Unique Field	392
Updating a Bean	393
Updating a Bean Without Changing the Date Modified	393
Deleting a Bean	394
Fetching Relationships	394
Fetching Related Records	394
Fetching a Parent Record	395
BeanFactory	396
Overview	396
Creating a SugarBean Object	396
newBeanByName()	396

Retrieving a SugarBean Object	396
retrieveBean()	397
Retrieving Module Keys	397
getBeanName()	397
Vardefs	398
Overview	398
Dictionary Array	398
Duplicate Check Array	399
Fields Array	399
Indices Array	401
Relationships Array	401
Visibility Array	402
Extending Vardefs	402
Manually Creating Custom Fields	402
Overview	403
Using ModuleInstaller to Create Custom Fields	403
Using the Vardef Extensions	406
Specifying Custom Indexes for Import Duplicate Checking	409
Overview	409
Resolution	409
Working With Indexes	410
Overview	411
Index Metadata	411
Creating Indexes	411
Removing Indexes	

Example	412
Creating Indexes for Import Duplicate Checking	413
Example	414
Fields	415
Overview	415
SugarField Widgets	415
Implementation	415
Relationships	416
Overview	416
Definitions	417
Database Structure	417
Relationship Cache	417
Custom Relationships	418
Overview	418
Creating Custom Relationships	418
Defining the Relationship MetaData	418
MetaData Example	419
Defining the Relationship in the TableDictionary	421
TableDictionary Example	422
Database	422
Overview	422
Primary Keys, Foreign Keys, and GUIDs	423
Indexes	424
DBManagerFactory	424
Overview	

.....	425
Instantiating a DB Object	425
Querying The Database	425
Retrieving a Single Result	425
Limiting Results	426
Generating SQL Queries	426
Select Queries	426
Count Queries	426
SugarQuery	427
Overview	427
Setup	427
Basic Usage	427
from()	428
Arguments	428
Returns	429
select()	429
Arguments	429
Returns	429
where()	430
Arguments	430
Returns	430
Relationships	430
Arguments	430
Returns	431
joinTable()	431
Arguments	431

Returns	432
Altering Results	432
distinct()	432
Arguments	433
Returns	433
limit()	433
Arguments	433
Returns	433
offset()	434
Arguments	434
Returns	434
orderBy()	434
Arguments	434
Returns	435
Execution	435
execute()	435
Arguments	436
Returns	436
compileSql()	436
Arguments	437
Returns	437
SugarQuery Conditions	437
Overview	437
Where Clause	437
equals() notEquals()	

	437
Arguments	438
Returns	438
equalsField() notEqualsField()	438
Arguments	438
Returns	438
isEmpty() isEmpty()	439
Arguments	439
Returns	439
isNull() notNull()	439
Arguments	439
Returns	440
contains() notContains()	440
Arguments	440
Returns	440
starts() ends()	440
Arguments	441
Returns	441
in() notIn()	441
Arguments	442
Returns	442
between()	442
Arguments	442
Returns	443
lt() lte() gt() gte()	443
Arguments	

Returns	443
dateRange()	444
Arguments	444
Returns	444
dateBetween()	444
Arguments	445
Returns	445
Combinations	445
queryAnd()	445
queryOr()	446
Advanced Techniques	446
Overview	447
Get First Record	447
getOne()	447
Aggregates	447
Arguments	448
Returns	448
Unions	448
union()	448
Arguments	449
Returns	449
Raw Methods	449
whereRaw()	450
Arguments	450
Returns	450

joinRaw()	450
Arguments	451
Returns	451
groupByRaw()	452
Arguments	452
Returns	452
orderByRaw()	452
Arguments	452
Returns	453
havingRaw()	453
Arguments	453
Returns	453
Architecture	453
Overview	453
Platform	454
Front-End Framework	455
Metadata	456
Extensions	456
Autoloader	457
Overview	457
SugarAutoLoader	457
Included File Extensions	457
Class Loading Directories	457
Ignored Directories	458
Initializing the AutoLoader	

Rebuilding the File Map	458
The buildCache Function	459
Configuration API	459
Overview	459
addDirectory(\$dir)	459
addPrefixDirectory(\$prefix, \$dir)	459
File Check API	460
Overview	460
existing(...)	460
existingCustom(...)	460
existingCustomOne(...)	460
fileExists(\$filename)	461
getDirFiles(\$dir, \$get_dirs = false, \$extension = null)	461
getFilesCustom(\$dir, \$get_dirs = false, \$extension = null)	461
lookupFile(\$paths, \$file)	462
requireWithCustom(\$file, \$both = false)	462
File Map Modification API	462
Overview	462
addToMap(\$filename, \$save = true)	463
delFromMap(\$filename, \$save = true)	463
put(\$filename, \$data, \$save = false)	463
touch(\$filename, \$save = false)	463
unlink(\$filename, \$save = false)	463
Metadata API	464
Overview	

.....	464
Metadata Loading	464
loadWithMetatables(\$module, \$varname)	464
loadPopupMeta(\$module, \$metadata = null)	465
loadExtension(\$extname, \$module = "application")	465
Caching	465
Overview	465
Developer Mode	466
Uploads	466
Overview	466
Uploads	466
Upload Extensions	467
How Files Are Stored	467
Email Attachments	467
Picture Fields	468
Document Attachments	468
Knowledge Base Attachments	468
Module Loadable Packages	469
CSV Imports	469
Working with File Uploads	469
Overview	469
Retrieving a Files Upload Location	470
Retrieving a Files Full File System Location	470
Retrieving a Files Contents	470
Duplicating a File	470

Mailer Factory	471
Overview	471
Mailers	471
System Mailer	471
Example	471
User Mailer	471
Example	472
Populating the Mailer	472
Example	472
Setting the Body	472
Example	472
Adding Recipients	472
Example	472
Clearing Recipients	473
Example	473
Adding Attachments	473
Example	473
Sending Emails	473
Example	473
Logging	475
Overview	475
Log Levels	475
Logging Messages	475
Debugging Messages	476
setLevel	477
Log Rotation	

.....	477
Creating Custom Loggers	477
Custom Loggers	477
Logic Hooks	479
Overview	479
Hook Definitions	479
Locations	479
Definition Properties	480
hook_version	480
hook_array	480
Hook Method	481
Namespaced Hooks	482
Hooks without Namespaces	483
Considerations	483
Application Hooks	484
after_entry_point	484
Overview	484
Definition	484
Arguments	484
Considerations	485
Change Log	485
Example	485
after_load_user	486
Overview	486
Definition	486
Arguments	486

Change Log	486
Example	487
after_session_start	488
Overview	488
Definition	488
Arguments	488
Change Log	489
Example	489
after_ui_footer	490
Overview	490
Definition	490
Arguments	490
Considerations	490
Change Log	491
Example	491
after_ui_frame	492
Overview	492
Definition	492
Arguments	492
Considerations	493
Change Log	493
Example	493
Application Hook	494
handle_exception	495
Overview	

Definition	495
Arguments	496
Considerations	496
Change Log	496
Example	496
server_round_trip	497
Overview	497
Definition	498
Arguments	498
Considerations	498
Change Log	498
Example	498
Module Hooks	499
after_delete	500
Overview	500
Definition	500
Arguments	500
Examples	500
Creating a Core Logic Hook	501
after_relationship_add	502
Overview	502
Definition	503
Arguments	503
Considerations	503
Change Log	

Examples	503
Examples	504
Creating a Core Logic Hook	505
after_relationship_delete	506
Overview	506
Definition	506
Arguments	506
Considerations	506
Change Log	507
Examples	507
Creating a Core Logic Hook	508
after_restore	509
Overview	509
Definition	509
Arguments	509
Examples	510
Creating a Core Logic Hook	511
after_retrieve	512
Overview	512
Definition	512
Arguments	512
Considerations	512
Examples	513
Creating a Core Logic Hook	514
after_save	515
Overview	

Definition	515
Arguments	515
Considerations	515
Change Log	516
Examples	516
Creating a Core Logic Hook	517
before_delete	518
Overview	518
Definition	518
Arguments	518
Examples	519
Creating a Core Logic Hook	520
before_relationship_add	521
Overview	521
Definition	521
Arguments	521
Considerations	521
Change Log	522
Creating a Logic Hook using the Extension Framework	522
before_relationship_delete	523
Overview	523
Definition	523
Arguments	523
Considerations	524
Change Log	

.....	524
Creating a Logic Hook using the Extension Framework	524
before_restore	525
Overview	525
Definition	525
Arguments	525
Examples	526
Creating a Core Logic Hook	527
before_save	528
Overview	528
Definition	528
Arguments	528
Considerations	529
Change Log	529
Examples	529
Creating a Core Logic Hook	530
process_record	531
Overview	531
Definition	531
Arguments	531
Considerations	532
Change Log	532
Examples	532
Creating a Core Logic Hook	533
User Hooks	534
after_login	

.	534
Overview	535
Definition	535
Arguments	535
Change Log	535
Example	535
after_logout	536
Overview	536
Definition	536
Arguments	537
Change Log	537
Example	537
before_logout	538
Overview	538
Definition	538
Arguments	538
Change Log	539
Example	539
login_failed	540
Overview	540
Definition	540
Arguments	540
Change Log	540
Example	541
Job Queue Hooks	542
job_failure	

.	542
Overview	542
Definition	542
Arguments	542
Change Log	542
Example	543
job_failure_retry	544
Overview	544
Definition	544
Arguments	544
Change Log	544
Example	544
SNIP Hooks	546
after_email_import	546
Overview	546
Definition	546
Arguments	546
Considerations	546
Change Log	546
Example	547
before_email_import	548
Overview	548
Definition	548
Arguments	548
Considerations	548
Change Log	

Example	548
Example	549
API Hooks	550
after_routing	550
Overview	550
Definition	550
Arguments	550
Considerations	550
Change Log	551
Example	551
before_api_call	552
Overview	552
Definition	552
Arguments	552
Considerations	553
Change Log	553
Example	553
before_filter	554
Overview	554
Definition	554
Arguments	554
Considerations	555
Change Log	555
Example	555
before_respond	556
Overview	

Definition	556
Arguments	557
Considerations	557
Change Log	557
Example	557
before_routing	558
Overview	559
Definition	559
Arguments	559
Considerations	559
Change Log	559
Example	559
Web Logic Hooks	561
Overview	561
Arguments	561
Example	562
Result	562
Logic Hook Release Notes	565
7.0.0RC1	565
6.6.0	565
6.5.0RC1	565
6.4.5	565
6.4.4	566
6.4.3	566
6.0.0RC1	

5.0.0a	566
4.5.0c	566
Examples	567
Comparing Bean Properties Between Logic Hooks	567
Overview	567
Storing and Comparing Values	567
Manipulating Logic Hook References	569
Overview	569
Adding Logic Hooks	569
Removing Logic Hooks	569
Preventing Infinite Loops with Logic Hooks	570
Overview	570
Saving in an After Save Hook	570
Related Record Save Loops	572
Languages	576
Overview	576
Language Keys	576
Change Log	578
Application Labels and Lists	579
Overview	579
Application Labels and Lists	580
Customizing Application Labels and Lists	580
Hierarchy Diagram	581
Retrieving Labels	581
Retrieving Lists	

.....	581
Accessing Application Strings in Sidecar	582
\$app_strings	582
\$app_list_strings	583
Module Labels	583
Overview	583
Module Labels	583
Customizing Labels	584
Label Cache	584
Hierarchy Diagram	584
Retrieving Labels	585
Accessing Module Strings in Sidecar	585
\$mod_strings	586
Managing Lists	586
Overview	586
Managing Lists With Studio	586
Directly Modifying Lists	587
Managing Lists With Dropdown Helper	587
Language Packs	588
Overview	588
Creating a Language Pack	588
Application and Module Strings	589
Dashlet Strings	592
Templates	593
Configuration	593
Module Loadable Packages	

Example Manifest File	593
Example Manifest File	594
Extensions	595
Overview	595
Extensions Properties	595
ActionFileMap	597
Overview	597
Properties	597
Implementation	597
File System	597
Module Loadable Package	598
Installdef Properties	598
ActionReMap	599
Overview	599
Properties	599
Implementation	600
File System	600
Module Loadable Package	600
Installdef Properties	601
ActionViewMap	602
Overview	602
Properties	602
Implementation	602
File System	602
Module Loadable Package	603
Installdef Properties	

Administration	603
Overview	605
Properties	605
Implementation	605
File System	605
Module Loadable Package	607
Installdef Properties	607
Application Schedulers	608
Overview	608
Properties	608
Implementation	609
File System	609
Module Loadable Package	610
Installdef Properties	610
Console	611
Overview	611
Properties	611
Implementation	612
File System	612
Module Loadable Package	612
Installdef Properties	612
	613
Dependencies	613
Overview	613
Properties	

.....	613
Implementation	614
File System	614
Module Loadable Package	615
Installdef Properties	615
EntryPointRegistry	616
Overview	616
Properties	616
Implementation	616
File System	617
Module Loadable Package	617
Installdef Properties	617
Extensions	619
Overview	619
Properties	619
Parameters	619
Implementation	620
File System	620
Module Loadable Package	620
Installdef Properties	621
FileAccessControlMap	621
Overview	622
Properties	622
Implementation	622
File System	622
Module Loadable Package	622

Installdef Properties	623
Include	624
Overview	624
Properties	624
Implementation	625
File System	625
Module Loadable Package	625
Installdef Properties	625
JSGroupings	627
Overview	627
Properties	627
Implementation	627
File System	628
Appending to Existing JSGroupings	629
Module Loadable Package	629
Installdef Properties	630
Considerations	631
Language	631
Overview	631
Properties	631
Implementation	632
File System	632
Creating New Module Label	633
Module Loadable Package	633
Installdef Properties	

Layoutdefs	633
Overview	634
Properties	634
Implementation	635
File System	635
Module Loadable Package	636
Installdef Properties	636
LogicHooks	637
Overview	637
Properties	637
Implementation	638
File System	638
Module Loadable Package	639
Installdef Properties	639
Alternative Installdef	640
Properties	641
ScheduledTasks	642
Overview	642
Properties	642
Implementation	643
File System	643
Module Loadable Package	644
Installdef Properties	644
Sidecar	645
Overview	

.....	645
Properties	645
Implementation	645
File System	646
Module Loadable Package	646
Installdef Properties	646
TinyMCE	647
Overview	648
Properties	648
Implementation	648
File System	648
Module Loadable Package	649
Installdef Properties	649
UserPage	650
Overview	650
Properties	650
Implementation	651
File System	651
Module Loadable Package	652
Installdef Properties	652
Utils	653
Overview	653
Properties	653
Implementation	653
File System	653
Module Loadable Package	653

Installdef Properties	654
Vardefs	655
Overview	655
Properties	655
Implementation	655
File System	656
Module Loadable Package	656
Installdef Properties	656
Creating Custom Fields	657
WirelessLayoutdefs	658
Overview	658
Properties	658
Implementation	658
File System	658
Module Loadable Package	659
Installdef Properties	659
WirelessModuleRegistry	660
Overview	660
Properties	660
Implementation	661
File System	661
Module Loadable Package	661
Installdef Properties	661
Filters	662
Overview	

.....	662
Operators	663
Example	669
Sub-Expressions	669
Module Expressions	670
Filter Examples	670
Adding Initial Filters to Lookup Searches	672
Adding Initial Filters to Drawers from a Controller	675
Duplicate Check	677
Overview	677
Default Strategy	677
Custom Filter	677
Example	678
Custom Strategies	680
Duplicate Check Class	680
Example	681
Vardef Settings	682
Programmatic Usage	682
Global Search	683
Overview	683
Configuring Search Results	683
Primary and Secondary Fields	683
Examples	685
Adding Fields to the Primary Row	685
Adding Fields to the Secondary Row	688

Sugar Logic	690
Overview	690
Terminology	690
Sugar Formula Engine	690
Types	691
Number Type	691
String Type	691
Boolean Type	691
Enum Type (list)	692
Link Type (relationship)	692
Functions	693
Triggers	693
Actions	693
notActions	693
Dependencies	693
Calculated Fields	694
Dependent fields	694
Dependent dropdowns	695
Clearing the Sugar Logic Cache	695
Dependency Actions	696
Overview	696
Dependency Parameters	696
ReadOnly	697
Overview	697
Implementation	697
ReadOnly Parameters	697

Example	697
Considerations	699
SetOptions	699
Overview	699
Implementation	699
Setoptions Parameters	700
Examples	700
Using an Existing DropDown List	700
Complex Dynamic Lists	701
SetPanelVisibility	702
Overview	702
Implementation	703
SetPanelVisibility Parameters	703
Example	703
SetRequired	704
Overview	704
Implementation	704
SetRequired Parameters	705
Example	705
SetValue	706
Overview	706
Implementation	706
SetValue Parameters	707
Examples	707
Dependency Extensions	

Chaining Dependencies	707
Dependencies in Field Definitions	708
SetVisibility	709
Overview	710
Implementation	710
SetVisibility Parameters	710
Examples	711
SetVisibility Dependency Extensions	711
Visibility Dependencies in Field Definitions	712
Extending Sugar Logic	713
Overview	713
Writing a Custom Formula Function	713
Writing a Custom Action	715
Using Sugar Logic Directly	718
Accessing an External API with a Sugar Logic Action	718
Creating a Custom Dependency for a View	721
Creating a Custom Dependency Using Metadata	723
Using Dependencies in Logic Hooks	724
Overview	725
Administration	725
Overview	725
The Administration Class	725
Creating / Updating Settings	726
Retrieving Settings	726
Considerations	

Configurator	726
Overview	727
Retrieving Settings	727
Creating / Updating Settings	727
Considerations	728
Core Settings	728
Overview	728
Settings Architecture	728
Settings	728
additional_js_config	728
additional_js_config.alertAutoCloseDelay	729
additional_js_config.authStorage	729
additional_js_config.disableOmnibarTypeahead	730
additional_js_config.skipTutorial (Deprecated in future release)	730
admin_access_control	731
admin_export_only	731
allow_oauth_via_get	732
allow_pop_inbound	732
allow_sendmail_outbound	733
analytics	733
analytics.connector	734
analytics.enabled	734
analytics.id	735
api	735
api.timeout	

authenticationClass	735
aws	736
aws.aws_key	736
aws.aws_secret	737
aws.upload_bucket	737
cache_dir	737
cache_expire_timeout	738
calendar	738
calendar.day_timestep	738
calendar.default_view	739
calendar.items_draggable	739
calendar.items_resizable	739
calendar.show_calls_by_default	740
calendar.week_timestep	740
check_query	740
check_query_cost	741
collapse_subpanels	741
cron	742
cron.enforce_runtime	742
cron.max_cron_jobs	742
cron.max_cron_runtime	743
cron.min_cron_interval	743
csrf	744
csrf.soft_fail_form	744
csrf.token_size	

default_language	754
default_number_grouping_seperator	755
default_permissions	756
default_permissions.dir_mode	756
default_permissions.file_mode	757
default_permissions.group	757
default_permissions.user	758
default_user_is_admin	758
developerMode	758
diagnostic_file_max_lifetime	759
disabled_languages	759
disable_count_query	760
disable_export	760
disable_related_calc_fields	761
disable_unknown_platforms	761
disable_uw_upload	762
disable_vcr	762
dump_slow_queries	763
email_address_separator	763
email_default_client	764
email_default_delete_attachments	764
email_default_editor	765
email_mailer_timeout	765
enable_inline_reports_edit	765
enable_mobile_redirect	

external_cache.memcache.host	766
external_cache.memcache.port	766
external_cache_db_gc_probability	767
external_cache_db_gc_threshold	767
external_cache_disabled	768
external_cache_disabled_apc	768
external_cache_disabled_db	769
external_cache_disabled_memcache	769
external_cache_disabled_memcached	769
external_cache_disabled_mongo	770
external_cache_disabled_smash	770
external_cache_disabled_wincache	771
external_cache_disabled zend	771
external_cache_force_backend	772
forms	772
forms.requireFirst	773
freebusy_use_vcal_cache	773
hide_admin_backup	774
hide_admin_licensing	774
hide_full_text_engine_config	775
hide_subpanels	775
hide_subpanels_on_login	776
history_max_viewed	776
installer_locked	777
jobs	

.....	777
jobs.hard_lifetime	778
jobs.max_retries	778
jobs.min_retry_interval	778
jobs.soft_lifetime	779
jobs.timeout	779
list_max_entries_per_page	779
list_report_max_per_page	780
logger	780
logger.file.dateFormat	780
logger.file.ext	781
logger.file.maxLogs	781
logger.file.maxSize	782
logger.file.name	782
logger.file.suffix	783
logger.level	783
logger.write_to_server	784
logger_visible	784
log_dir	785
log_file	785
log_memory_usage	786
maintenanceMode	786
mark_emails_seen	787
mass_actions	787
mass_actions.mass_link_chunk_size	787
max_session_time	

minify_resources	788
moduleInstaller	789
moduleInstaller.disableFileScan	789
moduleInstaller.packageScan	789
moduleInstaller.validExt	790
mso_fixup_paragraph_tags	791
noPrivateTeamUpdate	791
oauth_token_expiry	792
oauth_token_life	792
passwordHash	792
passwordHash.algo	793
passwordHash.allowLegacy	793
passwordHash.backend	794
passwordHash.options	794
passwordHash.options.cost	795
passwordHash.options.rounds	795
passwordHash.rehash	796
passwordsetting	796
passwordsetting.forgotpasswordON	797
passwordsetting.linkexpiration	797
passwordsetting.onelower	797
passwordsetting.onenumber	798
passwordsetting.oneupper	798
passwordsetting.systemexpirationtype	799
pdf_file_max_lifetime	

perfProfile	799
perfProfile.TeamSecurity	800
perfProfile.TeamSecurity.default	800
perfProfile.TeamSecurity.default.teamset_prefetch	801
perfProfile.TeamSecurity.default.teamset_prefetch_max	801
perfProfile.TeamSecurity.default.where_condition	802
pmse_settings_default	803
pmse_settings_default.error_number_of_cycles	803
pmse_settings_default.error_timeout	803
pmse_settings_default.logger_level	804
require_accounts	804
rest_response_etag_cache_age	805
roleBasedViews	805
SAML_provisionUser	805
SAML_X509Cert	806
search_engine	806
search_engine.max_bulk_delete_threshold	806
search_engine.max_bulk_query_threshold	807
search_engine.max_bulk_threshold	807
search_engine.postpone_job_time	807
search_wildcard_infront	808
session_dir	808
showThemePicker	809
show_download_tab	809
site_url	

.....	810
slow_query_time_msec	810
stack_trace_errors	810
studio_max_history	811
sugar_version	811
tmp_dir	812
tmp_file_max_lifetime	812
tracker_max_display_length	813
unique_key	813
upload_badext	813
upload_dir	814
upload_maxsize	814
upload_wrapper_class	815
use_common_ml_dir	815
use_php_code_json	816
use_real_names	816
use_sprites	816
validation	817
validation.compat_mode	817
validation.soft_fail	818
vcal_time	819
verify_client_ip	819
wl_list_max_entries_per_page	820
wl_list_max_entries_per_subpanel	820
xhprof_config	820
xhprof_config.enable	

xhprof_config.filter_wt	821
xhprof_config.flags	821
xhprof_config.ignored_functions	822
xhprof_config.log_to	822
xhprof_config.manager	822
xhprof_config.memory_limit	823
xhprof_config.mongodb_collection	823
xhprof_config.mongodb_db	824
xhprof_config.mongodb_options	824
xhprof_config.mongodb_uri	824
xhprof_config.sample_rate	825
xhprof_config.save_to	825
xhprof_config.track_elastic	826
xhprof_config.track_elastic_backtrace	826
xhprof_config.track_sql	826
xhprof_config.track_sql_backtrace	827
xhprof_config.track_sql_backtrace	827
Module Loader	827
Introduction to the Manifest	828
Overview	828
Manifest Definitions	828
key	828
name	828
description	828
built_in_version	828

version	829
acceptable_sugar_versions	829
acceptable_sugar_flavors	829
author	829
readme	830
icon	830
is_uninstallable	830
published_date	830
remove_tables	830
type	830
dependencies	831
Manifest Example	831
Installation Definitions	832
\$installdef Actions	832
action_file_map	832
action_remap	832
action_view_map	832
administration	832
appscheduledefs	832
beans	833
connectors	833
copy	833
custom_fields	833
console	834
dashlets	

dependencies	834
entrypoints	835
extensions	835
file_access	835
hookdefs	835
id	835
image_dir	835
jsgroups	836
language	836
layoutdefs	836
layoutfields	836
logic_hooks	836
pre_execute	836
post_execute	837
pre_uninstall	837
post_uninstall	837
relationships	838
scheduledefs	840
sidecar	840
tinymce	840
user_page	840
utils	840
vardefs	840
wireless_modules	840
wireless_subpanels	

.....	840
.....	841
Module Loader Restrictions	841
Overview	841
Access Controls	841
Enabling Package Scan	841
Enabling File Scan	842
Valid Extension Types	842
Blacklisted Classes	843
Blacklisted Function Calls	843
Disabling File Scan	849
Extending the List of Valid Extension Types	849
Blacklisting Additional Function Calls	850
Overriding Blacklisted Function Calls	850
Disabling Restricted Copy	850
Module Loader Actions	851
Disabling Module Loader Actions	851
Disabling Upgrade Wizard	852
Module Loader Restriction Alternatives	852
Overview	852
Blacklisted Functions	852
array_filter()	853
copy()	854
file_exists()	854
file_get_contents()	855
fwrite()	

.....	855
Adding/Removing Logic Hooks	855
getimagesize()	856
Sugar Exchange Package Guidelines	857
Overview	857
Best Practices	857
Use a consistent coding style	858
Invest in unit testing during development	858
Use Sugar REST APIs	858
Use Module Builder when possible	858
Use Extension framework and Dashlets for Sugar code customizations	859
Security Guidelines	859
Use TLS/SSL for web services calls	859
Do not hardcode sensitive information	859
User Interface Guidelines	859
Use the Sugar 7 Styleguide	860
Don't use incompatible UI frameworks or external CSS libraries	860
Encapsulation Guidelines	860
Use Extensions framework and Custom Modules as much as possible	860
Avoid customizations to core Sugar application files	861
Use Package Scanner to ensure your Sugar Package is ready for Sugar Cloud	861
Performance Guidelines	861
Index for large frequently used queries	862
Add scheduled jobs to prune large database tables	862
Ensure your application does not block user interface during long running processes	862
Examples	

.....	862
Creating an Installable Package for a Logic Hook	863
Overview	863
Package Example	863
Creating an Installable Package That Copies Files	865
Overview	865
Manifest Example	865
Creating an Installable Package that Creates New Fields	866
Overview	866
Manifest Example	866
Language Example	870
Removing Files with an Installable Package	870
Overview	870
Manifest Example	871
Module Builder	872
Overview	872
Creating New Modules	873
Understanding Object Templates	873
Editing Module Fields	874
Editing Module Layouts	874
Building Relationships	874
Publishing and Uploading Packages	874
Adding Custom Logic using Code	875
Logic Hooks	875
Custom Bean files	875
Using the New Module	

Best Practices	876
Overview	876
Build your module with Module Builder	877
Never re-deploy a package in a production environment	877
Only create one module per package	877
Wait to create relationships until after deploying the module	878
Delete the package from Module Builder after it is deployed	878
Entry Points	878
Overview	878
Accessing Entry Points	878
Creating Custom Entry Points	879
Overview	879
Custom Entry Points	879
Example	879
Job Queue	880
Overview	880
Stages	881
Execution Stage	881
Cleanup Stage	881
Schedulers	882
Overview	882
Config Settings	883
Considerations	883
Creating Custom Schedulers	884
Overview	

Defining the Job Label	884
Defining the Job Function	884
Using the New Job	885
Scheduler Jobs	885
Overview	885
Properties	885
Creating a Job	886
Job Targets	886
Running the Job	886
Job status	887
Job Resolution	887
Job Logic Hooks	887
Creating Custom Jobs	887
Overview	887
How it Works	888
Creating the Job	888
Queuing Logic Hook Actions	889
Overview	889
Example	889
Access Control Lists	891
Overview	891
Checking Access	892
Parameters	892
Example	892

Teams	892
Overview	892
Database Tables	893
Module Team Fields	893
Team Sets (team_set_id)	894
Primary Team (team_id)	894
Team Security	894
TeamSetLink	895
Manipulating Teams Programmatically	896
Overview	896
Fetching Teams	896
Adding Teams	896
Considerations	897
Removing Teams	897
Considerations	897
Replacing Team Sets	897
Considerations	898
Creating and Retrieving Team Set IDs	899
Visibility Framework	899
Overview	899
Custom Row Visibility	900
Visibility Class	900
Example	901
TinyMCE	909
Modifying the TinyMCE Editor	909
Overview	

.....	909
Overriding Buttons	910
Example File	910
Overriding Default Settings	911
Example File	911
Creating Plugins	912
SugarPDF	912
Overview	912
Key PDF Classes	912
Sugarpdf	913
ViewSugarpdf	914
SugarpdfFactory	915
SugarpdfHelper	915
PdfManagerHelper	915
FontManager	916
Generating PDFs	916
Overview	916
Generating a PDF	916
PDF Settings	919
Overview	919
PDF Setting Framework	919
Settings	919
Displaying and Editing Settings	920
Fonts	921
Overview	921
Font Manager	

Font Cache	921
Font Framework	922
Font Pack	922
DateTime	922
Overview	922
Setting	922
Formatting	923
TimeDate Class	924
Setting	924
Formatting	925
getNow	925
Parameters	925
Example	926
fromUser	926
Parameters	926
Example	926
asUser	927
Parameters	927
Example	927
fromDb	927
Parameters	927
Example	928
fromString	928
Parameters	928
Example	928

.....	928
get_db_date_time_format	928
Parameters	929
Example	929
to_display_date_time	929
Parameters	929
Example	930
Parsing	930
parseDateRange	930
Parameters	930
Example	931
Shortcuts	932
Overview	932
Shortcut Sessions	932
Register	933
Shortcut IDs	933
Global shortcuts	933
Shortcut Keys	933
Additional Features	934
Input Focus	934
Limitations	934
Shortcuts Help	934
Advanced Features	934
Dynamically saving, creating new, and restoring sessions	935
Example	935

Web Accessibility	937
Overview	937
Introduction	937
How It Works	938
Plugins	938
Click	939
Label	940
API	940
SUGAR.accessibility.run()	940
Arguments	941
Returns	941
SUGAR.accessibility.whichHelpers()	941
Returns	941
SUGAR.accessibility.getElementTag()	941
Arguments	942
Returns	942
CLI	942
Overview	942
Commands	942
Usage	943
Help	943
Custom Commands	945
Best Practices	945
Command Namespace	946
PHP Namespace	946
Mode	

.....	947
Class Definition	947
Methods	948
Registering Command	948
Example	949
Performance Tuning	951
Sugar Performance	951
Overview	951
General Settings in Sugar	952
Sugar Performance Settings	952
BWC Modules	953
General Environment Checks	953
PHP Caching	954
PHP Profiling	955
Overview	955
Integrating Sugar With New Relic APM for Performance Management	956
Overview	956
Prerequisites	956
Steps to Complete	956
Installing the New Relic for PHP Agent	956
Configuring Sugar to Work With New Relic for PHP	957
Using New Relic for PHP	957
Overview Dashboard	958
Transactions	959
Summary	963

Backward Compatibility	963
Overview	963
URL Differences	964
Studio Differences	964
Enabling Backward Compatibility	965
Overview	965
Enabling Backward Compatibility	965
Enabling BWC	965
Updating the MegaMenu Module Link	966
Updating the MegaMenu Sub-Navigation Links	966
Verifying BWC Is Enabled	968
Using Developer Tools in Google Chrome	968
Using Firebug in Google Chrome or Mozilla Firefox	969
Converting Legacy Modules To Sidecar	969
Overview	969
Steps to Complete	969
Advanced Workflow	970
Introduction	970
Advanced Workflow Modules	971
Process Definitions	971
Business Rules	971
Email Templates	972
Processes	972
Database Tables	972
Relationships	974
Flows	

.....	976
Extension and Customization	978
Caveats	979
Extending Advanced Workflow (Process Manager)	979
Introduction	979
Process Manager	979
Limitations	980
.....	980
Examples	981
Getting an Advanced Workflow Element Object	982
Getting and Using a Process Manager Field Evaluator Object	984
Getting an Exception Object with Logging	985
Maintaining State Throughout a Process	987
Integration	988
Overview	988
Best Practices	989
Overview	989
Latency When Posting Data To Sugar	989
Disabling Related Calculation Fields	989
Disabling Logic Hooks	990
Disabling Workflows	990
Queuing Data in the Job Queue	990
Web Services	991
Overview	991
Versioning	991
Quick Reference	

v10	992
Overview	992
What Is REST?	992
Getting Started	993
Authentication	993
Avoiding Login Conflicts	995
Input / Output Data Types	995
Date Handling	995
Endpoints	995
/Accounts/:record/link/:link_name/filter GET	996
Overview	996
Summary	996
Request Arguments	996
Filter Expressions	997
Basic	998
Example	998
Full	998
Example	998
Sub-expressions	999
Example	999
Modules	1.000
Example	1.000
Response Arguments	1.000
Response	1.001
Change Log	

	1.006
/Activities GET	1.006
Summary:	1.006
Query Parameters:	1.007
Input Example:	1.007
Output Example:	1.007
/Activities/filter GET	1.008
Summary:	1.008
Query Parameters:	1.008
Input Example:	1.008
Output Example:	1.008
/Administration/elasticsearch/indices GET	1.009
Overview	1.009
Summary	1.010
Response	1.010
Change Log	1.012
/Administration/elasticsearch/mapping GET	1.012
Overview	1.012
Summary	1.012
Response	1.012
Change Log	1.014
/Administration/elasticsearch/queue GET	1.014
Overview	1.014
Summary	1.014
Response	1.014
Change Log	1.014

.....	1.015
/Administration/elasticsearch/routing GET	1.015
Overview	1.015
Summary	1.015
Response	1.016
Change Log	1.016
/Administration/search/fields GET	1.017
Overview	1.017
Summary	1.017
Request Arguments	1.017
Response	1.018
Response using order_by_boost	1.018
Change Log	1.018
/Administration/search/reindex POST	1.019
Overview	1.019
Summary	1.019
Request Arguments	1.019
Response	1.020
Change Log	1.020
/Administration/search/status GET	1.020
Overview	1.020
Summary	1.020
Response	1.021
Change Log	1.021
/Calendar/invitee_search GET	1.021
Overview	

.....	1.021
Request Arguments	1.021
Response Arguments	1.022
Response	1.022
Change Log	1.024
/Calls POST	1.024
Overview	1.024
Request Arguments	1.024
Request	1.025
Response Arguments	1.025
Response	1.025
Change Log	1.025
/Calls/:record DELETE	1.026
Overview	1.026
Request Arguments	1.026
Request	1.026
Response Arguments	1.026
Response	1.026
Change Log	1.027
/Calls/:record PUT	1.027
Overview	1.027
Request Arguments	1.027
Request	1.027
Response Arguments	1.028
Response	1.028
Change Log	1.028

.....	1.028
/Contacts/:record/freebusy GET	1.028
Summary:	1.028
Request	1.028
Response	1.029
/Dashboards/Activities GET	1.029
Summary:	1.029
Query Parameters:	1.029
Request	1.031
Response	1.031
/Dashboards GET	1.031
Summary:	1.031
Query Parameters:	1.031
Request	1.032
Response	1.033
/Dashboards POST	1.033
Summary:	1.033
Request Arguments	1.033
Request	1.033
Response	1.034
/Dashboards/<module> GET	1.034
Summary:	1.034
Query Parameters:	1.034
Request	1.035
Response	1.035
/Dashboards/<module> POST	

Summary:	1.036
Request Arguments	1.036
Request	1.036
Response	1.036
/Documents/:record/file/:field POST	1.037
Overview	1.037
Request Arguments	1.037
Request	1.037
Response Arguments	1.038
Response	1.038
Change Log	1.038
/Documents/:record/file/:field PUT	1.038
Overview	1.038
Request Arguments	1.039
Request	1.039
Response Arguments	1.039
Response	1.040
Change Log	1.040
/EmbeddedFiles/:record/file/:field GET	1.041
Overview	1.041
Request Arguments	1.041
Response Arguments	1.041
Response	1.041
Change Log	1.041
/EmbeddedFiles/:record/file/:field POST	

Overview	1.042
Request Arguments	1.042
Request	1.042
Response Arguments	1.043
Response	1.043
Change Log	1.043
/EmbeddedFiles/:record/file/:field PUT	1.043
Overview	1.043
Request Arguments	1.044
Request	1.044
Response Arguments	1.044
Response	1.045
Change Log	1.045
/Employees GET	1.046
Overview	1.046
Summary	1.046
Request Arguments	1.046
Filter Expressions	1.049
Basic	1.049
Example	1.049
Full	1.049
Example	1.049
Sub-expressions	1.051
Example	1.051
Modules	

Example	1.051
Response Arguments	1.052
Response	1.052
Change Log	1.053
/ExpressionEngine/:record/related GET	1.053
Summary:	1.053
Query Parameters:	1.054
Input Example:	1.054
Output Example:	1.054
/Filters GET	1.055
Overview	1.055
Summary	1.055
Request Arguments	1.056
Filter Expressions	1.058
Basic	1.058
Example	1.059
Full	1.059
Example	1.059
Sub-expressions	1.060
Example	1.060
Modules	1.061
Example	1.061
Response Arguments	1.061
Response	1.062
Change Log	

.....	1.063
/ForecastManagerWorksheets GET	1.063
Summary:	1.063
Url Parameters:	1.063
Possible Errors	1.063
Url Example:	1.064
Output Example:	1.064
/ForecastManagerWorksheets/assignQuota POST	1.065
Summary:	1.065
Parameters:	1.065
Input Example:	1.066
Output Example:	1.066
/ForecastManagerWorksheets/export GET	1.066
Overview	1.066
Request Arguments	1.066
Request	1.067
Exporting Records by a List of IDs	1.067
Exporting Records Using a Filter	1.067
Exporting a Sample Result Set	1.067
Response Arguments	1.067
Response	1.068
Change Log	1.069
/ForecastManagerWorksheets/filter GET	1.069
Summary:	1.069
Query Parameters:	1.069
Possible Errors	

Url Example:	1.069
Output Example:	1.070
/ForecastManagerWorksheets/filter POST	1.070
Summary:	1.070
Query Parameters:	1.071
Possible Errors	1.071
Url Example:	1.071
Output Example:	1.071
/ForecastManagerWorksheets/:timeperiod_id GET	1.072
Summary:	1.072
Url Parameters:	1.072
Possible Errors	1.072
Url Example:	1.073
Output Example:	1.073
/ForecastManagerWorksheets/:timeperiod_id/:user_id GET	1.074
Summary:	1.074
Url Parameters:	1.074
Possible Errors	1.075
Url Example:	1.075
Output Example:	1.075
/ForecastWorksheets GET	1.076
Summary:	1.077
Url Parameters:	1.077
Query Parameters:	1.077
Possible Errors	

Url Example:	1.077
Output Example:	1.078
/ForecastWorksheets/export GET	1.078
Overview	1.078
Request Arguments	1.079
Request	1.079
Exporting Records by a List of IDs	1.079
Exporting Records Using a Filter	1.079
Exporting a Sample Result Set	1.080
Response Arguments	1.080
Response	1.080
Change Log	1.081
/ForecastWorksheets/filter GET	1.081
Summary:	1.081
Query Parameters:	1.082
Possible Errors	1.082
Url Example:	1.082
Output Example:	1.082
/ForecastWorksheets/filter POST	1.083
Summary:	1.083
Query Parameters:	1.083
Possible Errors	1.084
Url Example:	1.084
Output Example:	1.084
/ForecastWorksheets/:record PUT	

Summary:	1.085
Query Parameters:	1.085
Input Example:	1.085
Output Example:	1.086
/ForecastWorksheets/:timeperiod_id GET	1.086
Summary:	1.086
Url Parameters:	1.086
Query Parameters:	1.087
Possible Errors	1.087
Url Example:	1.087
Output Example:	1.087
/ForecastWorksheets/:timeperiod_id/:user_id GET	1.088
Summary:	1.088
Url Parameters:	1.088
Query Parameters:	1.089
Possible Errors	1.089
Url Example:	1.089
Output Example:	1.089
/Forecasts GET	1.090
Overview	1.090
Request Arguments	1.090
Response Arguments	1.090
Response	1.090
Change Log	1.093
/Forecasts/config POST	

Summary:	1.093
Query Parameters:	1.093
Input Example:	1.093
Output Example:	1.094
/Forecasts/config PUT	1.094
Summary:	1.094
Query Parameters:	1.095
Input Example:	1.095
Output Example:	1.095
/Forecasts/enum/selectedTimePeriod GET	1.096
Summary:	1.096
Query Parameters:	1.096
Input Example:	1.096
Output Example:	1.096
/Forecasts/init GET	1.097
Summary:	1.097
Query Parameters:	1.097
Input Example:	1.097
Output Example:	1.097
/Forecasts/reportees/:user_id GET	1.098
Summary:	1.098
Query Parameters:	1.098
Input Example:	1.099
Output Example:	1.099
/Forecasts/:timeperiod_id/progressManager/:user_id GET	

Summary:	1.100
Query Parameters:	1.100
Input Example:	1.100
Output Example:	1.100
/Forecasts/:timeperiod_id/progressRep/:user_id GET	1.101
Summary:	1.101
Query Parameters:	1.101
Input Example:	1.101
Output Example:	1.101
/Forecasts/:timeperiod_id/quotas/direct/:user_id GET	1.101
Summary:	1.101
Query Parameters:	1.102
Input Example:	1.102
Output Example:	1.102
/Forecasts/:timeperiod_id/quotas/rollup/:user_id GET	1.102
Summary:	1.102
Query Parameters:	1.102
Input Example:	1.103
Output Example:	1.103
/Forecasts/:timeperiod_id/:user_id/chart/:display_manager GET	1.103
Summary:	1.103
Query Parameters:	1.103
Input Example:	1.104
Output Example:	1.104
/Forecasts/user/:user_id GET	

Summary:	1.105
Query Parameters:	1.105
Input Example:	1.105
Output Example:	1.105
/KBContents GET	1.106
Overview	1.106
Summary	1.106
Request Arguments	1.106
Filter Expressions	1.109
Basic	1.109
Example	1.109
Full	1.109
Example	1.110
Sub-expressions	1.111
Example	1.111
Modules	1.111
Example	1.112
Response Arguments	1.112
Response	1.112
Change Log	1.113
/KBContents/config POST	1.114
Summary:	1.114
Query Parameters:	1.114
Input Example:	1.114
Output Example:	

	1.114
/KBContents/config PUT	1.114
Summary:	1.114
Query Parameters:	1.115
Input Example:	1.115
Output Example:	1.115
/KBContents/duplicateCheck POST	1.115
Overview	1.115
Request Arguments	1.115
Request	1.116
Response Arguments	1.116
Response	1.116
Change Log	1.118
/KBContents/filter GET	1.119
Overview	1.119
Summary	1.119
Request Arguments	1.119
Filter Expressions	1.122
Basic	1.122
Example	1.122
Full	1.123
Example	1.123
Sub-expressions	1.124
Example	1.124
Modules	1.125
Example	1.125

Response Arguments	1.125
Response	1.125
Change Log	1.126
/KBContents/:record/link POST	1.127
Overview	1.127
Request Arguments	1.127
Request	1.127
Response Arguments	1.128
Response	1.128
Change Log	1.133
/KBContents/:record/notuseful PUT	1.134
Overview	1.134
Request Arguments	1.134
Response Arguments	1.134
Response	1.134
Change Log	1.135
/KBContents/:record/useful PUT	1.135
Overview	1.136
Request Arguments	1.136
Response Arguments	1.136
Response	1.136
Change Log	1.137
/Leads POST	1.137
Summary:	1.137
Query Parameters:	

.....	1.138
/Leads/:leadId/convert POST	1.138
Summary:	1.138
Post Parameters:	1.138
/Leads/:record/freebusy GET	1.143
Summary:	1.144
Request	1.144
Response	1.144
/Leads/register POST	1.144
Overview	1.144
Request Arguments	1.145
Request	1.145
Response Arguments	1.145
Response	1.145
Change Log	1.146
/Mail POST	1.146
Overview	1.146
Query String Parameters	1.146
Input Parameters	1.146
Input Example	1.148
Result	1.149
Output Example	1.150
Change Log	1.150
/Mail/archive POST	1.151
Overview	1.151
Query String Parameters	

Input Parameters	1.151
Input Example	1.153
Result	1.154
Output Example	1.154
Change Log	1.155
/Mail/attachment POST	1.155
Overview	1.155
Query String Parameters	1.155
Input Parameters	1.155
Input Example	1.156
Result	1.156
Output Example	1.156
Change Log	1.156
/Mail/attachment/cache DELETE	1.157
Overview	1.157
Query Parameters:	1.157
Input Example:	1.157
Output Example:	1.157
Change Log	1.157
/Mail/attachment/:file_guid DELETE	1.157
Overview	1.157
Query Parameters:	1.157
Input Example:	1.158
Output Example:	1.158
Change Log	

.....	1.158
/Meetings POST	1.158
Overview	1.158
Request Arguments	1.158
Request	1.158
Response Arguments	1.159
Response	1.159
Change Log	1.159
/Meetings/:record DELETE	1.160
Overview	1.160
Request Arguments	1.160
Request	1.160
Response Arguments	1.160
Response	1.160
Change Log	1.160
/Meetings/:record PUT	1.161
Overview	1.161
Request Arguments	1.161
Request	1.161
Response Arguments	1.162
Response	1.162
Change Log	1.162
/Meetings/:record/external GET	1.162
Summary:	1.162
Query Parameters:	1.162
Input Example:	

Output Example:	1.163
/Notifications GET	1.163
Overview	1.163
Summary	1.163
Request Arguments	1.163
Filter Expressions	1.166
Basic	1.166
Example	1.167
Full	1.167
Example	1.167
Sub-expressions	1.168
Example	1.168
Modules	1.169
Example	1.169
Response Arguments	1.169
Response	1.170
Change Log	1.171
/Opportunities/config POST	1.171
Overview	1.171
Summary	1.171
Request Arguments	1.171
Request	1.172
Response Arguments	1.172
Response	1.172
Change Log	

.....	1.172
/Opportunities/enum/:field GET	1.172
Overview	1.172
Request Arguments	1.173
Response Arguments	1.173
Response	1.173
Change Log	1.173
/PdfManager GET	1.174
Overview	1.174
Summary	1.174
Request Arguments	1.174
Filter Expressions	1.177
Basic	1.177
Example	1.177
Full	1.177
Example	1.178
Sub-expressions	1.179
Example	1.179
Modules	1.179
Example	1.180
Response Arguments	1.180
Response	1.180
Change Log	1.181
/RevenueLineItems/:record/quote POST	1.182
Summary:	1.182
Query Parameters:	

Valid Urls:	1.182
Output Example:	1.182
/Tags/:record PUT	1.182
Overview	1.182
Request Arguments	1.182
Request	1.183
Link fields	1.183
Response Arguments	1.183
Response	1.184
Change Log	1.186
/Teams/:record/link POST	1.186
Overview	1.186
Request Arguments	1.186
Request	1.187
Response Arguments	1.187
Response	1.187
Change Log	1.193
/Teams/:record/link/:link_name/:remote_id DELETE	1.193
Overview	1.193
Request Arguments	1.193
Response Arguments	1.193
Response	1.194
Change Log	1.197
/Teams/:record/link/:link_name/:remote_id POST	1.198
Overview	

.....	1.198
Query String Parameters	1.198
Response Arguments	1.198
Response	1.198
Change Log	1.202
/TimePeriods GET	1.203
Overview	1.203
Summary	1.203
Request Arguments	1.203
Filter Expressions	1.206
Basic	1.206
Example	1.206
Full	1.207
Example	1.207
Sub-expressions	1.208
Example	1.208
Modules	1.209
Example	1.209
Response Arguments	1.209
Response	1.209
Change Log	1.211
/TimePeriods/current GET	1.211
Summary:	1.211
Output Example:	1.211
/TimePeriods/:date GET	1.211
Summary:	

.....	1.212
Output Example:	1.212
/TimePeriods/filter GET	1.212
Overview	1.212
Summary	1.212
Request Arguments	1.213
Filter Expressions	1.216
Basic	1.216
Example	1.216
Full	1.216
Example	1.216
Sub-expressions	1.217
Example	1.218
Modules	1.218
Example	1.218
Response Arguments	1.219
Response	1.219
Change Log	1.220
/Users GET	1.220
Overview	1.220
Summary	1.221
Request Arguments	1.221
Filter Expressions	1.224
Basic	1.224
Example	1.224
Full	1.224

.....	1.224
Example	1.224
.....	1.225
Sub-expressions	1.225
.....	1.226
Example	1.226
.....	1.226
Modules	1.226
.....	1.226
Example	1.226
.....	1.227
Response Arguments	1.227
.....	1.227
Response	1.227
.....	1.227
Change Log	1.227
.....	1.228
/Users/:record DELETE	1.228
.....	1.229
Summary:	1.229
.....	1.229
/Users/:record/freebusy GET	1.229
.....	1.229
Summary:	1.229
.....	1.229
Request	1.229
.....	1.229
Response	1.229
.....	1.229
/Users/:record/link POST	1.229
.....	1.230
Overview	1.230
.....	1.230
Request Arguments	1.230
.....	1.230
Request	1.230
.....	1.231
Response Arguments	1.231
.....	1.231
Response	1.231
.....	1.231
Change Log	1.231
.....	1.237
/Users/:record/link/:link_name/:remote_id DELETE	1.237
.....	1.237
Overview	1.237
.....	1.237
Request Arguments	1.237
.....	1.237
Response Arguments	1.237

Response	1.237
Change Log	1.242
/Users/:record/link/:link_name/:remote_id POST	1.242
Overview	1.242
Query String Parameters	1.242
Response Arguments	1.242
Response	1.242
Change Log	1.246
/VCardDownload GET	1.247
Overview	1.247
Request Arguments	1.247
Request	1.247
Response Arguments	1.247
Response	1.248
Change Log	1.248
/bulk POST	1.248
Overview	1.248
Summary	1.248
Request Arguments	1.249
Request Example	1.249
Response	1.250
Response Example	1.250
Change Log	1.251
/collection/:collection_name GET	1.251
Overview	

Summary	1.251
Request Arguments	1.252
Response Arguments	1.253
Response	1.253
Change Log	1.254
/connector/dnb/Accounts/bal POST	1.254
D&B API -- Post Requests Overview	1.254
Query Parameters	1.254
Query Parameters	1.255
/connector/dnb/Contacts/bal POST	1.256
D&B API -- Post Requests Overview	1.256
Query Parameters	1.256
Query Parameters	1.256
/connector/dnb/meter GET	1.257
D&B API -- Get Requests Overview	1.257
Query Parameters	1.258
Query Parameters	1.258
/connector/dnb/:module/bulkimport POST	1.260
D&B Bulk Import API Overview	1.260
Request Arguments	1.260
Request Example	1.261
Response	1.261
Response Example	1.261
/connector/dnb/:qtype POST	1.262
D&B API -- Post Requests Overview	

	1.262
Query Parameters	1.262
Query Parameters	1.262
/connector/dnb/:qtype/:qparam GET	1.263
D&B API -- Get Requests Overview	1.263
Query Parameters	1.263
Query Parameters	1.264
/connectors GET	1.266
Overview	1.266
Request Arguments	1.266
Result Arguments	1.266
Change Log	1.269
/connector/twitter/currentUser GET	1.269
Overview	1.269
Request Arguments	1.269
Response Arguments	1.269
Response	1.269
Change Log	1.273
/connector/twitter/:twitterId GET	1.273
Overview	1.273
Request Arguments	1.273
Response Arguments	1.273
Response	1.273
Change Log	1.277
/css GET	1.277
Overview	

.....	1.277
Request Arguments	1.277
Request	1.277
Response	1.278
Change Log	1.278
/css/preview GET	1.278
Overview	1.278
Request Arguments	1.278
Request	1.279
Response Arguments	1.279
Change Log	1.280
/globalsearch GET	1.280
Overview	1.280
Summary	1.280
Request Arguments	1.280
Request	1.282
Response Arguments	1.283
Response	1.283
Change Log	1.283
/globalsearch POST	1.283
Overview	1.284
Summary	1.284
Request Arguments	1.284
Request	1.286
Response Arguments	1.286
Response	1.286

Change Log	1.287
Change Log	1.287
/help GET	1.287
Overview	1.287
Request Arguments	1.287
Response Arguments	1.287
Response	1.288
Change Log	1.288
/help/exceptions GET	1.288
Overview	1.288
Request Arguments	1.288
Response Arguments	1.288
Response	1.289
Change Log	1.289
/logger POST	1.289
Overview	1.289
Request Arguments	1.289
Response Arguments	1.290
Response	1.290
Change Log	1.290
/me GET	1.290
Overview	1.290
Request Arguments	1.290
Response Arguments	1.290
Response Portal User	1.291
Change Log	

.....	1.291
/me PUT	1.292
Overview	1.292
Request Arguments	1.292
Response Arguments	1.292
Response Portal User Example	1.292
Change Log	1.293
/me/following GET	1.293
Overview	1.293
Request Arguments	1.293
Response Arguments	1.293
Response	1.293
Change Log	1.294
/me/password POST	1.294
Overview	1.294
Summary	1.294
Request Arguments	1.294
Request	1.294
Response Arguments	1.295
Response	1.295
Change Log	1.295
/me/password PUT	1.295
Overview	1.295
Summary	1.295
Request Arguments	1.296
Request	

.....	1.296
Response Arguments	1.296
Response	1.296
Change Log	1.297
/me/preference/:preference_name DELETE	1.297
Overview	1.297
Request Arguments	1.297
Response Arguments	1.297
Response	1.297
Change Log	1.298
/me/preference/:preference_name GET	1.298
Overview	1.298
Request Arguments	1.298
Response Arguments	1.298
Response	1.298
Change Log	1.299
/me/preference/:preference_name POST	1.299
Overview	1.299
Request Arguments	1.299
Response Arguments	1.299
Response	1.299
Change Log	1.300
/me/preference/:preference_name PUT	1.300
Overview	1.300
Request Arguments	1.300
Response Arguments	1.300

Response	1.300
Change Log	1.301
/me/preferences GET	1.301
Overview	1.301
Request Arguments	1.301
Response Arguments	1.301
Response	1.301
Change Log	1.302
/me/preferences PUT	1.302
Overview	1.302
Request Arguments	1.302
Response Arguments	1.302
Response	1.302
Change Log	1.303
/<module>/Activities GET	1.303
Summary:	1.303
Query Parameters:	1.303
Input Example:	1.303
Output Example:	1.303
/<module>/Activities/filter GET	1.304
Summary:	1.305
Query Parameters:	1.305
Input Example:	1.305
Output Example:	1.305
/<module>/MassUpdate DELETE	

Overview	1.306
Query String Parameters	1.306
Request	1.307
Response Arguments	1.307
Output Done Example	1.307
Change Log	1.307
/<module>/MassUpdate PUT	1.308
Overview	1.308
Query String Parameters	1.308
Request	1.308
Mass Updating Records with Teams	1.309
Mass Add Leads, Contacts or Prospects to a Target List	1.309
Response Arguments	1.310
Output Done Example	1.310
Change Log	1.310
/<module> GET	1.311
Overview	1.311
Summary	1.311
Request Arguments	1.311
Filter Expressions	1.314
Basic	1.314
Example	1.314
Full	1.315
Example	1.315
Sub-expressions	

Example	1.316
Modules	1.316
Example	1.317
Response Arguments	1.317
Response	1.318
Change Log	1.319
/<module> POST	1.319
Overview	1.319
Request Arguments	1.319
Request	1.320
Link fields	1.320
Response Arguments	1.320
Response	1.321
Change Log	1.323
/<module>/append/:target POST	1.323
Overview	1.323
Request Arguments	1.323
Request	1.324
Response Arguments	1.324
Response	1.324
Change Log	1.324
/<module>/config GET	1.325
Overview	1.325
Summary	1.325
Request Arguments	

Response Arguments	1.325
Response	1.325
Change Log	1.327
/<module>/config POST	1.327
Overview	1.327
Summary	1.327
Request Arguments	1.328
Request	1.328
Response Arguments	1.328
Response	1.328
Change Log	1.329
/<module>/config PUT	1.329
Overview	1.329
Summary	1.329
Request Arguments	1.329
Request	1.329
Response Arguments	1.330
Response	1.330
Change Log	1.330
/<module>/count GET	1.330
Overview	1.330
Summary	1.330
Request Arguments	1.331
Filter Expressions	1.334
Basic	

Example	1.334
Full	1.334
Example	1.335
Sub-expressions	1.335
Example	1.336
Modules	1.336
Example	1.337
Response Arguments	1.337
Response	1.337
Change Log	1.338
Change Log	1.339
/<module>/duplicateCheck POST	1.339
Overview	1.340
Request Arguments	1.340
Request	1.340
Response Arguments	1.340
Response	1.340
Change Log	1.340
Change Log	1.343
/<module>/enum/:field GET	1.343
Overview	1.344
Request Arguments	1.344
Response Arguments	1.344
Response	1.344
Change Log	1.344
/<module>/export/:record_list_id GET	1.345
Overview	1.345

.....	1.345
Request Arguments	1.345
Request	1.346
Exporting Records by a List of IDs	1.346
Exporting Records Using a Filter	1.346
Exporting a Sample Result Set	1.346
Response Arguments	1.347
Response	1.347
Change Log	1.348
/<module>/file/vcard_import POST	1.348
Overview	1.348
Request Arguments	1.348
Request	1.349
Response Arguments	1.349
Response	1.349
Change Log	1.350
/<module>/filter GET	1.350
Overview	1.350
Summary	1.350
Request Arguments	1.350
Filter Expressions	1.354
Basic	1.354
Example	1.354
Full	1.355
Example	1.355
Sub-expressions	

Example	1.356
Modules	1.357
Example	1.357
Response Arguments	1.357
Response	1.358
Change Log	1.359
/<module>/filter POST	1.359
Overview	1.360
Summary	1.360
Request Arguments	1.360
Filter Expressions	1.362
Basic	1.362
Example	1.362
Full	1.362
Example	1.363
Sub-expressions	1.364
Example	1.364
Modules	1.365
Example	1.365
Response Arguments	1.365
Response	1.366
Change Log	1.372
/<module>/filter/count GET	1.373
Overview	1.373
Summary	

.....	1.373
Request Arguments	1.373
Filter Expressions	1.375
Basic	1.375
.....	1.375
Example	1.375
Full	1.376
.....	1.376
Example	1.376
Sub-expressions	1.377
.....	1.377
Example	1.377
Modules	1.378
.....	1.378
Example	1.378
Response Arguments	1.378
Response	1.379
Change Log	1.385
/<module>/filter/count POST	1.386
Overview	1.386
Summary	1.386
Request Arguments	1.386
Filter Expressions	1.388
Basic	1.388
.....	1.388
Example	1.388
Full	1.389
.....	1.389
Example	1.389
Sub-expressions	1.390
.....	1.390
Example	1.390
Modules	1.390

Example	1.391
Response Arguments	1.391
Response	1.392
Change Log	1.392
Change Log	1.399
/<module>/filter/:record GET	1.399
Overview	1.399
Summary	1.399
Request Arguments	1.399
Filter Expressions	1.401
Basic	1.401
Example	1.401
Full	1.402
Example	1.402
Sub-expressions	1.403
Example	1.403
Modules	1.404
Example	1.404
Response Arguments	1.404
Response	1.405
Change Log	1.405
Change Log	1.413
/<module>/globalsearch GET	1.413
Overview	1.413
Summary	1.413
Request Arguments	1.413
Request	1.413

.....	1.416
Response Arguments	1.416
Response	1.416
Change Log	1.417
/<module>/globalsearch POST	1.417
Overview	1.417
Summary	1.417
Request Arguments	1.417
Request	1.420
Response Arguments	1.420
Response	1.420
Change Log	1.421
/<module>/insertafter/:target POST	1.421
Overview	1.421
Request Arguments	1.421
Request	1.422
Response Arguments	1.422
Response	1.422
Change Log	1.422
/<module>/insertbefore/:target POST	1.423
Overview	1.423
Request Arguments	1.423
Request	1.423
Response Arguments	1.424
Response	1.424
Change Log	1.424

.....	1.424
./<module>/prepend/:target POST	1.424
Overview	1.424
Request Arguments	1.425
Request	1.425
Response Arguments	1.425
Response	1.425
Change Log	1.426
./<module>/:record DELETE	1.426
Overview	1.426
Request Arguments	1.426
Response Arguments	1.426
Response	1.427
Change Log	1.427
./<module>/:record GET	1.427
Overview	1.427
Request Arguments	1.427
Response Arguments	1.428
Response	1.428
Change Log	1.431
./<module>/:record PUT	1.431
Overview	1.431
Request Arguments	1.431
Request	1.431
Link fields	1.432
Response Arguments	

Response	1.432
Change Log	1.433
/<module>/record_list POST	1.435
Overview	1.436
Summary	1.436
Request Arguments	1.436
Request Example	1.436
Response Arguments	1.436
Output Example	1.437
/<module>/record_list/:record_list_id DELETE	1.437
Overview	1.437
Summary	1.438
Request Arguments	1.438
Request Example	1.438
Response Arguments	1.438
/<module>/record_list/:record_list_id GET	1.438
Overview	1.438
Summary	1.439
Request Arguments	1.439
Request Example	1.439
Response Arguments	1.439
Output Example	1.440
/<module>/:record/audit GET	1.440
Overview	1.440
Request Arguments	

Response Arguments	1.440
Response	1.441
Change Log	1.442
/<module>/:record/children GET	1.442
Overview	1.442
Request Arguments	1.442
Response Arguments	1.443
Response	1.443
Change Log	1.444
/<module>/:record/collection/:collection_name GET	1.445
Overview	1.445
Summary	1.445
Request Arguments	1.445
Response Arguments	1.446
Response	1.447
Change Log	1.448
/<module>/:record/collection/:collection_name/count GET	1.448
Overview	1.448
Summary	1.448
Request Arguments	1.449
Response Arguments	1.449
Response	1.449
Change Log	1.450
/<module>/:record/favorite DELETE	1.450
Overview	

.....	1.450
Request Arguments	1.450
Response Arguments	1.450
Response	1.450
Change Log	1.453
/<module>/:record/favorite PUT	1.454
Overview	1.454
Request Arguments	1.454
Response Arguments	1.454
Response	1.454
Change Log	1.457
/<module>/:record/file GET	1.457
Overview	1.457
Request Arguments	1.457
Response Arguments	1.457
Response	1.458
Change Log	1.461
/<module>/:record/file/:field DELETE	1.461
Overview	1.461
Request Arguments	1.461
Response Arguments	1.462
Response	1.462
Change Log	1.462
/<module>/:record/file/:field GET	1.463
Overview	1.463
Request Arguments	

Response Arguments	1.463
Response	1.463
Change Log	1.463
/<module>/:record/file/:field POST	1.464
Overview	1.464
Request Arguments	1.464
Request	1.465
Response Arguments	1.465
Response	1.465
Change Log	1.466
/<module>/:record/file/:field PUT	1.466
Overview	1.466
Request Arguments	1.467
Request	1.467
Response Arguments	1.467
Response	1.468
Change Log	1.468
/<module>/:record/link POST	1.469
Overview	1.469
Request Arguments	1.469
Request	1.470
Response Arguments	1.470
Response	1.470
Change Log	1.477
/<module>/:record/link/activities GET	

Summary:	1.477
Query Parameters:	1.477
Input Example:	1.478
Output Example:	1.478
/<module>/:record/link/activities/filter GET	1.479
Summary:	1.479
Query Parameters:	1.479
Input Example:	1.479
Output Example:	1.480
/<module>/:record/link/history GET	1.481
Overview	1.481
Summary	1.481
Request Arguments	1.481
/<module>/:record/link/:link_name GET	1.482
Overview	1.482
Summary	1.483
Request Arguments	1.483
Filter Expressions	1.485
Basic	1.485
Example	1.485
Full	1.485
Example	1.485
Sub-expressions	1.487
Example	1.487
Modules	

Example	1.487
Response Arguments	1.488
Response	1.488
Change Log	1.495
/<module>/:record/link/:link_name POST	1.495
Overview	1.495
Request Arguments	1.496
Request	1.496
Response Arguments	1.496
Response	1.496
Change Log	1.501
/<module>/:record/link/:link_name/add_record_list/:remote_id POST	1.501
Overview	1.501
URL Arguments	1.501
Request	1.502
Response Arguments	1.502
Response	1.502
Change Log	1.504
/<module>/:record/link/:link_name/count GET	1.504
Overview	1.504
Summary	1.505
Request Arguments	1.505
Filter Expressions	1.507
Basic	1.507
Example	1.507

Full	1.507
Example	1.507
Sub-expressions	1.509
Example	1.509
Modules	1.510
Example	1.510
Response Arguments	1.510
Reponse	1.511
Change Log	1.517
/<module>/:record/link/:link_name/filter GET	1.518
Overview	1.518
Summary	1.518
Request Arguments	1.518
Filter Expressions	1.520
Basic	1.520
Example	1.520
Full	1.521
Example	1.521
Sub-expressions	1.522
Example	1.522
Modules	1.523
Example	1.523
Response Arguments	1.524
Reponse	1.524
Change Log	1.524

.....	1.531
/<module>/:record/link/:link_name/filter/count GET	1.531
Overview	1.531
Summary	1.531
Request Arguments	1.532
Filter Expressions	1.534
Basic	1.534
Example	1.534
Full	1.534
Example	1.534
Sub-expressions	1.536
Example	1.536
Modules	1.537
Example	1.537
Response Arguments	1.537
Response	1.538
Change Log	1.545
/<module>/:record/link/:link_name/:remote_id DELETE	1.545
Overview	1.545
Request Arguments	1.545
Response Arguments	1.545
Response	1.546
Change Log	1.551
/<module>/:record/link/:link_name/:remote_id GET	1.551
Overview	1.551
Request Arguments	

Response Arguments	1.551
Response	1.552
Change Log	1.555
/<module>/:record/link/:link_name/:remote_id POST	1.555
Overview	1.556
Query String Parameters	1.556
Response Arguments	1.556
Response	1.556
Change Log	1.561
/<module>/:record/link/:link_name/:remote_id PUT	1.562
Overview	1.562
Request Arguments	1.562
Request	1.562
Response Arguments	1.562
Response	1.563
Change Log	1.567
/<module>/:record/moveafter/:target PUT	1.568
Overview	1.568
Request Arguments	1.568
Response Arguments	1.568
Response	1.569
Change Log	1.569
/<module>/:record/movebefore/:target PUT	1.569
Overview	1.569
Request Arguments	

Response Arguments	1.569
Response	1.570
Change Log	1.570
/<module>/:record/movefirst/:target PUT	1.571
Overview	1.571
Request Arguments	1.571
Response Arguments	1.571
Response	1.571
Change Log	1.572
/<module>/:record/movelast/:target PUT	1.572
Overview	1.572
Request Arguments	1.572
Response Arguments	1.573
Response	1.573
Change Log	1.573
/<module>/:record/next GET	1.573
Overview	1.573
Request Arguments	1.574
Response Arguments	1.574
Response	1.574
Change Log	1.575
/<module>/:record/parent GET	1.575
Overview	1.575
Request Arguments	1.575
Response Arguments	

Response	1.576
Response	1.576
Change Log	1.577
/<module>/:record/path GET	1.577
Overview	1.577
Request Arguments	1.577
Response Arguments	1.578
Response	1.578
Change Log	1.579
/<module>/:record/prev GET	1.579
Overview	1.579
Request Arguments	1.579
Response Arguments	1.580
Response	1.580
Change Log	1.581
/<module>/:record/subscribe POST	1.581
Summary:	1.581
Query Parameters:	1.581
Input Example:	1.581
Output Example:	1.582
/<module>/:record/unfavorite PUT	1.582
Overview	1.582
Request Arguments	1.582
Response Arguments	1.582
Response	1.583
Change Log	

.....	1.586
/<module>/:record/unsubscribe DELETE	1.586
Summary:	1.586
Query Parameters:	1.586
Input Example:	1.586
Output Example:	1.586
/<module>/:record/vcard GET	1.587
Overview	1.587
Request Arguments	1.587
Response Arguments	1.587
Response	1.587
Change Log	1.588
/<module>/:root/tree GET	1.588
Overview	1.588
Request Arguments	1.588
Response Arguments	1.589
Response	1.589
Change Log	1.595
/<module>/temp/file/:field POST	1.595
Overview	1.595
Request Arguments	1.595
Request	1.596
Response Arguments	1.596
Response	1.597
Change Log	1.597
/<module>/temp/file/:field/:temp_id GET	

Overview	1.597
Request Arguments	1.598
Response Arguments	1.598
Response	1.598
Change Log	1.599
/mostactiveusers GET	1.600
Overview	1.600
Request Arguments	1.600
Request	1.600
Response Arguments	1.601
Change Log	1.602
/oauth2/bwc/login POST	1.602
ATTENTION: FOR INTERNAL USAGE ONLY	1.602
Overview	1.602
/oauth2/logout POST	1.602
Overview	1.602
Request Arguments	1.603
Response Arguments	1.603
Response	1.603
Change Log	1.603
/oauth2/sudo:/user_name POST	1.604
Overview	1.604
Request Arguments	1.604
Request	1.604
Response Arguments	1.604

Response	1.605
Response	1.605
Change Log	1.605
/oauth2/token POST	1.606
Overview	1.606
Request Arguments	1.606
Request for Password Grant Types	1.608
Request for Refresh Token Grant Types	1.608
Response Arguments	1.608
Response	1.609
Change Log	1.610
/password/request GET	1.610
Overview	1.610
Request Arguments	1.610
Request	1.610
Response Arguments	1.611
Response	1.611
Change Log	1.611
/ping GET	1.612
Overview	1.612
Request Arguments	1.612
Response Arguments	1.612
Response	1.612
Change Log	1.612
/ping/whattimeisit GET	1.613
Overview	

.....	1.613
Request Arguments	1.613
Response Arguments	1.613
Response	1.613
Change Log	1.613
/pmse_Business_Rules GET	1.614
Overview	1.614
Summary	1.614
Request Arguments	1.614
Filter Expressions	1.618
Basic	1.618
Example	1.618
Full	1.619
Example	1.619
Sub-expressions	1.620
Example	1.621
Modules	1.621
Example	1.621
Response Arguments	1.622
Response	1.622
Change Log	1.624
/pmse_Emails_Templates GET	1.624
Overview	1.624
Summary	1.624
Request Arguments	1.625
Filter Expressions	

Basic	1.628
Example	1.629
Full	1.629
Example	1.629
Sub-expressions	1.629
Example	1.631
Modules	1.631
Example	1.632
Response Arguments	1.632
Response	1.632
Change Log	1.633
/pmse_Project GET	1.634
Overview	1.634
Summary	1.635
Request Arguments	1.635
Filter Expressions	1.635
Basic	1.639
Example	1.639
Full	1.639
Example	1.639
Sub-expressions	1.640
Example	1.641
Modules	1.641
Example	1.642
Response Arguments	1.642

	1.643
Response	1.643
Change Log	1.644
/recent GET	1.645
Overview	1.645
Request Arguments	1.645
Request	1.646
Response Arguments	1.646
Response	1.647
Change Log	1.653
/rssfeed GET	1.654
Overview	1.654
Request Arguments	1.654
Request	1.655
Response Arguments	1.655
Response	1.656
Change Log	1.657
/search GET	1.657
Overview	1.657
Request Arguments	1.657
Response Arguments	1.659
Response	1.659
Change Log	1.661
/theme GET	1.661
Overview	1.661
Request Arguments	

Request	1.661
Response Arguments	1.662
Response	1.663
Change Log	1.663
/theme POST	1.664
Overview	1.664
Request Arguments	1.664
Request	1.665
Response Arguments	1.665
Response	1.665
Change Log	1.665
Extending Endpoints	1.666
Overview	1.666
Custom Endpoints	1.666
Defining New Endpoints	1.666
registerApiRest() Method	1.668
Endpoint Method	1.671
Help Documentation	1.672
Quick Repair and Rebuild	1.675
Example	1.675
Redefining Existing Endpoints	1.677
Endpoint Scoring	1.679
Endpoint Versioning	1.681
API Exceptions	1.684
Overview	

.....	1.684
Stock Exceptions	1.684
Using Exceptions	1.688
Custom Exceptions	1.689
Example	1.689
Examples	1.691
Bash	1.691
How to Export a List of Records	1.691
Overview	1.691
Exporting Records	1.692
Filtering Records	1.692
Response	1.693
Creating a Record List	1.694
Response	1.695
Exporting Records	1.695
Response	1.695
How to Filter a List of Records	1.697
Overview	1.697
Filtering Records	1.698
Filtering Records	1.698
Response	1.699
How to Favorite a Record	1.700
Overview	1.700
Favoriting a Record	1.701
Favoriting a Record	1.701
Response	1.701

.....	1.702
How to Manipulate File Attachments	1.706
Overview	1.706
Manipulating File Attachments	1.706
Submitting a File Attachment	1.707
Create Note	1.707
Add An Attachment to the Note	1.707
Response	1.708
Getting a File Attachment	1.710
Response	1.710
How to Fetch Related Records	1.711
Overview	1.711
Fetching Related Records	1.711
Fetching Related Records	1.712
Response	1.713
How to Manipulate Records (CRUD)	1.724
Overview	1.724
CRUD Operations	1.724
Creating a Record	1.725
Response	1.725
Getting a Record	1.730
Response	1.730
Updating a Record	1.731
Response	1.731
Deleting a Record	1.736
Response	1.736

How to Follow a Record	1.736
Overview	1.737
Following a Record	1.737
Following a Record	1.737
Response	1.738
How to Unfavorite a Record	1.738
Overview	1.738
Unfavoriting a Record	1.738
Unfavoriting a Record	1.739
Response	1.739
How to Unfollow a Record	1.744
Overview	1.744
Unfollowing a Record	1.744
Unfollowing a Record	1.745
Response	1.745
How to Get the Most Active Users	1.746
Overview	1.746
Get Most Active Users	1.746
Active Users	1.746
Response	1.747
How to Authenticate and Log Out	1.748
Overview	1.748
Authenticating	1.748
Response	1.749
Logout	

Response	1.749
How to Ping	1.750
Overview	1.750
Pinging	1.750
Pinging	1.751
Response	1.751
How to Fetch the Current Users Time	1.751
Overview	1.751
Authenticating	1.751
Getting the Current Time and Date for the User	1.752
Response	1.752
How to Fetch Recently Viewed Records	1.752
Overview	1.752
Authenticating	1.753
Recently Viewed Records	1.753
Response	1.754
How to Use the Global Search	1.765
Authenticating	1.766
Searching Records	1.766
Response	1.767
How to Check for Duplicate Records	1.780
Overview	1.780
Duplicate Records	1.780
Retrieving Duplicates	1.780
Response	1.780

	1.781
	1.788
PHP	1.789
How to Export a List of Records	1.789
Overview	1.789
Exporting Records	1.789
Filtering Records	1.791
Request Payload	1.793
Response	1.794
Creating a Record List	1.795
Request Payload	1.796
Response	1.796
Exporting Records	1.796
Response	1.798
Download	1.799
How to Filter a List of Records	1.799
Overview	1.800
Filtering Records	1.800
Filtering Records	1.801
Request Payload	1.803
Response	1.804
Download	1.805
How to Favorite a Record	1.805
Overview	1.805
Favoriting a Record	1.806
Favoriting a Record	1.806

Request Payload	1.807
Response	1.808
Download	1.813
How to Manipulate File Attachments	1.813
Overview	1.813
Manipulating File Attachments	1.813
Submitting a File Attachment	1.815
Request	1.818
Response	1.818
Getting a File Attachment	1.820
Request	1.821
Response	1.822
Download	1.823
How to Fetch Related Records	1.823
Overview	1.823
Fetching Related Records	1.823
Fetching Related Records	1.825
Request	1.826
Response	1.826
Download	1.837
How to Manipulate Records (CRUD)	1.838
Overview	1.838
CRUD Operations	1.838
Creating a Record	1.840
Request Payload	

Response	1.841
Getting a Record	1.841
Updating a Record	1.846
Request Payload	1.847
Response	1.848
Deleting a Record	1.849
Request Payload	1.853
Response	1.854
Download	1.855
How to Follow a Record	1.855
Overview	1.855
Following a Record	1.855
Following a Record	1.855
Request Payload	1.857
Response	1.858
Download	1.858
Download	1.858
Download	1.858
How to Unfavorite a Record	1.859
Overview	1.859
Unfavoriting a Record	1.859
Unfavoriting a Record	1.859
Request Payload	1.861
Response	1.862
Download	1.862
Download	1.866
How to Unfollow a Record	

Overview	1.866
Unfollowing a Record	1.867
Unfollowing a Record	1.868
Request Payload	1.869
Response	1.869
Download	1.870
How to Get the Most Active Users	1.870
Overview	1.870
Get Most Active Users	1.870
Active Users	1.872
Request	1.873
Response	1.873
Download	1.874
How to Authenticate and Log Out	1.874
Overview	1.874
Authenticating	1.875
Request Payload	1.876
Response	1.877
Logout	1.877
Response	1.878
Download	1.878
How to Ping	1.879
Overview	1.879
Pinging	1.879
Pinging	1.879

Response	1.881
Download	1.882
How to Fetch the Current Users Time	1.882
Overview	1.882
Authenticating	1.882
Getting the Current Time and Date for the User	1.884
Request	1.884
Response	1.885
Downloads	1.885
How to Fetch Recently Viewed Records	1.885
Overview	1.885
Authenticating	1.885
Recently Viewed Records	1.887
Request	1.888
Response	1.888
Downloads	1.900
How to Use the Global Search	1.900
Overview	1.900
Authenticating	1.900
Searching Records	1.902
Request	1.903
Response	1.904
Downloads	1.916
How to Check for Duplicate Records	1.916

Overview	1.917
Duplicate Records	1.917
Retrieving Duplicates	1.919
Request Payload	1.920
Response	1.920
Download	1.928
v1 - v4.1	1.928
Overview	1.928
SOAP VS REST	1.928
REST	1.929
How do I access the REST service?	1.929
Input / Output Datatypes	1.929
Defining your own Datatypes	1.930
SOAP	1.931
How do I access the SOAP service?	1.931
WS-I 1.0 Compliancy	1.931
URL Parameters	1.932
use	1.932
Validation	1.932
SOAP Failure Response	1.932
What is NuSOAP?	1.933
Overview	1.933
Where Can I Get It?	1.933
How Do I Use It?	1.933

Example	1.933
Methods	1.934
get_available_modules	1.934
Overview	1.934
Available APIs	1.934
Definition	1.934
Parameters	1.935
Result	1.935
Change Log	1.936
PHP	1.936
get_document_revision	1.936
Overview	1.936
Available APIs	1.937
Definition	1.937
Parameters	1.937
Result	1.937
Change Log	1.938
PHP	1.938
get_entries	1.939
Overview	1.939
Available APIs	1.939
Definition	1.939
Parameters	1.939
Result	1.940
Change Log	

PHP	1.941
get_entries_count	1.943
Overview	1.943
Available APIs	1.943
Definition	1.943
Parameters	1.943
Result	1.944
Change Log	1.944
PHP	1.944
get_entry	1.945
Overview	1.945
Available APIs	1.945
Definition	1.945
Parameters	1.946
Result	1.946
Change Log	1.947
PHP	1.947
get_entry_list	1.949
Overview	1.949
Available APIs	1.949
Definition	1.949
Parameters	1.949
Result	1.950
Change Log	1.951
PHP	

.....	1.952
get_language_definition	1.953
Overview	1.953
Available APIs	1.953
Definition	1.954
Parameters	1.954
Result	1.954
Change Log	1.955
PHP	1.955
get_last_viewed	1.956
Overview	1.956
Available APIs	1.956
Definition	1.956
Parameters	1.956
Result	1.956
Change Log	1.957
PHP	1.957
get_modified_relationships	1.958
Overview	1.958
Available APIs	1.958
Definition	1.958
Parameters	1.959
Result	1.960
Change Log	1.961
Considerations	1.961
PHP	

.....	1.961
get_module_fields	1.963
Overview	1.963
Available APIs	1.963
Definition	1.963
Parameters	1.963
Result	1.964
Change Log	1.964
PHP	1.964
get_module_fields_md5	1.965
Overview	1.965
Available APIs	1.965
Definition	1.965
Parameters	1.966
Result	1.966
Change Log	1.966
PHP	1.966
get_module_layout	1.967
Overview	1.967
Available APIs	1.967
Definition	1.967
Parameters	1.967
Result	1.968
Change Log	1.969
PHP	1.969
get_module_layout_md5	

Overview	1.970
Available APIs	1.970
Definition	1.970
Parameters	1.971
Result	1.971
Change Log	1.972
PHP	1.972
get_note_attachment	1.973
Overview	1.973
Available APIs	1.973
Definition	1.973
Parameters	1.974
Result	1.974
Change Log	1.975
PHP	1.975
get_quotes_pdf	1.975
Overview	1.976
Available APIs	1.976
Definition	1.976
Parameters	1.976
Result	1.976
Change Log	1.977
PHP	1.977
get_relationships	1.977
Overview	

.....	1.978
Available APIs	1.978
Definition	1.978
Parameters	1.978
Result	1.979
Change Log	1.980
PHP	1.980
get_report_entries	1.982
Overview	1.982
Available APIs	1.982
Definition	1.983
Parameters	1.983
Result	1.983
Change Log	1.984
Considerations	1.984
PHP	1.984
get_report_pdf	1.985
Overview	1.985
Available APIs	1.985
Definition	1.985
Parameters	1.986
Result	1.986
Change Log	1.986
PHP	1.986
get_server_info	1.987
Overview	

Available APIs	1.987
Definition	1.987
Parameters	1.987
Result	1.988
Change Log	1.988
PHP	1.989
get_upcoming_activities	1.989
Overview	1.989
Available APIs	1.989
Definition	1.989
Parameters	1.990
Result	1.990
Change Log	1.990
PHP	1.991
get_user_id	1.991
Overview	1.991
Available APIs	1.991
Definition	1.991
Parameters	1.991
Result	1.992
Change Log	1.992
PHP	1.992
get_user_team_id	1.993
Overview	1.993
Available APIs	

Definition	1.993
Parameters	1.993
Result	1.993
Change Log	1.994
PHP	1.994
job_queue_cycle	1.994
Overview	1.994
Available APIs	1.995
Definition	1.995
Parameters	1.995
Result	1.995
Change Log	1.996
PHP	1.996
job_queue_next	1.996
Overview	1.996
Available APIs	1.997
Definition	1.997
Parameters	1.997
Result	1.997
Change Log	1.998
PHP	1.998
job_queue_run	1.998
Overview	1.998
Available APIs	1.998
Definition	

Parameters	1.999
Result	1.999
Change Log	2.000
PHP	2.000
login	2.000
Overview	2.001
Available APIs	2.001
Definition	2.001
Parameters	2.001
Result	2.002
Change Log	2.003
PHP	2.004
logout	2.005
Overview	2.005
Available APIs	2.005
Definition	2.006
Parameters	2.006
Result	2.006
Change Log	2.006
PHP	2.007
oauth_access	2.007
Overview	2.007
Available APIs	2.007
Definition	2.007
Parameters	

Result	2.008
Change Log	2.008
PHP	2.008
seamless_login	2.009
Overview	2.009
Available APIs	2.009
Definition	2.009
Parameters	2.009
Result	2.010
Change Log	2.010
Considerations	2.010
PHP	2.010
search_by_module	2.011
Overview	2.011
Available APIs	2.011
Definition	2.011
Parameters	2.011
Result	2.013
Change Log	2.013
PHP	2.013
set_campaign_merge	2.015
Overview	2.015
Available APIs	2.015
Definition	2.015
Parameters	2.015

Result	2.015
Change Log	2.016
PHP	2.017
set_document_revision	2.017
Overview	2.017
Available APIs	2.018
Definition	2.018
Parameters	2.018
Result	2.019
Change Log	2.019
PHP	2.019
set_entries	2.020
Overview	2.020
Available APIs	2.020
Definition	2.021
Parameters	2.021
Result	2.022
Change Log	2.022
Considerations	2.022
PHP	2.023
set_entry	2.025
Overview	2.025
Available APIs	2.025
Definition	2.025
Parameters	2.025

Result	2.026
Change Log	2.027
Considerations	2.027
PHP	2.027
set_note_attachment	2.029
Overview	2.029
Available APIs	2.029
Definition	2.029
Parameters	2.029
Result	2.030
Change Log	2.031
PHP	2.031
set_relationship	2.032
Overview	2.032
Available APIs	2.032
Definition	2.033
Parameters	2.033
Result	2.034
Change Log	2.035
PHP	2.036
set_relationships	2.037
Overview	2.037
Available APIs	2.037
Definition	2.038
Parameters	

Result	2.038
Change Log	2.039
PHP	2.040
snip_import_emails	2.041
Overview	2.044
Available APIs	2.044
Definition	2.044
Parameters	2.044
Result	2.046
Change Log	2.046
snip_update_contacts	2.047
Overview	2.047
Available APIs	2.047
Definition	2.047
Parameters	2.047
Result	2.048
Change Log	2.048
Extending v1 - v4.1 Web Services	2.049
Overview	2.049
Extending the API	2.049
Defining the Entry Point Location	2.049
Define the SugarWebServiceImpl Class	2.050
Define the Registry Class	2.052
Define the REST Entry Point	2.053
Define the SOAP Entry Point	

REST Release Notes	2.054
Overview	2.055
Release Notes	2.055
v4	2.056
v3_1	2.056
v3	2.057
v2_1	2.057
v2 (REST API was introduced into SugarCRM)	2.058
SOAP Release Notes	2.059
Overview	2.059
Release Notes	2.059
v4	2.060
v3_1	2.060
v3	2.060
v2_1	2.061
v2	2.061
Examples	2.065
REST	2.066
PHP	2.066
Creating Documents	2.066
Overview	2.066
Example	2.067
Result	2.072
Creating Notes with Attachments	2.073
Overview	

Example	2.073
Result	2.073
Creating or Updating a Record	2.078
Overview	2.080
Example	2.080
Result	2.080
Creating or Updating Multiple Records	2.084
Overview	2.085
Example	2.085
Result	2.085
Creating or Updating Teams	2.089
Overview	2.090
Example	2.090
Result	2.090
Logging In	2.095
Overview	2.096
Standard Authentication Example	2.096
LDAP Authentication Example	2.096
Result	2.099
Relating Quotes and Products	2.102
Overview	2.105
Example	2.106
Result	2.106
Retrieving a List of Fields From a Module	2.115
Overview	2.122

Example	2.122
Result	2.122
Retrieving a List of Records	2.126
Overview	2.128
Example	2.128
Result	2.129
Retrieving a List of Records With Related Info	2.133
Overview	2.136
Example	2.137
Result	2.137
Retrieving Email Attachments	2.142
Overview	2.154
Example	2.154
Result	2.154
Retrieving Multiple Records by ID	2.160
Overview	2.174
Example	2.174
Result	2.174
Retrieving Records by Email Domain	2.179
Overview	2.182
Example	2.182
Result	2.182
Retrieving Related Records	2.189
Overview	2.216
Example	2.216

	Results	2.216
	Searching Records	2.221
	Overview	2.224
	Example	2.224
	Result	2.225
	Result	2.229
SOAP		2.233
C#		2.233
Creating or Updating a Record		2.233
Overview		2.233
Example		2.234
Result		2.238
Logging In		2.238
Overview		2.239
Example		2.239
Result		2.241
PHP		2.242
Creating Documents		2.242
Overview		2.242
Example		2.243
Result		2.247
Creating Notes with Attachments		2.248
Overview		2.248
Example		2.248
Result		2.252
Creating or Updating a Record		

Overview	2.253
Example	2.253
Result	2.253
Creating or Updating Multiple Records	2.257
Overview	2.257
Example	2.257
Result	2.261
Creating or Updating Teams	2.261
Overview	2.262
Example	2.262
Result	2.266
Logging In	2.266
Overview	2.266
Standard Authentication Example	2.266
LDAP Authentication Example	2.268
Result	2.270
Relating Quotes and Products	2.274
Overview	2.274
Example	2.274
Result	2.284
Retrieving a List of Fields From a Module	2.285
Overview	2.285
Example	2.286
Result	2.289
Retrieving a List of Records	

Overview	2.291
Example	2.291
Result	2.295
Retrieving a List of Records With Related Info	2.298
Overview	2.298
Example	2.299
Result	2.303
Retrieving Multiple Records by ID	2.315
Overview	2.315
Example	2.315
Result	2.319
Retrieving Records by Email Domain	2.322
Overview	2.322
Example	2.322
Result	2.328
Retrieving Related Records	2.355
Overview	2.355
Example	2.355
Result	2.359
Searching Records	2.362
Overview	2.362
Example	2.363
Result	2.367
SugarHttpClient	2.370
Overview	

.....	2.370
The SugarHttpClient Class	2.370
Making a Request	2.371
Migration	2.373
Importing Records	2.373
Importing Email Addresses	2.374
Overview	2.374
Importing via API	2.374
Importing New Records	2.374
Email1 Field	2.375
Email Link Field	2.375
Updating Existing Records	2.377
Email1 Field	2.377
Email Link Field	2.378
Importing via Database	2.380
Checking for Duplicates	2.381
Removing Duplicates	2.383
Migrating From a Broken Instance to a Clean Install	2.384
Overview	2.385
Requirements	2.385
Migrating to a New Instance	2.386
Security	2.387
Endpoints and Entry Points	2.387
Overview	2.387
Description	2.388
Legacy Entry Points	

REST API Endpoints	2.388
Publication History	2.390
Web Server Configuration	2.391
Overview	2.391
SugarCRM .htaccess file	2.391
Securing .htaccess	2.392
Prevent version exposure	2.394
Directory listing and index page	2.395
Additional Options directives	2.396
Web server permissions	2.397
Disable unnecessary loadable modules	2.397
HTTP methods	2.398
Secure HTTP traffic	2.400
Cipher suite hardening	2.402
Public trusted root CA's	2.403
HSTS header	2.404
CSP header	2.405
Additional security headers	2.406
Secure cookie settings	2.406
Security modules	2.407
Additional resources	2.408
Publication History	2.408
XSS Prevention	2.409
Overview	2.409
Best Practices	

.....	2.409
Creating Safe Variables	2.410
JavaScript	2.410
Vulnerable Setup	2.411
Protected Setup	2.411
Handlebars Templates	2.412

Sugar Developer Guide 7.8

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

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

Last Modified: 03/15/2018 07:53pm

Introduction

Overview

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

Prerequisites

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

Understanding Sugar's Framework

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

key goals of this Developer Guide. For more information on Sugar's structure, please review the architecture section.

Supported Platforms

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

Sugar Editions

Sugar 7.x is available in three editions: Professional, Enterprise, and Ultimate, which are all sold under a commercial subscription agreement. These editions are developed by the same development team using the same source tree with extra modules available in the Enterprise and Ultimate editions.

Note: Some customers may subscribe to the Corporate edition, a legacy offering that is comparable to the Professional edition.

Basic Development Rules for Sugar Products

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

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

Development Tools

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

Developer Mode

Developer Mode will allow for Sugar to recompile 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. For more information, please refer to the System documentation.

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

Diagnostic Tool

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

The diagnostic tool has the ability to export the following:

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

-
- Sugar schema output (VARDEFS)

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

Composer

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

Last Modified: 03/15/2018 06:49pm

Development Methodology

Overview

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

Development Best Practices

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

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

Using `.gitignore` Files

Today, many developers choose to use Git as their source control management. There are certain Sugar application files that you do not want to track; most of

these are generated files that are created at runtime or are Sugar instance-specific configuration files.

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

```
*.log
/cache
/config.php
/config_override.php
/.htaccess
/custom/application
/custom/history
/custom/modules/*/Ext
/custom/blowfish
```

Recommendations for Development Teams

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

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

Deploying Sugar Code

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

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

-
- Sugar on-site projects : Develop these customizations using the exact version and flavor of Sugar that you plan to use in production.
 - Sugar cloud projects : Develop these customizations on the latest available version of Sugar for the particular flavor the customer has purchased.
 - Custom modules or integrations : If you plan on distributing your customization to many Sugar customers via SugarExchange or channel partners, design your customization with Sugar's cloud service in mind.
 - Sugar's cloud service is more restrictive than our on-site installs regarding supported customizations.
 - A customization designed for Sugar's cloud service can be supported in Sugar on-site instances, but the inverse is not always true.

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

Packaging

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

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

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

For these situations:

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

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

DML changes (if necessary) should be managed within SQL or PHP scripts.

Deployment

If using the traditional Git-based deployment, then deploying new Sugar code is as easy as checking out the latest branch and then running Quick Repair and Rebuild

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

In a coordinated deployment, database changes should be deployed first, followed by filesystem changes, followed by a Quick Repair (if permitted) to clear system caches. You can clear caches manually by deleting the contents of the `c./cache/` directory and then truncating the `metadata_cache` table in the Sugar database (in Sugar 7.6 and later). The Sugar application regenerates these caches as needed.

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

Managing Multiple Environments

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

SugarCRM recommends 4 different environments:

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

Code changes that flow upstream from a development environment should not be

allowed into production without going through quality assurance (QA) and user acceptance testing (UAT) first. The intermediate environments serve as gates between development and production that help intercept problems before they reach production.

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

Consistency

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

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

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

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

Testing

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

- Unit testing should be applied to ensure that even the smallest code units within your application are behaving as expected.
- Functional testing should be performed to ensure that each feature and

function behaves the way it was designed.

- System Integration testing should be performed to ensure that Sugar co-exists with external systems and that data flows between all these systems successfully.
- User acceptance testing should be performed by key stakeholders to ensure that your project is meeting business requirements.
- Performance testing should be performed to ensure that the Sugar application's responsiveness meets user expectations and that the application continues to scale.

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

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

Sugar Test Tools

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

Sugar Unit Tests

The Sugar 7 Unit Test suites are the automated unit tests developed and maintained by SugarCRM Engineering during the process of building and releasing each new Sugar 7 release. As part of our development process, these tests are required to run "green" (100% passing) at all times on each master release branch. Essentially, these tests form a regression test suite for an uncustomized version of Sugar 7 running in our controlled build environment.

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

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

continues to pass.

- Create new tests for new code customizations that you create.

Sugar Performance Tests

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

We also provide Apache JMeter scenarios to Sugar customers and partners who request access. These JMeter scenarios can be used to drive a simulated load against the Sugar REST API interface used by Sugar 7. They can validate that your customizations have not had an unexpected impact on the performance of a Sugar instance.

DevOps

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

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

Co-Existing with Studio Customizations

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

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

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

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

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

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

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

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

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

Last Modified: 03/19/2018 12:07pm

Composer

Overview

As of Sugar 7.5, the Composer dependency management integration has been made publicly available for Sugar developers. Because Composer is the de facto standard for managing PHP dependencies, this enhancement to the Sugar platform will make customizations based on external libraries more intuitive.

Prerequisites

- The Composer package must be available on your system if you wish to make any changes to the dependencies. Please refer to the Getting Started guide on Composer's website for more details on installation and usage.
- Composer is a command-line tool, so you must have access to the command-line interface on the system.
- You must have the proper file-system permissions to alter files in the Sugar instance directory.

Note: Before customizing the Composer configuration, be sure to read the documentation on this page in its entirety and feel confident in your knowledge of Composer. We strongly recommend reviewing the Composer website's documentation for more information.

Restrictions

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

Note: The list of restrictions may change at any time, opening up certain configuration keys or making them more restrictive. These changes will be communicated through release notes, and this page will be updated to reflect the latest state.

Root Element	Restriction Level	Explanation
<code>name</code>	Prohibited	These elements are owned by SugarCRM and should never be modified.
<code>description</code>	Prohibited	
<code>type</code>	Prohibited	
<code>license</code>	Prohibited	

homepage	Prohibited	These elements are owned by SugarCRM and should not be modified. They are under consideration for relaxed restriction levels in the future.
support	Prohibited	
autoload	Prohibited	
minimum-stability	Prohibited	
config	Prohibited	
require	Restricted	<ul style="list-style-type: none"> • You may add new packages without restriction. • You may add new repositories as long as they do not conflict with SugarCRM-owned packages. • You may remove packages if they are not owned by SugarCRM. • You may remove repositories if they are not owned by SugarCRM.
repositories	Restricted	
elements not listed here	No Restrictions	All other root elements may be added/changed at the developer's discretion.

Please contact Sugar Support if the current restrictions block you from using the Composer integration in Sugar. Also, review the Frequently Asked Questions section of this page for additional information.

Finding Packages

Packages that are available to Composer are published through Packagist. This is the primary place to look for third-party libraries. Composer is not restricted to packages published on Packagist, so you may refer to public or private repositories that are not found on Packagist. For more information, please refer to the Repositories section of Composer's documentation.

Adding a New Package

To add a new package, you must modify the "require" element and then update the dependencies. To learn more about Composer's "require" element, please refer to the Composer documentation. For information on version constraints, please refer to Composer's package versions documentation.

Follow these steps to add a new package:

1. Back up the files `./composer.json` and `./composer.lock`.
2. Modify the "require" section for `./composer.json`. For example, to add the `monolog/monolog` package to `./composer.json`, add the package name and the required version constraint in the "require" section, as shown 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"
  },
  ...
}
```

3. Next, update the dependencies. The easiest way to do this is by running `composer update --no-dev` from the directory where `composer.json` lives. This is the recommended method for production systems.
 - Composer will automatically determine which packages need to be fetched and updated.
 - All dependencies are stored in the `./vendor/` folder.
 - The `./composer.lock` file is updated with the installed versions and repository information.
 - The `--no-dev` switch skips the packages in the `require-dev` section because those packages are only needed for development.
 - Using the example in step 2, you will see the Monolog package has a dependency of its own: the `psr/log` package. Composer automatically installed this dependency.
4. Clear the cache used by Sugar's autoloader to be sure the newly installed dependencies are available. Manually remove these two files:
 - `./cache/file_map.php`
 - `./cache/class_map.php`

Alternatively, when developing new code in Sugar, you could enable

Developer Mode in Sugar. When enabled, the autoloader will automatically refresh itself.

Removing Packages

To remove a package, follow the steps in Adding a New Package, but this time, remove the package name and the required version constraint from the "require" section in step 2.

Note: Keep in mind the restrictions on the "require" section. Only remove packages that you control. Never remove packages that are owned by SugarCRM.

Adding and Removing Repositories

To configure additional repositories in `./composer.json`, you must first remove `{ "packagist.org": false }` from the "repositories" section. You can then add or remove repository references and update composer as needed. 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

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

Integrations

When updating the Composer configuration, Composer will generate different mappings based on the settings of every dependency:

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

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

Internals

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

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

Optimization

When updating dependencies through Composer, 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 and Rebuild.

Handling Upgrades

Once you take control of the Composer configuration, there may be complications

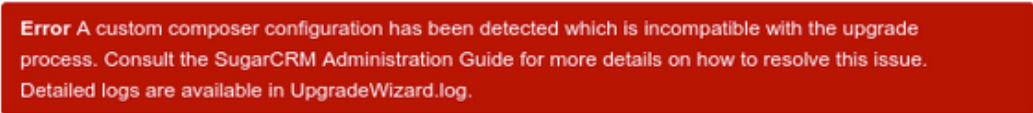
during upgrades. The upgrade logic will determine whether or not the Composer configuration has any customizations and whether a custom config is compatible for the upgrade.

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 before running the upgrade again.

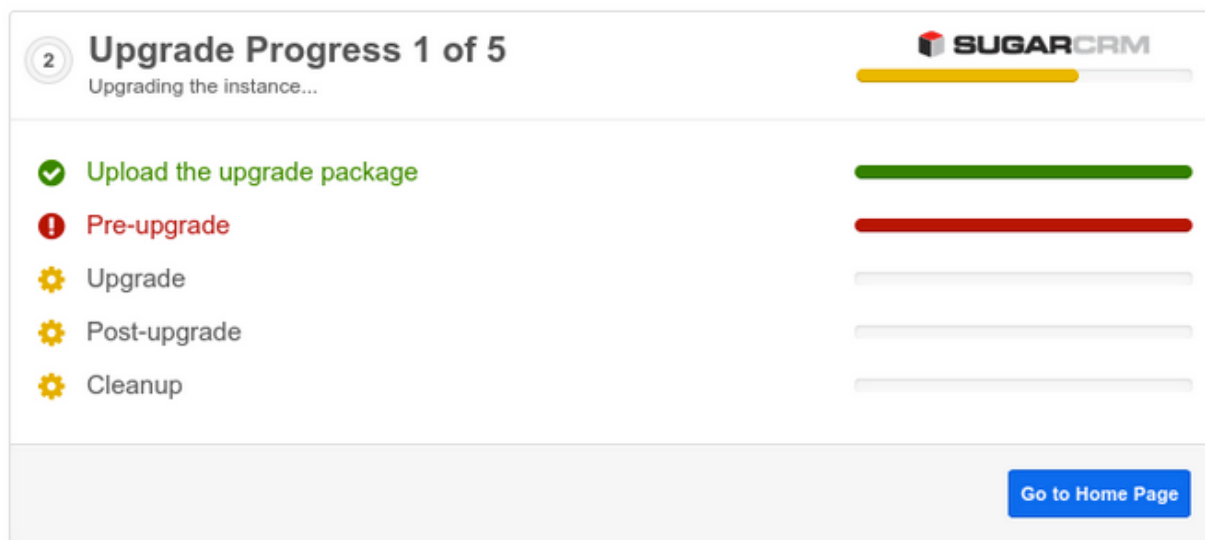
Upgrade Failure Notification

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

Web Upgrader



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.



The screenshot shows the 'Upgrade Progress 1 of 5' interface. It features a progress bar at the top with the SugarCRM logo. Below the progress bar, there is a list of steps: 'Upload the upgrade package' (completed, green), 'Pre-upgrade' (failed, red), 'Upgrade' (pending, grey), 'Post-upgrade' (pending, grey), and 'Cleanup' (pending, grey). A 'Go to Home Page' button is located at the bottom right.

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. Sections that are highlighted in red indicate 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. Log in and test the instance.
 6. Perform upgrade again.

What's 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

These definitions are owned by SugarCRM. In case one of these settings were already present before they 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, follow the procedure to run composer update, clear the autoloader cache, and test drive the Sugar instance. When approved, you can try to upgrade again.

Retry Upgrade

After resolving all conflicts, the upgrade process will successfully resolve. 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 79626e71bad09bflc09585c89a28875e 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
```

Frequently Asked Questions

Is the composer package required to install a Sugar instance?

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

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

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

the ability to expand from them.

Is the Composer package required to upgrade a Sugar instance?

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

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

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

May I change the version number of a package?

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

Can I use Composer's autoloader?

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

How can I optimize the autoloader?

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

Can I load customizations in Sugar through Composer?

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

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

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

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

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

Last Modified: 11/28/2017 04:25pm

Delivery and Deployment Guide for Enterprises

Overview

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

Deploying Application Configuration and Metadata

Use Case: Deployment of System Settings

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

Storage types:

- `config_override.php`
 - This is a file stored in the Sugar root directory that allows for overriding

core config values (found in config.php). In the UI, the main place to make changes to this is via Admin -> System Settings

- database 'config' table
 - It is loaded, used, and accessible throughout the Sugar application through the Config API.
 - System Tab Settings
 - Forecasting Settings
 - Portal Settings

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

config_override.php

System considerations:

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

Steps to migrate:

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

database 'config' table

System considerations:

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

Steps to migrate:

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

Screenshots:

Figure 1:

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

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

Use Case: Deployment of Reports

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

System considerations:

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

Additional notes:

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

Steps to migrate:

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

Use Case: Deployment of Dashboards

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

System considerations:

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

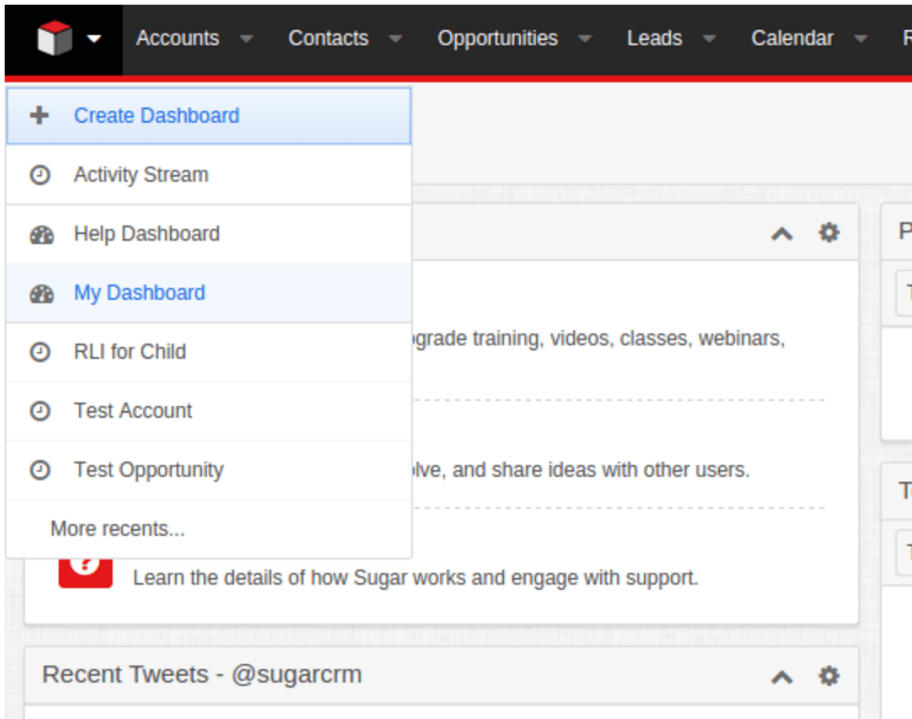
Steps to migrate:

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

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

Screenshots:

Figure 1



Use Case: Deployment of Roles

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

System considerations:

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

Steps to migrate:

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

Use Case: Deployment of Teams

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

System considerations:

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

Additional notes:

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

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

Steps to migrate:

1. Write a script using our Teams object API
 1. Create needed team object
 2. Set appropriate data on object and/or POST data
 3. After saving, potentially add users to the team
3. Execute script on dev instance and confirm changes
4. Use script to promote to next instance

Use Case: Deployment of User Settings

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

Storage types:

- User Preferences

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

- Users module settings

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

User preferences

System considerations:

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

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

Steps to Migrate:

1. Write a script using our User Preferences API
 1. Query the database to retrieve the row for a given user
 2. base64 decode the value
 3. unserialize the value
 4. update the data required
 5. serialize the data
 6. base64 encode
 7. rewrite the row to the database
 8. (repeat for all applicable users)
 1. (See modules/UserPreferences/UserPreference.php or modules/Users/User.php, specifically getPreference() and setPreference())
 2. Load the User object
 3. Call getPreference for the specified value
 4. Make changes
 5. Call setPreference for the specified value

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

User table

System considerations:

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

Steps to Migrate:

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

Use Case: Deployment of custom fields

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

System considerations:

- File System:
 - Files are potentially created in the following directories:
 - Custom field vardef:
custom/Extension/modules/<module_name>/Ext/Vardefs/sugarfield_<field_name>.php
 - Custom field label (and app_list_string if necessary):
./custom/Extension/modules/Accounts/Ext/Language/en_us.lang.php
- Database:
 - The <module_name>_cstm table is created, if it doesn't already exist.
 - The field <field_name>_c is created against that table
- Scripts required
 - A Quick Repair and Rebuild is required after copying the files and fields_meta_data table values.
 - SQL
 - You will need to insert the relevant entries from the fields_meta_data table

Steps to migrate:

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

Use Case: Deployment of custom modules

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

System considerations:

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

Additional notes:

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

Steps to migrate:

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

4. Test functionality

Use Case: Deployment of custom Relationships

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

Follow the same instructions as for Custom Fields, but:

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

Otherwise, the same process applies.

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

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

System considerations:

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

Additional notes:

- For layouts(record) you have the option to simply save the modified layout. In this case the metadata for the layout can be found in

`./custom/working/modules/<module>/<platform>/views/<view>/<view>.php`

- Layouts created from studio create views in `./custom/modules/<module>/views`
The created views can be record | list | selection-list
- For the two popup layouts(created from studio), extra metadata is provided in the `./custom/modules/<module>/metadata/popupdefs.php`

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

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

System considerations:

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

Additional notes:

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

Deploying Application Code and Integrations

Use Case: Deployment of custom CSS (LESS)

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

System considerations:

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

Use Case: Deployment of Logic Hooks and Web Logic Hooks

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

System considerations:

Logic Hook:

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

Web Logic Hook:

- Database: YES
 - Only for weblogic hooks : row is inserted into the weblogichooks table.
- Scripts required
 - After deploying the custom hooks and database entry in the weblogichooks table, you will need to run the Quick Repair and Rebuild script found in the admin section.
 - SQL
 - Script required to import the entry from the weblogichooks table.

Use Case: Deployment of custom API endpoints

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

System considerations:

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

Additional notes:

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

Use Case: Deployment of custom Administration Panels

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

System considerations:

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

and depending on the admin panel url, either:

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

or

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

Additional notes:

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

Use Case: Deployment of custom Jobs / Schedulers

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

System considerations:

- File System: YES:
 - ./custom/Extension/modules/Schedulers/Ext/ScheduledTasks/<jobname>.php
 - ./custom/Extension/modules/Schedulers/Ext/Language/<langtype.jobname>.php
- Database: YES
 - Row is inserted into the schedulers table, if a job is added as a scheduled job using Administration > Scheduler.
- Scripts required
 - After deploying the custom jobs, you will need to run the Quick Repair and Rebuild script found in the admin section.
 - SQL
 - Script required to import the entry from the schedulers table.

Last Modified: 03/16/2018 03:19pm

Migration Guide

Purpose

The purpose of this document is to provide insight to Sugar Developers for upgrading custom Sugar code, extensions, and integrations to Sugar 7.8. This guide focuses on changes in Sugar 7.8 that could cause an immediate impact on Sugar customizations and integrations built for Sugar 7.7.

Expected to Affect Most Developers

This section explains items expected to have widest impact on Sugar customizations and integrations when migrated to Sugar 7.8. We expect that these items will affect most developers.

Support for PHP 5.6 only

In this release, Sugar has dropped support for PHP versions that had reached their end of life. This means we have dropped support for PHP 5.3, 5.4, and 5.5. Sugar Developers should ensure their development environments are setup with PHP 5.6 and that any on-site hosting environments are also using PHP 5.6 when running Sugar 7.8.

For more details on the benefits of PHP 5.6 and what it means for Sugar Developers please review the following blog posts.

<https://developer.sugarcrm.com/2016/07/25/sugar-now-supports-php-5-6-in-sugar-7-7-1/>

<https://developer.sugarcrm.com/2016/08/08/preparing-sugar-customizations-for-php-5-6-and-7-0/>

Upgrade to Backbone v1.2.3

Backbone.js is the key library that serves as basis for the Sidecar framework used for Sugar client (front-end) development. This library underwent a long overdue upgrade from v0.9.10 to v1.2.3 in the Sugar 7.8 release. There were a significant number of API changes to Backbone that can affect custom Sidecar code. It is therefore essential that Sugar Developers test their custom Sidecar controllers in Sugar 7.8 to assure they continue to behave as expected.

A Sugar Developer blog was posted on July 13th, 2016 that covers in deep technical detail a number of impactful changes. Please review this blog post for more details.

<https://developer.sugarcrm.com/2016/07/13/prepare-for-the-backbone-js-upgrade-in-sugar-7-8/>

Custom code that relies on Backbone v0.9.10 will break in Sugar 7.8.

Expected to Affect Some Developers

This section explains items expected to have a moderate impact on Sugar customizations and integrations when migrated to Sugar 7.8. It is expected that these items will affect some developers.

Removal of deprecated or empty Sidecar controllers, classes, and methods

Where these APIs are in use, this change will cause breakages.

A number of previously deprecated Sidecar classes and methods have been removed in this Sugar release. We have also removed a number of empty Sidecar controllers that were unnecessarily included in the codebase. If you are using any of the following then you need to update your code to remove these references. Where there is an alternative API, we have identified it in the format: Old API → Alternative API

Sugar Dashlets

`./clients/base/views/news`

Sidecar Methods

`app.Bean.setDefaultAttributes()` ? `app.Bean.setDefault()`

`app.Bean.setDefaultAttribute()` ? `app.Bean.setDefault()`

`app.Bean.removeDefaultAttribute()`

`app.Bean.getDefaultAttribute()` ? `app.Bean.getDefault()`

`app.Bean.getDefaultAttributes()` ? `app.Bean.getDefault()`

`app.date.compareDates()` ? `app.date.compare()`

`app.date.isDateAfter()` ? `app.date.isAfter()`

`app.date.isDateBefore() ? app.date.isBefore()`

`app.date.isDateOn() ? app.date.isSame()`

`app.date.isDateBetween() ? app.date.isBetween()`

Sidecar Controllers

`./clients/base/layouts/subpanels-create/subpanels-create`

`./clients/base/views/history-summary-preview-header`

`./clients/base/views/passwordmodal`

`./clients/portal/views/passwordmodal`

`./modules/ACLRoles/clients/base/layouts/records`

`./modules/Administration/clients/base/layouts/records`

`./modules/Calendar/clients/base/layouts/records`

`./modules/Campaigns/clients/base/layouts/records`

`./modules/Contracts/clients/base/layouts/records`

`./modules/ContractTypes/clients/base/layouts/records`

`./modules/KBContents/clients/base/views/prefilteredlist`

`./modules/Quotes/clients/base/views/panel-top-for-accounts`

`./modules/RevenueLineItems/clients/base/views/subpanel-list-with-massupdate`

Enforcement of CSRF Tokens

In this Sugar release, the `$sugar_config['csrf']['soft_fail_form']` setting will default to `false` and the `$sugar_config['csrf']['opt_in']` setting has been removed. This means that CSRF tokens are now enabled out of the box.

For more details on Sugar's CSRF implementation, please review the following post on the Sugar Developer blog.

<https://developer.sugarcrm.com/2015/12/01/csrf-tokens-in-sugar-7-7/>

HTML forms used in BWC modules need to be authenticated in order to continue to function. If necessary, Sugar Developers can temporarily set the following configuration variable to disable enforcement of CSRF tokens.

```
$sugar_config['csrf']['soft_fail_form'] = true
```

The above setting will generate FATAL messages in the `sugarcrm.log` whenever a CSRF authentication failure happens. Sugar Developers and Administrators should not use this configuration in production instances.

jQuery library upgrade

In this Sugar Release, we have upgraded jQuery from version 1.7.1 to version 1.11.3. This jQuery affects both Sidecar and BWC modules. Note that the jQuery 1.9 release was a major release that modified many existing jQuery APIs. So we have included jQuery Migrate 1.2.1 plug-in to preserve behavior for custom code using pre-1.9 jQuery APIs.

Please review the jQuery 1.9 Upgrade Guide for more details:
<http://jquery.com/upgrade-guide/1.9/>

For more details on the jQuery Migrate plug-in, please review the README:
<https://github.com/jquery/jquery-migrate#readme>

Sugar Developers migrate code using old jQuery APIs and re-test code that makes use of jQuery to ensure there are no unexpected changes in behavior.

Underscore library upgrade in Sidecar framework

The Underscore.js library has been upgraded from v1.4.4 to v1.8.3. Sugar Developers that have written code that uses Underscore APIs should test their customizations to ensure regressions have not occurred.

An Underscore API change introduced in version 1.5.0 means that using `_.bindAll(object)` no longer works. You must specify the specific method names to bind using `_.bindAll(object, methodNames)`. Any custom Sugar code that relies on `_.bindAll(object)` needs to be updated. We recommend Sugar Developers use Underscore `_.bind()` or some other alternative to pass context to individual callback methods instead.

There are, in other cases, a change in order of parameters used in some Underscore APIs. Usage of Sidecar provided APIs is unaffected by these changes

in Underscore.

For full details, please refer to Underscore.js changelogs:
<http://underscorejs.org/#changelog>

Changes in application footer Handlebars template

There has been a change to how the SugarCRM logo in the footer of the application is rendered. The logo URL is no longer part of the FooterLayout Handlebars template. It is now set using JavaScript within the FooterLayout controller each time the layout is rendered. These changes may affect those instances that have customized the theme and branding of the Sugar application.

Changes to the Preview layout

Previously, the Preview components were initialized only when first rendering the page. This sometimes meant the Preview components were not properly initialized for the correct module when previewing a record. The Preview components (such as the PreviewView) are now re-initialized and re-rendered whenever a preview icon is clicked to ensure it is properly initialized.

This change in behavior can affect code that relied on the old Preview initialization procedure.

Additionally, the Preview Layout metadata has changed to support lazy loading for performance improvements. Any custom Preview Layout metadata should have the `lazy_loaded` flag set to true.

For example,

```
$viewdefs['base']['layout']['preview'] = array(
    'lazy_loaded' => 'true',
    'components' => array(
        ...
    ),
    ...
)
```

The `oauth_token` URL parameter is no longer enabled by default

A convenience feature that allowed an OAuth 2.0 access token to be passed via the `oauth_token` URL parameter instead of using the OAuth-Token HTTP header is no

longer enabled by default in Sugar 7.8. This was disabled due to security concerns with passing and accepting session identifiers using URL parameters since this is against OWASP guidelines. The examples in this Sugar Developer Guide use the OAuth-Token HTTP header.

For reference:

[Session Management Cheat Sheet - OWASP](#)

This HTTP request is no longer supported

```
GET /sugar/rest/v10/Contacts?oauth_token={access-token} HTTP/1.1
Host: localhost:8080
Content-Type: application/json
Cache-Control: no-cache
```

This HTTP request continues to be supported

```
GET /sugar/rest/v10/Contacts HTTP/1.1
Host: localhost:8080
Content-Type: application/json
OAuth-Token: {access-token}
Cache-Control: no-cache
```

If you need to enable this feature temporarily then you can use the following SugarConfig setting.

```
$sugar_config['allow_oauth_via_get'] = true;
```

Expected to Affect Few Developers

This section explains items expected to have a low impact on Sugar customizations and integrations when migrated to Sugar 7.8. It is expected that these items will affect few developers.

Removal of external Job Queue framework classes added in Sugar 7.7

Where these APIs are in use, this change will cause breakages.

The partial implementation for an external job queue framework that was added in Sugar 7.7 has been removed in the Sugar 7.8 release. We decided to not go forward with the Sugar 7.7 based implementation of the external job queue. The cron.php based job queue and scheduler continues to be supported.

```
./queueManager.php
```

```
./src/JobQueue/
```

Removal of unused code in Advanced Workflow

Where these APIs are in use, this change will cause breakages.

Unused Advanced Workflow related code has been removed in this Sugar release. This does not have an impact on Advanced Workflow behavior.

```
./modules/pmse_Inbox/engine/Crypt.php
```

```
./modules/pmse_Inbox/engine/PMSEAccessManager.php
```

```
./modules/pmse_Inbox/engine/PMSELicenseManager.php
```

```
./modules/pmse_Project/pmse_BpmAccessManagement/
```

Removal of unused SugarMVC Notifications controller

Where these APIs are in use, this change will cause breakages.

In this Sugar release, we have removed the empty SugarMVC Notifications module controller that has no longer been in use by Sugar.

```
./modules/Notifications/controller.php
```

Custom code that used or included these classes or files will need to be updated.

Removal of setFields parameter from Sidecar loadData() methods

Where these APIs are in use, this change will cause breakages.

There has been a change in the supported parameters for Sidecar's `Layout#loadData()` and `View#loadData()` methods in this Sugar release. The optional `setFields` parameter could be used to change the fields set on the current Context. This allowed a single Sidecar component on the page to alter the behavior of other Sidecar components on the page which is unwise. So support for the `setFields` parameter has been removed.

The new method signature for Sidecar `loadData()` functions is now:

```
loadData(options);
```

TinyMCE library upgrade for Sidecar

The TinyMCE library has been upgraded from v3.5.8 to v4.1.8 for Sidecar modules. The TinyMCE version used in BWC modules has not changed. This may affect custom code that uses TinyMCE. For more details view the TinyMCE changelog. <https://www.tinymce.com/docs/changelog/>

Timepicker library upgrade

The jQuery timepicker plugin has been upgraded from v1.4.10 to v1.8.8 in this Sugar release. No changelog is available so please use the jQuery timepicker Github repository to review API changes between these versions. <https://github.com/jonthornton/jquery-timepicker/tree/1.8.8>

Extraneous render in Record View

The Sugar Record view (RecordView) was being rendered twice on each page load. This performance issue has been corrected. Now the Record view renders once and the individual Field components are re-rendered when data is available. Sugar Developer should test their record view customizations to ensure this does not introduce any regressions in code that relied on this behavior.

Specifically, Developers should test if custom Sidecar code relies on field data being available during Record view render then this will break because Views are rendered before Fields.

Change in default model used for Sidecar's BeanCollection class

The default model class for Sidecar's Data.BeanCollection is now Data.Bean instead of Backbone.Model. This change can affect code that expected Backbone.Model.

Removal of oracle_enable_ci setting

The Sugar Config setting oracle_enable_ci is being removed. It was a setting previously because case insensitive query behavior on Oracle was only obtained at a performance cost. This setting was removed because case insensitivity does not impact performance anymore for Oracle.

This can affect Sugar deployments using this setting but an internal survey of Oracle customers determined that use of this setting is limited.

Deprecation of Tooltip and EllipsisInline Sidecar plug-ins

The Tooltip and EllipsisInline Sidecar plug-ins have been deprecated in this release. As of Sugar 7.8, these plug-ins are ignored and the client will instead simply look for `ellipsis_inline` class and `rel=tooltip` attribute on DOM elements. Tooltips are now initialized during mouseover event.

Sugar Developers should verify that `ellipsis_inline` class and `rel=tooltip` attribute is not used where it is not needed otherwise ellipsis and tooltips may begin to appear in places where it did not appear in the past.

Sidecar Relate Field changes

The Sidecar Field `def` property that merged a field's `viewdefs` and `vardefs` has been deprecated in this Sugar release. This was necessary, in part, to address issues in Sidecar's Relate field.

The Sidecar `RelateField` has been changed in this release to rely on `this.viewDefs` and `this.fieldDefs` properties instead of the deprecated `this.def` property. This may affect some code customizations that have been made to Relate fields. For example, custom fields that extend from the `RelateField` controller.

Sidecar Alert auto-close changes

The default automatic close timeout when using Sidecar Alerts is now 5 seconds instead of 9 seconds. This may affect users expecting a longer timeout. Sugar Developers can optionally providing a delay time in milliseconds to change this behavior.

For example,

```
// This Sidecar Alert message will automatically close after 10
seconds
app.alert.show('message-id', {
  level: 'success',
  messages: 'Task completed!',
  autoClose: true
  autoCloseDelay : 10000
```

```
});
```

Last Modified: 04/25/2018 04:09am

User Interface

Overview

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

Clients

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

base

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

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

mobile

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

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

portal

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

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

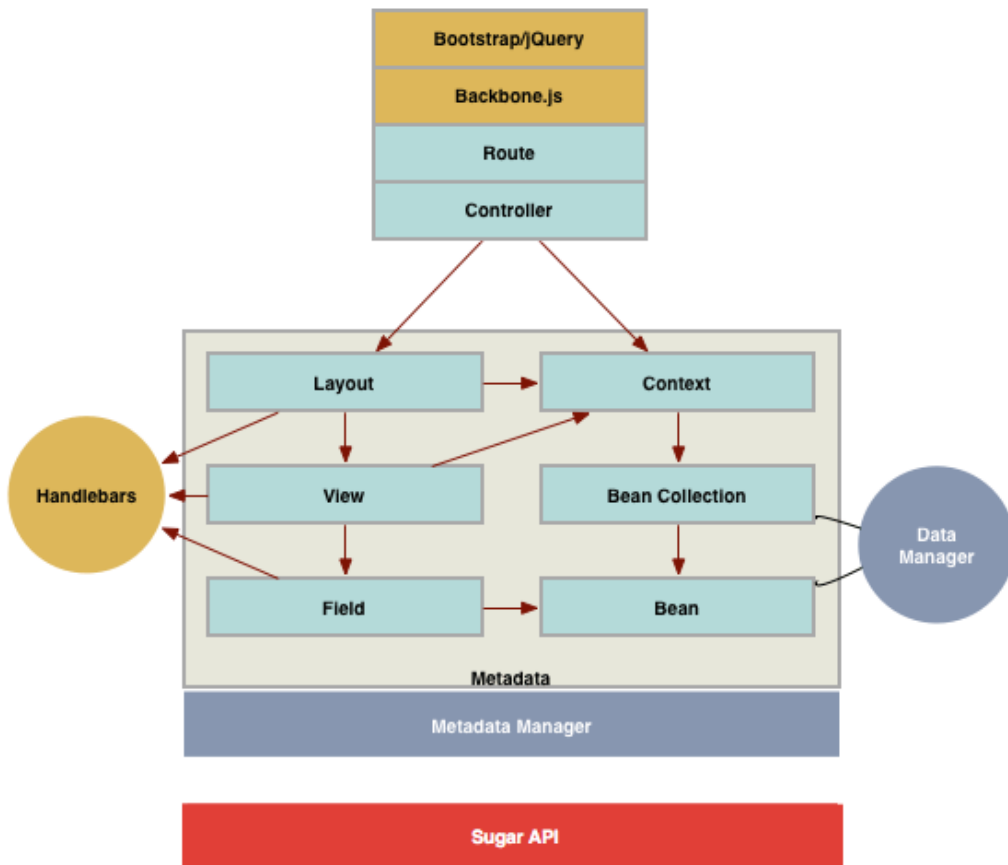
Last Modified: 11/27/2017 12:24pm

Sidecar

Overview

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

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



Composition

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

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

Backbone.js

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

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

Components

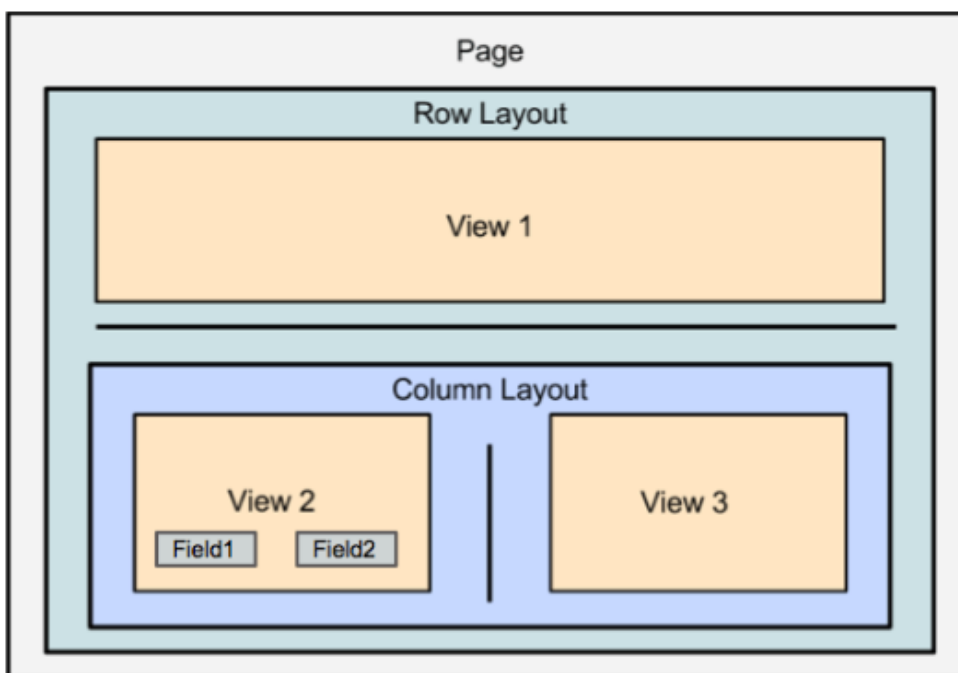
Everything that is renderable on the page is a component. A layout is a component that serves as a canvas for one or more views and other layouts. All pages will have at least one master layout, and that master layout can contain multiple nested layouts.

Layouts

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

Example layouts include:

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



For more information about the various layouts, please refer to the [Layouts](#) page of this documentation.

Views

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

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

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

Fields

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

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

Context

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

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

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

Last Modified: 10/14/2016 04:12pm

Events

Overview

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

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

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

Existing Backbone Event Catalog

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

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

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

Sidecar Events

Global Application Events

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

Name	Description
app:init	Triggered after the Sidecar application initializes Note: Initialization registers events,

	builds out Sidecar API objects, loads public metadata and config, and initializes modules.
app:start	Triggered after the Sidecar application starts
app:sync	Triggered when metadata is being synced with the user interface, for example, after login has occurred
app:sync:complete	Triggered after metadata has completely synced
app:sync:error	Triggered when metadata sync fails
app:sync:public:error	Triggered when public metadata sync fails during initialization
app:view:change	Triggered when a new view is loaded
app:locale:change	Triggered when the local changes
lang:direction:change	Triggered when the local changes and the direction of the language is different
app:login	Triggered when the "Login" route is called
app:login:success	Triggered after a successful login
app:logout	Triggered when the application is logging out
app:logout:success	Triggered after a successful logout

Bean Events

The following table lists bean object events.

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

error:validation:<fieldName>	Triggered when a particular field has a bean validation error
attributes:revert	Triggered when the bean reverts to the previous attributes

Context Events

The context object is used to facilitate communication between different Sidecar components on the page using events. For example, the `button:save_button:click` event triggers whenever a user clicks the Save button. The record view uses this event to run Save routines without being tightly coupled to a particular Save button. A list of these contextual events is not plausible because the user interface is continuously changing between versions, and there are many more possibilities based on the views and layouts in each version.

Utilizing Events

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

```
./custom/include/javascript/myAppLogoutSuccessEvent.js
```

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

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

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

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

$js_groupings[$key]['custom/include/javascript/myAppLogoutSuccessEvent
```

```
.js'] = 'include/javascript/sugar_grp7.min.js';
    }
    break;
}
}
```

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

Last Modified: 04/25/2018 04:09am

Routes

Overview

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

Routes

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

Route Definitions

The router accepts route definitions in the following format:

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

```
    route: "pattern/:variable",
    callback: "<callback name>"
  }
]
```

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

Route Patterns

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

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

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

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

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

- `module`
- `id`

Custom Routes

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

`route()`

Arguments

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

Example

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

`./custom/javascript/customRoutes.js`

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

addRoutes()

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

contain a variable.

Arguments

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

Example

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

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

`./custom/javascript/customRoutes.js`

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

```
        app.router.addRoutes(routes);
    })
})(SUGAR.App);
```

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

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

```
<?php

foreach ($js_groupings as $key => $groupings) {
    foreach ($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/customRoutes.js'] =
'include/javascript/sugar_grp7.min.js';
        }
        break;
    }
}
```

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

Last Modified: 10/30/2017 11:21am

Handlebars

Overview

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

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

Templates

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

Debugging Templates

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

This is an example of using log in an HBS template:

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

Helpers

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

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

Creating Helpers

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

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

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

```
/**  
 * Handlebars helpers.  
 *  
 * These functions are to be used in handlebars templates.
```

```

* @class Handlebars.helpers
* @singleton
*/
(function(app) {
  app.events.on("app:init", function() {

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

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

  });
})(SUGAR.App);

```

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

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

```

<?php

//Loop through the groupings to find
include/javascript/sugar_grp7.min.js
foreach ($js_groupings as $key => $groupings)
{
  foreach ($groupings as $file => $target)
  {
    if ($target == 'include/javascript/sugar_grp7.min.js')
    {
      //append the custom helper file

$js_groupings[$key]['custom/JavaScript/my-handlebar-helpers.js'] =
'include/javascript/sugar_grp7.min.js';
    }
  }
}

```

```
        break;
    }
}
```

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

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

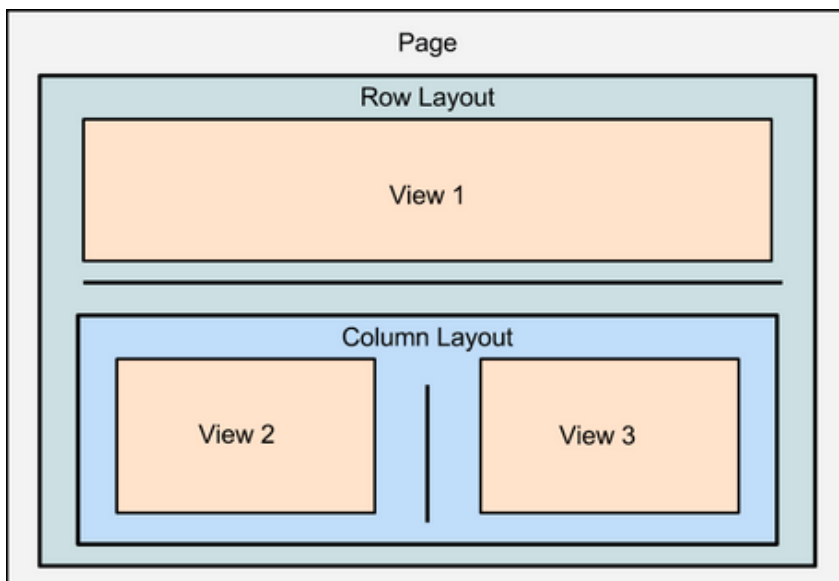
```
{{customUpperCase "MyString"}}
{{customLowerCase "MyString"}}
```

Last Modified: 11/27/2017 11:11am

Layouts

Overview

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

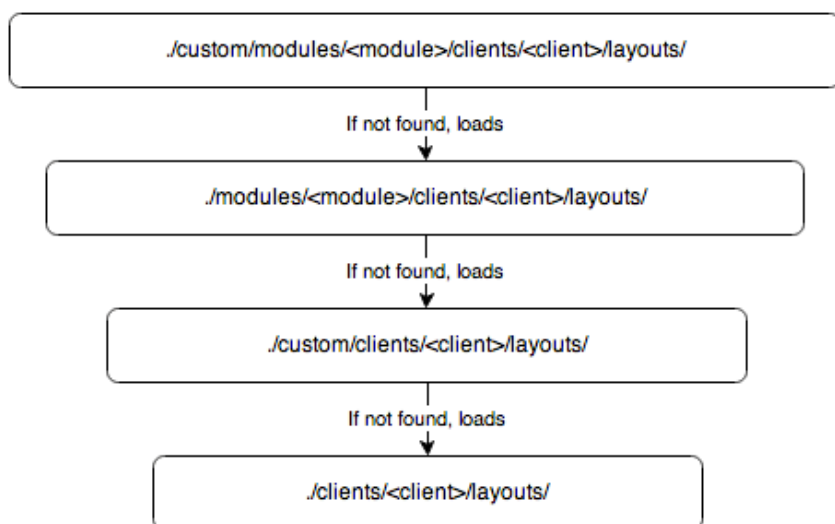


Layout components are typically made up of a controller JavaScript file (.js) and a

PHP file (.php), however, layout types vary and are not dependent on having both files.

Hierarchy Diagram

The layout components are loaded in the following manner:



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

Sidecar Layout Routing

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

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

Layout Example

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

JavaScript

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

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

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

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

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

    // Helper to create boiler plate layout containers
    function createLayoutContainers(self) {
      // Only creates the containers once
      if (!self.$el.children()[0]) {
        comp.$el.addClass('list');
      }
    }
    createLayoutContainers(this);
    // All components of this layout will be placed within the
    // innermost container div.
    this.$el.append(comp.el);
  }
})
```

Layout Definition

The layout definition is contained as an array in `list.php`. This layout definition

contains four views:

- massupdate
- massaddtolist
- recordlist
- list-bottom

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

```
<?php
```

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

Application

For information on working with layouts, please refer to the [Creating Layouts and](#)

Last Modified: 10/14/2016 04:12pm

Creating Layouts

Overview

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

Creating the Layout

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

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

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

Creating a View

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

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

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

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

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

<h1>My View</h1>

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

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

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

Navigating to the Layout

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

Last Modified: 10/14/2016 04:12pm

Overriding Layouts

Overview

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

-
1. Override the Layout
 2. Extend the View

These steps are explained in the following sections.

Overriding the Layout

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

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

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

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

```
<?php
```

```
$viewdefs['base']['layout']['record'] = array(  
    'components' => array(  
        array(  
            'layout' => array(  
                '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',
'subpanels',
'icon-table',
'LBL_DATA_VIEW',
'icon-table',
'LBL_LISTVIEW',
'activitystream',
'icon-th-list',
'LBL_ACTIVITY_STREAM',
'.filter',
'prepend'
'filter-actions',
'.filter-options'

'defaults' => array(
  'toggle-view' =>
),
),
'availableToggles' => array(
  array(
    'name' => 'subpanels',
    'icon' =>
    'label' =>
  ),
  array(
    'name' => 'list',
    'icon' =>
    'label' =>
  ),
  array(
    'name' =>
    'icon' =>
    'label' =>
  ),
),
'components' => array(
  array(
    'layout' => 'filter',
    'targetEl' =>
    'position' =>
  ),
  array(
    'view' =>
    "targetEl" =>
  ),
  array(

```

```

'filter-rows',
'.filter-options'
'activitystream',
'Activities',
'subpanels',

        'view' =>
        "targetEl" =>
    ),
    array(
        'layout' =>
        'context' =>
        array(
            'module' =>
        ),
    ),
    array(
        'layout' =>
    ),
    ),
    ),
    ),
    '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, navigate to Admin > Repair > Quick Repair and Rebuild.

Last Modified: 10/14/2016 04:12pm

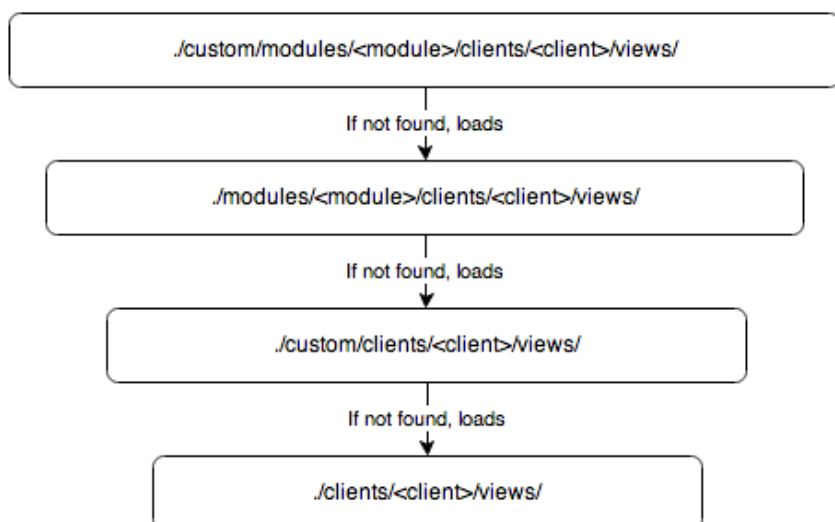
Views

Overview

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

Load Order Hierarchy Diagram

The view components are loaded in the following manner:



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

Components

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

Controller

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

Handlebar Template

The views template is built on Handlebars and is what adds the display markup for the data. The template is typically named after the view or an action in the view. In the example of the record view, the main template is 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
```

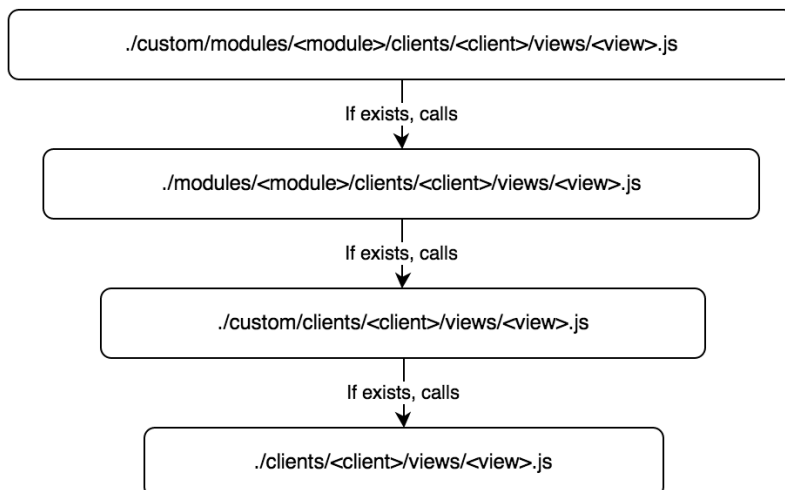
```
( {  
  extendsFrom: 'RecordView',  
  
  /**
```

```

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

Controller

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

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

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

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

Attributes

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

Handlebar Template

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

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

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

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

Helpers

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

Last Modified: 04/26/2018 12:00am

Metadata

Overview

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

View Metadata Framework

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

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

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

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

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

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

View Metadata

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

```

        '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 browser's console window by using the global App Identifier:

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

Last Modified: 11/27/2017 12:24pm

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',
        ...

```

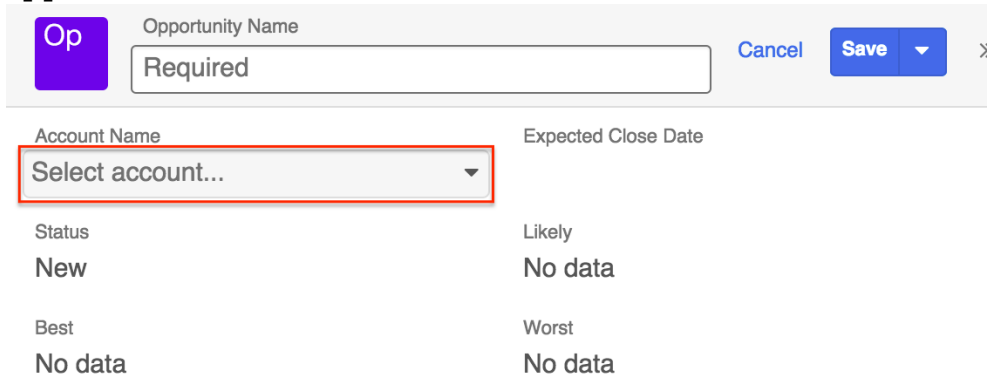
- These changes will only affect the record view for Opportunities. You may need to modify additional module layouts based on your use case. A list of other views you may need to modify for your module are shown below.
 - ./custom/modules/<module>/clients/base/views/list/list.php
 - ./custom/modules/<module>/clients/base/views/record/record.php
 - ./custom/modules/<module>/clients/base/views/dupecheck-list/dupecheck-list.php
 - ./custom/modules/<module>/clients/base/views/resolve-conflicts-list/resolve-conflicts-list.php
 - ./custom/modules/<module>/clients/base/views/selection-list/selection-list.php
 - ./custom/modules/<module>/clients/base/views/subpanel-list/subpanel-list.php

Once completed, you will need to navigate to Admin > Repairs and run a Quick Repair & Rebuild

Application

Now that the appropriate changes have been made, navigate to the Opportunities module and create a new opportunity record. You will notice that the Account

Name field no longer indicates "Required" in the field. In addition, the Account Name field will no longer be marked as required when importing records into the Opportunities module.



Opportunity Name: Required

Account Name: Select account...

Status: New

Expected Close Date: No data

Likely: No data

Worst: No data

Last Modified: 11/27/2017 12:24pm

Adding Buttons to the Record View

Overview

This example explains how to create additional buttons on the record view and add events. We will extend and override the stock Accounts record view to add a custom button. The custom button will be called "Validate Postal Code" and ping the Zippopotamus REST service to validate the records billing state and postal code.

Steps To Complete

This tutorial requires the following steps, which are explained in the sections below:

1. Defining the Metadata
2. Adding Custom Buttons
3. Defining the Button Label
4. Extending and Overriding the Controller

Defining the Metadata

To add a button to the record view, you will first need to create the custom metadata for the view if it doesn't exist. You can easily do this by opening and

saving your modules record layout in studio. Depending on your module, the path will then be `./custom/modules/<module>/clients/base/views/record/record.php`. Once your file is in place, you will need to copy the button array from `./clients/base/views/record/record.php` and add it to the `$viewdefs['<module>']['base']['view']['record']` portion of your metadata array. An example of the button array is shown below:

```
'buttons' => array(
    array(
        'type' => 'button',
        'name' => 'cancel_button',
        'label' => 'LBL_CANCEL_BUTTON_LABEL',
        'css_class' => 'btn-invisible btn-link',
        'showOn' => 'edit',
    ),
    array(
        'type' => 'rowaction',
        'event' => 'button:save_button:click',
        'name' => 'save_button',
        'label' => 'LBL_SAVE_BUTTON_LABEL',
        'css_class' => 'btn btn-primary',
        'showOn' => 'edit',
        'acl_action' => 'edit',
    ),
    array(
        'type' => 'actiondropdown',
        'name' => 'main_dropdown',
        'primary' => true,
        'showOn' => 'view',
        'buttons' => array(
            array(
                'type' => 'rowaction',
                'event' => 'button:edit_button:click',
                'name' => 'edit_button',
                'label' => 'LBL_EDIT_BUTTON_LABEL',
                'acl_action' => 'edit',
            ),
            array(
                'type' => 'shareaction',
                'name' => 'share',
                'label' => 'LBL_RECORD_SHARE_BUTTON',
                'acl_action' => 'view',
            ),
            array(
                'type' => 'pdfaction',
                'name' => 'download-pdf',
```

```

        'label' => 'LBL_PDF_VIEW',
        'action' => 'download',
        'acl_action' => 'view',
    ),
    array(
        'type' => 'pdfaction',
        'name' => 'email-pdf',
        'label' => 'LBL_PDF_EMAIL',
        'action' => 'email',
        'acl_action' => 'view',
    ),
    array(
        'type' => 'divider',
    ),
    array(
        'type' => 'rowaction',
        'event' => 'button:find_duplicates_button:click',
        'name' => 'find_duplicates_button',
        'label' => 'LBL_DUP_MERGE',
        'acl_action' => 'edit',
    ),
    array(
        'type' => 'rowaction',
        'event' => 'button:duplicate_button:click',
        'name' => 'duplicate_button',
        'label' => 'LBL_DUPLICATE_BUTTON_LABEL',
        'acl_module' => $module,
    ),
    array(
        'type' => 'rowaction',
        'event' => 'button:audit_button:click',
        'name' => 'audit_button',
        'label' => 'LNK_VIEW_CHANGE_LOG',
        'acl_action' => 'view',
    ),
    array(
        'type' => 'divider',
    ),
    array(
        'type' => 'rowaction',
        'event' => 'button:delete_button:click',
        'name' => 'delete_button',
        'label' => 'LBL_DELETE_BUTTON_LABEL',
        'acl_action' => 'delete',
    ),
),
),

```



```

    ),
    array(
        'name' => 'sidebar_toggle',
        'type' => 'sidebartoggle',
    ),
),

```

Note: When copying this array into your metadata, you will need to replace `$module` with the text string of your module's name.

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

Property	Potential Values	Description
type	button, rowaction, shareaction, actiondropdown	The widget type
name		The name of the button
label		The label string key for the display text of the button
css_class		The CSS class to append to the button
showOn	edit, view	The ACL action of the button

For this example, we will add the custom button to the main dropdown. For actiondropdown types, there is an additional buttons array for you to specify the dropdown list of buttons. The button definitions in this array will contain the following properties:

Property	Potential Values	Description
type	button, rowaction, shareaction, actiondropdown	The widget type; Most custom buttons are 'rowaction'
event	button:button_name:click	The event name of the button
name		The name of the button
label		The label string key for the display text of the button
acl_action	edit, view	The ACL action of the button

Adding Custom Buttons

For this example, modify the accounts' metadata to add the button definition to main_dropdown:

```
array(  
    'type' => 'rowaction',  
    'event' => 'button:validate_postal_code:click',  
    'name' => 'validate_postal_code',  
    'label' => 'LBL_VALIDATE_POSTAL_CODE',  
    'acl_action' => 'view',  
)
```

A full example is shown below:

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

```
<?php  
  
$viewdefs['Accounts'] =  
array (  
    'base' =>  
    array (  
        'view' =>  
        array (  
            'record' =>  
            array (  
                'buttons' =>  
                array (  
                    0 =>  
                    array (  
                        'type' => 'button',  
                        'name' => 'cancel_button',  
                        'label' => 'LBL_CANCEL_BUTTON_LABEL',  
                        'css_class' => 'btn-invisible btn-link',  
                        'showOn' => 'edit',  
                    ),  
                ),  
                1 =>  
                array (  
                    'type' => 'rowaction',  
                    'event' => 'button:save_button:click',  
                    'name' => 'save_button',  
                    'label' => 'LBL_SAVE_BUTTON_LABEL',  
                    'css_class' => 'btn btn-primary',
```

```

        'showOn' => 'edit',
        'acl_action' => 'edit',
    ),
    2 =>
    array (
        'type' => 'actiondropdown',
        'name' => 'main_dropdown',
        'primary' => true,
        'showOn' => 'view',
        'buttons' =>
        array (
            0 =>
            array (
                'type' => 'rowaction',
                'event' => 'button:edit_button:click',
                'name' => 'edit_button',
                'label' => 'LBL_EDIT_BUTTON_LABEL',
                'acl_action' => 'edit',
            ),
            1 =>
            array (
                'type' => 'shareaction',
                'name' => 'share',
                'label' => 'LBL_RECORD_SHARE_BUTTON',
                'acl_action' => 'view',
            ),
            2 =>
            array (
                'type' => 'rowaction',
                'event' => 'button:validate_postal_code:click',
                'name' => 'validate_postal_code',
                'label' => 'LBL_VALIDATE_POSTAL_CODE',
                'acl_action' => 'view',
            ),
            3 =>
            array (
                'type' => 'divider',
            ),
            4 =>
            array (
                'type' => 'rowaction',
                'event' => 'button:duplicate_button:click',
                'name' => 'duplicate_button',
                'label' => 'LBL_DUPLICATE_BUTTON_LABEL',
                'acl_module' => 'Accounts',
            ),
        ),
    ),

```

```

5 =>
array (
    'type' => 'rowaction',
    'event' => 'button:audit_button:click',
    'name' => 'audit_button',
    'label' => 'LNK_VIEW_CHANGE_LOG',
    'acl_action' => 'view',
),
6 =>
array (
    'type' => 'divider',
),
7 =>
array (
    'type' => 'rowaction',
    'event' => 'button:delete_button:click',
    'name' => 'delete_button',
    'label' => 'LBL_DELETE_BUTTON_LABEL',
    'acl_action' => 'delete',
),
),
),
3 =>
array (
    'name' => 'sidebar_toggle',
    'type' => 'sidebartoggle',
),
),
'panels' =>
array (
    0 =>
array (
    'name' => 'panel_header',
    'header' => true,
    'fields' =>
array (
    0 =>
array (
        'name' => 'picture',
        'type' => 'avatar',
        'width' => 42,
        'height' => 42,
        'dismiss_label' => true,
        'readonly' => true,
    ),
    1 => 'name',

```

```
2 =>
array (
    'name' => 'favorite',
    'label' => 'LBL_FAVORITE',
    'type' => 'favorite',
    'dismiss_label' => true,
),
3 =>
array (
    'name' => 'follow',
    'label' => 'LBL_FOLLOW',
    'type' => 'follow',
    'readonly' => true,
    'dismiss_label' => true,
),
),
1 =>
array (
    'name' => 'panel_body',
    'columns' => 2,
    'labelsOnTop' => true,
    'placeholders' => true,
    'fields' =>
array (
    0 => 'website',
    1 => 'industry',
    2 => 'parent_name',
    3 => 'account_type',
    4 => 'assigned_user_name',
    5 => 'phone_office',
),
),
2 =>
array (
    'name' => 'panel_hidden',
    'hide' => true,
    'columns' => 2,
    'labelsOnTop' => true,
    'placeholders' => true,
    'fields' =>
array (
    0 =>
array (
        'name' => 'fieldset_address',
        'type' => 'fieldset',
```

```
'css_class' => 'address',
'label' => 'LBL_BILLING_ADDRESS',
'fields' =>
array (
  0 =>
  array (
    'name' => 'billing_address_street',
    'css_class' => 'address_street',
    'placeholder' => 'LBL_BILLING_ADDRESS_STREET',
  ),
  1 =>
  array (
    'name' => 'billing_address_city',
    'css_class' => 'address_city',
    'placeholder' => 'LBL_BILLING_ADDRESS_CITY',
  ),
  2 =>
  array (
    'name' => 'billing_address_state',
    'css_class' => 'address_state',
    'placeholder' => 'LBL_BILLING_ADDRESS_STATE',
  ),
  3 =>
  array (
    'name' => 'billing_address_postalcode',
    'css_class' => 'address_zip',
    'placeholder' => 'LBL_BILLING_ADDRESS_POSTALCODE',
  ),
  4 =>
  array (
    'name' => 'billing_address_country',
    'css_class' => 'address_country',
    'placeholder' => 'LBL_BILLING_ADDRESS_COUNTRY',
  ),
),
),
1 =>
array (
  'name' => 'fieldset_shipping_address',
  'type' => 'fieldset',
  'css_class' => 'address',
  'label' => 'LBL_SHIPPING_ADDRESS',
  'fields' =>
  array (
    0 =>
    array (
```

```

        'name' => 'shipping_address_street',
        'css_class' => 'address_street',
        'placeholder' => 'LBL_SHIPPING_ADDRESS_STREET',
    ),
    1 =>
    array (
        'name' => 'shipping_address_city',
        'css_class' => 'address_city',
        'placeholder' => 'LBL_SHIPPING_ADDRESS_CITY',
    ),
    2 =>
    array (
        'name' => 'shipping_address_state',
        'css_class' => 'address_state',
        'placeholder' => 'LBL_SHIPPING_ADDRESS_STATE',
    ),
    3 =>
    array (
        'name' => 'shipping_address_postalcode',
        'css_class' => 'address_zip',
        'placeholder' =>
'LBL_SHIPPING_ADDRESS_POSTALCODE',
    ),
    4 =>
    array (
        'name' => 'shipping_address_country',
        'css_class' => 'address_country',
        'placeholder' => 'LBL_SHIPPING_ADDRESS_COUNTRY',
    ),
    5 =>
    array (
        'name' => 'copy',
        'label' => 'NTC_COPY_BILLING_ADDRESS',
        'type' => 'copy',
        'mapping' =>
        array (
            'billing_address_street' =>
'shipping_address_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, define the label for the button:

```
./custom/Extension/modules/Accounts/Ext/Language/en_us.validatePostalCode.php
```

```
<?php
```

```
$mod_strings['LBL_VALIDATE_POSTAL_CODE'] = 'Validate Postal Code';
```

Extending and Overriding the Controller

Once the button has been added to the metadata, extend and override the record view controller:

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

```
( {  
  
  extendsFrom: 'RecordView',  
  
  zipJSON: {},  
  
  initialize: function (options) {  
    this._super('initialize', [options]);  
  
    //add listener for custom button  
    this.context.on('button:validate_postal_code:click',  
this.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,
```


When throwing a validation error, Sugar has several stock messages you may choose to use. They are shown below:

Error Key	Label	Description
maxValue	ERROR_MAXVALUE	This maximum value of this field
minValue	ERROR_MINVALUE	This minimum value of this field
maxLength	ERROR_MAX_FIELD_LENGTH	The max length of this field
minLength	ERROR_MIN_FIELD_LENGTH	The min length of this field
datetime	ERROR_DATETIME	This field requires a valid date
required	ERROR_FIELD_REQUIRED	This field is required
email	ERROR_EMAIL	There is an invalid email address
primaryEmail	ERROR_PRIMARY_EMAIL	No primary email address is set
duplicateEmail	ERROR_DUPLICATE_EMAIL	There is a duplicate email address
number	ERROR_NUMBER	This field requires a valid number
isBefore	ERROR_IS_BEFORE	The date of this field can not be after another date
isAfter	ERROR_IS_AFTER	The date of this field can not be before another date

You also have the option of displaying multiple error messages at a time. The example below would throw an error message notifying the user that the selected field is required and that it is also not a number.

```
errors['<field name>'] = errors['<field name>'] || {};
errors['<field name>'].required = true;
errors['<field name>'].number = true;
```

Custom Error Messages

Custom error message can be used by appending custom language keys to

`app.error.errorName2Keys` when initializing an extended controller. To accomplish this, create a new language key in the `$app_strings`. An example of this is shown below:

```
./custom/Extension/application/Ext/Language/en_us.error_custom_message.php
```

```
<?php
```

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

Next, you will need to update your controller to use the new language key. To accomplish this, add your custom language key to the `app.error.errorName2Keys` array in the `initialize` method:

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

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

    ....
},
```

Once completed, you can call your custom message by setting the `app.error.errorName2Keys` entry to `true` as shown below:

```
errors['phone_office'] = errors['phone_office'] || {};
errors['phone_office'].custom_message = true;
```

Method 1: Extending the RecordView and CreateView Controllers

One way to validate fields on record view is by creating record and create view controllers. This method requires a duplication of code due to the hierarchy design, however, it does organize the code by module and extend the core functionality. To accomplish this, override and extend the create view controller. This handles the validation when a user is creating a new record. Once the controller has been properly extended, define the validation check and use the `model.addValidationTask` method to append your function to the save validation.

```
./custom/modules/Accounts/clients/base/views/create/create.js
```

```

({
  extendsFrom: 'CreateView',
  initialize: function (options) {

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

    //add validation tasks
    this.model.addValidationTask('check_account_type',
    _.bind(this._doValidateCheckType, this));
    this.model.addValidationTask('check_email',
    _.bind(this._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, duplicate the validation logic for the record view. This handles the validation when editing existing records.

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

```

({
  /* because 'accounts' already has a record view, we need to extend
  it */
  extendsFrom: 'AccountsRecordView',

```

```

initialize: function (options) {

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

    //add validation tasks
    this.model.addValidationTask('check_account_type',
    _.bind(this._doValidateCheckType, this));
    this.model.addValidationTask('check_email',
    _.bind(this._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, navigate to Admin > Repair > Quick Repair and Rebuild. More information on displaying custom error messages can be found in the Error Messages section.

Method 2: Overriding the RecordView and CreateView Layouts

Another method for defining your own custom validation is to override a module's record and create layouts to append a new view with your logic. The benefits of this method are that you can use the single view to house the validation logic, however, this means that you will have to override the layout. When overriding a layout, verify that the layout has not changed when upgrading your instance. Once the layouts are overridden, define the validation check and use the `model.addValidationTask` method to append the function to the save validation.

First, create the custom view. For the accounts example, create the view `validate-account`:

```
./custom/modules/Accounts/clients/base/views/validate-account/validate-account.js
```

```
({
  className: "hidden",

  _render: function() {
    //No-op, Do nothing here
  },

  bindDataChange: function() {
    //add validation tasks
    this.model.addValidationTask('check_account_type',
    _.bind(this._doValidateCheckType, this));
    this.model.addValidationTask('check_email',
    _.bind(this._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);
},
}))

```

More information on displaying custom error messages can be found in the Error Messages section. Next, depending on the selected module, duplicate its create layout to the modules custom folder to handle record creation. In our Accounts example, we have an existing `./modules/Accounts/clients/base/layouts/create/create.php` file so we need to duplicate this file to `./custom/modules/Accounts/clients/base/layouts/create/create.php`. After this has been completed, append the new custom view to the components. append:

```

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

```

As shown below:

`./custom/modules/Accounts/clients/base/layouts/create/create.php`

```
<?php
```

```

$viewdefs['Accounts']['base']['layout']['create'] = array(
    'components' =>
        array(
            array(
                'layout' =>
                    array(
                        'components' =>
                            array(
                                array(
                                    'layout' =>
                                        array(
                                            'components' =>
                                                array(
                                                    array(
                                                        'view' =>
'create',
                                                    ),
                                                array(
'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',
                                'width' => 12,
                                'label' => 'DNB Account Create',
                            ),
                        'type' => 'simple',
                        'name' =>
'dashboard-pane',
                        'span' => 4,
                    ),
                ),
            '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 the selected module, duplicate its record layout to the modules custom folder to handle editing a record. In the accounts example, we do not have an existing `./modules/Accounts/clients/base/layouts/record/record.php` file so we duplicated the core `./clients/base/layouts/record/record.php` to `./custom/modules/Accounts/clients/base/layouts/record/record.php`. Since we are copying from the `./clients/` core directory, modify:

```

$viewdefs['base']['layout']['record'] = array(
    ...
);

```

To:

```

$viewdefs['Accounts']['base']['layout']['record'] = array(
    ...
);

```

After this has been completed, append the new custom view to the components:

```

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

```

The resulting file is shown below:

`./custom/modules/Accounts/clients/base/layouts/record/record.php`

```

<?php

$viewdefs['Accounts']['base']['layout']['record'] = array(
    'components' => array(
        array(
            'layout' => array(
                'components' => array(
                    array(
                        'layout' => array(
                            'components' => array(
                                array(
                                    'view' => 'record',
                                    'primary' => true,
                                ),
                                array(
                                    'view' => 'validate-account',
                                ),
                                array(
                                    'layout' => 'extra-info',
                                ),
                                array(
                                    'layout' => array(
                                        'name' => 'filterpanel',
                                        'span' => 12,
                                        'last_state' => array(
                                            'id' =>
'record-filterpanel',
                                            'defaults' => array(
                                                'toggle-view' =>
'subpanels',
                                            ),
                                        ),
                                        'availableToggles' => array(
                                            array(
                                                'name' => 'subpanels',
                                                'icon' =>
'icon-table',
                                                'label' =>
'LBL_DATA_VIEW',
                                            ),
                                            array(
                                                'name' => 'list',
                                                'icon' =>
'icon-table',
                                                'label' =>
'LBL_LISTVIEW',
                                            )
                                        )
                                    )
                                )
                            )
                        )
                    )
                )
            )
        )
    )
)

```

```

        ),
        'type' => 'simple',
        'name' => 'main-pane',
        'span' => 8,
    ),
),
array(
    'layout' => array(
        'components' => array(
            array(
                'layout' => 'sidebar',
            ),
        ),
        'type' => 'simple',
        'name' => 'side-pane',
        'span' => 4,
    ),
),
array(
    'layout' => array(
        'components' => array(
            array(
                'layout' => array(
                    'type' => 'dashboard',
                    'last_state' => array(
                        'id' => 'last-visit',
                    )
                ),
            ),
            'context' => array(
                'forceNew' => true,
                'module' => 'Home',
            ),
        ),
        'type' => 'simple',
        'name' => 'dashboard-pane',
        'span' => 4,
    ),
),
array(
    'layout' => array(
        'components' => array(
            array(
                'layout' => 'preview',
            ),
        ),
    ),
),

```

```
        'type' => 'simple',
        'name' => 'preview-pane',
        'span' => 8,
    ),
),
),
    'type' => 'default',
    'name' => 'sidebar',
    'span' => 12,
),
),
),
    'type' => 'simple',
    'name' => 'base',
    'span' => 12,
);
```

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

Last Modified: 04/25/2018 04:09am

Passing Data to Templates

Overview

This page explains how to create a custom view component that passes data to the Handlebars template.

Steps to Complete

The view component will render the actual content on the page. The view below will display data passed to it from the controller.

```
./custom/clients/base/views/my-dataview/my-dataview.js
```

```
({
  className: 'tcenter',
  loadData: function (options) {
    //populate your data
    myData = {};
    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);
        */

        //reset flags on reload
        if (options && __.isFunction(options.complete)) {
            options.complete();
        }
        this.render();
    }
})
```

Next, add the 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, add the view to a layout and then navigate to Admin > Repair > Quick Repair and Rebuild.

Last Modified: 05/31/2017 06:34pm

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'][$outerKey]['buttons'] as $innerKey => $innerButton)
    {
        if (
            isset($innerButton['name'])
            && $innerButton['name'] == 'button_name'
        )
        {

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

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

$viewdefs[$module]['base']['view']['record']['buttons'][$outerKey]['buttons'][$innerKey]=$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: 10/14/2016 04:12pm

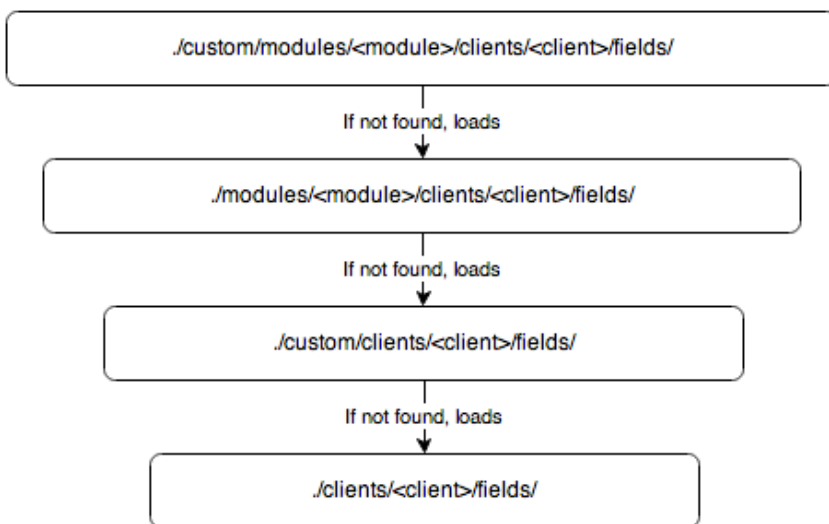
Fields

Overview

Fields are component plugins that render and format field values. They are made up of a controller JavaScript file (.js) and at least one Handlebars template (.hbt). For more information regarding the data handling of a field, please refer the data framework fields documentation. For information on creating custom field types, please refer the Creating Custom Fields documentation.

Hierarchy Diagram

The field components are loaded in the following manner:



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

Field Example

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

Controller

The `bool.js`, file shown below, overrides the base `_render` function 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
<code>_render</code>	Function to render the field.
<code>unformat</code>	Function to dynamically check the checkbox based on the value.
<code>format</code>	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: 10/14/2016 04:12pm

Creating Custom Field Types

Overview

In this example, we create a custom field type called "Highlightfield", which will mimic the base text field type with the added feature that the displayed text for the field will be highlighted in a color chosen when the field is created in Studio.

This example requires the following steps, which are covered in the sections and subsections below:

1. Creating the JavaScript Controller
2. Defining the Handlebar Templates
3. Adding the Field Type to Studio
4. Enabling Search and Filtering

Naming a Custom Field Type in Sugar

When choosing a name for a custom field type, keep in mind the following rules:

- The first letter of a custom field type's name must be capitalized.
- All subsequent letters in the field type's name must be lowercase.
- A field type's name cannot contain non-letter characters such as 0-9, hyphens, or dashes.

Therefore, in this example, the field type "Highlightfield" cannot be called HighLightField or highlightfield.

Creating the JavaScript Controller

First, create a controller file. Since we are starting from scratch, we need to extend the base field template. To accomplish this, create `./custom/clients/base/fields/Highlightfield/Highlightfield.js`. This file will contain the JavaScript needed to render the field and format the values. By default, all fields extend the base template and do not require you to add the `extendsFrom` property. An example template is shown below:

```
./custom/clients/base/fields/Highlightfield/Highlightfield.js
```

```
({  
  /**  
   * Called when initializing the field  
   * @param options  
   */  
  initialize: function(options) {  
    this._super('initialize', [options]);  
  },  
  /**
```

```

    * Called when rendering the field
    * @private
    */
  _render: function() {
    this._super('_render');
  },

  /**
   * Called when formatting the value for display
   * @param value
   */
  format: function(value) {
    return this._super('format', [value]);
  },

  /**
   * Called when unformatting the value for storage
   * @param value
   */
  unformat: function(value) {
    return this._super('unformat', [value]);
  }
}
}))

```

Note: This controller file example contains methods for initialize, `_render`, `format`, and `unformat`. These are shown for your ease of use but are not necessarily needed for this customization. For example, you could choose to create an empty controller consisting of nothing other than `{}` and the field would render as expected in this example.

Defining the Handlebar Templates

Next, define the handlebar templates. The templates will nearly match the base template found in `./clients/base/fields/base/` with the minor difference of an additional attribute of `style="background:{{def.backcolor}}; color:{{def.textcolor}}"` for the detail and list templates.

Detail View

The first template to define is the Sidecar detail view. This template handles the display of data on the record view.

```
./custom/clients/base/fields/Highlightfield/detail.hbs
```

```
{{!
```

The data for field colors are passed into the handlebars template through the def array. For this example, the def.backcolor and def.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}}
```

Edit View

Now define the Sidecar edit view. This template handles the display of the field in the edit view.

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

```
{{!
```

We have not made any edits to this file that differ from stock, 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">
```

List View

Finally, define the list view. This template handles the display of the custom field in list views.

```
./custom/clients/base/fields/Highlightfield/list.hbs
```

```
{{!  
    The data for field colors are passed into the handlebars template  
    through the def array. Our case  
    being the def.backcolor and def.textcolor properties. These  
    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>
```

Adding the Field Type to Studio

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

```
./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 the custom field type, the ext1, ext2, ext3, and ext4 database fields in

the `fields_meta_data` table offer additional property storage.

Creating the Form Controller

Next, set up the field's form controller. This controller will handle the field's form template in Admin > Studio > Fields and allow us to assign color values to the Smarty template.

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

```
<?php
```

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

```
require_once
```

```
'custom/modules/DynamicFields/templates/Fields/TemplateHighlightfield.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');
}
?>

```

Creating the Smarty Template

Once the form controller is in place, create the Smarty template. The .tpl below will define the center content of the field's Studio edit view. The template includes a coreTop.tpl and coreBottom.tpl file. These files add the base field properties such as "Name" and "Display Label" that are common across all field types. This allows us to focus on the fields that are specific to our new field type.

`./custom/modules/DynamicFields/templates/Fields/Forms/Highlightfield.tpl`

```

{include
file="modules/DynamicFields/templates/Fields/Forms/coreTop.tpl"}
<tr>
    <td class='mbLBL'>{sugar_translate module="DynamicFields"
label="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.getElementById(
'default'),this.value);">
            <input type='hidden' id="orig_len" name='orig_len'
value='{$vardef.len}'>
                {if $action=="saveSugarField"}
                    <input type='hidden' name='customTypeValidate'
id='customTypeValidate' value='{$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=' {$svardef.len}'>{$svardef.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"}

```

Registering the Field Type

Once you have defined the Studio template, register the field as a valid field type. For this example, the field will inherit the base field type as it is a text field. In doing this, we are able to override the core functions. The most used override is the save function shown below.

```
./custom/include/SugarFields/Fields/Highlightfield/SugarFieldHighlightfield.php
```

```
<?php

require_once 'include/SugarFields/Fields/Base/SugarFieldBase.php';
require_once 'data/SugarBean.php';

class SugarFieldHighlightfield extends SugarFieldBase
{
    //this function is called to format the field before saving. For
    //example we could put code in here
    // to check spelling or to change the case of all the letters
    public function save(&$bean, $params, $field, $properties, $prefix
= '')
    {
        $GLOBALS['log']->debug("SugarFieldHighlightfield::save()
function called.");
        parent::save($bean, $params, $field, $properties, $prefix);
    }
}
?>
```

Creating Language Definitions

For the new field, you must define several language extensions to ensure everything is displayed correctly. This section includes three steps:

- Adding the Custom Field to the Type List
- Creating Labels for Studio
- Defining Dropdown Controls

Adding the Custom Field to the Type List

Define the new field type in the field types list. This will allow an Administrator to select the Highlightfield field type in Studio when creating a new field.

```
./custom/Extension/modules/ModuleBuilder/Ext/Language/en_us.Highlightfield.php
```

```
<?php

$mod_strings['fieldTypes']['Highlightfield'] = 'Highlighted Text';
```

Creating Labels for Studio

Once the field type is defined, add the labels for the Studio template.

```
./custom/Extension/modules/DynamicFields/Ext/Language/en_us.Highlightfield.php
```

```
<?php

$mod_strings['LBL_HIGHLIGHTFIELD'] = 'Highlighted Text';
$mod_strings['LBL_HIGHLIGHTFIELD_FORMAT_HELP'] = '';

$mod_strings['LBL_HIGHLIGHTFIELD_BACKCOLOR'] = 'Background Color';
$mod_strings['LBL_HIGHLIGHTFIELD_TEXTCOLOR'] = 'Text Color';
```

Defining Dropdown Controls

For this example, we must define dropdown lists, which will be available in Studio for an administrator to add or remove color options for the field type.

```
./custom/Extension/application/Ext/Language/en_us.Highlightfield.php
```

```
<?php

$app_strings['LBL_HIGHLIGHTFIELD_OPERATOR_CONTAINS'] = 'contains';
$app_strings['LBL_HIGHLIGHTFIELD_OPERATOR_NOT_CONTAINS'] = 'does not
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'  
);
```

Enabling Search and Filtering

To enable Highlightfield for searching and filtering, define the filter operators and the Sugar widget.

Defining the Filter Operators

The filter operators, defined in `./custom/clients/base/filters/operators/operators.php`, allow the custom field be used for searching in Sidecar listviews.

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

```
<?php  
  
require 'clients/base/filters/operators/operators.php';  
  
$viewdefs['base']['filter']['operators']['Highlightfield'] = array(  
    '$contains' => 'LBL_HIGHLIGHTFIELD_OPERATOR_CONTAINS',  
    '$not_contains' => 'LBL_HIGHLIGHTFIELD_OPERATOR_NOT_CONTAINS',  
    '$starts' => 'LBL_HIGHLIGHTFIELD_OPERATOR_STARTS_WITH',  
);
```

Note: The labels for the filters in this example are defined in `./custom/Extension/application/Ext/Language/en_us.Highlightfield.php`. For the full list of filters in the system, you can look at `./clients/base/filters/operators/operators.php`.

Defining the Sugar Widget

Finally, define the Sugar widget. The widget will help our field for display on Reports and subpanels in backward compatibility. It also controls how search filters are applied to our field.

```
./custom/include/generic/SugarWidgets/SugarWidgetFieldHighlightfield.php
```

```
<?php  
  
if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry
```

Point');

require_once

'include/generic/SugarWidgets/SugarWidgetFieldvarchar.php';

```
class SugarWidgetFieldHighlightfield extends SugarWidgetFieldVarchar
{
    function SugarWidgetFieldText(&$layout_manager)
    {
        parent::SugarWidgetFieldVarchar($layout_manager);
    }

    function queryFilterEquals($layout_def)
    {
        return
        $this->reporter->db->convert($this->_get_column_select($layout_def),
        "text2char") .
        " = " .
        $this->reporter->db->quoted($layout_def['input_name0']);
    }

    function queryFilterNot_Equals_Str($layout_def)
    {
        $column = $this->_get_column_select($layout_def);
        return "($column IS NULL OR " .
        $this->reporter->db->convert($column, "text2char") . " != " .
        $this->reporter->db->quoted($layout_def['input_name0']) . ")";
    }

    function queryFilterNot_Empty($layout_def)
    {
        $column = $this->_get_column_select($layout_def);
        return "($column IS NOT NULL AND " .
        $this->reporter->db->convert($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, navigate to Admin > Repair > Quick Repair and Rebuild. It is also best practice to clear your browser cache before using the new field type in Studio.

Last Modified: 04/25/2018 04:09am

Converting Address' Country Field to a Dropdown

Overview

Address fields in Sugar are normally text fields, which allow users to enter in the appropriate information (e.g. street, city, and country) for the record. However, with multiple users working in Sugar, it is possible for data (e.g. country) to be entered in a variety of different ways (e.g. USA, U.S.A, and United States) when creating or editing the record. This can cause some issues when creating a report grouped by the Billing Country field, for example, as records with the same country will be grouped separately based on the different ways the country was entered.

This article will cover how to change the address' Billing Country field to a dropdown list which will allow users to select a single value (e.g. USA) and maintain consistency in data throughout the system.

Note: This article pertains to Sugar versions 6.x and 7.x.

Use Case

In this example, we will convert the Billing Country field in the Accounts module to a dropdown-type field to allow values to be selected from a dropdown list.

Prerequisites

A part of making this change involves mapping the "countries_dom" dropdown list to your existing Billing Country field's values. This dropdown list can be accessed and modified via Admin > Dropdown Editor. For more information on editing dropdown lists, please refer to the Developer Tools documentation. It is very

important that the existing values in the Billing Country field exactly match the Item Name values in the "countries_dom" dropdown list in order for the values to convert properly. If the existing value (e.g. United States) does not match one of the country options (e.g. USA) in the dropdown list, then the new dropdown version of the Billing Country field will most likely default to a blank value for that record record.

To avoid such issues, please review and update all of your existing Billing Country text field values prior to making this change. For more information on updating many records at once via import, please refer to the Updating Records Via Import article.

Steps to Complete

Converting Field to Dropdown

Use the following steps to change the Billing Country field to a dropdown list:

1. First, we will associate the Billing Country field in the Accounts module with the "countries_dom" dropdown. Locate the following directory in your Sugar file system: `./custom/Extension/modules/Accounts/Ext/Vardefs/`
2. Locate a file called `sugarfield_billing_address_country.php` which controls the `billing_address_country` field and add the following lines to the file. This will set the field type to 'enum' and define the dropdown (`countries_dom`) to use for the field.

Note: If this file or location does not exist, then you will need to first create this path and file.

Your file should look like this:

3. `<?php`

```
$dictionary['Account']['fields']['billing_address_country']['type']
='enum';
$dictionary['Account']['fields']['billing_address_country']['optio
ns']='countries_dom';
```

4. Now, since the address block is handled uniquely in Sugar, we will also need to modify the template file in order to display the field as a dropdown anywhere the layout is being used in backward compatibility mode. To do this in an upgrade safe way, you will need to copy the file located in `./include/SugarFields/Fields/Address/en_us.EditView.tpl` and place it in the following directory: `./custom/include/SugarFields/Fields/Address/`.

If you are using multiple languages in Sugar, you will need to make this change for each language-type in your system. For our example, we will be focusing on the "en_us" language.

Note: You will most likely need to create this directory as it likely does not

already exist.

5. Once the en_us.EditView.tpl file is in the custom directory, locate the line for the input html element for the country field. The line of code should look similar to this:

```
<input type="text" name="{{ $country }}" id="{{ $country }}"
size="{{ $displayParams.size|default:30 }}" {{if
!empty($vardef.len)}}maxlength='{{ $vardef.len }}' {{/if}}
value='{{ $fields.{{ $country }}.value}' tabindex="{{ $tabindex }}">
```

6. Directly between the list shown above and the <td> appearing before it, add the following lines to display the field as a dropdown:

```
{if (!isset($config.enable_autocomplete) ||
$config.enable_autocomplete==false) &&
isset($fields.{{ $country }}.options)}
<select name="{{ $country }}" id="{{ $country }}" title=''>
{if isset($fields.{{ $country }}.value) &&
$fields.{{ $country }}.value != ''}
{html_options options=$fields.{{ $country }}.options
selected=$fields.{{ $country }}.value}
{else}
{html_options options=$fields.{{ $country }}.options
selected=$fields.{{ $country }}.default}
{/if}
</select>
{else}
<input type="text" name="{{ $country }}" id="{{ $country }}"
size="{{ $displayParams.size|default:30 }}" {{if
!empty($vardef.len)}}maxlength='{{ $vardef.len }}' {{/if}}
value='{{ $fields.{{ $country }}.value}' tabindex="{{ $tabindex }}">
{/if}
```

7. Save the changes to the file.

Once the necessary change has been made, please navigate to Admin > Repair and perform a Quick Repair and Rebuild in order for the change to take effect. Your Billing Country field will now display as a dropdown field in the Accounts module.

Please note that only the Account module's Billing Country field will be converted to a dropdown field once this change is applied. If you wish to have the Shipping Country field converted to a dropdown as well, please follow the steps above, but be sure to make the change specific to the "shipping_address_country" field. In addition, you can make this change for other modules (e.g. Contacts and Leads) as well by specifying the appropriate module name (e.g. Contacts) and field name (e.g. Primary Address Country) when going through the steps.

Updating the Field Type for List View Filter

Once the appropriate changes have been made per the section above, use the following steps to get the list view filter to recognize the change:

1. Navigate to Admin > Studio > Accounts > Layouts > Search.
2. Drag the Billing Country field from the Default column to the Hidden column and click "Save & Deploy".

The screenshot shows the 'Search' layout configuration in SugarCRM Studio. The interface is divided into two columns: 'Default' and 'Hidden'. The 'Default' column contains the following fields: Name [name], Type [account_type], Industry [industry], Annual Revenue [annual_revenue], Street [address_street], City [address_city], State [address_state], Postal Code [address_postalcode], and Billing Country [billing_address_country]. The 'Hidden' column contains the following fields: Modified By [modified_by_name], Created By [created_by_name], Description [description], Facebook Account [facebook], Twitter Account [twitter], Google Plus ID [googleplus], Fax [phone_fax], Billing Street 2 [billing_address_street_2], and Billing Street 3 [billing_address_street_3]. A red box highlights the 'Billing Country' field in the Default column, and a red arrow points to the 'Billing Street 2' field in the Hidden column, indicating the drag-and-drop action. The 'Save & Deploy' button is also highlighted with a red box.

3. Now drag the Billing Country field back to the Default column and click "Save & Deploy" again.

This will retrieve the Billing Country field in its new state and add it to the account's list view filter.

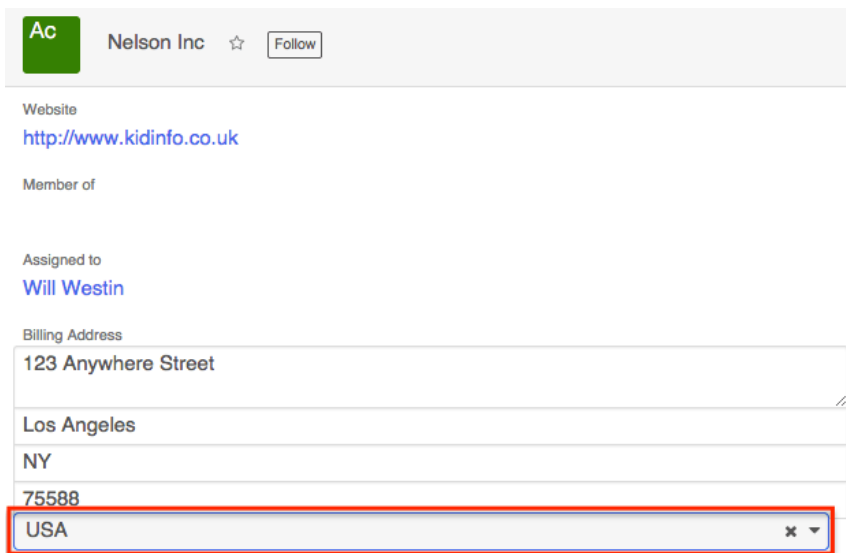
Application

Once the Billing Country field is converted successfully to a dropdown, all records that had an existing value in the field should have the matching country automatically selected. Going forward, users will simply need to select the appropriate country when entering address information in Sugar.

Please note that administrators can add additional country values as necessary to the "countries_dom" dropdown via Admin > Dropdown Editor. For more information on editing dropdown lists, please refer to the Developer Tools documentation.

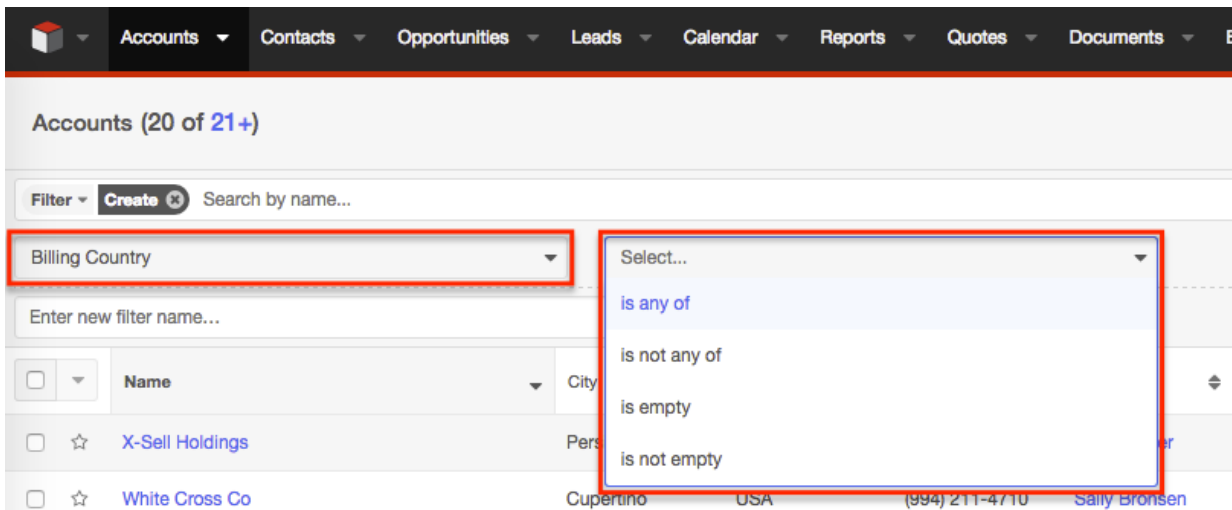
The Billing Country field will now appear as follows:

For Accounts record view:



The screenshot shows the record view for 'Nelson Inc'. The 'Billing Address' section contains the following fields: '123 Anywhere Street', 'Los Angeles', 'NY', '75588', and 'USA'. The 'USA' field is highlighted with a red border.

For Accounts list view filter:



The screenshot shows the Accounts list view filter. The 'Billing Country' dropdown menu is highlighted in red, and its options are visible: 'Select...', 'is any of', 'is not any of', 'is empty', and 'is not empty'. Below the filter, a table of accounts is visible, including 'X-Sell Holdings' and 'White Cross Co'.

Last Modified: 04/07/2017 03:44pm

Subpanels

Overview

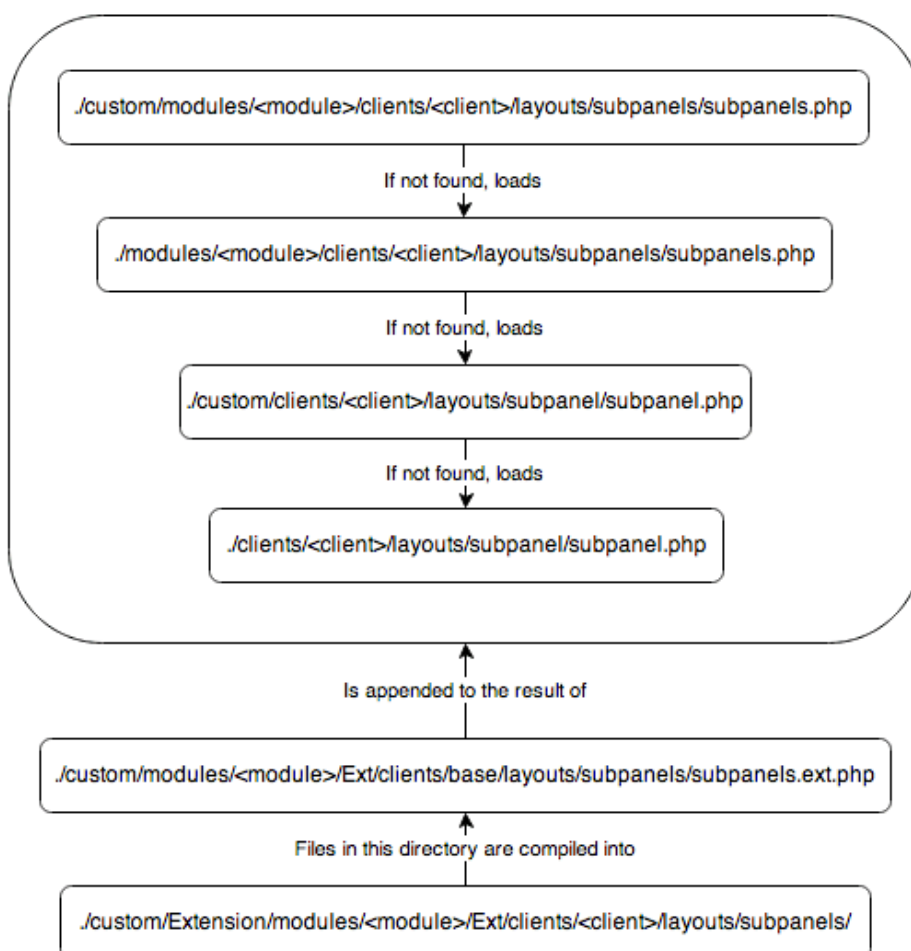
For Sidecar, Sugar's subpanel layouts have been modified to work as simplified metadata. This page is an overview of the metadata framework for subpanels.

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

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

Hierarchy Diagram

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



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

Subpanels and Subpanel Layouts

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

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

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

```
<?php

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

```
);
```

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

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

```
<?php
```

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

Adding Subpanel Layouts

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

Sidecar Layouts

Custom Sidecar layouts, located in `./custom/Extension/modules/<module>/Ext/clients/<client>/layouts/subpanels/`, are compiled into

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

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

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

```
<?php

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

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

```
<?php

$viewdefs['Bugs']['mobile']['layout']['subpanels']['components'][] =
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

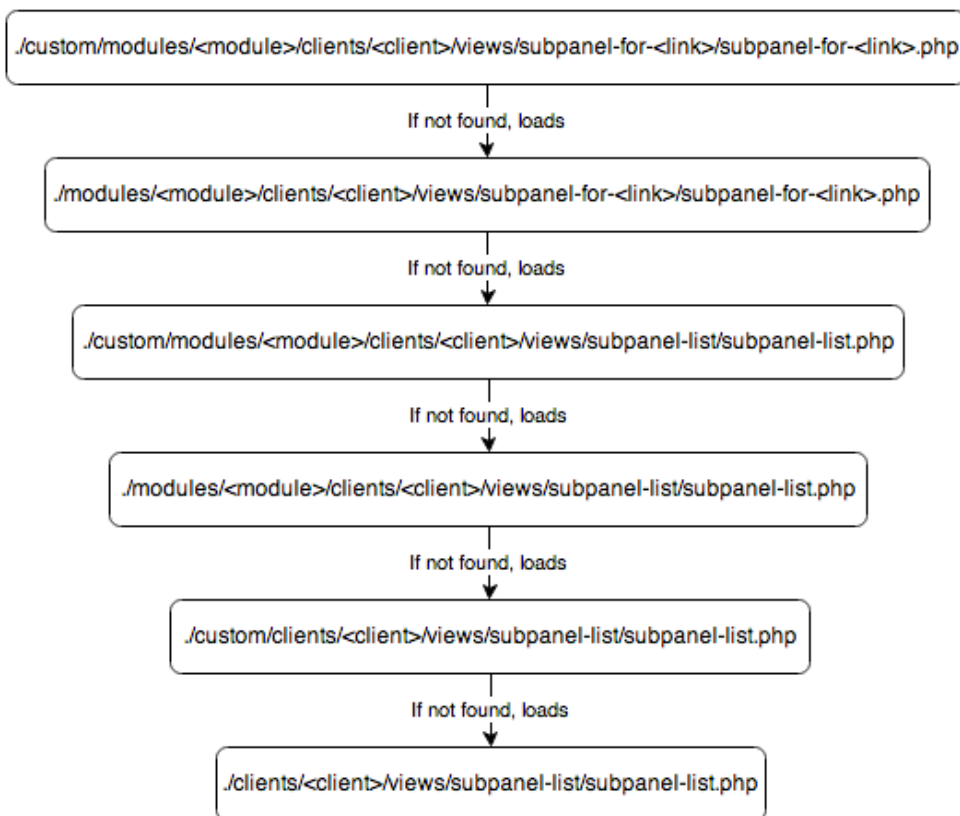
$layout_defs["Bugs"]["subpanel_setup"]['bugs_leads_1'] = array (
    'order' => 100,
    'module' => 'Leads',
    'subpanel_name' => 'default',
    'title_key' => 'LBL_BUGS_LEADS_1_FROM_LEADS_TITLE',
    'get_subpanel_data' => 'bugs_leads_1',
);
```

Fields Metadata

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

Hierarchy Diagram

The subpanel list metadata is loaded in the following manner:



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

Subpanel List Views

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

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

```
<?php
```

```
$subpanel_layout['list_fields'] = array (
```

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

```

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

```

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

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

Last Modified: 11/27/2017 12:24pm

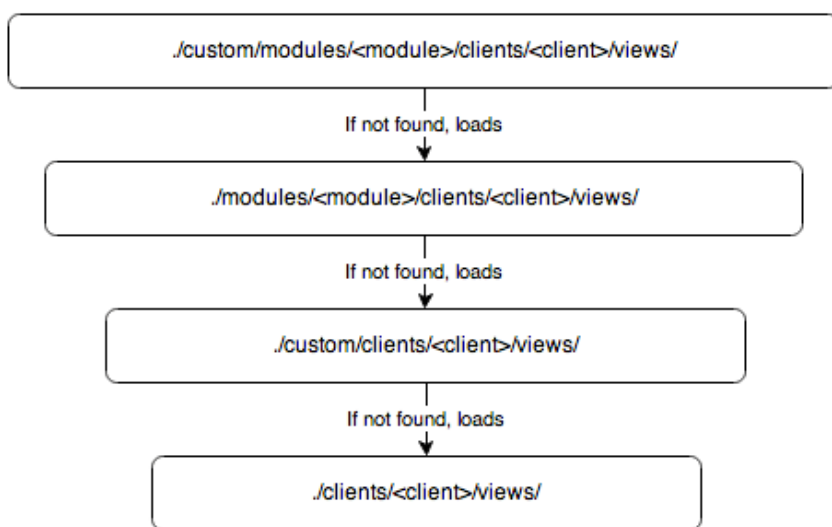
Dashlets

Overview

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

Hierarchy Diagram

Sugar loads the dashlet view components in the following manner:



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

Dashlet Views

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

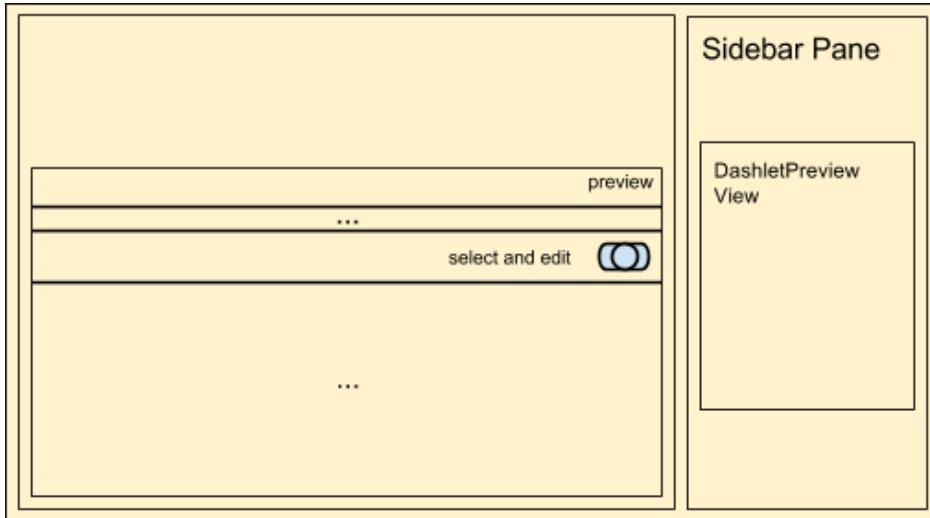
Preview

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

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

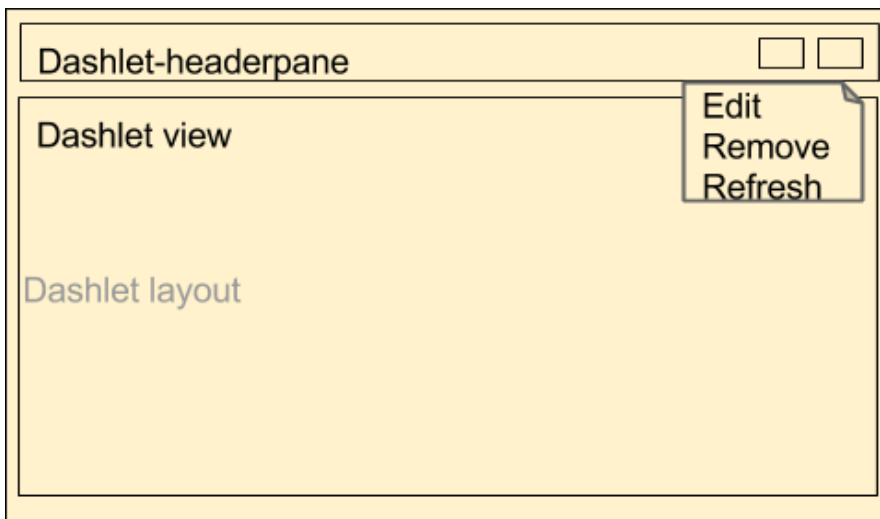
The values in the preview metadata can be retrieved using:

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



Dashlet View

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



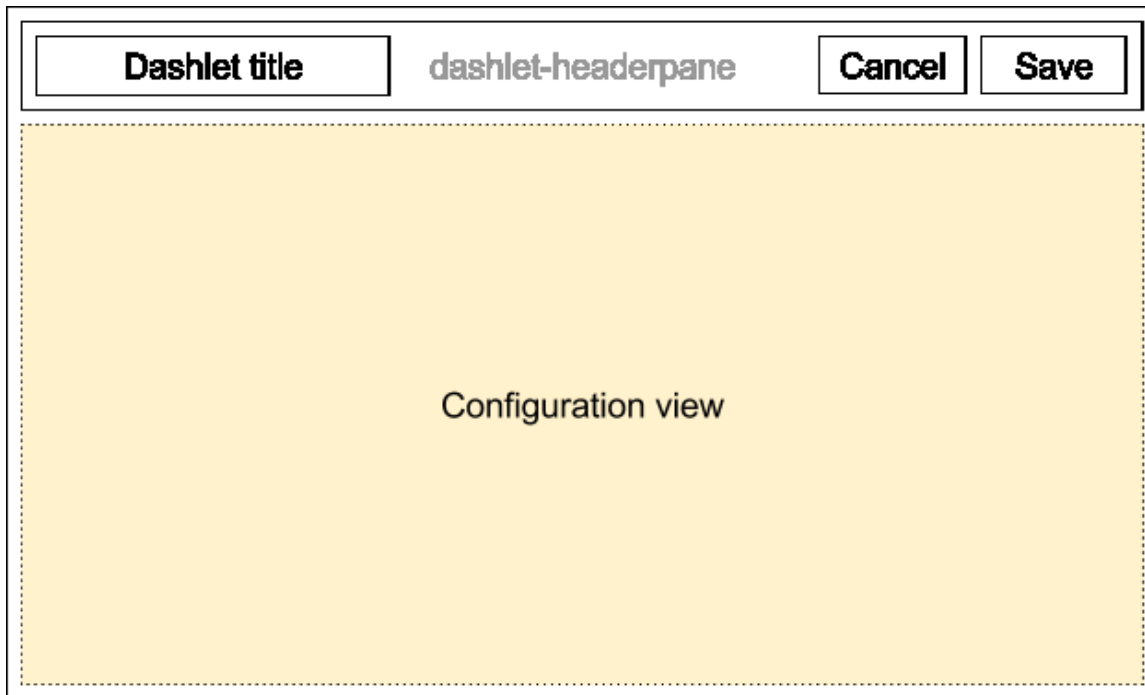
Configuration View

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

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

The values in the config metadata can be retrieved using:

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



Dashlet Example

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

Metadata

The Dashlet view contains the 'dashlets' metadata:

Parameters	Type	Required	Description
label	String	yes	The name of the dashlet
description	String	no	A description of the dashlet

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

The RSS feed dashlets metadata is located in:

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

```
<?php

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

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

```

```

        'auto_refresh' => 0,
        'feed_url' => 'http://blog.sugarcrm.com/feed/',
    ),
),
'panels' => array(
    array(
        'name' => 'panel_body',
        'columns' => 2,
        'labelsOnTop' => true,
        'placeholders' => true,
        'fields' => array(
            array(
                'name' => 'feed_url',
                'label' => 'LBL_RSS_FEED_URL',
                'type' => 'text',
                'span' => 12,
                'required' => true,
            ),
            array(
                'name' => 'limit',
                'label' => 'LBL_RSS_FEED_ENTRIES_COUNT',
                'type' => 'enum',
                'options' => 'tasks_limit_options',
            ),
            array(
                'name' => 'auto_refresh',
                'label' => 'LBL_DASHLET_REFRESH_LABEL',
                'type' => 'enum',
                'options' =>
'sugar7_dashlet_reports_auto_refresh_options',
            ),
        ),
    ),
),
);

```

Controller

The `rssfeed.js` controller file, shown below, contains the JavaScript to render the news articles on the dashlet. The Dashlet view must include 'Dashlet' plugin and can override `initDashlet` to add additional custom process while it is initializing.

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

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

  /**
   * Default options used when none are supplied through metadata.
   *
   * Supported options:
   * - timer: How often (minutes) should refresh the data
collection.
   * - limit: Limit imposed to the number of records pulled.
   *
   * @property {Object}
   * @protected
   */
  _defaultOptions: {
    limit: 5,
    auto_refresh: 0
  }
})
```

```

    },

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

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

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

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

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

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

            // Only set up the interval handler if there is a
refreshRate higher
            // than 0
            if (refreshRate > 0) {
                if (this.timerId) {

```

```

        clearInterval(this.timerId);
    }
    this.timerId = setInterval(_.bind(function() {
        if (self.context) {
            self.context.resetLoadFlag();
            self.loadData(options);
        }
    }, this), refreshRate);
}
}

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

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

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

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

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

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

/**
 * Handles the response of the feed consumption request and sets

```

```

data from
  * the result
  *
  * @param {Object} data Response from the rssfeed API call
  */
handleFeed: function (data) {
  if (this.disposed) {
    return;
  }

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

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

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

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

```

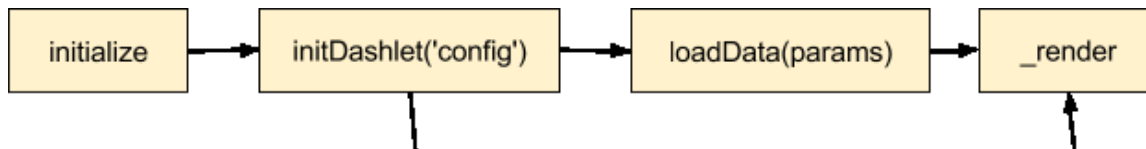
```

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

```

Workflow

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



Retrieving Data

Use the following commands to retrieve the corresponding data:

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

Handlebar Template

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

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

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

```

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

Last Modified: 03/19/2018 09:20am

Drawers

Overview

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

Methods

`app.drawer.open(layoutDef, onClose)`

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

Parameters

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

		reference, when the drawer is closed, the context is destroyed.
onClose	no	Optional callback handler for when the drawer is closed.

Example

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

app.drawer.close(callbackOptions)

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

Parameters

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

Standard Example

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

Callback Example

```
//open drawer
app.drawer.open({
  layout: 'my-layout',
```

```
},
function(message1, message2) {
    alert(message1);
    alert(message2);
});

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

app.drawer.load(options)

Loads a new layout into an existing drawer.

Parameters

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

Example

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

app.drawer.reset(triggerBefore)

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

Parameters

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

Example

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

Last Modified: 10/14/2016 04:12pm

Alerts

Overview

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

Methods

`app.alert.show(id, options)`

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

Parameters

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

Default Alert Values

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

Alert Examples

Standard Alert

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

Confirmation Alert

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

Process Alert

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

```
});
```

app.alert.dismiss(id)

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

Parameters

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

Example

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

app.alert.dismissAll

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

Example

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

Testing in Console

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

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

Last Modified: 01/11/2018 04:36pm

Language

Overview

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

Methods

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

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

Parameters

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

Example

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

`app.lang.getAppString(key)`

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

Parameters

Name	Required	Description
key	yes	The key of the string to

		retrieve
--	--	----------

Example

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

app.lang.getAppListStrings(key)

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

Parameters

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

Example

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

app.lang.getModuleSingular(moduleKey)

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

Parameters

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

Example

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

app.lang.getLanguage()

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

Example

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

app.lang.updateLanguage(languageKey)

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

Parameters

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

Example

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

Testing in Console

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

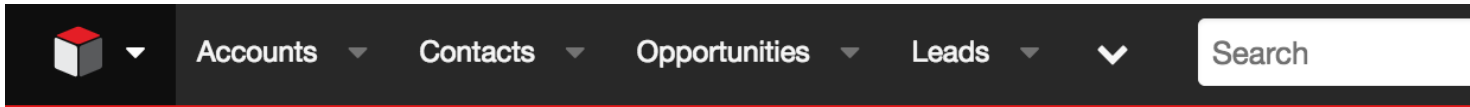
```
App.lang.getAppListStrings('sales_stage_dom');
```

Last Modified: 10/14/2016 04:12pm

MegaMenu

Overview

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



Layout Components

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

Link Actions

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

Name	Description
<code>acl_action</code>	The ACL action is used to verify the user has access to a specific action required for the link
<code>acl_module</code>	The ACL module is used to verify if the user has access to a specific module required for the link
<code>icon</code>	The bootstrap icon to display next to the link (the full list of icons are listed in Admin > Styleguide > Core Elements > Base CSS > Icons)
<code>label</code>	The label key that contains your link's display text
<code>openwindow</code>	Specifies whether or not the link should open in a new window
<code>route</code>	The route to direct the user. For sidecar modules, this is <code>#<module></code> , but modules in backward compatibility

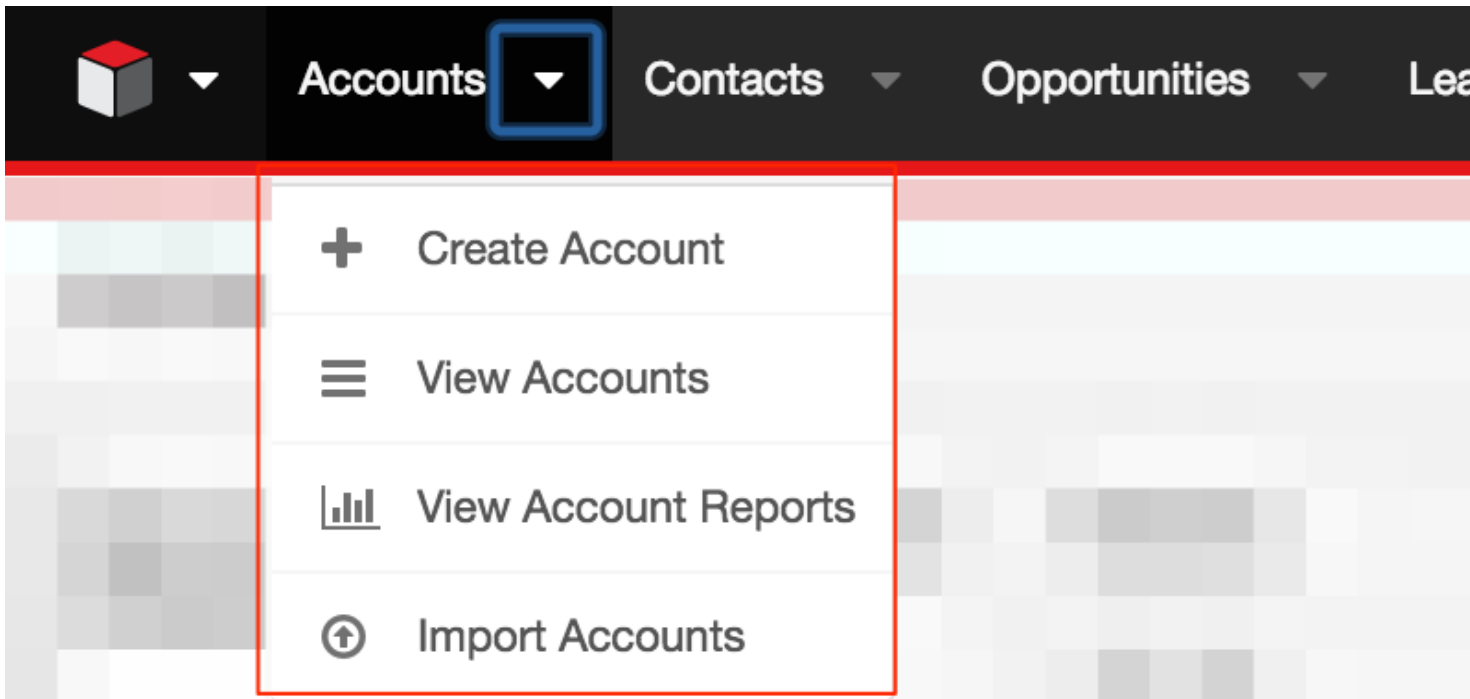
	mode are routed as <code>#bwc/index.php?module=<module></code> . Note: External links require the full URL as well as <code>openwindow</code> set to <code>true</code> .
submenu	An array of sub-navigation links Note: Sub-navigation links contain these same basic link properties.

Module Links

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

Module Action Links

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



Adding Module Action Links

The example below demonstrates how to add a module action link that points to

the Leads module. To define your own module action link, you must create your own label extension for the link's display label:

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

```
<?php

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

Now create the module action link extension:

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

```
<?php

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

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

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

Removing Module Action Links

To remove a module action link, loop through the list of module actions and remove the item by one of its properties. For your reference, the stock module actions can be found in

```
./modules/<module>/clients/base/menus/header/header.php.
```

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

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

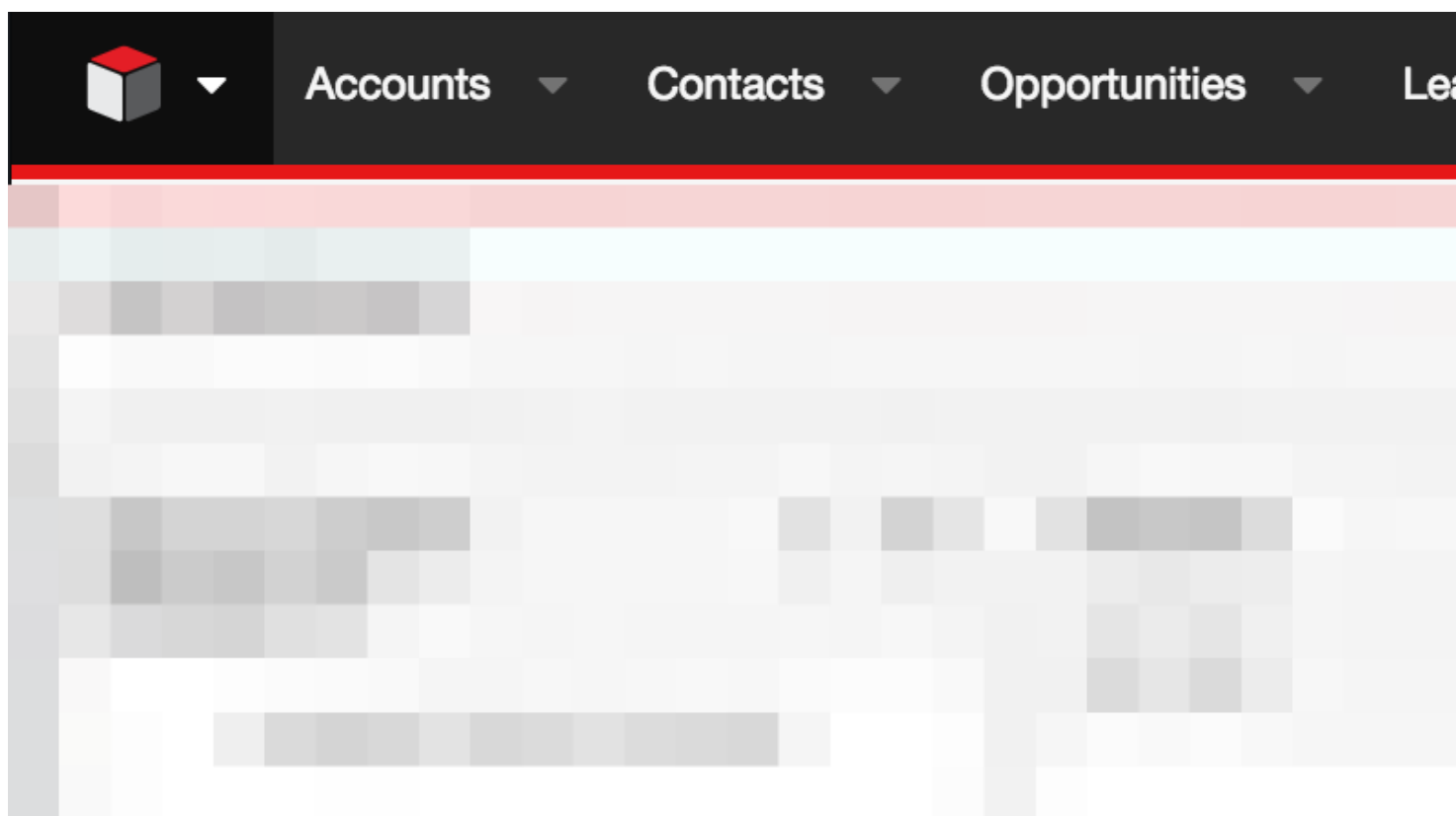
```
key>'')) {  
  
unset($viewdefs['<module>']['base']['menu']['header'][$key]);  
    }  
    }  
}
```

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

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

Profile Action Links

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



Adding Profile Action Links

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

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

```
<?php
```

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

Next, create the profile action link extension:

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

```
<?php
```

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

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

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

Removing Profile Action Links

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

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

```
<?php
```

```
if (isset($viewdefs['base']['view']['profileactions'])) {
    foreach ($viewdefs['base']['view']['profileactions'] as $key =>
        $profileAction) {
```

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

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

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

Last Modified: 01/31/2018 10:44am

Administration Links

Overview

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

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

Example

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

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

```
<?php
```

```
$admin_option_defs = array();
$admin_option_defs['Administration']['<section key>'] = array(
```

```

        //Icon name. Available icons are located in
./themes/default/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'
);

```

To define labels for administration links in the new panel:

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

```
<?php
```

```

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

```

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

Last Modified: 10/14/2016 04:12pm

Legacy MVC

Overview

The legacy MVC Architecture.

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

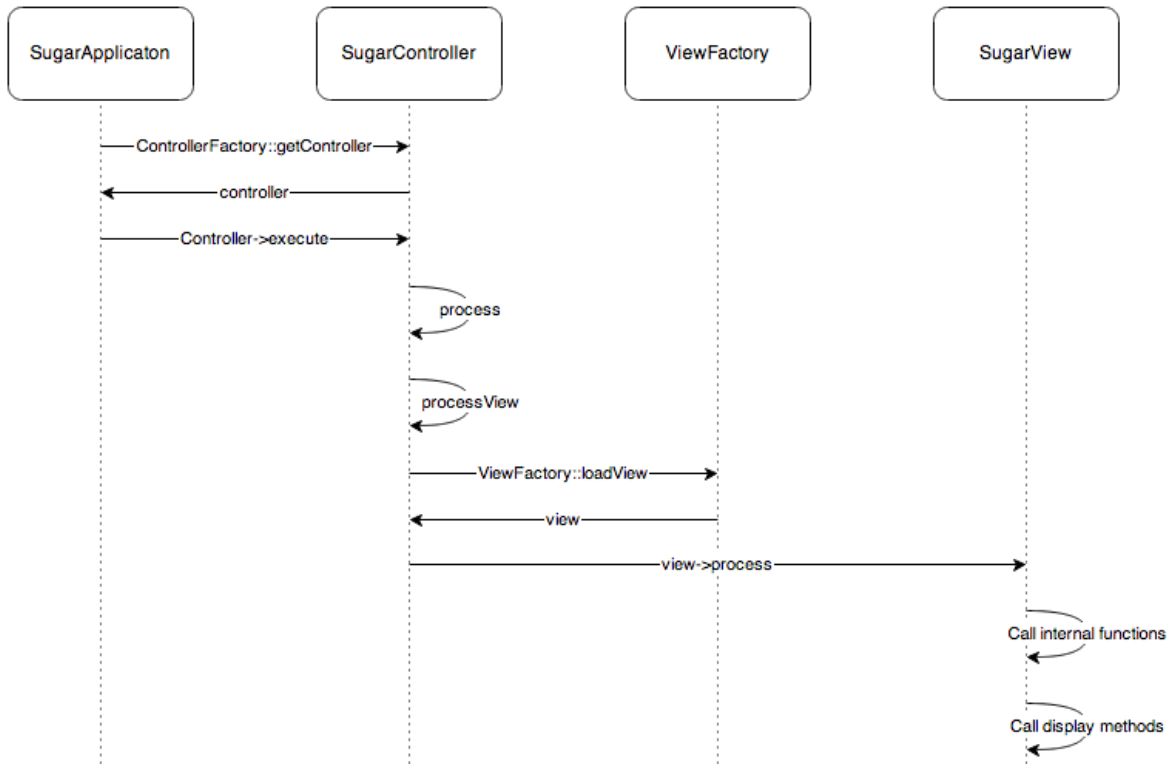
Model-View-Controller (MVC) Overview

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

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

SugarCRM MVC Implementation

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



Last Modified: 10/14/2016 04:12pm

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: 10/14/2016 04:12pm

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 rebuilding the file map can be found in the SugarAutoLoader.

Mapping Actions to Files

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

```
$action_file_map['subpanelviewer'] =  
'include/SubPanel/SubPanelViewer.php';  
$action_file_map['save2'] = 'include/generic/Save2.php';  
$action_file_map['deleterelationship'] =  
'include/generic/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: 10/14/2016 04:12pm

Metadata

Overview

An overview of the legacy MVC metadata framework.

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

Metadata Framework

Background

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

Application Metadata

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

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

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

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

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

Module Metadata

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

File	Description
<code>additionalDetails.php</code>	Used to render the popup information displayed when a user hovers the mouse cursor over a row in the List View.
<code>editviewdefs.php</code>	Used to render a record's EditView.
<code>detailviewdefs.php</code>	Used to render a record's DetailView.
<code>listviewdefs.php</code>	Used to render the List View display for a module.
<code>metafiles.php</code>	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 `$searchDefs` array is another array that describes the search form layout and fields.

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

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

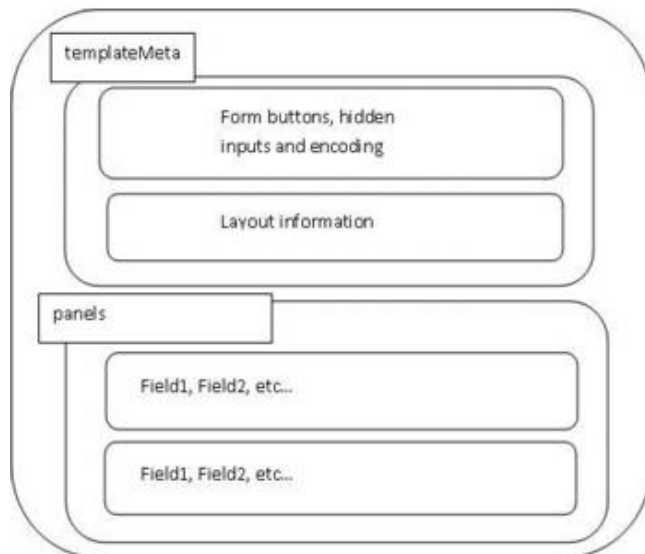
The `searchdefs.php` file is invoked from the MVC framework whenever a module's list view is rendered (see `include/MVC/View/views/view.list.php`). Within `view.list.php`, checks are made to see if the module has defined a `SearchForm.html` file. If this file exists, the MVC will run in classic mode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

The processing flow for the search form using the `metadata subpaneldefs.php` file

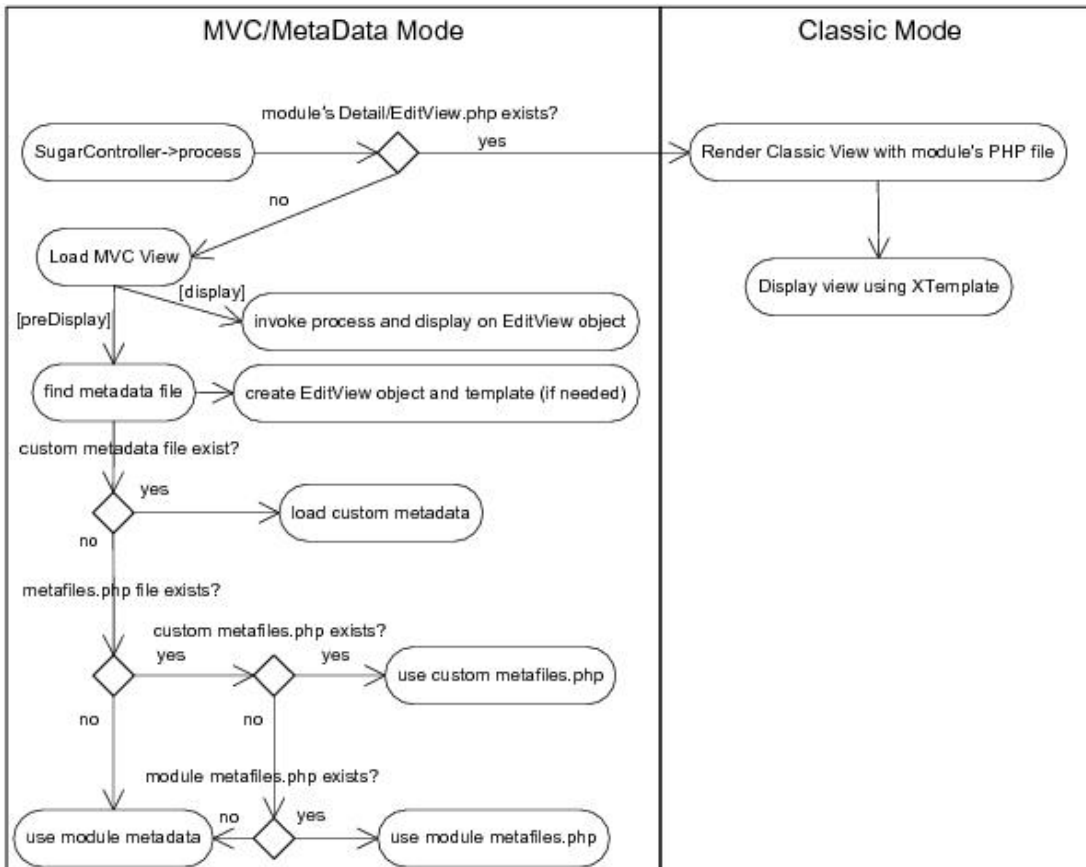
is similar to that of `EdiView` and `DetailView`.

DetailView and EditView Metadata

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



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



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

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

```

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

```

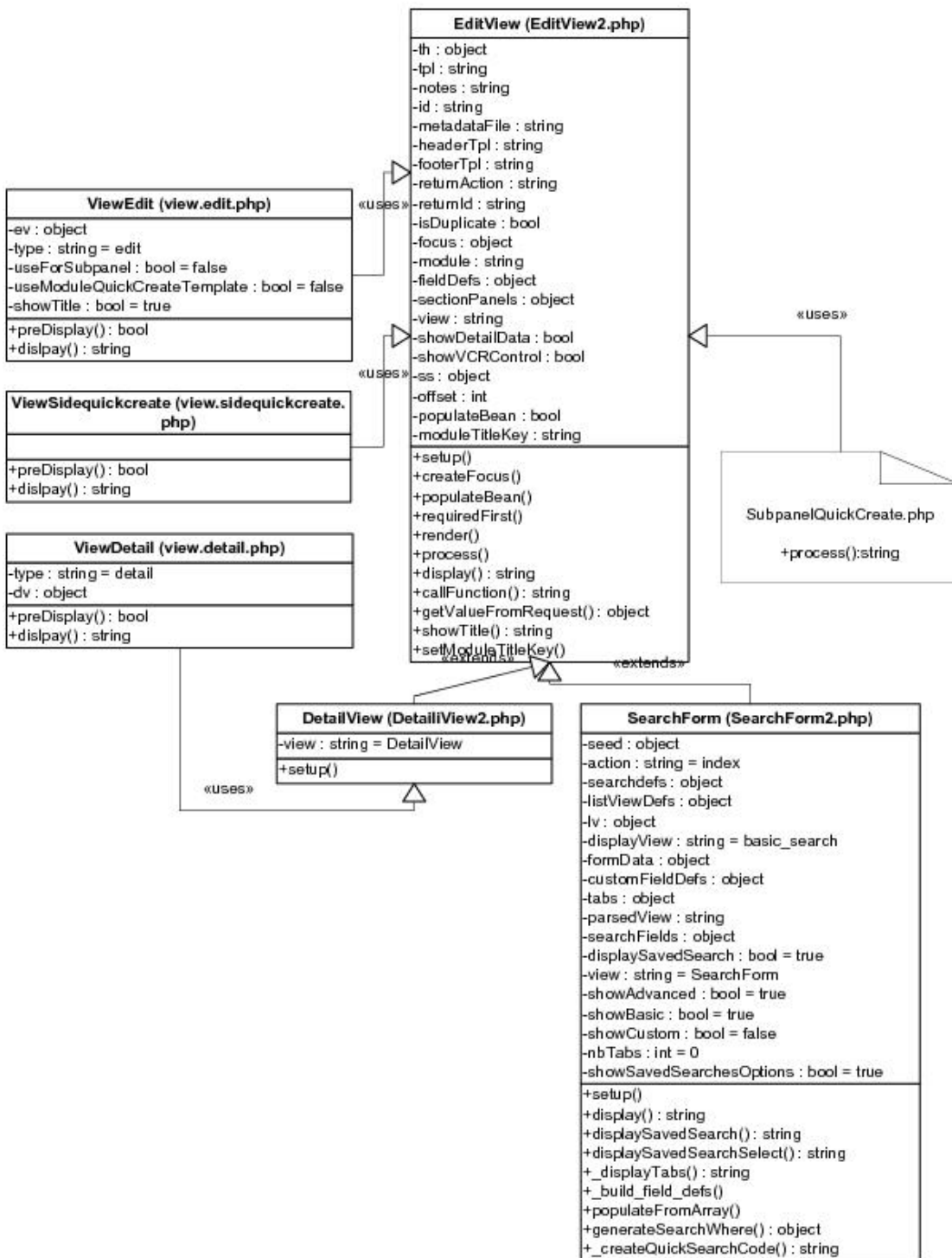
```
'searchfields' => 'modules/Accounts/metadata/SearchFields.php',
);
```

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

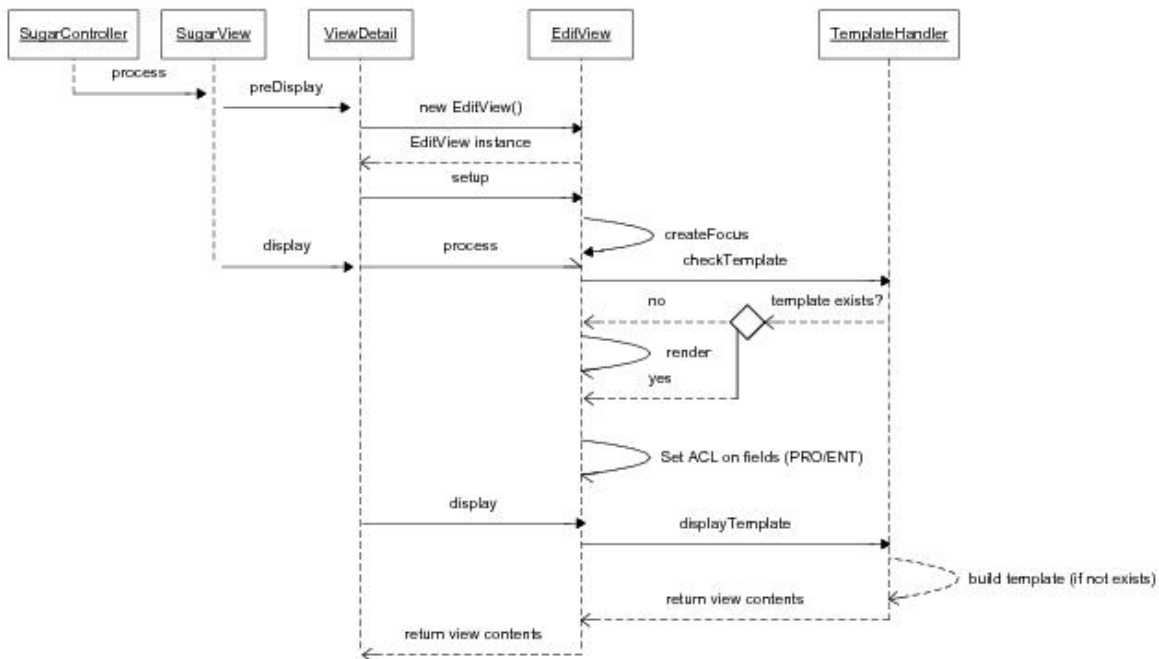
After the `preDisplay` method is called in the view code, the `display` method is called, resulting in a call to the `EditView` object's `process` method, as well as the `EditView` object's `display` method.

The `EditView` class is responsible for the bulk of the Metadata file processing and creation of the resulting display. The `EditView` class also checks to see if the resulting Smarty template is already created. It also applies the field level ACL controls for 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.



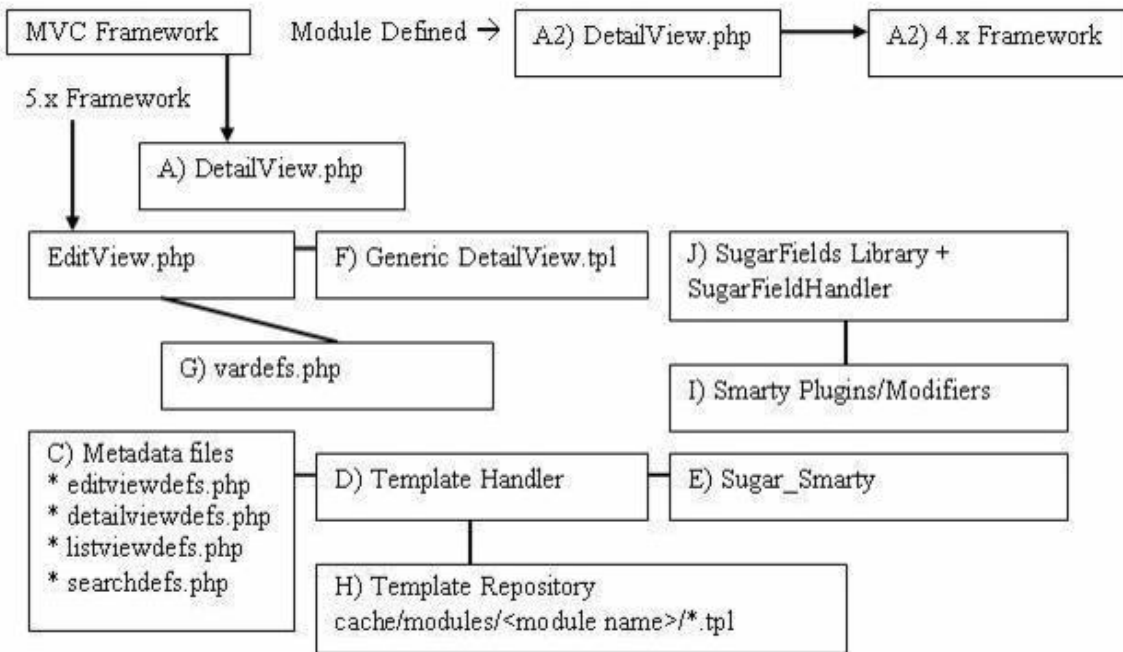
The following diagram highlights the EditView class's main responsibilities and their relationships with other classes in the system. We will use the example of a DetailView request although the sequence will be similar for other views that use the EditView class.



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 DetailView.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:

100%			
50%			
12.5%		37.5%	
Label 1	Value 1	Label 2	Value 2
Label 3	Value 3	Label 4	Value 4
...		...	
...		...	

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.

100%		
50%		
12.5%	37.5%	
Name	Fee	Amount \$100,000
Account	Test Account	
...		Type New Business
...		...

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 repository (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: 10/14/2016 04:12pm

Examples

Legacy MVC metadata examples.



Last Modified: 10/14/2016 04:12pm

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.

Hiding 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 copying 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: 05/19/2017 09:38am

Manipulating Buttons on Legacy MVC Layouts

Overview

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

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

Metadata

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

Custom Layouts

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

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

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

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

Editing Layouts

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

Developer Mode

You can turn on Developer Mode:

```
Admin > System Settings
```

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

Quick Repair and Rebuild

You can run a Quick Repair and Rebuild:

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

Doing this will rebuild the cache for the metadata.

Saving & Deploying the Layout in Studio

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

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

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

Adding Custom Buttons

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

JavaScript Actions

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

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

Example

```
<?php
```

```
$viewdefs['Accounts'] =
```

```
array (
```

```
    'DetailView' =>
```

```
    array (
```

```
        'templateMeta' =>
```

```
        array (
```

```
            'form' =>
```

```
            array (
```

```
'buttons' =>
array (
    0 => 'EDIT' ,
    1 => 'DUPLICATE' ,
    2 => 'DELETE' ,
    3 => 'FIND_DUPLICATES' ,
    4 => 'CONNECTOR' ,
    5 =>
        array (
            'customCode' => '<input id="JavaScriptButton"
title="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' ] [ 'closeFormBeforeCustomButtons' ]
```

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: 10/14/2016 04:12pm

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: 10/14/2016 04:12pm

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

Sugar Cloud

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',
```

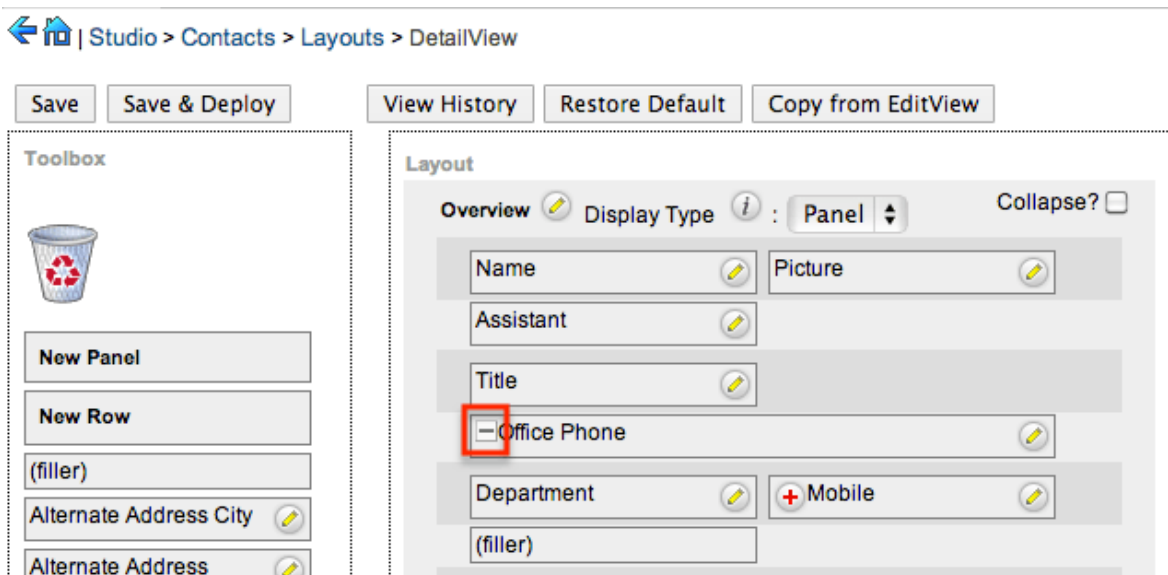
Once that is updated, locate the 'widths' array to define the spacing for your new column(s). You should have a label and field entry for each column in your layout:

```
'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 Sugar cloud 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.

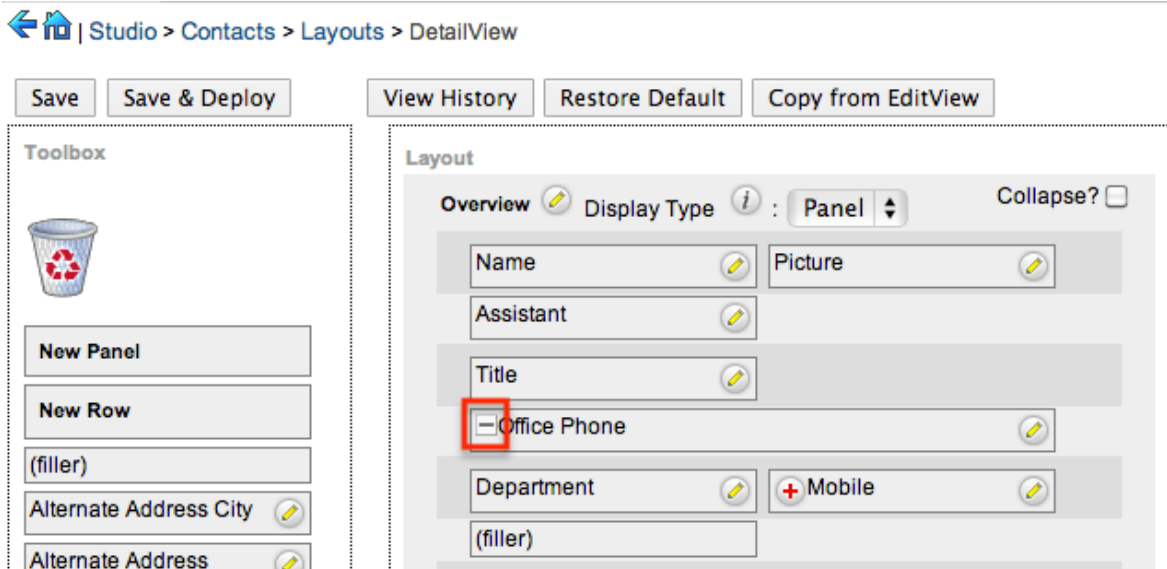
Next, open the custom layout file, locate the 'maxColumns' value, and change it to the number of columns you would like to have on screen:

```
'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: 03/15/2018 07:22pm

Examples

Provides an overview of example MVC customizations.

Last Modified: 10/14/2016 04:12pm

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 rebuilding 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 original 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: 10/14/2016 04:12pm

Data Framework

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

Last Modified: 10/14/2016 04:12pm

Modules

Overview

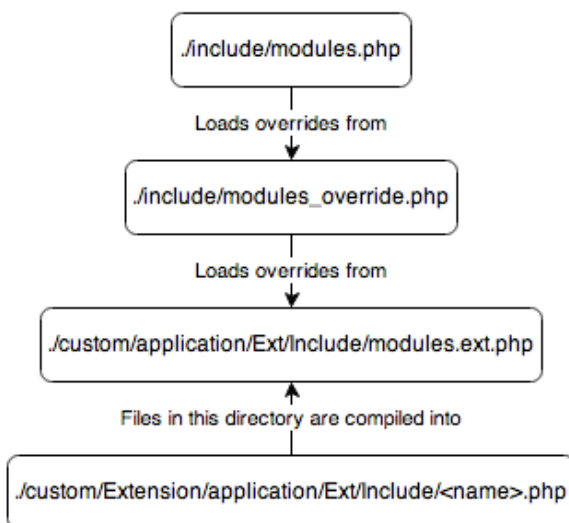
How modules are defined and used within the system

Module Definitions

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

Hierarchy Diagram

The modules metadata are loaded in the following manner:



`$moduleList`

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

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

`$beanList`

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

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

`$beanFiles`

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

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

`$modInvisList`

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

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

`$modules_exempt_from_availability_check`

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

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

`$report_include_modules`

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

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

\$adminOnlyList

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

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

\$bwcModules

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

Last Modified: 10/14/2016 04:12pm

Models

Overview

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

SugarObject Templates

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

To further illustrate this, let's say the Basic object type has a field 'name' with length 10 and the Company object has a field 'name' with length 20. If you inherit from Basic first and then Company, your field will inherit the Company object's length of 20. However, the module-level setting always overrides any values provided by Sugar objects, so, if the field 'name' in your module has been set to length 60, then the field's length will ultimately be 60.

There are six types of SugarObject templates:

- Basic : Contains the basic fields required by all Sugar modules
- Person : Based on the Contacts, Prospects, and Leads modules
- Issue : Based on the Bugs and Cases modules
- Company : Based on the Accounts module
- File : Based on the Documents module
- Sale : Based on the Opportunities module

SugarObject Interfaces

In addition to the object templates above, "assignable" object templates can be used for modules with records that should contain an Assigned To field. Although every module does not use this, most modules allow assignment of records to users. SugarObject interfaces allow us to add the assignable attribute to these modules.

SugarObject interfaces and SugarObject templates are very similar to one another, but the main distinction is that templates have a base class you can subclass while interfaces do not. If you look into the file structure, templates include many additional files, including a full metadata directory. This is used primarily for Module Builder.

File Structure

- ./include/SugarObjects/interfaces
- ./include/SugarObjects/templates

Implementation

There are two things you need to do to take advantage of Sugar objects:

1. Subclass the SugarObject class you wish to extend for your class:

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

2. Add the following line to the end of the vardefs.php file:

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

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

Performance Considerations

VardefManager caches the generated vardefs into a single file that is loaded at run time. If that file is not found, Sugar loads the vardefs.php file, located in your module's directory. The same is true for language files. This caching also includes data for custom fields and any vardef or language extensions that are dropped into the extension framework.

Cache Files

- ./cache/modules/<module>/<object_name>vardefs.php
- ./cache/modules/<module>/languages/en_us.lang.php

Last Modified: 10/14/2016 04:12pm

SugarBean

Overview

The SugarBean class supports all the model operations to and from the database. Any module that interacts with the database utilizes the SugarBean class, which contains methods to create, read/retrieve, update, and delete records in the database (CRUD), as well as fetch related records.

CRUD Handling

The SugarBean handles the most basic functions of the Sugar database, allowing you to create, retrieve, update, and delete data.

Creating and Retrieving Records

The BeanFactory class is used for bean loading. The class should be used whenever you are creating or retrieving bean objects. It will automatically handle any classes required for the bean. More information on this can be found in the BeanFactory section.

Obtaining the Id of a Recently Saved Bean

For new records, a GUID is generated and assigned to the record id field. Saving new or existing records returns a single value to the calling routine, which is the id attribute of the saved record. The id has the following format:

```
aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee
```

You can retrieve a newly created record's id with the following:

```
//Create bean
$bean = BeanFactory::newBean($module);

//Populate bean fields
$bean->name = 'Example Record';

//Save
$bean->save();

//Retrieve the bean id
$record_id = $bean->id;
```

Saving a Bean with a Specific ID

Saving a record with a specific id requires the id and new_with_id attribute of the bean to be set. When doing so, it is important that you use a globally unique identifier. Failing to do so may result in unexpected behavior in the system. An example setting an id is shown below:

```
//Create bean
$bean = BeanFactory::newBean($module);

//set the new flag
$bean->new_with_id = true;

//Set the record id with a static id
$id = '38c90c70-7788-13a2-668d-513e2b8df5e1';
$bean->id = $id;

//or have the system generate an id for you
//$id = Sugarcrm\Sugarcrm\Util\Uuid::uuid1()
//$bean->id = $id;

//Populate bean fields
$bean->name = 'Example Record';

//Save
$bean->save();
```

Setting `new_with_id` to true prevents the save method from creating a new id value and uses the assigned id attribute. If the id attribute is empty and the `new_with_id` attribute is set to true, the save will fail. If you would like for the system to generate an id for you, you can use `Sugarcrm\Sugarcrm\Util\Uuid::uuid1()`.

Distinguishing New from Existing Records

To identify whether or not a record is new or existing, you can check the `fetches_rows` property. If the `$bean` has a `fetches_row`, it was loaded from the database. An example is shown below:

```
if (!isset($bean->fetches_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
}
```

Updating a Bean

You can update a bean by fetching a record and then updating the property:

```
//Retrieve bean
$bean = BeanFactory::retrieveBean($module, $id);

//Fields to update
$bean->name = 'Updated Name';

//Save
$bean->save();
```

Note: Disabling row-level security when accessing a bean should be set to true only when it is absolutely necessary to bypass security, for example, when updating a Lead record from a custom Entry Point. An example of accessing the bean while bypassing row security is:

```
$bean = BeanFactory::retrieveBean($module, $record_id,
array('disable_row_level_security' => true));
```

Updating a Bean Without Changing the Date Modified

The SugarBean class contains an attribute called `update_date_modified`, which is set to true when the class is instantiated and means that the `date_modified` attribute is updated to the current database date timestamp. Setting `update_date_modified` to false would result in the `date_modified` attribute not being set with the current database date timestamp.

```
//Retrieve bean
$bean = BeanFactory::retrieveBean($module, $id);

//Set modified flag
$bean->update_date_modified = false;

//Fields to update
$bean->name = 'Updated Name';

//Save
$bean->save();
```

Note: Disabling row-level security when accessing a bean should be set to true only when it is absolutely necessary to bypass security, for example, when updating a Lead record from a custom Entry Point. An example of accessing the bean while bypassing row security is:

```
$bean = BeanFactory::retrieveBean($module, $record_id,  
array('disable_row_level_security' => true));
```

Deleting a Bean

Deleting a bean can be done by fetching it then calling the `mark_deleted()` method which makes sure any relationships with related records are removed as well:

```
//Retrieve bean  
$bean = BeanFactory::retrieveBean($module, $id);  
  
//Set deleted to true  
$bean->mark_deleted();  
  
//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));
```

Fetching Relationships

This section explains how the `SugarBean` class can be used to fetch related information from the database.

Fetching Related Records

To fetch related records, load the relationship using the link:

```
//If relationship is loaded  
if ($bean->load_relationship($link)) {  
    //Fetch related beans
```

```
    $relatedBeans = $bean->$link->getBeans();
}
```

An example of this is to load the contacts related to an account:

```
//Load Account
$bean = BeanFactory::getBean('Accounts', $id);

//If relationship is loaded
if ($bean->load_relationship('contacts')) {
    //Fetch related beans
    $relatedBeans = $bean->contacts->getBeans();
}
```

Fetching a Parent Record

Fetching a parent record is similar to fetching child records in that you will still need to load the relationship using the link:

```
//If relationship is loaded
if ($bean->load_relationship($link)) {
    //Fetch related beans
    $relatedBeans = $bean->$link->getBeans();

    $parentBean = false;
    if (!empty($relatedBeans)) {
        //order the results
        reset($relatedBeans);

        //first record in the list is the parent
        $parentBean = current($relatedBeans);
    }
}
```

An example of this is to load a contact and fetch its parent account:

```
//Load Contact
$bean = BeanFactory::getBean('Contacts', $id);

//If relationship is loaded
if ($bean->load_relationship('accounts')) {
    //Fetch related beans
    $relatedBeans = $bean->accounts->getBeans();

    $parentBean = false;
```

```
if (!empty($relatedBeans)) {
    //order the results
    reset($relatedBeans);

    //first record in the list is the parent
    $parentBean = current($relatedBeans);
}
}
```

Last Modified: 04/25/2018 04:09am

BeanFactory

Overview

The BeanFactory class, located in `./data/BeanFactory.php`, is used for loading an instance of a SugarBean. This class should be used any time you are creating or retrieving bean objects. It will automatically handle any classes required for the bean.

Creating a SugarBean Object

`newBean()`

To create a new, empty SugarBean, use the `newBean()` method. This method is typically used when creating a new record for a module or to call properties of the module's bean object.

```
$bean = BeanFactory::newBean($module);
```

`newBeanByName()`

Used to fetch a bean by its beanList name.

```
$bean = BeanFactory::newBeanByName($name);
```

Retrieving a SugarBean Object

`getBean()`

The `getBean()` method can be used to retrieve a specific record from the database.

If a record id is not passed, a new bean object will be created.

```
$bean = BeanFactory::getBean($module, $record_id);
```

Note: Disabling row-level security when accessing a bean should be set to true only when it is absolutely necessary to bypass security, for example when updating a Lead record from a custom Entry Point. An example of accessing the bean while bypassing row security is:

```
$bean = BeanFactory::getBean($module, $record_id,  
array('disable_row_level_security' => true));
```

retrieveBean()

The `retrieveBean()` method can also be used to retrieve a specific record from the database. The difference between this method and `getBean()` is that null will be returned instead of an empty bean object if the retrieve fails.

```
$bean = BeanFactory::retrieveBean($module, $record_id);
```

Note: Disabling row-level security when accessing a bean should be set to true only when it is absolutely necessary to bypass security, for example, when updating a Lead record from a custom Entry Point. An example of accessing the bean while bypassing row security is:

```
$bean = BeanFactory::retrieveBean($module, $record_id,  
array('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 a name of 'Case'.

```
$moduleKey = BeanFactory::getObjectName($moduleName);
```

getBeanName()

The `getBeanName()` method will retrieve the bean class name given a module name.

```
$moduleClass = BeanFactory::getBeanName($module);
```

Last Modified: 10/14/2016 04:12pm

Vardefs

Overview

Vardefs (Variable Definitions) provide the Sugar application with information about SugarBeans. Vardefs specify information on the individual fields, relationships between beans, and the indexes for a given bean.

Each module that contains a SugarBean file has a vardefs.php file located in it, which specifies the fields for that SugarBean. For example, the vardefs for the Contact bean are located in `./modules/Contacts/vardefs.php`.

Dictionary Array

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

- `table` : The name of the database table (usually, the name of the module) for this bean that contains the fields
- `audited` : Specifies whether the module has field-level auditing enabled
- `duplicate_check` : Determines if duplicate checking is enabled on the module, and what duplicate check strategy will be used if enabled.
- `fields` : A list of fields and their attributes
- `indices` : A list of indexes that should be created in the database
- `optimistic_locking` : Determines whether the module has optimistic locking enabled
 - Optimistic locking prevents loss of data by using the bean's modified date to ensure that it is not being modified simultaneously by more than one person or process.
- `unified_search` : Determines whether the module can be searched via Global Search
 - This setting defaults to false and has no effect if all of the fields in the `fields` array have the `'unified_search'` field attribute set to false.
- `unified_search_default_enabled` : Determines whether the module should be searched by default for new users via Global Search
 - This setting defaults to true but has no effect if `unified_search` is set to

false.

- `visibility` : A list of visibility classes enabled on the module

Duplicate Check Array

The `duplicate_check` array contains two properties, that control if duplicate checking is enabled on the module, and which duplicate check strategy will be used to check for duplicates.

The two properties for the array are as follows:

Name	Type	Description
<code>enabled</code>	Boolean	Specifies whether or not the Bean is utilizing the duplicate check framework
<code><class_name></code>	Array	<code><class_name></code> is the name of the duplicate check strategy class that is handling the duplicate checking. It is set to an array of Metadata, specific to the strategy defined in the key. Review the Duplicate Check Framework for further information.

Fields Array

The `fields` array contains one array for each field in the SugarBean. At the top level of this array, the key is the name of the field, and the value is an array of attributes about that field.

The list of possible attributes are as follows:

- `name` : The name of the field
- `vname` : The language pack ID for the label of this field
- `type` : The type of the attribute
 - `assigned_user_name` : A linked user name
 - `bool` : A boolean value

-
- char : A character array
 - date : A date value with no time
 - datetime : A date and time
 - email : An email address field
 - enum : An enumeration (dropdown list from the language pack)
 - id : A database GUID
 - image : A photo-type field
 - link : A link through an explicit relationship. This attribute should only be used when working with related fields. It does not make the field render as a link.
 - name : A non-database field type that concatenates other field values
 - phone : A phone number field to utilize with callto:// links
 - relate : Related bean
 - team_list : A team-based ID
 - text : A text area field
 - url : A hyperlinked field on the detail view
 - varchar : A variable-sized string
 - table : The database table the field comes from.
 - The table attribute is only needed to join fields from another table outside of the module in focus.
 - isnull : Whether the field can contain a null value
 - len : The length of the field (number of characters if a string)
 - options : The name of the enumeration (dropdown list) in the language pack for the field
 - dbtype : The database type of the field (if different than the type)
 - reportable : Determines whether the field will be available in the Reports and Workflow modules (for commercial editions)
 - default : The default value for this field
 - massupdate : Determines whether the field will show up in the mass-update panel on its module's list view
 - Some field types are restricted from mass update regardless of this setting.
 - rname : For relate-type fields, the field from the related variable that contains the text
 - id_name : For relate-type fields, the field from the bean that stores the ID for the related bean
 - source : Set to 'non-db' if the field value does not come from the database
 - The source attribute can be used for calculated values or values retrieved in some other way.
 - sort_on : The field to sort by if multiple fields are used in the presentation of field's information
 - fields : For concatenated values, an array containing the fields that are concatenated
 - db_concat_fields : For concatenated values, an array containing the fields to concatenate in the database
 - unified_search : Determines whether the field can be searched via Global

Search

- This has no effect if the dictionary array setting 'unified_search' is not set to true.
- `enable_range_search` : For date, datetime, and numeric fields, determines if this field should be searchable by range in module searches
- `dependency` : Allows a field to have a predefined formula to control the field's visibility
- `studio` : Controls the visibility of the field in the Studio editor
 - If set to false, then the field will not appear in any studio screens for the module. Otherwise, you may specify an Array of view keys from which the field's visibility should be removed (e.g. `array('listview'=>false)` will hide the field in the listview layout screen).

The following example illustrates a standard ID field for a bean:

```
'id' => array (  
    'name' => 'id',  
    'vname' => 'LBL_ID',  
    'type' => 'id',  
    'required' => true,  
),
```

Indices Array

This array contains a list of arrays that are used to create indices in the database. The fields in this array are:

- `name` : The unique name of the index
- `type` : The type of index (primary, unique, or index)
- `fields` : An ordered array of the fields to index
- `source` : Set to 'non-db' if you are creating an index for added application functionality such as duplication checking on imports

The following example creates a primary index called 'userspk' on the 'id' column:

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

Relationships Array

The relationships array specifies relationships between beans. Like the indices array entries, it is a list of names with array values.

-
- `lhs_module` : The module on the left-hand side of the relationship
 - `lhs_table` : The table on the left-hand side of the relationship
 - `lhs_key` : The primary key column of the left-hand side of the relationship
 - `rhs_module` : The module on the right-hand side of the relationship
 - `rhs_table` : The table on the right-hand side of the relationship
 - `rhs_key` : The primary key column of the right-hand side of the relationship
 - `relationship_type` : The type of relationship (e.g. one-to-many, many-to-many)
 - `relationship_role_column` : The type of relationship role
 - `relationship_role_column_value` : Defines the unique identifier for the relationship role

The following example creates a relationship between a contact and the contact that they report to. The `reports_to_id` field maps to the id of the record that belongs to the higher-ranked contact. This is a one-to-many relationship in that a contact may only report to one person, but many people may report to the same contact.

```
'contact_direct_reports' => array(  
  'lhs_module' => 'Contacts',  
  'lhs_table' => 'contacts',  
  'lhs_key' => 'id',  
  'rhs_module' => 'Contacts',  
  'rhs_table' => 'contacts',  
  'rhs_key' => 'reports_to_id',  
  'relationship_type' => 'one-to-many'  
)
```

Visibility Array

The visibility array specifies the row level visibility classes that are enabled on the bean. Each entry in the array, is a key-value pair, where the key is the name of the Visibility class and the value is set to boolean True. More information on configuring custom Visibility strategies can be found in the Architecture section under Visibility Framework.

Extending Vardefs

More information about extending and overriding vardefs can be found in the Extensions Framework documentation under Vardefs.

Last Modified: 11/27/2017 12:24pm

Manually Creating Custom Fields

Overview

The most common way to create custom fields in Sugar is via Studio inside the application. This page describes how to use the `ModuleInstaller` class or `vardef` extensions as alternative methods of creating custom fields.

Using ModuleInstaller to Create Custom Fields

There are two ways to create a field using the `ModuleInstaller` class: via installer package or programmatically. An example of creating a field from a module-loadable package is explained in the [Module Loader documentation](#), [Creating an Installable Package that Creates New Fields](#). The following example shows how to programmatically add custom fields using the `ModuleInstaller` class with the `install_custom_fields()` method:

```
<?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);
```

Add labels for custom fields by creating a corresponding language extension file:

```
./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';
```

Finally, navigate to Admin > Repair > Quick Repair and Rebuild to make the new field available for users.

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 field's custom vardef. This situation is outlined in the following section.
- You must run a Quick Repair and Rebuild and then execute the generated SQL after the vardef is installed.
- You must correctly define the properties of a vardef. If you miss any, the field may not work properly.
- Your field name must end with "_c" and have the property 'source' set to 'custom_fields'. This is required as you should not modify core tables in Sugar and it is not permitted on Sugar's cloud service.

-
- Your vardef must specify the exact indexes of the properties you want to set. For example, 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 challenge when creating your own custom vardef is getting the system to recognize the vardef and generate the database field. This issue is 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 the vardef is in place, determine whether the custom field's module already contains any other custom fields. If there are not any existing custom fields, 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, 1, 'true');

```

Finally, 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. Execute the scripts generated to create the Save 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*/

```

```

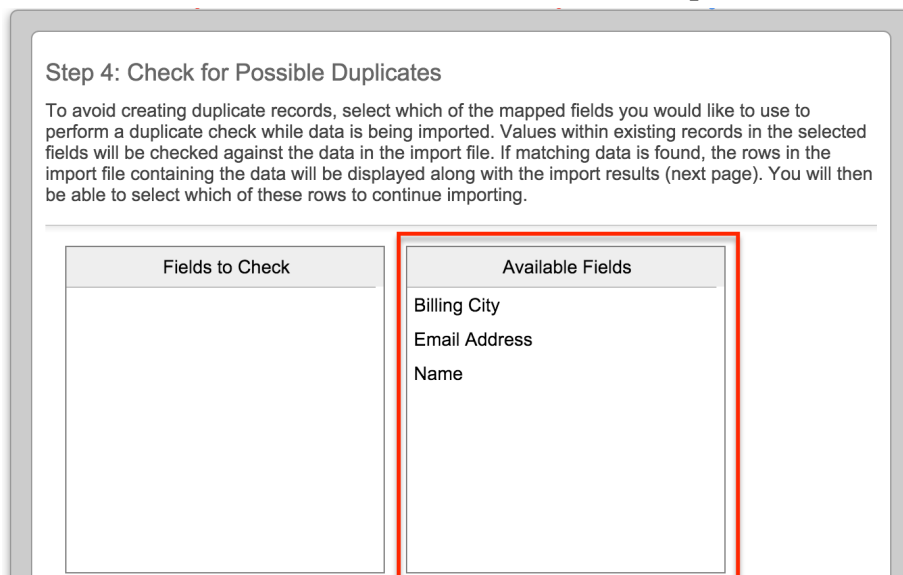
ALTER TABLE <module>_cstm add COLUMN example_c varchar(255) NULL ;

```


Specifying Custom Indexes for Import Duplicate Checking

Overview

When importing records to Sugar via the Import Wizard, users can select which of the mapped fields they would like to use to perform a duplicate check and thereby avoid creating duplicate records. This article explains how to enable an additional field or set of fields for selection in this step.



Resolution

The import wizard's duplicate check operates based on indices defined for that module. You can create a non-database index to check for a field. It is important that it is non-database as single column indices on your database can hamper overall performance. The following is an example to add the home phone field to the Contact module's duplicate check.

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

```
./custom/Extension/modules/Contacts/Ext/Vardefs/custom_import_index.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: 10/14/2016 04:12pm

Working With Indexes

Overview

Sugar provides a simple method for creating custom indexes through the vardef framework. Indexes can be built on one or more fields within a module. Indexes can be saved down to the database level or made available only in the application for functions such as Import Duplicate Checking.

Index Metadata

Indexes have the following metadata options that can be configured per index:

Key	Value	Description
name	string	A Unique identifier to define the index. Best practices recommend indexes start with idx and contain the suffix cstm to avoid conflicting with a stock index. Note : Some databases have restrictions on the length of index names. Please check with your database vendor to avoid any issues.
type	string	All indexes should use the type of "index"
fields	array	A PHP array of the fields for the index to utilize
source	string	Specify as "non-db" to avoid creating the index in the database

Creating Indexes

Stock indexes are initially defined in the module's vardefs file under the indices array. For reference, you can find them using the vardef path of your module. The path will be `./modules/<module>/vardefs.php`.

Custom indexes should be created using the Extension Framework. First, create a PHP file in the extension directory of your desired module. The path should be similar to `./custom/Extension/modules/<module>/Ext/Vardefs/<name>.php`.

In the new file, add the appropriate `$dictionary` reference to define the custom index:

```
<?php

$dictionary['<module>']['indices'][] = array(
    'name' => '<index name>',
    'type' => 'index',
    'fields' => array(
        'field1',
        'field2',
    )
);
```

Note : For performance reasons, it is not recommended to create an index on a single field unless the source is set to non-db.

Once installed, you will need to navigate to Admin > Repair > Quick Repair and Rebuild to enable the custom index. You will need to execute any scripts generated by the rebuild process.

Removing Indexes

Stock indexes are initially defined in the module's `vardefs` file under the `indices` array. For reference, you can find them using the `vardef` path of your module. The path will be `./modules/<module>/vardefs.php`.

Stock indexes should be removed using the Extension Framework. First, create a PHP file in the extension directory of your desired module. The path should be similar to `./custom/Extension/modules/<module>/Ext/Vardefs/<name>.php`.

In the new file, loop through the existing `'indices'` sub-array of the `$dictionary` to locate the stock index to remove, and use `unset()` to remove it from the array.

Example

The following is an example to remove the `idx_calls_date_start` index from the `Call` module's `vardefs`.

First, create

`./custom/Extension/modules/Calls/Ext/Vardefs/remove_idx_calls_date_start.php` from the root directory of your Sugar installation on the web server. When creating the file, keep in mind the following requirements:

- The name of the file is not important, as long as it ends with a `.php` extension.
- The rest of the directory path is case sensitive so be sure to create the directories as shown.
- If you are removing the index for a module other than Calls, then substitute the corresponding directory name with that module.
- Ensure that the entire directory path and file have the correct ownership and sufficient permissions for the web server to access the file.

The contents of the file should look similar to the following code:

```
<?php

$call_indexes = $dictionary['Call']['indices'];
$remove_index = "idx_calls_date_start";

foreach($call_indexes as $index_key => $index_item) {
    if( $index_item['name'] == $remove_index ) {
        unset($dictionary['Call']['indices'][$index_key]);
    }
}
```

Note : Removing the reference to the index from the module's indices array does not actually remove the index from the module's database table. Removing the reference from the indices array ensures that the index is not added back to the module's database table when performing any future Quick Repair and Rebuilds. The database index must be removed directly at the database level. On MySQL, with the current example, this could be done with a query like:

```
ALTER TABLE calls DROP INDEX idx_calls_date_start;
```

Once installed, you will need to navigate to Admin > Repair > Quick Repair and Rebuild to remove the index from the `$dictionary` array. You will need to execute any scripts generated by the 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.

Step 4: Check for Possible Duplicates

To avoid creating duplicate records, select which of the mapped fields you would like to use to perform a duplicate check while data is being imported. Values within existing records in the selected fields will be checked against the data in the import file. If matching data is found, the rows in the import file containing the data will be displayed along with the import results (next page). You will then be able to select which of these rows to continue importing.

Fields to Check	Available Fields
	Billing City Email Address Name

Example

The following is an example to add the home phone field to the Contact module's duplicate check.

First, create

`./custom/Extension/modules/Contacts/Ext/Vardefs/custom_import_index.php` from the root directory of your Sugar installation on the web server. When creating the file, keep in mind the following requirements:

- The name of the file is not important, as long as it ends with a `.php` extension.
- The rest of the directory path is case sensitive so be sure to create the directories as shown.
- If you are creating the import index for a module other than Contacts, then substitute the corresponding directory name with that module.
- Ensure that the entire directory path and file have the correct ownership and sufficient permissions for the web server to access the file.

The contents of the file should look similar to the following code:

```
<?php
```

```
$dictionary['Contact']['indices'][] = array(  
    'name' => 'idx_home_phone_cstm',  
    'type' => 'index',  
    'fields' => array(  
        0 => 'phone_home',
```

```
    ),  
    'source' => 'non-db',  
);
```

Please note that the module name in line 2 of the code is singular (i.e. Contact, not Contacts). If you are unsure of what to enter for the module name, you can verify the name by opening the `./cache/modules/<module_name>/<module_name>vardefs.php` file. The second line of that file will have text like the following:

```
$GLOBALS["dictionary"]["Contact"] = array (...);
```

The parameter following "dictionary" is the same parameter you should use in the file defining the custom index. To verify duplicates against a combination of fields (i.e. duplicates will only be flagged if the values of multiple fields match those of an existing record), then simply add the desired fields to the 'fields' array in the code example.

Finally, navigate to Admin > Repair > Quick Repair and Rebuild to enable the custom index for duplicate verification when importing records in the module.

Last Modified: 11/27/2017 12:24pm

Fields

Overview

How fields interact with the various aspects of Sugar.

SugarField Widgets

The SugarField widgets, located in `./include/SugarFields/Fields/`, define the data formatting and search query structure for the various field types. They also define the rendering of fields for modules running in backward compatibility mode. When creating or overriding field widgets, developers should place their customization in `./custom/include/SugarFields/Fields/`. For information on how Sidecar renders fields, please refer to the fields section in our user interface documentation.

Creating Custom Fields

Implementation

All fields for a module are defined within vardefs. Within this definition, the type attribute will determine all of the logic applied to the field. For example, the Contacts module has a 'Do Not Call' field. In the vardefs, this field is defined as follows:

```
'do_not_call' => array (  
  'name' => 'do_not_call', // the name of the field  
  'vname' => 'LBL_DO_NOT_CALL', // the label for the field name  
  'type' => 'bool', // the fields type  
  'default' => '0', // the fields default value  
  'audited'=>true, // whether the field is audited  
  'duplicate_on_record_copy' => 'always', // whether to duplicate  
the fields value when being copied  
  'comment' => 'An indicator of whether contact can be called' //  
admin context of the field  
) ,
```

The bool type field is rendered in the UI from the `./clients/base/fields/bool/bool.js` field controller which renders the appropriate handlebars template as defined by the users current view for sidecar enabled modules. When the user saves data, the controller formats the data for the API and passes it to an endpoint. Once the data is received by the server, The SugarField definition calls any additional logic in the `apiSave` function to format the data for saving to the database. The same concept is applied in the `apiFormatField` function when retrieving data from the database to be passed back to the user interface through the API. For modules running in backward compatibility mode, the bool field is rendered using the Smarty `.tpl`) templates located in `./include/SugarFields/Fields/Bool/`.

While the vardefs define the default type for a field, this value can be overridden in the metadata of the view rendering the field. The example being that in `./custom/modules/Contacts/clients/base/views/record/record.php`, you can modify the `do_not_call` field array to point to a custom field type you have created. For more information on creating custom field types, please refer to [Creating Custom Fields](#) documentation.

Last Modified: 10/14/2016 04:12pm

Relationships

Overview

Relationships are the basis for linking information within the system. This page

explains the various aspects of relationships. For information on custom relationships, please refer to the Custom Relationships documentation.

Definitions

Relationships are initially defined in the module's vardefs file under the relationships array. For reference, you can find them using the vardef path `./modules/<module>/vardefs.php`.

Database Structure

In Sugar, most relationships are stored using a joining table. This applies to both one-to-many (1:M) relationships as well as many-to-many (M:M) relationships. An example of this is the relationship between Accounts and Opportunities where there are three tables: accounts, accounts_opportunities, and opportunities. You will find that the joining table, accounts_opportunities, will contain the fields needed in order to establish the relationship link.

The fields on the accounts_opporuntities table are listed below:

Fields	Description
id	A unique identifier for the relationship row (not typically used)
opportunitiy_id	The ID for the related opportunity record. This is named uniquely based on the relationship
account_id	The ID for the related account record. This is named uniquely based on the relationship
date_modified	The date the row was last modified
deleted	Whether or not the relationship still exists

Relationship Cache

All relationships in Sugar are compiled into the cache directory `./cache/Relationships/relationships.cache.php`. If needed, the relationships cache can be rebuilt by navigating to Admin > Repair > Rebuild Relationships.

Last Modified: 01/19/2018 10:21am

Custom Relationships

Overview

This page needs an overview

Creating Custom Relationships

Relationships are initially defined in the module's vardefs file under the relationships array. For reference, you can find them using the vardef path as follows:

```
./modules/<module>/vardefs.php
```

Custom relationships are created in a different way using the Extension Framework. The process requires two steps which are explained in the following sections:

1. Defining the Relationship MetaData
2. Defining the Relationship in the TableDictionary

Defining the Relationship MetaData

The definitions for custom relationships will be found in a path similar to:

```
./custom/metadata/<relationship name>MetaData.php
```

This file will contain the \$dictionary information needed for the system to generate the relationship. The \$dictionary array will contain the following:

Index	Type	Description
true_relationship_type	String	The relationship's structure (possible values: 'one-to-many' or 'many-to-many')
from_studio	Boolean	Whether the relationship was created in Studio
table	String	The name of the table that

		is created in the database to contain the link ids
fields	Array	An array of fields to be created on the relationship join table
indices	Array	The list of indexes to be created
relationships	Array	An array defining relationships
relationships.<rel>	Array	The array defining the relationship
relationships.<rel>.lhs_module	String	Left-hand module (should match \$beanList index)
relationships.<rel>.lhs_table	String	Left-hand table name
relationships.<rel>.lhs_key	String	The key to use from the table on the left
relationships.<rel>.rhs_module	String	Right-hand module (should match \$beanList index)
relationships.<rel>.rhs_table	String	Right-hand table name
relationships.<rel>.rhs_key	String	The key to use from the table on the right
relationships.<rel>.relationship_type	String	The relationship type, typically stored as 'many-to-many'
relationships.<rel>.join_table	String	The join table
relationships.<rel>.join_key_lhs	String	Left table key, should exist in table field definitions above
relationships.<rel>.join_key_rhs	String	Right table key, should exist in table field definitions above

MetaData Example

Creating a custom 1:M relationship between Accounts and Contacts will yield the following metadata file:

./custom/metadata/accounts_contacts_1MetaData.php

<?php

// created: 2013-09-20 15:15:47

```
$dictionary["accounts_contacts_1"] = array (
  'true_relationship_type' => 'one-to-many',
  'from_studio' => true,
  'relationships' =>
  array (
    'accounts_contacts_1' =>
    array (
      'lhs_module' => 'Accounts',
      'lhs_table' => 'accounts',
      'lhs_key' => 'id',
      'rhs_module' => 'Contacts',
      'rhs_table' => 'contacts',
      'rhs_key' => 'id',
      'relationship_type' => 'many-to-many',
      'join_table' => 'accounts_contacts_1_c',
      'join_key_lhs' => 'accounts_contacts_1accounts_ida',
      'join_key_rhs' => 'accounts_contacts_1contacts_idb',
    ),
  ),
  'table' => 'accounts_contacts_1_c',
  'fields' =>
  array (
    0 =>
    array (
      'name' => 'id',
      'type' => 'varchar',
      'len' => 36,
    ),
    1 =>
    array (
      'name' => 'date_modified',
      'type' => 'datetime',
    ),
    2 =>
    array (
      'name' => 'deleted',
      'type' => 'bool',
      'len' => '1',
      'default' => '0',
      'required' => true,
```

```

),
3 =>
array (
  'name' => 'accounts_contacts_1accounts_ida',
  'type' => 'varchar',
  'len' => 36,
),
4 =>
array (
  'name' => 'accounts_contacts_1contacts_idb',
  'type' => 'varchar',
  'len' => 36,
),
),
'indices' =>
array (
  0 =>
array (
  'name' => 'accounts_contacts_1spk',
  'type' => 'primary',
  'fields' =>
array (
  0 => 'id',
),
),
1 =>
array (
  'name' => 'accounts_contacts_1_ida1',
  'type' => 'index',
  'fields' =>
array (
  0 => 'accounts_contacts_1accounts_ida',
),
),
2 =>
array (
  'name' => 'accounts_contacts_1_alt',
  'type' => 'alternate_key',
  'fields' =>
array (
  0 => 'accounts_contacts_1contacts_idb',
),
),
),
);

```

Defining the Relationship in the TableDictionary

Once a relationship's metadata has been created, the metadata file will have a reference placed in the TableDictionary:

```
./custom/Extension/application/Ext/TableDictionary/<relationship name>.php
```

This file will contain an 'include' reference to the metadata file:

```
<?php
    include('custom/metadata/<relationship name>MetaData.php');
?>
```

TableDictionary Example

The custom 1:M relationship between Accounts and Contacts will yield the following TableDictionary file:

```
./custom/Extension/application/Ext/TableDictionary/accounts_contacts_1.php
```

```
<?php
    //WARNING: The contents of this file are auto-generated
    include('custom/metadata/accounts_contacts_1MetaData.php');
?>
```

If you have created the relationship through Studio, the files above will be automatically created. If you are manually creating the files, run a Quick Repair and Rebuild and run any SQL scripts generated. The Quick Repair and Rebuild will rebuild the file map and relationship cache as well as populate the relationship in the relationships table.

Last Modified: 10/14/2016 04:12pm

Database

Overview

All 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 or recommend database triggers or stored procedures. This design simplifies coding and testing across different database vendors. The only implementation difference across the various supported databases is column types.

Primary Keys, Foreign Keys, and GUIDs

By default, Sugar uses globally unique identification values (GUIDs) for primary keys for all database records. Sugar provides a `Sugarcrm\Sugarcrm\Util\Uuid::uuid1()` utility function for creating these GUIDs in the following format: `aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee`. The primary key's column length is 36 characters.

The GUID format and value has no special meaning (relevance) in Sugar other than the ability to match records in the database. Sugar links two records (such as an Accounts record with a Contacts record) with a specified ID in the record type relationship table (e.g. `accounts_contacts`).

Primary keys in Sugar may contain any unique string such as a GUID algorithm, a key that has some meaning (e.g. bean type first, followed by info), an external key, or auto-incrementing numbers converted to strings. Sugar chose GUIDs over auto-incrementing keys to enable easier data synchronization across databases and avoid primary-key collisions.

You can also import data from a previous system with one primary key format and make all new records in Sugar use the GUID primary key format. All keys must be stored as globally unique strings with no more than 36 characters.

Notice If multiple records between modules contain matching ids, you may experience undesired behaviors within the system.

To implement a new primary key method or to import data with a different primary key format (based on the existing GUID mechanism for new records), keep in mind the following rules of primary key behavior:

- Quote characters : Sugar expects primary keys to be string types and will format the SQL with quotes. If you change the primary key types to an integer type, SQL errors may occur since Sugar stores all ID values in quotes in the generated SQL. The database may be able to ignore this issue. MySQL running in Safe mode experiences issues, for instance.

-
- **Case sensitivity** : The ID values abc and ABC are treated the same in MySQL but represent different values in Oracle. When migrating data to Sugar, some CRM systems may use case-sensitive strings as their IDs on export. If this is the case, and you are running MySQL, you must run an algorithm on the data to make sure all of the IDs are unique. One simple algorithm is to MD5 the ID values that they provide. A quick check will let you know if there is a problem. If you imported 80,000 leads and there are only 60,000 in the system, some may have been lost due to non-unique primary keys caused by case insensitivity.
 - **Key size** : Sugar only tracks the first 36 characters in the primary key. Any replacement primary key will either require changing all of the ID columns with one of an appropriate size or to make sure you do not run into any truncation or padding issues. MySQL in some versions has had issues with Sugar where the IDs were not matching because it was adding spaces to pad the row out to the full size. MySQL's handling of char and varchar padding has changed in later versions. To protect against this, make sure the GUIDs are not padded with blanks in the database by removing any leading or trailing space characters.

Indexes

Indexes can be defined in the main or custom vardefs.php for a module in an array under the key indices. See below for an example of defining several indices:

```
'indices' => array(
    array(
        'name' => 'idx_modulename_name',
        'type' => 'index',
        'fields' => array('name'),
    ),
    array(
        'name' => 'idx_modulename_assigned_deleted',
        'type' => 'index',
        'fields' => array('assigned_user_id', 'deleted'),
    ),
),
```

The name of the index must start with idx_ and must be unique across the database. Possible values for type include primary for a primary key or index for a normal index. The fields list matches the column names used in the database.

Last Modified: 01/23/2017 02:09pm

DBManagerFactory

Overview

Learn more about the DBManagerFactory class, which allows 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 for a result set, use the `query()` and `fetchByAssoc()` methods. The `query()` method will retrieve the results while the `fetchByAssoc()` method will iterate through the results:

```
$sql = "SELECT id FROM accounts WHERE deleted = 0";
```

```
$result = $GLOBALS[ 'db' ]->query($sql);
```

```
while($row = $GLOBALS[ 'db' ]->fetchByAssoc($result) )  
{  
    //Use $row['id'] to grab the id fields value  
    $id = $row['id'];  
}
```

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 or use the `limitQuery()` method:

```
$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 the following methods from the bean class.

Select Queries

To create a select query you can use the `create_new_list_query()` method:

```
$bean = BeanFactory::newBean($module);  
  
$order_by = '';  
$where = '';  
$fields = array(  
    'id',  
    'name',  
);  
  
$sql = $bean->create_new_list_query($order_by, $where, $fields);
```

Count Queries

You can also run the generated SQL through the `create_list_count_query()` method to generate a count query:

```
$bean = BeanFactory::newBean('Accounts');  
  
$sql = "SELECT * FROM accounts WHERE deleted = 0";  
  
$count_sql = $bean->create_list_count_query($sql);
```

Last Modified: 10/14/2016 04:12pm

SugarQuery

Overview

SugarQuery provides an object-oriented approach to working with the database. This allows developers to generate the applicable SQL for a Sugar system without having to know which database backend the instance is using. SugarQuery supports all databases supported by Sugar.

Note: SugarQuery only supports Reading data from the database at this time (i.e. SELECT statements).

Setup

To use SugarQuery, simply include the SugarQuery files, and create a new SugarQuery object.

```
require_once('include/SugarQuery/SugarQuery.php');  
$sugarQuery = new SugarQuery();
```

Basic Usage

Using the SugarQuery object to retrieve records or generate SQL queries is very simple. At a minimum you need to set the Module you are working with, using the `from()` method, however, there are helper methods for just about any operation you would need in a SQL query. The methods listed below will outline the major methods you should consider utilizing on the SugarQuery object in order to

achieve your development goals.

from()

The from() method is used to set the primary module the SugarQuery object will be querying from. It is also used to set some crucial options for the query, such as whether Team Security should be used or if only non-deleted records should be queried. The following example will set the SugarQuery object to query from the Accounts module.

```
$sugarQuery->from(BeanFactory::newBean('Accounts'));
```

Arguments

| Name | Type | Required | Description |
|-----------|------------------|----------|--|
| \$bean | SugarBean Object | true | The SugarBean object for a specified module. The SugarBean object does not have to be a blank or new Bean as seen in the example above, but can be a previously instantiated SugarBean object. |
| \$options | Array | false | An associative array that can specify any of the following options: <ul style="list-style-type: none">• alias - string -• team_security - boolean - Whether or not Team Security should be added to the generated SQL query |

| | | | |
|--|--|--|--|
| | | | <ul style="list-style-type: none"> • <code>add_deleted</code> - boolean - Whether or not 'deleted' = 0 should be added to Where clause of generated SQL query |
|--|--|--|--|

Returns

SugarQuery Object

Allows for method chaining on the SugarQuery object.

select()

The example above demonstrates the most basic example of retrieving records from a module. The `select()` method can be used on the SugarQuery object to specify the specific fields you wish to retrieve from the query.

```
//Alter the Selected Fields
$sugarQuery->select(array('id','name'));
```

Arguments

| Name | Type | Required | Description |
|-----------------------|-------|----------|---|
| <code>\$fields</code> | Array | false | Sets the fields that should be added to the SELECT portion of the SQL query |

Returns

SugarQuery_Builder_Select Object

You cannot chain SugarQuery methods off of the `select()` method, however, you can use the returned Select object to modify the SELECT portion of the statement.

Review the `SugarQuery_Builder_Select` object in `./include/SugarQuery/Builder/Select.php` for additional information on usage.

where()

To add a WHERE clause to the query, use the `where()` method to generate the Where Object, and then use method chaining with the various helper methods to add conditions. To add a WHERE clause for records with the name field containing the letter "I", you could add the following code.

```
//add the where clause
$sugarQuery->where()->contains('name', 'I');
```

Arguments

None

Returns

`SugarQuery_Builder_Where` Object

Allows for method chaining on the Where Object as shown above. Review the `SugarQuery Conditions` documentation for full spectrum of `where()` method usage.

Relationships

join()

To add data from a related module to the SugarQuery, use the `join()` method. Adding to the same SugarQuery code example in this page, the following code would add the JOIN from Accounts module tables to Contacts table:

```
//add join
$sugarQuery->join('contacts');
```

Arguments

| Name | Type | Required | Description |
|--------------------------|--------|----------|------------------------------|
| <code>\$link_name</code> | String | true | The name of the relationship |

| | | | |
|-----------|-------|-------|---|
| \$options | Array | false | <p>An associative array that can specify any of the following options:</p> <ul style="list-style-type: none"> • alias - string - • relatedJoin - string - If joining to a secondary table (related to a related module), such as joining on Opportunities related to Contacts, when querying from Accounts, you can specify either the name of the relationship or the alias you specified for that relationship table. |
|-----------|-------|-------|---|

Returns

SugarQuery_Builder_Join Object

Allows for method chaining on the SugarQuery_Builder_Join Object, to add additional conditions to the WHERE clause of the SQL condition.

joinTable()

To add data from a related table to the SugarQuery, use the joinTable() method. Adding to the same SugarQuery code example in this page, the following code

would add the JOIN from Accounts module tables to the accounts_contacts table:

```
//add join
$sugarQuery->joinTable('accounts_contacts', array('alias' =>
'ac'))->on()
    ->equalsField('accounts.id','ac.account_id')
    ->equals('ac.primary_account',1);
```

Arguments

| Name | Type | Required | Description |
|--------------|--------|----------|--|
| \$table_name | String | true | The name of the database table to join. |
| \$options | Array | false | An associative array that can specify any of the following options: <ul style="list-style-type: none">• alias - string - |

Returns

SugarQuery_Builder_Join Object

Allows for method chaining on the SugarQuery_Builder_Join Object, to add additional conditions to the ON clause using the on() method.

Altering Results

Altering the result set of a query can help performance of the queries, as well as be crucial to finding the correct data. The following methods provide ways to limit the result set and change the order.

distinct()

To group the query on a field, you can use the corresponding distinct() method.

```
//add group by
$sugarQuery->distinct(true);
```

Arguments

| Name | Type | Required | Description |
|---------|---------|----------|--|
| \$value | Boolean | true | Set whether or not the DISTINCT statement should be added to the query |

Returns

Current SugarQuery Object

Allows for method chaining on the SugarQuery Object.

limit()

To limit the results of the query, you can use the limit() method.

```
//set the limit
$sugarQuery->limit(10);
```

Arguments

| Name | Type | Required | Description |
|----------|---------|----------|---|
| \$number | Integer | true | The max amount of rows that should be returned by the query |

Returns

Current SugarQuery Object

Allows for method chaining on the SugarQuery Object.

offset()

Adding a limit to the query limits the rows returned, however when doing so, you may need to alter the offset of the query to account for pagination or access other portions of the result set. To set an offset, you can use the offset() method.

```
//set the offset
$sugarQuery->offset(5);
```

Arguments

| Name | Type | Required | Description |
|----------|---------|----------|---|
| \$number | Integer | true | The offset amount of rows, or starting point, of the result |

Returns

Current SugarQuery Object

Allows for method chaining on the SugarQuery Object.

orderBy()

To order the query on a field, you can use the corresponding orderBy() method. This method can be called multiple times, to add multiple fields to the order by clause of the query.

```
//add group by
$sugarQuery->orderBy('account_type');
```

Arguments

| Name | Type | Required | Description |
|----------|--------|----------|------------------------------------|
| \$column | String | true | The field you want the query to be |

| | | | |
|-------------|--------|-------|--|
| | | | grouped on |
| \$direction | String | false | Sets the direction of sorting. Must be 'ASC' or 'DESC'. The default is 'DESC'. |

Returns

Current SugarQuery Object

Allows for method chaining on the SugarQuery Object.

Execution

Once you have the SugarQuery object setup and configured for your statement, you will want to retrieve the results of the query, or simply get the generated query for the object. The following methods are used for executing the SugarQuery object.

execute()

To query the database for a result set, you will use the execute() method. The execute() method will retrieve the results and return them as a raw string, db object, json, or an array depending on the \$type parameter. By default, results are returned as an array. An example of fetching records from an account is below:

```
//fetch the result
$result = $sugarQuery->execute();
```

The execute() function will return an array of results that you can iterate through as shown below:

```
Array
(
    [0] => Array
        (
            [id] => f39593da-3f88-3059-4f18-524b4d23d07b
            [name] => International Art Inc
        )
)
```

Note: An empty resultset will return an empty array.

Arguments

| Name | Type | Required | Description |
|--------|--------|----------|---|
| \$type | String | false | <p>How you want the results of the Query returned. Can be one of the following options:</p> <ul style="list-style-type: none">• raw - Returns the compiled SQL statement. Does not run the query. See compileSql() method below• db - Returns the result directly from the DatabaseManager resource• array - Default• json - Returns the results encoded as JSON |

Returns

Default: Array. See above argument details for details on other Return options.

compileSql()

If you want to log the query being generated or want to output the query without running it during development, the compileSql() method is what should be used.

```
//get the compiled SQL query
$query = $sugarQuery->compileSql();
```

Arguments

No arguments

Returns

String

The compiled SQL Query built by the SugarQuery object.

Last Modified: 01/08/2018 01:02pm

SugarQuery Conditions

Overview

Learn about the various methods that can be utilized with SugarQuery to add conditional statements to a query.

Where Clause

Manipulating the WHERE clause of a SugarQuery object is crucial for getting the correct results. To create a WHERE clause on the query, use the where() method on the SugarQuery object, as outlined in the SugarQuery documentation. Once you have the Where Object, you can utilize the following methods on the Where Object to add conditional statements.

equals() | notEquals()

Used to equate a field to a given value. Wildcards will not work with this function, as it is looking for an exact match.

```
//add equals
$sugarQuery->where()->equals('name', 'Test');
```

```
//add Not Equals
```

```
$SugarQuery->where()->notEquals('name','Tester');
```

Arguments

| Name | Type | Required | Description |
|---------|--------|----------|--|
| \$field | String | true | The field you are checking against |
| \$value | String | true | The value the field should be equal to |

Returns

SugarQuery_Builder_Where Object

Allows for method chaining on the Where Object to add additional conditions.

equalsField() | notEqualsField()

Used to equate a field to another field in the result set.

```
//add an Equals Field statement  
$SugarQuery->where()->equalsField('industry','account_type');
```

```
//add a Not Equals Field statement  
$SugarQuery->where()->notEqualsField('name','account_type');
```

Arguments

| Name | Type | Required | Description |
|---------|--------|----------|---|
| \$field | String | true | The field you are checking against |
| \$field | String | true | The other field you want the first field to be equal to |

Returns

SugarQuery_Builder_Where Object

Allows for method chaining on the Where Object to add additional conditions.

isEmpty() | isEmpty()

Used to check if a field is or isn't empty.

```
//add an isEmpty statement
$SugarQuery->where()->isEmpty('industry');

//add an isEmpty statement
$SugarQuery->where()->isEmpty('name');
```

Arguments

| Name | Type | Required | Description |
|---------|--------|----------|------------------------------------|
| \$field | String | true | The field you are checking against |

Returns

SugarQuery_Builder_Where Object

Allows for method chaining on the Where Object to add additional conditions.

isNull() | notNull()

Used to check if a field is or isn't equal to NULL.

```
//add an isNull statement
$SugarQuery->where()->isNull('industry');
//add a notNull statement
$SugarQuery->where()->notNull('name');
```

Arguments

| Name | Type | Required | Description |
|---------|--------|----------|------------------------------------|
| \$field | String | true | The field you are checking against |

Returns

SugarQuery_Builder_Where Object

Allows for method chaining on the Where Object to add additional conditions.

contains() | notContains()

Used to check if a field has or doesn't have a specified string in it's value. Utilizes the LIKE statement, and wildcards on both sides of the provided string.

```
//add an isNull statement
$SugarQuery->where()->contains('name','Test');
```

```
//add a notNull statement
$SugarQuery->where()->notContains('industry','Test');
```

Arguments

| Name | Type | Required | Description |
|---------|--------|----------|---|
| \$field | String | true | The field you are checking against |
| \$value | String | true | The string being searched for in the value of the field |

Returns

SugarQuery_Builder_Where Object

Allows for method chaining on the Where Object to add additional conditions.

starts() | ends()

Similar to the above contains() method, these methods use the LIKE statement in the SQL query and wildcards for searching for a specified string in the field's value. However the starts() and ends() methods only wildcard the right side and the left side, respectively. The following example code, would be searching for records where the Name field starts with A, and ends with E.

```
//add an starts and ends statement
$SugarQuery->where()->starts('name','A')->ends('name','e');
```

Arguments

| Name | Type | Required | Description |
|---------|--------|----------|---|
| \$field | String | true | The field you are checking against |
| \$value | String | true | The string being searched for in the value of the field |

Returns

SugarQuery_Builder_Where Object

Allows for method chaining on the Where Object to add additional conditions.

in() | notIn()

Used to check if a field's value is or isn't one of a set of specified values. The following examples looks for records where the industry field is in a list of values, and not in a separate list of values.

```
$values = array(
    'Support',
    'Sales',
    'Engineering'
);
```

```
//add in statement
$SugarQuery->where()->in('industry',$values);
```

```
$values = array(
```

```

        'Marketing',
        'Accounting'
    );

//add NotIn Statement
$SugarQuery->where()->notIn('industry',$values);

```

Arguments

| Name | Type | Required | Description |
|----------|--------|----------|--|
| \$field | String | true | The field you are checking |
| \$values | Array | true | The array of values which the field is being checked against |

Returns

SugarQuery_Builder_Where Object

Allows for method chaining on the Where Object to add additional conditions.

between()

Used primarily for numeric type fields, to check if the value is greater than the minimum number specified and less than the maximum number specified. The following code would check for records where the employees field is between 50 and 1000.

```

//add Between statement
$SugarQuery->where()->between('employees',50,1000);

```

Arguments

| Name | Type | Required | Description |
|---------|--------|----------|------------------------------------|
| \$field | String | true | The field you are checking against |
| | | | |

| | | | |
|-------|--------|------|--|
| \$min | Number | true | The lowest number the field's value should be |
| \$max | Number | true | The highest number the field's value should be |

Returns

SugarQuery_Builder_Where Object

Allows for method chaining on the Where Object to add additional conditions.

lt() | lte() | gt() | gte()

These methods are primarily for numeric fields, to check if a field's value is less than (<), less than or equal (<=), greater than (>), or greater than or equal (>=) to a specified value.

```
//Add Less Than Statement
$SugarQuery->where()->lt('gross_revenue',1000000);
```

```
//Add Less Than or Equal to Statement
$SugarQuery->where()->lte('net_revenue', '500000');
```

```
//Add Greater Than Statement
$SugarQuery->where()->gt('gross_revenue',500000);
```

```
//Add Greater Than or Equal to Statement
$SugarQuery->where()->gte('net_revenue',100000);
```

Arguments

| Name | Type | Required | Description |
|---------|--------|----------|------------------------------------|
| \$field | String | true | The field you are checking against |
| \$value | Number | true | The numeric value for comparison |

Returns

SugarQuery_Builder_Where Object

Allows for method chaining on the Where Object to add additional conditions.

dateRange()

Used to check if a field's value is between a preset date range from the current time. See the TimeDate documentation on the available date range keys.

```
//add DateRange statement
$SugarQuery->where()->dateRange('date_modified','last_30_days');
```

Arguments

| Name | Type | Required | Description |
|---------|--------|----------|--|
| \$field | String | true | The field you are checking against |
| \$value | String | true | The string specifying the date range key that will be used for comparison.
Example
'next_7_days' |

Returns

SugarQuery_Builder_Where Object

Allows for method chaining on the Where Object to add additional conditions.

dateBetween()

To group the query on a field, you can use the corresponding groupBy() method. This method can be called multiple times, to add multiple fields to the grouping of the query.

```
//add group by
$SugarQuery->where()->dateBetween('date_created',array('2016-01-01','2016-03-01'));
```

Arguments

| Name | Type | Required | Description |
|---------|--------|----------|--|
| \$field | String | true | The field you are checking against |
| \$value | Array | true | An array containing the minimum date in the first key, and the maximum date in the second. |

Returns

SugarQuery_Builder_Where Object

Allows for method chaining on the Where Object to add additional conditions.

Combinations

Now that you have reviewed all of the available conditional statements for SugarQuery, you may want to combine them using AND and OR all within the same query. By default when the where() method is called, chained conditional methods will be added with AND to the where clause. You can specify an OR where clause on the main SugarQuery object by using the orWhere() method, which works the same as the where() method, just adds conditional statements with OR instead. The following methods allow for adding internal AND and OR logic to conditional statements on the Where Object.

queryAnd()

To start a group of conditional statements that should all evaluate to True, use the queryAnd() method. For example, if you want to query for Accounts, where the name contains 'Test' AND description contains 'Test', you might use the following code:

```
$SugarQuery = new SugarQuery();
$SugarQuery->select(array('name'));
$SugarQuery->from(BeanFactory::newBean('Accounts'));

//Using queryAnd
$SugarQuery->where()->queryAnd()->contains('name','Test')->contains('description','Test');
```

The above use of queryAnd() method isn't entirely needed, as the main Where Object would be using AND for all conditions anyway, but it does group the two conditions inside of their own parenthesis in the compiled query, as shown below, to demonstrate how it can be used for altering query logic.

```
SELECT accounts.name name FROM accounts WHERE accounts.deleted = 0
AND (accounts.name LIKE '%Test%' AND accounts.description LIKE
'%Test%')
```

queryOr()

To start a group of conditional statements that should evaluate to true, if any condition is true, you can use the queryOr() method. For example, if you want to query for Accounts, where the name contains 'Test' or where the description contains 'Test', you might use the following code:

```
$SugarQuery = new SugarQuery();
$SugarQuery->select(array('name'));
$SugarQuery->from(BeanFactory::newBean('Accounts'));

//Using queryOr
$SugarQuery->where()->queryOr()->contains('name','Test')->contains('description','Test');
```

This will group the two conditions inside of their own parenthesis in the compiled query, as shown below, so that if either of the conditions are True, it will return a record.

```
SELECT accounts.name name FROM accounts WHERE accounts.deleted = 0
AND (accounts.name LIKE '%Test%' OR accounts.description LIKE
'%Test%')
```

Last Modified: 11/27/2017 12:24pm

Advanced Techniques

Overview

Learn about some of the advanced methods that SugarQuery has to offer, that are not as commonly used.

Get First Record

Getting the first record in a result set, can be accomplished by using the `limit()` method. The `getOne()` method is similar in that it gets the first record, but it also returns the first piece of data for that record.

`getOne()`

Get the first piece of data on the first record returned by the generated query. In this example we want the 'name' from the Account with a given ID.

```
$SugarQuery = new SugarQuery();
$SugarQuery->select(array('name'));
$SugarQuery->from(BeanFactory::newBean('Accounts'));
$SugarQuery->where()->equals('id',$id);

//Get the Name of the account
$accountName = $SugarQuery->getOne();
```

Aggregates

`setCountQuery()`

Currently the only method available for creating an aggregate column, is the `setCountQuery()` method on the `SugarQuery_Builder_Select` Object. You can add this method to your `select()` method chain, to add `count(0)` as `record_count` to the SQL `SELECT` statement.

```
$SugarQuery = new SugarQuery();
$SugarQuery->select(array('name'))->setCountQuery();
$SugarQuery->from(BeanFactory::newBean('Accounts'));
$SugarQuery->groupByRaw('accounts.name');
```

The above example will output the following SQL query:

```
SELECT  accounts.name name, count(0) AS record_count FROM accounts
```

GROUP BY accounts.name

Arguments

No arguments

Returns

SugarQuery_Builder_Select Object

Allows for method chaining on the Select Object.

Unions

Unions allow joining multiple queries with the same selected fields to be combined during output. You can use Unions in SugarQuery by using the union() method.

union()

To add a union, you can use the corresponding union() method. The example below will join two SQL queries:

```
//fetch the bean of the module to query
$bean = BeanFactory::newBean('Accounts');

//specify fields to fetch
$fields = array(
    'id',
    'name'
);

//create first query
$sql1 = new SugarQuery();
$sql1->select($fields);
$sql1->from($bean, array('team_security' => false));
$sql1->Where()->in('account_type', array('Customer'));

//create second query
$sql2 = new SugarQuery();
$sql2->select($fields);
```

```

$sq2->from($bean, array('team_security' => false));
$sq2->Where()->in('account_type', array('Investor'));

//create union
$sqUnion = new SugarQuery();
$sqUnion->union($sql);
$sqUnion->union($sq2);
$sqUnion->limit(5);

```

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

```

Arguments

| Name | Type | Required | Description |
|----------|------------|----------|---|
| \$select | SugarQuery | true | The SugarQuery Object you wish to add to the UNION query |
| \$all | Boolean | false | Whether to use UNION ALL or just UNION in the query. Default is TRUE. |

Returns

SugarQuery_Builder_Union Object

Allows for method chaining on the Union Object.

Raw Methods

The SugarQuery Object has a few helper methods that allow raw SQL statement parts to be passed into. This allows for more complex statements or edge case scenarios where a helper function may not have met the requirements for the query.

whereRaw()

To add to the WHERE clause of SugarQuery Object with raw SQL syntax, you can utilize the whereRaw() method. This method will append the specified statement, to the WHERE clause using an AND operator, and will wrap the entire statement in parenthesis. The following is an example use with the output :

```
$SugarQuery = new SugarQuery();  
$SugarQuery->select(array('name'));  
$SugarQuery->from(BeanFactory::newBean('Accounts'));  
$SugarQuery->whereRaw("name LIKE '%T%'");
```

The above example will output the following SQL query:

```
SELECT accounts.name name FROM accounts WHERE accounts.deleted = 0  
AND (name LIKE '%T%')
```

Arguments

| Name | Type | Required | Description |
|-------|--------|----------|---|
| \$sql | String | true | The WHERE clause SQL to be appended to the where clause on the SugarQuery object. All conditions passed in are wrapped in parenthesis and appended using AND (if other conditions exist on where clause). |

Returns

SugarQuery_Builder_Where Object

Allows for method chaining on the Where Object.

joinRaw()

To add a JOIN statement that may not align with the relationships stated in Vardefs, you can use the joinRaw() method. The following example, joins to the contacts_cstm table from Accounts, on a custom relate field.

```
$SugarQuery = new SugarQuery();  
$SugarQuery->select(array('name'));  
$SugarQuery->from(BeanFactory::newBean('Accounts'));  
$SugarQuery->joinRaw("INNER JOIN contacts_cstm ccl ON accounts.id =  
ccl.account_id_c");
```

The above example will output the following SQL query:

```
SELECT accounts.name name FROM accounts INNER JOIN contacts_cstm ccl  
ON accounts.id = ccl.accounts_id_c WHERE accounts.deleted = 0
```

Arguments

| Name | Type | Required | Description |
|-----------|--------|----------|--|
| \$sql | String | true | The JOIN statement to be added to the SQL query |
| \$options | Array | false | An array containing any of the following options: <ul style="list-style-type: none">• alias - String - The alias used for the JOIN |

Returns

SugarQuery_Builder_Join Object

Allows for method chaining on the SugarQuery_Builder_Join Object.

groupByRaw()

To add multiple fields to the GROUP BY statement on the SugarQuery Object, it may sometime's be easiest to use the groupByRaw() method.

```
$SugarQuery->groupByRaw("accounts.account_type,accounts.industry");
```

Arguments

| Name | Type | Required | Description |
|-------|--------|----------|---|
| \$sql | String | true | The GROUP BY statement, without the GROUP BY keyword. |

Returns

SugarQuery Object

Allows for method chaining on the SugarQuery Object.

orderByRaw()

Using the oderBy() method only allows for adding a single field to the SugarQuery object at a time. In some cases, you might consider using the orderByRaw() method to add multiple fields or the entire ORDER BY statement to the SugarQuery object.

```
$SugarQuery->orderByRaw("accounts.name DESC, accounts.date_modified");
```

Arguments

| Name | Type | Required | Description |
|-------|--------|----------|---|
| \$sql | String | true | The ORDER BY statement, without the ORDER BY keyword. |

Returns

SugarQuery Object

Allows for method chaining on the SugarQuery Object.

havingRaw()

Using the havingRaw() method allows for adding a having statement to the SugarQuery object.

```
$SugarQuery->havingRaw("record_count >= 3");
```

Arguments

| Name | Type | Required | Description |
|-------|--------|----------|---|
| \$sql | String | true | The HAVING statement, without the HAVING keyword. |

Returns

SugarQuery Object

Allows for method chaining on the SugarQuery Object.

Last Modified: 11/27/2017 12:24pm

Architecture

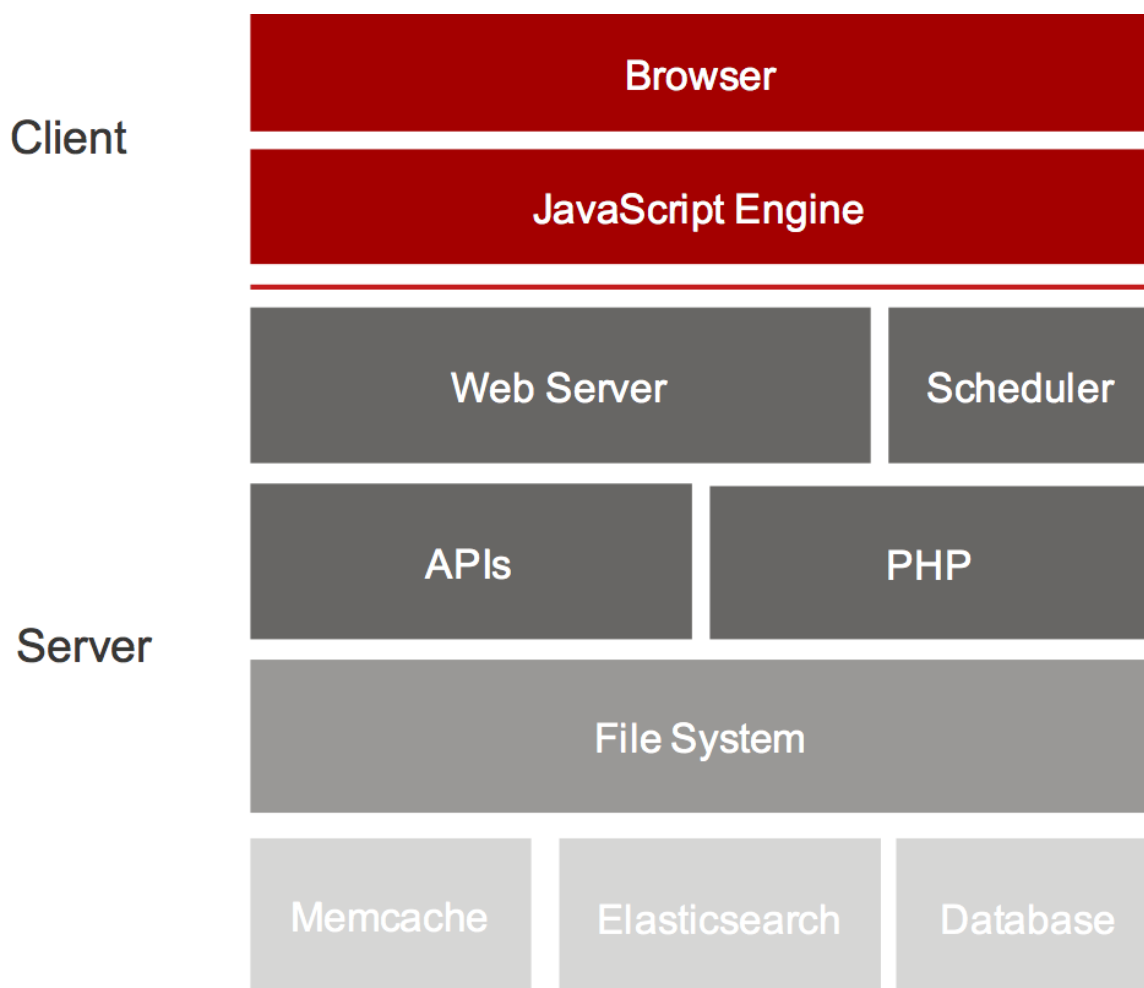
Overview

This section of Sugar's Developer Guide begins with a high-level overview of the Sugar platform's architecture and contains documentation on granular concepts in Sugar such as logic hooks, caching, logging, extensions, job queue, and more.

Please continue to the bottom of this page or use the navigation on the left to explore the related content.

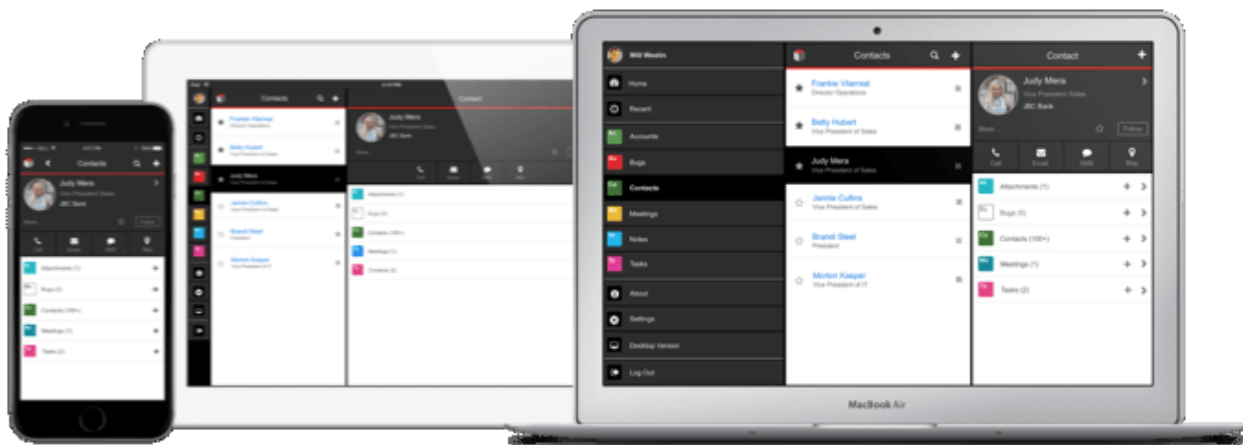
Platform

Sugar is built on open standards and technology such as HTML5, PHP, and JavaScript, and runs on a variety of free and open-source technology like Linux, MySQL, and Elasticsearch. The Sugar platform also supports common proprietary databases such as Oracle, IBM DB2, and Microsoft SQL Server.



All of Sugar's customers and partners have access to source code that they can choose to deploy on-premise or utilize Sugar's cloud service for a SaaS deployment.

Out of the box, Sugar uses a consistent platform across all clients and devices (e.g. mobile, web, plug-ins, etc.).

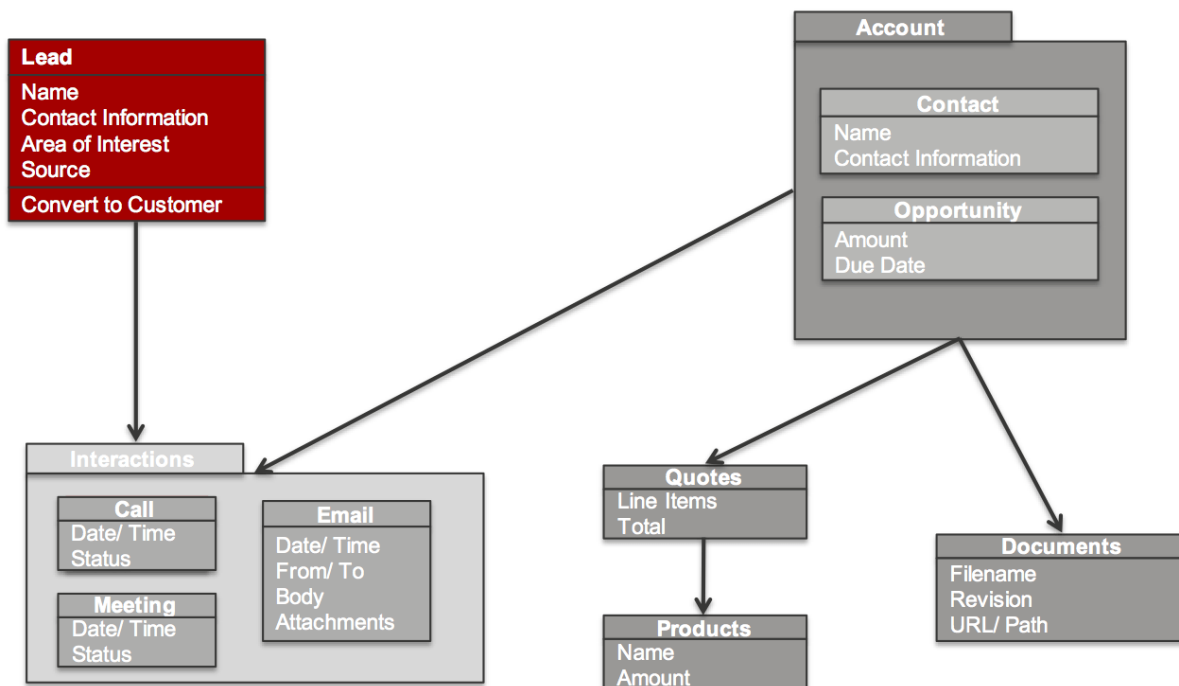


Front-End Framework

Our clients are based on a front-end framework called Sidecar. Sidecar is built on open source technology: Backbone.js, jQuery, Handlebars.js, and Bootstrap. The Sidecar framework provides a responsive UI (to support a variety of form factors) and uses modern, single-page client architecture. Sugar clients connect to Sugar server application via our client REST API. The REST API is implemented in PHP and drives server-side business logic and interacts with a database. If it can be accomplished via one of our clients, then its equivalent functionality can be accomplished using our REST API.

The Sugar platform uses modules. Modules are a vertically integrated application component that is traditionally organized around a single feature or record type (or underlying database table). For example, contact records are managed via a Contacts module that contains all the business logic, front-end interface definitions, REST APIs, data schema, and relationships with other modules.

Custom modules can be created and deployed as needed in order to add new features to a Sugar application instance.



Metadata

Sugar's modules are defined primarily using Metadata. There are two types of metadata definitions within Sugar: Vardefs, which define the data model for Sugar modules; and Viewdefs, which define the user interface components that are used with a module.

Sugar Metadata is implemented as PHP files that can be modified directly by a Sugar Developer making filesystem changes, or indirectly through the use of Sugar Studio and Module Builder by a Sugar Administrator.

Metadata allows you to configure solutions instead of having to write countless lines of custom code in order to implement common customizations such as adding custom fields, calculated values, and changing user interface layouts.

Extensions

Beyond metadata, Sugar is highly customizable and includes an extensive Extensions Framework that provides Sugar Developers the capability to contribute to pre-defined extension points within the application in a way that is upgrade-safe and will not conflict with other customizations that exist in the system.

Last Modified: 03/15/2018 05:01pm

Autoloader

Overview

The autoloader is an API that allows the unified handling of customizations and customizable metadata while reducing the number of filesystem accesses and improving performance.

SugarAutoLoader

The SugarAutoLoader class, located in `./include/utils/autoloader.php`, keeps a map of files within the Sugar directory that may be loaded. 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.

Included File Extensions

The autoloader will only map files with the following extensions:

- bmp
- css
- gif
- hbs
- html
- ico
- jpg
- js
- less
- override
- php
- png
- tif
- tpl
- xml

*All other file extensions are excluded from the mapping.

Class Loading Directories

The autoloader will scan and autoload classes in the following directories:

- ./clients/base/api/
- ./data/duplicatecheck/
- ./data/Relationships/
- ./data/visibility/
- ./include/
- ./include/api/
- ./include/CalendarEvents/
- ./include/SugarSearchEngine/
- ./modules/Calendar/
- ./modules/Mailer/

Ignored Directories

The following directories in Sugar are ignored by the autoloader mapping:

- ./idea/
- ./cache/
- ./custom/backup/
- ./custom/blowfish/
- ./custom/Extension/
- ./custom/history/
- ./custom/modulebuilder/
- ./docs/
- ./examples/
- ./portal/
- ./tests/
- ./upload/
- ./vendor/bin/
- ./vendor/HTMLPurifier/
- ./vendor/log4php/
- ./vendor/nusoap/
- ./vendor/pclzip/
- ./vendor/reCaptcha/
- ./vendor/ytree/

Initializing the AutoLoader

The `init()` function will initialize the loader, register `autoload()` as an autoloader function, and 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 during a Quick Repair and Rebuilds (Admin > Repair > Quick Repair and Rebuild), module installs, and can rebuild 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, however, be aware that this will update the metadata hash and force a reload for users. You should also note that if you are hosting your instance on Sugar's cloud service, you are subject to the Module Loader Restrictions.

The `buildCache` Function

To programmatically rebuild the file map, you can use the function as shown below:

```
SugarAutoloader::buildCache();
```

Last Modified: 03/15/2018 05:18pm

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: 10/14/2016 04:12pm

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 `existingCustomOne()`, 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 its 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: 10/14/2016 04:12pm

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: 10/14/2016 04:12pm

Metadata API

Overview

Methods to load metadata for the AutoLoader API.

Metadata Loading

For the specific sets of metadata, such as detailviewdefs, editviewdefs, 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: 10/14/2016 04:12pm

Caching

Overview

Much of Sugar's user interface is built dynamically using a combination of templates, metadata and language files. A file caching mechanism improves the performance of the system by reducing the number of static metadata and language files that need to be resolved at runtime. This cache directory stores the

compiled files for JavaScript files, Handlebars templates, and language files.

In a stock instance, the cache is located in the `./cache/` directory. If you would like to move this directory to a new location, you can update the config parameter `cache_dir` in `config.php` or `config_override.php` to meet your needs. It is not advisable to move the cache to another network server as it may impact system performance.

Developer Mode

To prevent caching while developing, a developer may opt to turn on Developer Mode by navigating to Admin > System Settings > Advanced > Developer Mode. This will disable caching so that developers can test code-level customizations without the need to manually rebuild the cache, which is especially helpful when developing templates, metadata, or language files. The system automatically refreshes the file cache. Make sure to deactivate Developer Mode after completing customizations because this mode degrades system performance.

Last Modified: 10/14/2016 04:12pm

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 Sugar's cloud service.

Upload Extensions

By default, several extension types are restricted due to security issues. Any files uploaded with these extensions will have '.txt' appended to it. The restricted extensions are listed below:

- php
- php3
- php4
- php5
- pl
- cgi
- py
- asp
- cfm
- js
- vbs
- html
- htm

You can add or remove extensions to this list by modifying sugar configuration setting for `upload_badext`. You should note that this setting cannot be modified for instances hosted on Sugar's cloud service.

How Files Are Stored

Note Attachments

When a file is uploaded to Sugar attached to a note, the file will be moved to the upload directory with a GUID name matching that of the notes id. The attributes of the file, such as `filename` and `file_mime_type`, will be stored in the note record.

The SQL to fetch information about a notes attachment is shown below:

```
SELECT filename, file_mime_type FROM notes;
```

Email Attachments

Email attachments are stored the same way as note attachments. When an email is

imported to Sugar, the file will be moved to the upload directory with a GUID file name matching that of the notes id. The attributes of the file, such as filename and file_mime_type, will be stored in the note record and the note will have a parent_type of 'Emails'. This relates the attachment to the 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 shown below:

```
SELECT * FROM upgrade_history;
```

CSV Imports

When importing records into Sugar, the most recent uploaded CSV file is stored in the upload directory as `IMPORT_<module>_<user id>`. Once the import has been run, the results of the import are stored in `./upload/import/` directory using a predefined format using the current 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: 03/15/2018 07:33pm

Working with File Uploads

Overview

The UploadFile class handles the various tasks when uploading a file.

Retrieving a Files Upload Location

To retrieve a files upload path, you can use the `get_upload_path` method and pass in the file's GUID id.

```
require_once('include/upload_file.php');
UploadFile::get_upload_path($file_id);
```

This method will normally return the path as:

```
upload://1d0fd9cc-02e5-f6cd-1426-51a509a63334
```

Retrieving a Files Full File System Location

To retrieve a files full system path 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-1426-51a509a63334
```

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: 10/14/2016 04:12pm

Mailer Factory

Overview

The Mailer Factory, located in `./modules/Mailer/MailerFactory.php`, helps developers generate outbound mailers for the system account as well as individual user accounts. The Mailer Factory is a replacement for SugarPHPMailer which is now deprecated.

Mailers

There are two types of outbound mailers: System and User. The follow sections will outline how to use each.

System Mailer

The system outbound mailer can be set using the `getSystemDefaultMailer` method. This will set the mailer to use the system outbound email account.

Example

```
$mailer = MailerFactory::getSystemDefaultMailer();
```

User Mailer

The user outbound mailer can be set using the `getMailerForUser` method. This will set the mailer to use the outbound email account for a specific user.

Example

```
$user = BeanFactory::getBean("Users", 1);  
$mailer = MailerFactory::getMailerForUser($user);
```

Populating the Mailer

Setting the Subject

To set the email subject, use the `setSubject` method. It accepts a plain text string.

Example

```
$mailer->setSubject("Test Mail Subject");
```

Setting the Body

Depending on your email type, you can use the `setTextBody` and/or `setHtmlBody` methods respectively to populate the content of the email body.

Example

```
// Text Body  
$mailer->setTextBody("This is a text body message");  
  
// HTML Body  
$mailer->setHtmlBody("This is an <b>HTML</b> body message. <br> You  
can use html tags.");
```

Note: The email HTML body is not necessary if you have populated the text body.

Adding Recipients

To add recipients to your email, you can use the `addRecipientsTo`, `addRecipientsCc`, or `addRecipientsBcc` methods. These methods require an `EmailIdentity` object as a parameter.

Example

```
$mailer->addRecipientsTo(new EmailIdentity('user1@yourcompany.crm',  
'User 1'));  
$mailer->addRecipientsCc(new EmailIdentity('user2@yourcompany.crm',
```

```
'User 2')));  
$mailer->addRecipientsBcc(new EmailIdentity('user3@yourcompany.crm',  
'User 3')));
```

Clearing Recipients

You can clear the current recipients specified in the mailer by using the `clearRecipients` method.

Example

```
$to = true;  
$cc = true;  
$bcc = true;  
  
$mailer->clearRecipients($to, $cc, $bcc);
```

Adding Attachments

To add attachments, use the `addAttachment` method.

Example

```
$path = "/path/to/your/document";  
$mailer->addAttachment(new Attachment($path));
```

Sending Emails

Once your email is populated, you can send it using the `send` method. The `send` method will return the content of the mail. If the Mailer Factory experiences an error, it will throw an exception. It is highly recommended to use a try and catch when sending emails.

Example

```
$mailSubject = "Test Mail Subject";  
$mailHTML = "<h1>SugarCRM</h1><br> Test body message";  
$mailTo = array(  
    0 => array(  
        'name' => 'Test User',  
        'email' => 'test@yourcompany.crm',
```

```

    ),
    1 => array(
        'name' => 'Other Recipient',
        'email' => 'email@addres'
    )
);

$mailAttachment = "/path/to/pdf/files/document.pdf";

try {
    $mailer = MailerFactory::getSystemDefaultMailer();
    $mailTransmissionProtocol =
$mailailer->getMailTransmissionProtocol();
    $mailer->setSubject($mailSubject);
    $body = trim($mailHTML);
    $textOnly = EmailFormatter::isTextOnly($body);
    if ($textOnly) {
        $mailer->setTextBody($body);
    } else {
        $textBody = strip_tags(br2nl($body)); // need to create the
plain-text part
        $mailer->setTextBody($textBody);
        $mailer->setHtmlBody($body);
    }
    $mailer->clearRecipients();
    foreach ($mailTo as $mailTo) {
        $mailer->addRecipientsTo(new \EmailIdentity($mailTo['email'],
$mailTo['name']));
    }
    $mailer->addAttachment(new \Attachment($mailAttachment));
    $result = $mailer->send();
    if ($result) {
        // $result will be the body of the sent email
    } else {
        // an exception will have been thrown
    }
} catch (MailerException $me) {
    $message = $me->getMessage();
    switch ($me->getCode()) {
        case \MailerException::FailedToConnectToRemoteServer:
            $GLOBALS["log"]->fatal("BeanUpdatesMailer :: error sending
email, system smtp server is not set");
            break;
        default:
            $GLOBALS["log"]->fatal("BeanUpdatesMailer :: error sending
e-mail (method: {$mailTransmissionProtocol}), (error: {$message})");
    }
}

```

```
        break;
    }
}
```

Last Modified: 11/27/2017 12:24pm

Logging

Overview

The SugarLogger class, located in `./include/SugarLogger/SugarLogger.php`, allows for developers and system administrators to log system events to a log file. Sugar then determines which events to write to the log based on the system's Log Level, set in Admin > System Settings.

Log Levels

The logging levels in order of most verbose to least are as follows:

- Debug : Logs events that help in debugging the application
- Info : Logs informational messages and database queries
- Warn : Logs potentially harmful events
- Error : Logs error events in the application
- Fatal : (Default) Logs severe error events that may cause the application to abort
- Security : Logs events that may compromise the security of the application
- Off : The logger will not log any events

When you specify a logging level, the system will record messages for the specified level as well as all higher levels. For example, if you specify "Error", the system records all Error, Fatal, and Security messages.

Note: The verbosity of higher levels of logging can impact the performance of your Sugar instance. We do not recommend leaving the log level any higher than the default Fatal level unless actively troubleshooting an issue.

When you are not troubleshooting Sugar, the log level should be set to Fatal to ensure that your environment is not wasting unnecessary resources to write 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 `_ppLogger()` 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. For example, if the Sugar log was named "sugarcrm.log", it will then be renamed "sugarcrm_1.log". The next log rotation after that would create "sugarcrm_2.log". This will occur until the maxLogs setting has been met. Once met, the log rollover will start over.

Last Modified: 10/14/2016 04:12pm

Creating Custom Loggers

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'
        )
        {
            LogManager::setLogger('default', 'FirePHPLogger');
        }
    }

    /** see LoggerTemplate::log() */
    public function log($level, $message)
    {
        // change to a string if there is just one entry
        if ( is_array($message) && count($message) == 1 )
        {
            $message = array_shift($message);
        }

        switch ($level)
        {
            case 'debug':
                FB::log($message);
            }
        }
    }
}
```

```
        break;
    case 'info':
        FB::info($message);
        break;
    case 'deprecated':
    case 'warn':
        FB::warn($message);
        break;
    case 'error':
    case 'fatal':
    case 'security':
        FB::error($message);
        break;
    }
}
}
```

The only method that needs to be implemented by default is the `log()` method, which writes the log message to the backend. You can specify which log levels this backend can use in the constructor by calling the `LoggerManager::setLogger()` method and specifying the level to use for this logger in the first parameter; passing 'default' makes it the logger for all logging levels.

You will then specify your default logger as 'FirePHP' in your `config_override.php` file.

```
$sugar_config['logger']['default'] = 'FirePHP';
```

Last Modified: 10/14/2016 04:12pm

Logic Hooks

Overview

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

Hook Definitions

Logic hook actions and trigger events are defined in the hook definition.

Locations

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

| Path | Description |
|--|--|
| <code>./custom/Extension/modules/<module>/Ext/LogicHooks/<file>.php</code> | 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. |
| <code>./custom/modules/<module>/logic_hooks.php</code> | 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. |
| <code>./custom/Extension/application/Ext/LogicHooks/<file>.php</code> | 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. |
| <code>./custom/modules/logic_hooks.php</code> | 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 to append the action to |
| process_index | Integer | The order of action execution |
| description | String | A short description of the hook action |
| file_path | String | The path to the logic hook file in the <code>./custom/</code> directory, or null if using namespaces |
| class_name | String | The name of the logic hook action class including any namespaces |
| method_name | String | The name of the logic hook action method |

Your definition should resemble the code block below:

```
<?php

$hook_version = 1;

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

Hook Method

The hook action class can be located anywhere you choose. We recommended grouping the hooks with the module they are associated within the `./custom/` directory. You can create a hook action method either with a namespace or without.

Namespaced Hooks

As of 7.7, developers can create namespaced logic hooks. When using namespaces, the file path in `./custom/` will be automatically built out when using the corresponding namespace. The filename defining the class must match the class name exactly to allow the autoloader to find the class definition.

| Namespace | File Path |
|--|-------------------------------------|
| <code>Sugarcrm\Sugarcrm\custom</code> | <code>./custom/</code> |
| <code>Sugarcrm\Sugarcrm\custom</code> | <code>./custom/src/</code> |
| <code>Sugarcrm\Sugarcrm\custom\any\path</code> | <code>./custom/any/path/</code> |
| <code>Sugarcrm\Sugarcrm\custom\any\path</code> | <code>./custom/src/any/path/</code> |

`./custom/Extension/modules/Accounts/Ext/LogicHooks/<file>.php`

```
<?php
```

```
    $hook_array['before_save'][] = array(
        1,
        'Hook description',
        null,
        'Sugarcrm\Sugarcrm\custom\modules\Accounts\className',
        'methodName'
    );
```

```
?>
```

`./custom/modules/Accounts/className.php`

```
<?php
```

```
namespace Sugarcrm\Sugarcrm\custom\modules\Accounts;
```

```
class className
{
    function methodName($bean, $event, $arguments)
    {
        //logic
    }
}
```

```
?>
```

The logic hook method signature may contain different arguments depending on the hook event you have selected.

Hooks without Namespaces

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

```
<?php

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

./custom/modules/Accounts/customLogicHook.php

```
<?php

class className
{
    function methodName($bean, $event, $arguments)
    {
        //logic
    }
}
?>
```

The logic hook method signature may contain different arguments depending on the hook event you have selected.

Considerations

When using logic hooks, keep in mind the following best practices:

- As of PHP 5.3, objects are automatically passed by reference. When creating logic hook signatures, do not append the ampersand (&) to the \$bean variable. Doing this will cause unexpected behavior.
- There is no hook that fires specifically for the ListView, DetailView or EditView events. Instead, use either the process_record or after_retrieve logic hooks.
- In order to compare new values with previous values, use fetched_row to

determine whether a change has occurred:

```
if ($bean->fetched_row['{field}'] != $bean->{field})
{
    //logic
}
```

- Make sure that the permissions on the logic_hooks.php file and the class file that it references are readable by the web server. Otherwise, Sugar will not read the files and your business logic extensions will not work. For example, *nix developers who are extending the Tasks logic should use the following command for the logic_hooks file and the same command for the class file that will be called:

```
chmod +r ./custom/modules/Tasks/logic_hooks.php
```

- Make sure that the entire ./custom/ directory is writable by the web server or else the logic hooks code will not work properly.

Last Modified: 10/14/2016 04:12pm

Application Hooks

Application hooks execute logic when working with the global application.

Last Modified: 10/14/2016 04:12pm

after_entry_point

Overview

The after_entry_point application hook executes at the start of every request.

Definition

```
function after_entry_point($event, $arguments){}
```

Arguments

| Name | Type | Description |
|------|------|-------------|
| | | |

| | | |
|-----------|--------|---|
| event | String | The current event |
| arguments | Array | Additional information related to the event (typically empty) |

Considerations

- The `after_entry_point` hook is a global logic hook where the logic hook reference must be placed in `./custom/modules/logic_hooks.php`.
- This hook is executed at the start of every request at the end of `./include/entryPoint.php`.
- This hook should not be used for any type of display output.
- The `after_entry_point` hook is ideally used for logging or loading libraries.
- Application hooks do not make use of the `$bean` argument.

Change Log

| Version | Note |
|---------|---|
| 6.4.3 | Added <code>after_entry_point</code> 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: 10/14/2016 04:12pm

after_load_user

Overview

The `after_load_user` hook executes after the current user is set for the current request. It handles actions that are dependent on the current user, such as setting ACLs or configuring user-dependent parameters.

Definition

```
function after_load_user($bean, $event, $arguments){}
```

Arguments

| Name | Type | Description |
|-----------|--------|---|
| bean | Object | The bean object |
| event | String | The current event |
| arguments | Array | Additional information related to the event (typically empty) |

Change Log

| Version | Note |
|---------|---|
| 6.6.0 | Added <code>after_load_user</code> hook |

Example

`./custom/modules/logic_hooks.php`

```
<?php
```

```
$hook_version = 1;
$hook_array = Array();

$hook_array['after_load_user'] = Array();
$hook_array['after_load_user'][] = Array(
    //Processing index. For sorting the array.
    1,

    //Label. A string value to identify the hook.
    'after_load_user example',

    //tde PHP file where your class is located.
    'custom/modules/application_hooks_class.php',

    //tde class the method is in.
    'application_hooks_class',

    //tde method to call.
    'after_load_user_method'
```

```
);  
  
?>  
  
./custom/modules/application_hooks_class.php  
  
<?php  
  
    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry  
Point');  
  
    class application_hooks_class  
    {  
        function after_load_user_method($bean, $event, $arguments)  
        {  
            //logic  
        }  
    }  
  
?>
```

Last Modified: 10/14/2016 04:12pm

after_session_start

Overview

The `after_session_start` hook executes before the user's session starts, but after the user's visibility rules have been set up and the `after_load_user` logic hook has executed.

Definition

```
function after_session_start($bean, $event, $arguments){}
```

Arguments

| Name | Type | Description |
|------|--------|-----------------|
| bean | Object | The bean object |
| | | |

| | | |
|-----------|--------|---|
| event | String | The current event |
| arguments | Array | Additional information related to the event (typically empty) |

Change Log

| Version | Note |
|---------|--------------------------------|
| 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: 10/14/2016 04:12pm

after_ui_footer

Overview

The `after_ui_footer` hook 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 (typically empty) |

Considerations

- The `after_ui_footer` hook is only applicable for modules in backward compatibility mode.
- This is a global logic hook where the logic hook reference must be placed in `./custom/modules/logic_hooks.php`.
- If you intend to write display logic to the screen in a module running in backward compatibility, you must first wrap the display logic in an IF condition to prevent the text breaking the Ajax page loads. This logic may vary and it is best to only run your code on specific pages rather than all pages.

```
if (!isset($_REQUEST["to_pdf"]) || $_REQUEST["to_pdf"] == false)
{
    //display logic
}
```

- This hook is executed on all page loads.
- Application hooks do not make use of the `$bean` argument.

Change Log

| Version | Note |
|---------|---|
| 5.0.0a | Added <code>after_ui_footer</code> 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: 10/14/2016 04:12pm

after_ui_frame

Overview

The `after_ui_frame` hook executes after the frame has been invoked and before the footer has been invoked for modules in backward compatibility mode.

Definition

```
function after_ui_frame($event, $arguments){}
```

Arguments

| Name | Type | Description |
|-----------|--------|---|
| event | String | The current event |
| arguments | Array | Additional information related to the event (typically empty) |

Considerations

- This hook is only applicable for modules in backward compatibility mode.
- This hook is executed on most views such as the DetailView, EditView, and Listview.
- Application hooks do not make use of the \$bean argument.
- This is logic hook can be used as a global reference (`./custom/modules/logic_hooks.php`) or as a module reference (`./custom/modules/<module>/logic_hooks.php`).

Change Log

| Version | Note |
|---------|--|
| 5.0.0a | Added <code>after_ui_frame</code> 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: 10/14/2016 04:12pm

handle_exception

Overview

The `handle_exception` logic hook executes when an exception is thrown.

Definition

```
function handle_exception($event, $exception){}
```

Arguments

| Name | Type | Description |
|-----------|--------|----------------------|
| event | String | The current event |
| exception | Object | The exception object |

Considerations

- This hook should not be used for any type of display output.
- You can retrieve the exception message by using `$exception->getMessage()`. All exception classes extend the PHP Exceptions class.

Change Log

| Version | Note |
|---------|--|
| 6.4.4 | Added <code>handle_exception</code> 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 11:46am

server_round_trip

Overview

Executes at the end of every page.

Definition

```
function server_round_trip($event, $arguments){}
```

Arguments

| Name | Type | Description |
|-----------|--------|---|
| event | String | The current event |
| arguments | Array | Additional information related to the event (typically empty) |

Considerations

- This is a global logic hook where the logic hook reference must be placed in `./custom/modules/logic_hooks.php`.
- If you are intending to write display logic to the screen, you must first wrap the display logic in an if condition to prevent breaking the Ajax page loads:

```
if (!isset($_REQUEST["to_pdf"]) || $_REQUEST["to_pdf"] == false)
{
    //display logic
}
```

- This hook is executed on all page loads.
- Application hooks do not make use of the `$bean` argument.

Change Log

| Version | Note |
|---------|--|
| 5.0.0a | Added <code>server_round_trip</code> 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: 10/14/2016 04:12pm

Module Hooks

Module hooks execute logic when working with Sugar modules (e.g. Accounts, Cases, etc.).

Last Modified: 10/14/2016 04:12pm

after_delete

Overview

The after_delete hook executes after a record is deleted.

Definition

```
function after_delete($bean, $event, $arguments){}
```

Arguments

| Name | Type | Description |
|--------------|--------|---|
| bean | Object | The bean object |
| event | String | The current event |
| arguments | Array | Additional information related to the event |
| arguments.id | String | ID of the deleted record |

Examples

Creating a Logic Hook using the Extension Framework

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

```
<?php
```

```
    $hook_array['after_delete'][] = Array(  
        //Processing index. For sorting the array.  
        1,
```

```

        //Label. A string value to identify the hook.
        'after_delete example',

        //The PHP file where your class is located.
        'custom/modules/<module>/logic_hooks_class.php',

        //The class the method is in.
        'logic_hooks_class',

        //The method to call.
        'after_delete_method'
    );
?>

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

<?php

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

    class logic_hooks_class
    {
        function after_delete_method($bean, $event, $arguments)
        {
            //logic
        }
    }
?>

```

Creating a Core Logic Hook

Prior to Sugar 6.3.x, logic hooks could only be created using the following method. Please note that this approach is still valid but is not recommended when building plugins as it may conflict with existing customizations.

```

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

<?php

$hook_version = 1;
$hook_array = Array();

```

```
$hook_array['after_delete'] = Array();
$hook_array['after_delete'][] = Array(
    //Processing index. For sorting the array.
    1,

    //Label. A string value to identify the hook.
    'after_delete example',

    //The PHP file where your class is located.
    'custom/modules/<module>/after_delete_class.php',

    //The class the method is in.
    'after_delete_class',

    //The method to call.
    'after_delete_method'
);
```

?>

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

<?php

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

```
class after_delete_class
{
    function after_delete_method($bean, $event, $arguments)
    {
        //logic
    }
}
```

?>

Last Modified: 10/14/2016 04:12pm

after_relationship_add

Overview

The `after_relationship_add` hook executes after a relationship has been added between two records.

Definition

```
function after_relationship_add($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
<code>bean</code>	Object	The bean object
<code>event</code>	String	The current event
<code>arguments</code>	Array	Additional information related to the event
<code>arguments.id</code>	String	Module ID
<code>arguments.module</code>	String	Module name
<code>arguments.related_id</code>	String	Related module ID
<code>arguments.related_module</code>	String	Related module name
<code>arguments.link</code>	String	Link field name
<code>arguments.relationship</code>	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 <code>after_relationship_add</code> 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: 10/14/2016 04:12pm

after_relationship_delete

Overview

The `after_relationship_delete` hook executes after a relationship between two records has been deleted.

Definition

```
function after_relationship_delete($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object
event	String	The current event
arguments	Array	Additional information related to the event
arguments.id	String	Module ID
arguments.module	String	Module name
arguments.related_id	String	Related module ID
arguments.related_module	String	Related module name
arguments.link	String	Link field name
arguments.relationship	String	Relationship name

Considerations

-
- This hook will be executed for each side of the relationship. For example, if you delete an association between an account and contact, the hook will run for both.
 - The 'arguments' parameter will have additional information regarding the records being modified. The \$bean variable will not contain this information.

Change Log

Version	Note
6.0.0	Added after_relationship_delete hook

Examples

Creating a Logic Hook using the Extension Framework

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

```
<?php
```

```
$hook_array['after_relationship_delete'][] = Array(
    //Processing index. For sorting the array.
    1,

    //Label. A string value to identify the hook.
    'after_relationship_delete example',

    //The PHP file where your class is located.
    'custom/modules/<module>/after_relationship_delete_class.php',

    //The class the method is in.
    'after_relationship_delete_class',

    //The method to call.
    'after_relationship_delete_method'
);
```

```
?>
```

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

```
<?php
```

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

class after_relationship_delete_class
{
    function after_relationship_delete_method($bean, $event,
$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: 10/14/2016 04:12pm

after_restore

Overview

The after_restore hook executes after a record gets undeleted (i.e. the deleted field's value changes from 1 to 0).

Definition

```
function after_restore($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object
event	String	The current event

arguments	Array	Additional information related to the event (typically empty)
-----------	-------	---

Examples

Creating a Logic Hook using the Extension Framework

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

```
<?php
```

```
$hook_array['after_restore'][] = Array(
    //Processing index. For sorting the array.
    1,

    //Label. A string value to identify the hook.
    'after_restore example',

    //The PHP file where your class is located.
    'custom/modules/<module>/after_restore_class.php',

    //The class the method is in.
    'after_restore_class',

    //The method to call.
    'after_restore_method'
);
```

```
?>
```

`./custom/modules/<module>/after_restore_class.php`

```
<?php
```

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

```
class after_restore_class
{
    function after_restore_method($bean, $event, $arguments)
    {
        //logic
    }
}
```

```
}
```

```
?>
```

Creating a Core Logic Hook

Prior to Sugar 6.3.x, logic hooks could only be created using the following method. Please note that this approach is still valid but is not recommended when building plugins as it may conflict with existing customizations.

```
./custom/modules/<module>/logic_hooks.php
```

```
<?php
```

```
$hook_version = 1;
$hook_array = Array();

$hook_array['after_restore'] = Array();
$hook_array['after_restore'][] = Array(
    //Processing index. For sorting the array.
    1,

    //Label. A string value to identify the hook.
    'after_restore example',

    //The PHP file where your class is located.
    'custom/modules/<module>/after_restore_class.php',

    //The class the method is in.
    'after_restore_class',

    //The method to call.
    'after_restore_method'
);
```

```
?>
```

```
./custom/modules/<module>/after_restore_class.php
```

```
<?php
```

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

class after_restore_class
```

```
{
  function after_restore_method($bean, $event, $arguments)
  {
    //logic
  }
}
```

?>

Last Modified: 10/14/2016 04:12pm

after_retrieve

Overview

The `after_retrieve` hook executes after a record has been retrieved from the database.

Definition

```
function after_retrieve($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object
event	String	The current event
arguments	Array	Additional information related to the event
arguments.id	String	ID of the record
arguments.encode	Boolean	Whether or not to encode

Considerations

- The `after_retrieve` hook does not fire when a new record is being created.
- Calling `save` on the bean in this hook can cause adverse side effects if not

handled correctly and should be avoided. (i.e. `$bean->save()`)

Examples

Creating a Logic Hook using the Extension Framework

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

```
<?php
```

```
$hook_array['after_retrieve'][] = Array(  
    //Processing index. For sorting the array.  
    1,  
  
    //Label. A string value to identify the hook.  
    'after_retrieve example',  
  
    //The PHP file where your class is located.  
    'custom/modules/<module>/after_retrieve_class.php',  
  
    //The class the method is in.  
    'after_retrieve_class',  
  
    //The method to call.  
    'after_retrieve_method'  
);
```

```
?>
```

`./custom/modules/{module}/{module}_hook.php`

```
<?php
```

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

```
class after_retrieve_class  
{  
    function after_retrieve_method($bean, $event, $arguments)  
    {  
        //logic  
    }  
}
```

```
?>
```

Creating a Core Logic Hook

Prior to Sugar 6.3.x, logic hooks could only be created using the following method. Please note that this approach is still valid but is not recommended when building plugins as it may conflict with existing customizations.

```
./custom/modules/<module>/logic_hooks.php
```

```
<?php

$hook_version = 1;
$hook_array = Array();

$hook_array['after_retrieve'] = Array();
$hook_array['after_retrieve'][] = Array(
    //Processing index. For sorting the array.
    1,

    //Label. A string value to identify the hook.
    'after_retrieve example',

    //The PHP file where your class is located.
    'custom/modules/<module>/after_retrieve_class.php',

    //The class the method is in.
    'after_retrieve_class',

    //The method to call.
    'after_retrieve_method'
);

?>
```

```
./custom/modules/<module>/after_retrieve_class.php
```

```
<?php

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

class after_retrieve_class
{
    function after_retrieve_method($bean, $event, $arguments)
    {
```

```
        //logic
    }
}
```

?>

Last Modified: 10/14/2016 04:12pm

after_save

Overview

The `after_save` hook executes after a record is saved.

Definition

```
function after_save($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object
event	String	The current event
arguments	Array	Additional information related to the event (typically empty)
arguments.isUpdate	Boolean	Whether or not the record is newly created or not. True is an existing record being saved. False is a new record being created
arguments.dataChanges	Array	A list of the 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: 10/14/2016 04:12pm

before_delete

Overview

The before_delete logic hook executes before a record is deleted.

Definition

```
function before_delete($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object
event	String	The current event
arguments	Array	Additional information related to the event

Name	Type	Description
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: 10/14/2016 04:12pm

before_relationship_add

Overview

The before_relationship_add logic hook executes before a relationship has been added between two records.

Definition

```
function before_relationship_add($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object
event	String	The current event
arguments	Array	Additional information related to the event
arguments.id	String	Module ID
arguments.module	String	Module name
arguments.related_id	String	Related module ID
arguments.related_module	String	Related module name
arguments.link	String	Link field name
arguments.relationship	String	Relationship name

Considerations

- This hook will be executed for each side of the relationship. For example, if you add an association between an account and a contact, the hook will run for both records.
- The arguments parameter will have additional information regarding the records being modified. The \$bean variable will not contain this information.

Change Log

Version	Note
6.4.5	Added before_relationship_add hook

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: 10/14/2016 04:12pm

before_relationship_delete

Overview

The `before_relationship_delete` logic hook executes before a relationship between two records is deleted.

Definition

```
function before_relationship_delete($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object
event	String	The current event
arguments	Array	Additional information related to the event
arguments.id	String	Module id
arguments.module	String	Module name
arguments.related_id	String	Related module id
arguments.related_module	String	Related module name

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. For example, if you delete an association between an account and a contact, the hook will run for both records.
- The arguments parameter will have additional information regarding the records being modified. The \$bean variable will not contain this information.

Change Log

Version	Note
6.4.5	Added before_relationship_delete hook.

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: 10/14/2016 04:12pm

before_restore

Overview

The before_restore logic hook executes before a record gets undeleted (i.e. the deleted field's value changes from 1 to 0).

Definition

```
function before_restore($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object

Name	Type	Description
event	String	The current event
arguments	Array	Additional information related to the event (typically empty)

Examples

Creating a Logic Hook using the Extension Framework

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

```
<?php
```

```
    //Processing index. For sorting the array.
    1,

    //Label. A string value to identify the hook.
    'before_restore example',

    //The PHP file where your class is located.
    'custom/modules/<module>/before_restore_class.php',

    //The class the method is in.
    'before_restore_class',

    //The method to call.
    'before_restore_method'
```

```
);
```

```
?>
```

`./custom/modules/<module>/before_restore_class.php`

```
<?php
```

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

```
class before_restore_class
```

```
{
```

```
    function before_restore_method($bean, $event, $arguments)
```

```
{
```

```
        //logic
    }
}
```

?>

Creating a Core Logic Hook

Prior to Sugar 6.3.x, logic hooks could only be created using the following method. Please note that this approach is still valid but is not recommended when building plugins as it may conflict with existing customizations.

`./custom/modules/<module>/logic_hooks.php`

```
<?php

$hook_version = 1;
$hook_array = Array();

$hook_array['before_restore'] = Array();
$hook_array['before_restore'][] = Array(
//Processing index. For sorting the array.
1,

//Label. A string value to identify the hook.
'before_restore example',

//The PHP file where your class is located.
'custom/modules/<module>/before_restore_class.php',

//The class the method is in.
'before_restore_class',

//The method to call.
'before_restore_method'
);

?>
```

`./custom/modules/<module>/before_restore_class.php`

```
<?php

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

```
class before_restore_class
{
function before_restore_method($bean, $event, $arguments)
{
//logic
}
}

?>
```

Last Modified: 10/14/2016 04:12pm

before_save

Overview

The before_save logic hook executes before a record is saved.

Definition

```
function before_save($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object
event	String	The current event
arguments	Array	Additional information related to the event
arguments.check_notify	Boolean	Whether or not to send notifications
arguments.isUpdate	Boolean	Whether or not the record is newly created <ul style="list-style-type: none">• true = this is an update to an existing

Name	Type	Description
		record false = a newly created record

Considerations

- For modules that contain a user-friendly record ID (e.g. the case_number field for the Cases module), the value of that field is not available for a before_save call. This is because this business logic has yet to be executed.
- Calling save on the bean in this hook will cause an infinite loop if not handled correctly. (i.e: \$bean->save())

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: 10/14/2016 04:12pm

process_record

Overview

The process_record logic hook executes when the record is being processed as a part of the ListView or subpanel list.

Definition

```
function process_record($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object
event	String	The current event
arguments	Array	Additional information related to the event (typically empty)

Considerations

- This can be used to set values in a record's fields prior to display in the ListView or in the subpanel of a DetailView.
- This event is not fired in the EditView.
- You should not call save on the referenced bean. The bean is only populated for list fields and will result in a loss of information.

Change Log

Version	Note
5.0.0a	Added process_record hook

Examples

Creating a Logic Hook using the Extension Framework

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

```
<?php
```

```
$hook_array['process_record'][] = Array(  
    //Processing index. For sorting the array.  
    1,  
  
    //Label. A string value to identify the hook.  
    'process_record example',
```

```

//The PHP file where your class is located.
'custom/modules/<module>/process_record_class.php',

//The class the method is in.
'process_record_class',

//The method to call.
'process_record_method'
);

?>

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

<?php

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

class process_record_class
{
    function process_record_method($bean, $event, $arguments)
    {
        //logic
    }
}

?>

```

Creating a Core Logic Hook

Prior to Sugar 6.3.x, logic hooks could only be created using the following method. Please note that this approach is still valid but is not recommended when building plugins as it may conflict with existing customizations.

```

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

<?php

$hook_version = 1;
$hook_array = Array();

$hook_array['process_record'] = Array();
$hook_array['process_record'][] = Array(
    //Processing index. For sorting the array.

```

```
1,

//Label. A string value to identify the hook.
'process_record example',

//The PHP file where your class is located.
'custom/modules/<module>/process_record_class.php',

//The class the method is in.
'process_record_class',

//The method to call.
'process_record_method'
);

?>

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

<?php

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

    class process_record_class
    {
        function process_record_method($bean, $event, $arguments)
        {
            //logic
        }
    }

?>
```

Last Modified: 12/05/2016 03:30pm

User Hooks

User hooks execute logic around user actions such as logging in and logging out.

Last Modified: 10/14/2016 04:12pm

after_login

Overview

The `after_login` hook executes after a user logs into the system.

Definition

```
function after_login($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object
event	String	The current event
arguments	Array	Additional information related to the event (typically empty)

Change Log

Version	Note
5.0.0a	Added <code>after_login</code> 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: 10/14/2016 04:12pm

after_logout

Overview

The after_logout hook executes after a user logs out of the system.

Definition

```
function after_logout($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object
event	String	The current event
arguments	Array	Additional information related to the event (typically empty)

Change Log

Version	Note
5.0.0a	Added after_logout hook

Example

```
./custom/modules/Users/logic_hooks.php
```

```
<?php
```

```
$hook_version = 1;
$hook_array = Array();

$hook_array['after_logout'] = Array();
$hook_array['after_logout'][] = Array(
    //Processing index. For sorting the array.
    1,

    //Label. A string value to identify the hook.
    'after_logout example',

    //The PHP file where your class is located.
    'custom/modules/Users/logic_hooks_class.php',

    //The class the method is in.
    'logic_hooks_class',
```

```
        //The method to call.
        'after_logout_method'
    );

?>

./custom/modules/Users/logic_hooks_class.php

<?php

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

    class logic_hooks_class
    {
        function after_logout_method($bean, $event, $arguments)
        {
            //logic
        }
    }

?>
```

Last Modified: 10/14/2016 04:12pm

before_logout

Overview

The before_logout hook executes before a user logs out of the system.

Definition

```
function before_logout($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object

Name	Type	Description
event	String	The current event
arguments	Array	Additional information related to the event (typically empty)

Change Log

Version	Note
5.0.0a	Added before_logout hook

Example

`./custom/modules/Users/logic_hooks.php`

```
<?php
```

```
$hook_version = 1;
$hook_array = Array();

$hook_array['before_logout'] = Array();
$hook_array['before_logout'][] = Array(
    //Processing index. For sorting the array.
    1,

    //Label. A string value to identify the hook.
    'before_logout example',

    //The PHP file where your class is located.
    'custom/modules/Users/logic_hooks_class.php',

    //The class the method is in.
    'logic_hooks_class',

    //The method to call.
    'before_logout_method'
);
```

```
?>
```

./custom/modules/Users/logic_hooks_class.php

```
<?php

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

class logic_hooks_class
{
    function before_logout_method($bean, $event, $arguments)
    {
        //logic
    }
}

?>
```

Last Modified: 10/14/2016 04:12pm

login_failed

Overview

The login_failed hook executes on a failed login attempt.

Definition

```
function login_failed($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The bean object
event	String	The current event
arguments	Array	Additional information related to the event (typically empty)

Change Log

Version	Note
5.0.0a	Added login_failed hook

Example

`./custom/modules/Users/logic_hooks.php`

```
<?php
```

```
    $hook_version = 1;
    $hook_array = Array();

    $hook_array['login_failed'] = Array();
    $hook_array['login_failed'][] = Array(
        //Processing index. For sorting the array.
        1,

        //Label. A string value to identify the hook.
        'login_failed example',

        //The PHP file where your class is located.
        'custom/modules/Users/logic_hooks_class.php',

        //The class the method is in.
        'logic_hooks_class',

        //The method to call.
        'login_failed_method'
    );
```

```
?>
```

`./custom/modules/Users/logic_hooks_class.php`

```
<?php
```

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

```
    class logic_hooks_class
    {
```

```
function login_failed_method($bean, $event, $arguments)
{
    //logic
}
}
```

?>

Last Modified: 10/14/2016 04:12pm

Job Queue Hooks

Job Queue hooks execute logic when working with the Job Queue module.

Last Modified: 10/14/2016 04:12pm

job_failure

Overview

The `job_failure` hook executes when a job's final failure occurs.

Definition

```
function job_failure($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The Scheduler's Job object
event	String	The current event
arguments	Array	Additional information related to the event (typically empty)

Change Log

Version	Note
6.5.0RC1	Added job_failure 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: 10/14/2016 04:12pm

job_failure_retry

Overview

The `job_failure_retry` hook executes each time a job fails before its final failure. If you only want action on the final failure, use the `job_failure` logic hook.

Definition

```
function job_failure_retry($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The Scheduler's Job object
event	String	The current event
arguments	Array	Additional information related to the event (typically empty)

Change Log

Version	Note
6.5.0RC1	Added <code>job_failure_retry</code> 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: 10/14/2016 04:12pm

SNIP Hooks

SNIP hooks execute logic when working with the SNIP email-archiving service.

Last Modified: 10/14/2016 04:12pm

after_email_import

Overview

The after_email_import hook executes after a SNIP email has been imported.

Definition

```
function after_email_import($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The Email object
event	String	The current event
arguments	Array	Additional information related to the event (typically empty)

Considerations

- This is a global logic hook where the logic hook reference must be placed in `./custom/modules/logic_hooks.php`.

Change Log

Version	Note
---------	------

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: 10/14/2016 04:12pm

before_email_import

Overview

The `before_email_import` hook executes before a SNIP email has been imported.

Definition

```
function before_email_import($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	The Email object
event	String	The current event
arguments	Array	Additional information related to the event (typically empty)

Considerations

- This is a global logic hook where the logic hook reference must be placed in `./custom/modules/logic_hooks.php`.

Change Log

Version	Note
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: 10/14/2016 04:12pm

API Hooks

API hooks execute logic when working with the REST API and routing.

Last Modified: 10/14/2016 04:12pm

after_routing

Overview

The after_routing hook executes when the v10+ REST Service has found the route for the request.

Definition

```
function after_routing($event, $arguments){}
```

Arguments

Name	Type	Description
event	String	The name of the logic hook event
arguments	Array	Additional information related to the event
arguments.api	Object	The RestService Object
arguments.request	Object	The RestResponse Object

Considerations

- This is a global logic hook where the logic hook reference must be placed in `./custom/modules/logic_hooks.php`.
- This hook can change request object parameters that influence routing.
- This hook should not be used for any type of display output.

Change Log

Version	Note
7.0.0RC1	Added <code>after_routing</code> 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: 10/14/2016 04:12pm

before_api_call

Overview

The before_api_call hook executes when the v10+ REST Service is about to call the API implementation.

Definition

```
function before_api_call($event, $arguments){}
```

Arguments

Name	Type	Description
event	String	The name of the logic hook event
arguments	Array	Additional information related to the event
arguments.api	Object	The RestService Object
arguments.request	Object	The RestResponse Object

Name	Type	Description
------	------	-------------

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 <code>before_api_call</code> 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: 10/14/2016 04:12pm

before_filter

Overview

The `before_filter` hook executes when the v10+ REST Service is about to route the request.

Definition

```
function before_filter($bean, $event, $arguments){}
```

Arguments

Name	Type	Description
bean	Object	An empty bean object of

Name	Type	Description
		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 <code>before_filter</code> hook

Example

`./custom/modules/{module}/logic_hooks.php`

```
<?php
```

```

$hook_version = 1;
$hook_array = Array();

$hook_array['before_filter'] = Array();
$hook_array['before_filter'][] = Array(
    //Processing index. For sorting the array.
    1,
```

```
//Label. A string value to identify the hook.
'before_filter example',

//The PHP file where your class is located.
'custom/modules/{module}/logic_hooks_class.php',

//The class the method is in.
'logic_hooks_class',

//The method to call.
'before_filter_method'
);
```

```
?>
```

```
./custom/modules/{module}/logic_hooks_class.php
```

```
<?php
```

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

```
class logic_hooks_class
{
    function before_filter_method($bean, $event, $arguments)
    {
        //logic
    }
}
```

```
?>
```

Last Modified: 10/14/2016 04:12pm

before_respond

Overview

The `before_respond` hook executes before the v10+ REST Service call returns data to the user.

Definition

```
function before_respond($event, $response){}
```

Arguments

Name	Type	Description
event	String	The name of the logic hook event
response	Object	The RestResponse Object

Considerations

- This is a global logic hook where the logic hook reference must be placed in `./custom/modules/logic_hooks.php`.
- It is not advised to remove or unset any data. Sugar may be using this data and it can adversely affect your instance.
- This hook should not be used for any type of display output.
- This hook should be used when you want to add data to all API responses. If you are looking to modify data received from one specific endpoint, It is a much better approach to override that specific endpoint rather than to use this logic hook.

Change Log

Version	Note
7.0.0RC1	Added before_respond hook

Example

```
./custom/modules/logic_hooks.php
```

```
<?php
```

```
    $hook_version = 1;
```

```
$hook_array = Array();

$hook_array['before_respond'] = Array();
$hook_array['before_respond'][] = Array(
    //Processing index. For sorting the array.
    1,

    //Label. A string value to identify the hook.
    'after_routing example',

    //The PHP file where your class is located.
    'custom/modules/logic_hooks_class.php',

    //The class the method is in.
    'logic_hooks_class',

    //The method to call.
    'before_respond_method'
);
```

?>

`./custom/modules/logic_hooks_class.php`

<?php

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

```
class logic_hooks_class
{
    function before_respond_method($event, $response)
    {
        //logic
    }
}
```

?>

Last Modified: 10/14/2016 04:12pm

before_routing

Overview

The `before_routing` hook executes when the v10+ REST Service is about to route the request.

Definition

```
function before_routing($event, $arguments){}
```

Arguments

Name	Type	Description
event	String	The name of the logic hook event
arguments	Array	Additional information related to the event
arguments.api	Object	The RestService Object
arguments.request	Object	The RestResponse Object

Considerations

- This is a global logic hook where the logic hook reference must be placed in `./custom/modules/logic_hooks.php`.
- This hook can change request object parameters that influence routing.
- This hook should not be used for any type of display output.

Change Log

Version	Note
7.0.0RC1	Added <code>before_routing</code> hook

Example

./custom/modules/logic_hooks.php

```
<?php

$hook_version = 1;
$hook_array = Array();

$hook_array['before_routing'] = Array();
$hook_array['before_routing'][] = Array(
    //Processing index. For sorting the array.
    1,

    //Label. A string value to identify the hook.
    'before_routing example',

    //The PHP file where your class is located.
    'custom/modules/logic_hooks_class.php',

    //The class the method is in.
    'logic_hooks_class',

    //The method to call.
    'before_routing_method'
);

?>
```

./custom/modules/logic_hooks_class.php

```
<?php

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

class logic_hooks_class
{
    function before_routing_method($event, $arguments)
    {
        //logic
    }
}

?>
```


Web Logic Hooks

Overview

Web logic hooks let administrators post record and event information to a specified URL when certain sugar events take place.

The administration panel for web logic hooks can be found by navigating to Admin > Web Logic Hooks in the Sugar application. When a web logic hook is triggered, Sugar creates a record in the job queue.

Note: You must have the cron set up and running in order to process the job queue for web logic hooks. The POST of the information to the external URL will happen when the cron runs and not when the actual event occurs.

Arguments

Name	Description
URL	The URL to post JSON-encoded information
Module Name	The name of the module to which the web logic hook will apply
Trigger Event	The event that will trigger the web logic hook The following events are applicable to a web logic hook: <ul style="list-style-type: none">• after_save• after_delete• after_relationship_add• after_relationship_delete• after_login• after_logout• after_login_failed
Request Method	The request method to use when sending the information

	The following methods may be used:
--	------------------------------------

- POST
- GET
- PUT
- DELETE

Example

The following example shows how to receive the information posted to the web logic hook URL parameter.

`http://{url}/receive.php`

```
<?php

    //get the posted JSON data
    $data = file_get_contents('php://input');
    //decode the data
    $decoded_data = json_decode(trim($data));

    //use the data
    $file = 'receivedData-'.time().'.txt';
    file_put_contents($file, print_r($decoded_data, true));

?>
```

Result

receivedData-1380158171.txt

```
stdClass Object
(
    [isUpdate] => 1
    [dataChanges] => Array
        (
        )

    [bean] => Account
```

```
[data] => stdClass Object
(
  [id] => e0623cdb-ac4b-e7ff-f681-5242e9116892
  [name] => Super Star Holdings Inc
  [date_modified] => 2013-09-25T21:16:06-04:00
  [modified_user_id] => 1
  [modified_by_name] => admin
  [created_by] => 1
  [created_by_name] => Administrator
  [description] =>
  [deleted] =>
  [assigned_user_id] => seed_will_id
  [assigned_user_name] => Will Westin
  [team_count] =>
  [team_name] => Array
  (
    [0] => stdClass Object
      (
        [id] => East
        [name] => East
        [name_2] =>
        [primary] => 1
      )
    [1] => stdClass Object
      (
        [id] => West
        [name] => West
        [name_2] =>
        [primary] =>
      )
  )
  [linkedin] =>
  [facebook] =>
  [twitter] =>
  [googleplus] =>
  [account_type] => Customer
  [industry] => Energy
  [annual_revenue] =>
  [phone_fax] =>
  [billing_address_street] => 111 Silicon Valley Road
  [billing_address_city] => Los Angeles
  [billing_address_state] => NY
```

```
[billing_address_postalcode] => 84028
[billing_address_country] => USA
[rating] =>
[phone_office] => (374) 791-2199
[phone_alternate] =>
[website] =>
[ownership] =>
[employees] =>
[ticker_symbol] =>
[shipping_address_street] => 111 Silicon Valley Road
[shipping_address_city] => Los Angeles
[shipping_address_state] => NY
[shipping_address_postalcode] => 84028
[shipping_address_country] => USA
[email] => Array
(
  [0] => stdClass Object
  (
    [email_address] => example@example.tw
    [invalid_email] =>
    [opt_out] =>
    [primary_address] => 1
    [reply_to_address] =>
  )

  [1] => stdClass Object
  (
    [email_address] => the.example@example.name
    [invalid_email] =>
    [opt_out] =>
    [primary_address] =>
    [reply_to_address] =>
  )

)
[email1] => example@example.tw
[parent_id] =>
[sic_code] =>
[parent_name] =>
[email_opt_out] =>
[invalid_email] =>
[campaign_id] =>
[campaign_name] =>
)
```

```
[event] => after_save  
)
```

Last Modified: 10/14/2016 04:12pm

Logic Hook Release Notes

This page documents historical changes to Sugar's logic hook libraries.

7.0.0RC1

- Web Logic Hooks were introduced in this release.
- The following hooks were also added in the 7.0.0RC1 release:
 - after_routing
 - before_api_call
 - before_filter
 - before_respond
 - before_routing
- The following parameters were added in the 7.0.0RC1 release:
 - after_save arguments.isUpdate
 - after_save arguments.dataChanges
 - before_save arguments.isUpdate

6.6.0

The after_load_user hook was added in the 6.6.0 release.

6.5.0RC1

The following hooks were added in the 6.5.0RC1 release:

- after_email_import
- before_email_import
- job_failure
- job_failure_retry

6.4.5

The following hooks were added in the 6.4.5 release:

- `before_relationship_add`
- `before_relationship_delete`

6.4.4

The `handle_exception` hook was added in the 6.4.4 release.

6.4.3

The `after_entry_point` hook was added in the 6.4.3 release.

6.0.0RC1

The following hooks were added in the 6.0.0RC1 release:

- `after_relationship_add`
- `after_relationship_delete`

5.0.0a

The following hooks were added in the 5.0.0a release:

- `after_login`
- `after_logout`
- `after_ui_footer`
- `after_ui_frame`
- `before_logout`
- `login_failed`
- `process_record`
- `server_round_trip`

4.5.0c

The `after_save` hook was added in the 4.5.0c release.

Last Modified: 11/30/2016 10:48pm

Examples

These pages demonstrate some common examples of logic hooks in Sugar.

Last Modified: 10/14/2016 04:12pm

Comparing Bean Properties Between Logic Hooks

Overview

How to compare the properties of a bean between the `before_save` and `after_save` logic hooks

Storing and Comparing Values

When working with a bean in the `after_save` logic hook, you may notice that the `after_save` fetched rows no longer match the `before_save` fetched rows. If your `after_save` logic needs to be able to compare values that were in the `before_save`, you can use the following example to help you store and use the values.

```
./custom/modules/<module>/logic_hooks.php
```

```
<?php
```

```
$hook_version = 1;
$hook_array = Array();

$hook_array['before_save'] = Array();
$hook_array['before_save'][] = Array(
    1,
    'Store values',
    'custom/modules/<module>/My_Logic_Hooks.php',
    'My_Logic_Hooks',
```

```
        'before_save_method'
    );

    $hook_array['after_save'] = Array();
    $hook_array['after_save'][] = Array(
        1,
        'Retrieve and compare values',
        'custom/modules/<module>/My_Logic_Hooks.php',
        'My_Logic_Hooks',
        'after_save_method'
    );
};
```

?>

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

<?php

```
if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry 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: 10/14/2016 04:12pm

Manipulating Logic Hook References

Overview

How to add and remove logic hook references with code

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

Remove a logic hook reference 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: 10/14/2016 04:12pm

Preventing Infinite Loops with Logic Hooks

Overview

Infinite loops often happen when a logic hook calls save on a bean in a scenario that triggers the same hook again. This example shows how to add a check to a logic hook to eliminate perpetual looping.

Saving in an After Save Hook

Infinite loops can sometimes happen when you have a need to update a record after it has been run through the workflow process in the after_save hook. Here is an example of a looping hook:

```
./custom/modules/Accounts/logic_hooks.php
```

```
<?php

$hook_version = 1;
$hook_array = Array();
$hook_array['after_save'] = Array();
$hook_array['after_save'][] = Array(
    1,
    'Update Account Record',
    'custom/modules/Accounts/Accounts_Hook.php',
    'Accounts_Hook',
    'update_self'
);
```

```
./custom/modules/Accounts/Accounts_Hook.php
```

```
<?php

if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry 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, there is a generic condition that, when met, will trigger the update of a field on the record. This will cause an infinite loop because calling the save() method will trigger once during the before_save hook and then again during the after_save hook. The solution to this problem is to add a new property on the bean to ignore any unneeded saves. For this example, we will name the property "ignore_update_c" and add a check to the logic hook to eliminate the perpetual loop. An example of this is shown below:

./custom/modules/Accounts/Accounts_Hook.php

```
<?php
```

```

if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry 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 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: 10/14/2016 04:12pm

Languages

Overview

Sugar as an application platform is internationalized and localizable. Data is stored and presented in the UTF-8 codepage, allowing for all character sets to be used. Sugar provides a language-pack framework that allows developers to build support for any language in the display of user interface labels. Each language pack has its own set of display strings which is the basis of language localization. You can add or modify languages using the information in this guide.

Please scroll to the bottom of this page for additional language topics.

Language Keys

Sugar differentiates languages with unique language keys. These keys prefix the files that correspond with particular languages. For example, the default language for the application is English (US), which is represented by the language key

en_us. Any file that contains data specific to the English (US) language begins with the characters en_us. Language label keys that are not recognized will default to the English (US) version.

The following table displays the list of current languages and their corresponding keys:

Language	Language Key
Albanian (Shqip)	sq_AL
Arabic (العربية)	ar_SA
Bulgarian (Български)	bg_BG
Catalan (Català)	ca_ES
Chinese (中文)	zh_CN
Croatian (Hrvatski)	hr_HR
Czech (Česky)	cs_CZ
Danish (Dansk)	da_DK
Dutch (Nederlands)	nl_NL
English (UK)	en_UK
English (US)	en_us
Estonian (Eesti)	et_EE
Finnish (Suomi)	fi_FI
French (Français)	fr_FR
German (Deutsch)	de_DE
Greek (Ελληνικά)	el_EL
Hebrew (עברית)	he_IL
Hungarian (Magyar)	hu_HU
Italian (Italiano)	it_it
Japanese (日本語)	ja_JP
Korean (한국어)	ko_KR
Latvian (Latviešu)	lv_LV
Lithuanian (Lietuvių)	lt_LT
Norwegian (Bokmål)	nb_NO
Polish (Polski)	pl_PL
Portuguese (Português)	pt_PT

Language	Language Key
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 (Latinoamérica))	es_LA
Swedish (Svenska)	sv_SE
Thai (ไทย)	th_TH
Traditional Chinese (繁體中文)	zh_TW
Turkish (Türkçe)	tr_TR
Ukrainian (Українська)	uk_UA

Change Log

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

Version	Change
7.8.0.0	Added Thai language pack.
7.8.0.0	Added Croatian language pack.
7.7.0.0	Added Traditional Chinese language pack.
7.6.0.0	Added Ukrainian language pack.
7.6.0.0	Added Arabic language pack.
7.2.0	Added Finnish language pack.
7.2.0	Added Spanish (Latin America) language pack.
6.6.0	Added Albanian language pack.
6.6.0	Added Slovak language pack.
6.6.0	Added Korean language pack.
6.6.0	Added Greek language pack.
6.5.1	Added Latvian language pack.

Version	Change
6.4.0	Added Serbian language pack.
6.4.0	Added Portuguese Brazilian language pack.
6.4.0	Added English (UK) language pack.
6.4.0	Added Catalan language pack.
6.2.0	Added Polish language pack.
6.2.0	Added Hebrew language pack.
6.2.0	Added Estonian language pack.
6.2.0	Added Czech language pack.
6.1.2	Added Turkish language pack.
6.1.2	Added Swedish language pack.
6.1.2	Added Norwegian language pack.
6.1.2	Added Lithuanian language pack.
6.1.0	Added Chinese language pack.
6.1.0	Added Russian language pack.
6.1.0	Added Romanian language pack.
6.1.0	Added Portuguese language pack.
6.1.0	Added Dutch language pack.
6.1.0	Added Japanese language pack.
6.1.0	Added Italian language pack.
6.1.0	Added Hungarian language pack.
6.1.0	Added French language pack.
6.1.0	Added Spanish language pack.
6.1.0	Added German language pack.
6.1.0	Added Danish language pack.
6.1.0	Added Bulgarian language pack.

Last Modified: 09/21/2017 01:42pm

Application Labels and Lists

Overview

Sugar, which is fully internationalized and localizable, differentiates languages with unique language keys. These keys prefix the files that correspond with particular languages. For example, the default language for the application is English (US), which is represented by the language key `en_us`. Any file that contains data specific to the English (US) language begins with the characters `en_us`. Language label keys that are not recognized will default to the English (US) version.

For more information on language keys, please refer to the [Languages](#) page.

Application Labels and Lists

`$app_list_strings` and `$app_strings`

The `$app_list_strings` array contains the various dropdown lists for the system while `$app_strings` contains the system application labels. The initial set of definitions can be found in `./include/language/<language key>.lang.php`. As you work within the system and deploy modules and lists through Studio, any changes to these lists will be reflected in the language's extension directory:
`./custom/Extension/application/Ext/Language/<language key>.<list name>.php`.

Customizing Application Labels and Lists

If you are developing a customization and want to be able to create or edit existing label or list values, you will need to work within the extension application directory. To do this, create
`./custom/Extension/application/Ext/Language/<language key>.<unique name>.php`.

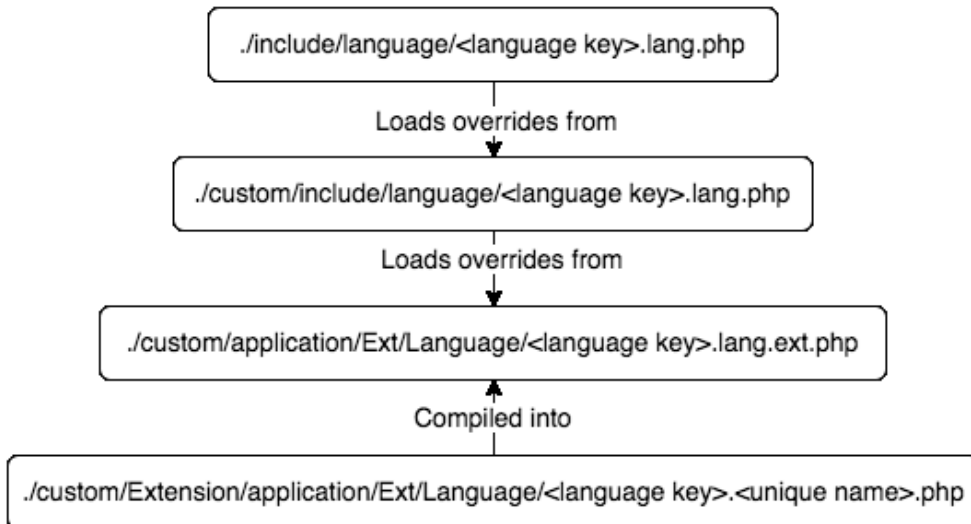
The file should contain your override values with each label index set individually. An example of this is:

```
<?php
```

```
    $app_strings['LBL_KEY'] = 'My Display Label';  
    $app_list_strings['LIST_NAME']['Key_Value'] = 'My Display Value';
```

Once the file is created with your adjustments, navigate to **Admin > Repair > Quick Rebuild & Repair**. This will compile all of the Extension files from `./custom/Extension/application/Ext/Language/` to `./custom/application/Ext/Language/<language key>.lang.ext.php`.

Hierarchy Diagram



Retrieving Labels

There are two ways to retrieve a label. The first is to use the `translate()` function found in `./include/utils.php`. This function will retrieve the label for the current user's language and can also be used to retrieve labels from `$mod_strings`, `$app_strings`, or `app_list_strings`.

An example of this is:

```
require_once('include/utils.php');
$label = translate('LBL_KEY');
```

Alternatively, you can also use the global variable `$app_strings` as follows:

```
global $app_strings;

$label = '';
if (isset($app_strings['LBL_KEY']))
{
    $label = $app_strings['LBL_KEY'];
}
```

Note: If a label key is not found for the user's preferred language, the system will default to "en_us" and pull the English (US) version of the label for display.

Retrieving Lists

There are two ways to retrieve a label. The first is to use the `translate()` function found in `include/utils.php`. This function will retrieve the label for the current user's language.

An example of this is:

```
require_once('include/utils.php');
$list = translate('LIST_NAME');

//You can also retrieve a specific list value this way
$displayValue = translate('LIST_NAME', '', 'Key_Value');
```

Alternatively, you can also use the global variable `$app_list_strings` as follows:

```
global $app_list_strings;

$list = array();
if (isset($app_list_strings['LIST_NAME']))
{
    $list = $app_list_strings['LIST_NAME'];
}
```

Note: If a list key is not found for the user's preferred language, the system will default to "en_us" and pull the English (US) version of the list for display.

Accessing Application Strings in Sidecar

All language-pack strings are accessible within the Sidecar framework.

`$app_strings`

To access `$app_strings` in Sidecar, use `app.lang.getAppString`:

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

To access `$app_strings` in your browser's console, use `SUGAR.App.lang.getAppString`:

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

For more information, please refer to the `app.lang.getAppString` section of the Languages page.

`$app_list_strings`

To access `$app_list_strings` in Sidecar, use `app.lang.getAppListStrings`:

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

To access `$app_list_strings` in your browser's console, use `SUGAR.App.lang.getAppListStrings`:

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

For more information, please refer to the `app.lang.getAppListStrings` section of the Languages page.

Last Modified: 11/27/2017 12:24pm

Module Labels

Overview

Sugar, which is fully internationalized and localizable, differentiates languages with unique language keys. These keys prefix the files that correspond with particular languages. For example, the default language for the application is English (US), which is represented by the language key `en_us`. Any file that contains data specific to the English (US) language begins with the characters `en_us`. Language label keys that are not recognized will default to the English (US) version.

For more information on language keys, please refer to the Languages page.

Module Labels

`$mod_strings`

The module language strings are stored in `$mod_strings`. This section explains how the `$mod_strings` are compiled. All modules, whether out-of-box or custom, will have an initial set of language files in `./modules/<module>/language/<language>`

key>.lang.php.

As you work with the system and modify labels through Studio, changes to the labels are reflected in the corresponding module's custom extension directory: `./custom/Extension/modules/<module>/Ext/Language/<language key>.lang.php`.

Customizing Labels

If you are developing a customization and want to be able to create new or override existing label values, you will need to work within the extension modules directory. To do this, create `./custom/Extension/modules/<module>/Ext/Language/<language key>.<unique_name>.php`.

The file should contain your override values with each label index set individually. An example of this is:

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

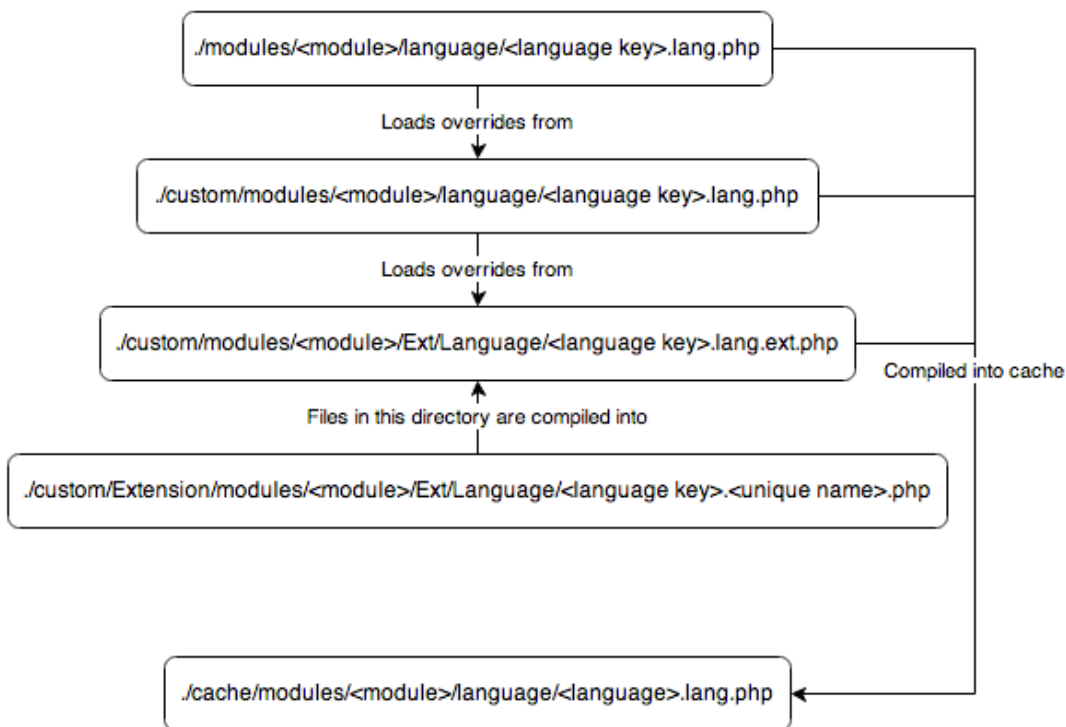
Once the file is created with your adjustments, navigate to Admin > Repair > Quick Rebuild & Repair. This will compile all of the Extension files from `./custom/Extension/modules/<module>/Ext/Language/` to `./custom/modules/<module>/Ext/Language/<language key>.lang.ext.php`.

Label Cache

The file locations discussed above are compiled into the cache directory, `./cache/modules/<module>/language/<language key>.lang.php`

The cached results of these files make up each module's `$mod_strings` definition.

Hierarchy Diagram



Retrieving Labels

There are two ways to retrieve a label. The first is to use the `translate()` function found in `include/utills.php`. This function will retrieve the label for the current user's language and can also be used to retrieve labels from `$mod_strings`, `$app_strings`, or `app_list_strings`.

An example of this is:

```
require_once('include/utills.php');
$label = translate('LBL_KEY', 'Accounts');
```

Alternatively, you can use the global variable `$mod_strings` as follows:

```
global $mod_strings;

$label = '';
if (isset($mod_strings['LBL_KEY']))
{
    $label = $mod_strings['LBL_KEY'];
}
```

Accessing Module Strings in Sidecar

All language-pack strings are accessible within the Sidecar framework.

`$mod_strings`

To access the `$mod_strings` in Sidecar, use `app.lang.get()`:

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

To access the `$mod_strings` in your browser's console, use `SUGAR.App.lang.get()`:

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

For more information, please refer to the `app.lang.get` section of the Languages page.

Last Modified: 10/14/2016 04:12pm

Managing Lists

Overview

There are three ways to manage lists in Sugar: by using Studio in the application, by directly modifying the list's language strings, and via the code-level Dropdown Helper. This page explains all three methods.

Managing Lists With Studio

If you know the name of the list you would like to edit, you can access the application's dropdown editor by navigating to Admin > Dropdown Editor. Alternatively, navigate to a field that uses the list (Admin > Studio > {Module} > Fields > {field_name}) and click "Edit" under the Dropdown List field:

Data Type: DropDown
Field Name: account_type
Display Label:
System Label:
Help Text:
Comment Text:
Drop Down List:
Default Value:

For information on using the dropdown editor, please refer to the Developer Tools documentation in the Administration Guide.

Directly Modifying Lists

There are two ways to directly modify the language strings. The first way is to modify the custom language file, located at `./custom/include/language/<language key>.lang.php`.

If you are developing a customization to be distributed and you want to be able to create new or override existing list values, you will need to work within the extension application directory. To do this you will create the following file: `./custom/Extension/application/Ext/Language/<language key>.<unique name>.php`.

The file will contain your override values. Please note that within this file you will set each label index individually. An example of this is:

```
<?php
    $app_list_strings['LIST_NAME']['Key_Value'] = 'My Display Value';
```

Once the file is created with your adjustments, navigate to Admin > Repair > Quick Rebuild & Repair. This will compile all of the Extension files from `./custom/Extension/application/Ext/Language/` to `./custom/application/Ext/Language/<language key>.lang.ext.php`.

Managing Lists With Dropdown Helper

You can use the dropdown helper to manage lists at the code level. This example demonstrates how to add and update values for a specific dropdown list:

```
require_once('modules/Studio/DropDowns/DropDownHelper.php');
```

```
$dropdownHelper = new DropDownHelper();

$parameters = array();
$parameters['dropdown_name'] = 'example_list';

$listValues = array(
    'Key_Value_1' => 'Display Value 1',
    'Key_Value_2' => 'Display Value 2',
    'Key_Value_3' => 'Display Value 3'
);

$count = 0;
foreach ($listValues as $key=>$value)
{
    $parameters['slot_' . $count] = $count;
    $parameters['key_' . $count] = $key;
    $parameters['value_' . $count] = $value;
    //set 'use_push' to true to update/add values while keeping old
values
    $parameters['use_push'] = true;
    $count++;
}

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

Last Modified: 10/14/2016 04:12pm

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/Opportunities

-
- ByLeadSourceDashlet.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: 05/17/2017 08:56pm

Extensions

Overview

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

Application extensions are stored under `./custom/Extension/application/Ext/` and module extensions are under `./custom/Extension/modules/<module>/Ext/`. The files in these directories are aggregated into a single file with a predefined name for the system to use. An example of this is the vardefs extension. The vardef extension directory for Accounts is located in `./custom/Extension/modules/Accounts/Ext/Vardefs/`. When a module is installed, uninstalled, enabled, or disabled, the files contained in this directory are merged into `./custom/modules/Accounts/Ext/Vardefs/vardefs.ext.php`. A Quick Repair & Rebuild will also cause the files to merge.

The core extension mappings are listed in the Topics section at the bottom of this page.

Extensions Properties

Each extension contains the following properties:

Property	Description
----------	-------------

Extension Scope	<ul style="list-style-type: none"> • All : Extension can be applied to the <code>./custom/application/</code> or <code>./custom/<module>/</code> directory • Application : Extension can only be applied to the <code>./custom/application/</code> directory • Module : Extension can only be applied to the <code>./custom/<module>/</code> directory
Definition Variable	The variable that Sugar for utilizing the extension definition. If defined, this variable must be set with the appropriate definition properties.
Extension Directory	<p>The directory that the extension compiles files from</p> <ul style="list-style-type: none"> • If the customization is for the application, the extension file should be placed in <code>./custom/Extension/application/Ext/<extension_directory>/</code> • If the customization is for a module, the extension file should be placed in <code>./custom/Extension/modules/<module>/Ext/<extension_directory>/</code>
Compiled Extension File	<p>The name of the compiled extension file</p> <ul style="list-style-type: none"> • If the extension is for the application, the compiled file will be located in <code>./custom/application/Ext/<extension>/<extension>.ext.php</code> • If the extension is for a module, the compiled file will be located in <code>./custom/modules/<module>/Ext/<extension>/<extension>.ext.php</code>
Manifest Installdef	The index of the <code>\$installdef</code> in the manifest file for module loadable packages.

ActionFileMap

Overview

The ActionFileMap extension maps actions to files, which helps you map a file to a view outside of `./custom/modules/<module>/views/view.<name>.php`. This page is only applicable to modules running in backward compatibility mode.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Module
Sugar Variable	<code>\$action_file_map</code>
Extension Directory	<code>./custom/Extension/modules/<module>/Ext/ActionFileMap/</code>
Compiled Extension File	<code>./custom/<module>/Ext/ActionFileMap/action_file_map.ext.php</code>
Manifest Installdef	<code>\$installdefs['action_file_map']</code>

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in `./custom/Extension/modules/<module>/Ext/ActionFileMap/` to map a new action in the system. The following example will create a new action called "example" in a module:

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

```
<?php  
  
$action_file_map['new_action'] =  
'custom/modules/<module>/new_action.php';
```

Next, create your action file:

```
./custom/modules/<module>/new_action.php
```

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

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and compile your customization into `./custom/modules/<module>/Ext/ActionFileMap/action_file_map.ext.php`

Module Loadable Package

When building a module-loadable package, you can use the `$installdefs['action_file_map']` index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The basepath of the file to be installed.
to_module	String	The key of the module the file is to be installed to.

The example below demonstrates the proper install definition that should be used in the `./manifest.php` file in order to add the Action File Map file to a specific module. You should note that when using this approach, you still need to use the `$installdefs['copy']` index for the Action file, but Sugar will automatically execute Rebuild Extensions to reflect the new Action in the system.

./manifest.php

```
<?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'ActionRemap_Example',
    'action_file_map' => array(
        array(
            'from' =>
'<basepath>/Files/custom/Extension/modules/<module>/Ext/ActionFileMap/
<file>.php',
            'to_module' => '<module>',
        )
    ),
    'copy' => array(
        array(
            'from' => '<basepath>/Files/custom/example.php',
            'to' => 'custom/example.php'
        )
    )
);
```

Alternatively, you may use the `$installdefs['copy']` index for the Action File Map Extension file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the `$installdefs['copy']` index and module-loadable packages, please refer to the Introduction to the Manifest page.

Last Modified: 01/04/2017 10:10am

ActionReMap

Overview

The ActionReMap extension maps new actions to existing actions. This extension is only applicable to modules running in backward compatibility mode.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Module
Sugar Variable	<code>\$action_remap</code>
Extension Directory	<code>./custom/Extension/modules/<module>/Ext/ActionReMap/</code>
Compiled Extension File	<code>./custom/<module>/Ext/ActionReMap/action_remap.ext.php</code>
Manifest Installdef	<code>\$installdefs['action_remap']</code>

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the file system, you can create a file in `./custom/Extension/modules/<module>/Ext/ActionReMap/` to map an action to another defined action. The following example will map the action 'example' to 'detailview':

```
./custom/Extension/modules/<module>/Ext/ActionReMap/<file>.php
```

```
<?php
```

```
$action_remap['example'] = 'detailview';
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and compile your customization into `./custom/modules/<module>/Ext/ActionReMap/action_remap.ext.php`.

Module Loadable Package

When building a module loadable package, you can use the `$installdefs['action_remap']` index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The basepath of the file to be installed
to_module	String	The key for the module where the file will be installed

The example below demonstrates the proper install definition that should be used in the `./manifest.php` file in order to add the Action Remap file to a specific module. You should note that when using this approach, Sugar will automatically execute Rebuild Extensions to reflect your changes in the system.

`./manifest.php`

```
<?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'ActionRemap_Example',
    'action_remap' => array(
        array(
            'from' =>
'<basepath>/Files/custom/Extension/modules/<module>/Ext/ActionReMap/<file>.php',
            'to_module' => '<module>',
        )
    )
);
```

Alternatively, you may use the `$installdefs['copy']` index. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the `$installdefs['copy']` index and module-loadable packages, please refer to the Introduction to the Manifest page.

ActionViewMap

Overview

The ActionViewMap extension maps additional actions for a module.

Note: Actions that apply to modules running in backward compatibility mode are mapped in `./custom/modules/<module>/controller.php`.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Module
Sugar Variable	<code>\$action_view_map</code>
Extension Directory	<code>./custom/Extension/modules/<module>/Ext/ActionViewMap/</code>
Compiled Extension File	<code>./custom/<module>/Ext/ActionViewMap/action_view_map.ext.php</code>
Manifest Installdef	<code>\$installdefs['action_view_map']</code>

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in `./custom/Extension/modules/<module>/Ext/ActionViewMap/` to map a new view in the system. The following example will map a new action called 'example' to the 'example' view:

`./custom/Extension/modules/<module>/Ext/ActionViewMap/<file>.php`

```
<?php
```

```
$action_view_map['example'] = 'example';
```

`./custom/modules/<module>/views/view.example.php`

```
<?php
```

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

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

```
class <module>ViewExample extends ViewDetail
{
    function <module>ViewExample()
    {
        parent::ViewDetail();
    }
    function display()
    {
        echo 'Example View';
    }
}
?>
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and compile your customization into `./custom/modules/<module>/Ext/ActionViewMap/action_view_map.ext.php`.

Module Loadable Package

When building a module-loadable package, use the `$installdefs['action_view_map']` index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The basepath of the file

to_module	String	The key for the module where the file will be installed
-----------	--------	---

The example below demonstrates the proper install definition for the `./manifest.php` file in order to add the Action View Map file to a specific module. You should note that when using this approach, you still need to use the `$installdefs['copy']` index for the View file, however, Sugar will automatically execute Rebuild Extensions to reflect the new Action View in the system.

`./manifest.php`

```
<?php
$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'actionView_example',
    'action_view_map' => array(
        array(
            'from' =>
'<basepath>/Files/custom/Extension/modules/<module>/Ext/ActionViewMap/
<file>.php',
            'to_module' => '<module>',
        )
    ),
    'copy' => array(
        array(
            'from' =>
'<basepath>/Files/custom/modules/<module>/views/view.example.php',
            'to' => 'custom/modules/<module>/views/view.example.php',
        )
    )
);
```

Alternatively, you may use the `$installdefs['copy']` index for the Action View Map Extension file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild.

For more information on module-loadable packages, please refer to the Introduction to the Manifest page .

Administration

Overview

The Administration extension adds new panels to Sugar's Administration page.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Module: Administration
Sugar Variable	\$admin_group_header
Extension Directory	./custom/Extension/modules/Administration/Ext/Administration/
Compiled Extension File	./custom/Administration/Ext/Administration/administration.ext.php
Manifest Installdef	\$installdefs['administration']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in `./custom/Extension/modules/Administration/Ext/Administration/<file>.php` to add new Administration Links in the system. The following example will add a new panel to the Administration page called "Example Admin Panel", which will contain one link called "Example Link":

The following example will create a new admin panel:

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

```
<?php

    $admin_option_defs = array();
    $admin_option_defs['Administration']['<section key>'] = array(
        //Icon name. Available icons are located in
        ./themes/default/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'] = 'Example Link';  
$mod_strings['LBL_LINK_DESCRIPTION'] = 'Link Description';  
$mod_strings['LBL_SECTION_HEADER'] = 'Example Admin Panel';  
$mod_strings['LBL_SECTION_DESCRIPTION'] = 'Section Description';
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions, compiling your customization into `./custom/modules/Administration/Ext/Administration/administration.ext.php`, and the new panel will appear in the Administration section.

Module Loadable Package

When building a module loadable package, use the `$installdefs['administration']` index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The base path of the file to be installed

The example below demonstrates the proper install definition that should be used in the `./manifest.php` file in order to add the Administration file to the system. If you are utilizing a Language file, as recommended above, you must use the `$installdefs['language']` index to install the Language definition. Using the `$installdefs['administration']` index will automatically execute Rebuild Extensions to reflect the new Administration Links in the system.

```
<?php
```

```
$manifest = array(  
    ...  
);  
  
$installdefs = array(  
    'id' => 'administration_example',  
    'administration' => array(  
        array(  

```

```
        'from' =>
'<basepath>/custom/Extension/modules/Administration/Ext/Administration
/<file>.php'
    )
),
'language' => array(
    array(
        'from' =>
'<basepath>/custom/Extensions/modules/Administration/Ext/Language/en_u
s.<file>.php',
        'to_module' => 'Administration',
        'language' => 'en_us'
    )
)
);
```

Alternatively, you may also choose to use the `$installdefs['copy']` index for the Administration Link Extension file. When using this approach, you may need to manually run a repair action such as a Quick Repair and Rebuild.

For more information on the `$installdefs['copy']` index and module-loadable packages, please refer to the Introduction to the Manifest page.

Last Modified: 01/04/2017 10:11am

Application Schedulers

Overview

Application Scheduler extensions add custom functions that can be used by scheduler jobs.

Note: This Extension is a duplicate of the ScheduledTasks extension, which typically places Scheduled Tasks in the Schedulers directory.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Application
Sugar Variable	\$job_strings
Extension Directory	./custom/Extension/application/Ext/ScheduledTasks/
Compiled Extension File	./custom/application/Ext/ScheduledTasks/scheduledtasks.ext.php
Manifest Installdef	\$installdefs['appscheduledefs']

Implementation

The following sections illustrates the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in `./custom/Extension/application/Ext/ScheduledTasks/` to add a new Scheduler Task to the system. The following example will create a new Scheduler Task 'example_job':

```
./custom/Extension/application/Ext/ScheduledTasks/<file>.php
```

```
<?php

$job_strings[] = 'exampleJob';

function exampleJob()
{
    //logic here

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

Next create the Language file for the Scheduler, so that the Job properly displays in Admin > Schedulers section:

```
./custom/Extension/modules/Schedulers/Ext/Language/<language>.<file>.php
```

```
<?php
//Label will be LBL_[upper case function name]
$mod_strings['LBL_EXAMPLEJOB'] = 'Example Job';
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and the customizations will be compiled into `./custom/application/Ext/ScheduledTasks/scheduledtasks.ext.php`

Module Loadable Package

When building a module loadable package, you can use the `$installdefs['appscheduleddefs']` index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The base path of the file to be installed.

The example below demonstrates the proper install definition that should be used in the `./manifest.php` file in order to add the custom Scheduler definition file to the system. You should note that, when using this approach, Sugar will automatically execute Rebuild Extensions to reflect the new scheduler in the system.

`./manifest.php`

```
<?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'appScheduledTasks_example',
    'appscheduleddefs' => array(
        array(
            'from' =>
'<basepath>/Files/custom/Extension/application/Ext/ScheduledTasks/<file>.php',
        )
    ),
    'language' => array(
```

```

        array(
            'from'
=>'<basepath>/Files/custom/Extension/modules/Schedulers/Ext/Language/<
file>.php',
            'to_module' => 'Schedulers',
            'language' => 'en_us'
        )
    )
);

```

Alternatively, you may use the `$installdefs['copy']` index for the Scheduled Task Extension file. When using this approach, you may need to manually run a repair action such as a Quick Repair and Rebuild.

For more information about the `$installdefs['copy']` index and module-loadable packages, please refer to the Introduction to the Manifest page.

Last Modified: 01/04/2017 10:11am

Console

Overview

The Console extension adds custom CLI commands to Sugar. More information on creating custom commands can be found in the CLI documentation.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Application
Extension Directory	./custom/Extension/application/Ext/Console/
Compiled Extension File	./custom/application/Ext/Console/console.ext.php
Manifest Installdef	<code>\$installdefs['console']</code>

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

To create a Sugar console extension, please refer to our CLI documentation.

Module Loadable Package

When building a module loadable package, you can use the `$installdefs['console']` index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The basepath of the file to be installed.

The example below demonstrates the proper install definition that should be used in the `./manifest.php` file in order to add the console command.

`./manifest.php`

```
<?php

$manifest = array(
    ...
);

$installdefs = array (
    'id' => 'console_Example',
    'console' => array(
        array(
            'from' =>
'<basepath>/Files/custom/Extension/application/Ext/Console/RegisterHelloWorldCommand.php'
        )
    )
)
```

```

),
'copy' => array (
    0 => array (
        'from' =>
'<basepath>/Files/custom/src/Console/Command/HelloWorldCommand.php',
        'to' => 'custom/src/Console/Command/HelloWorldCommand.php',
    ),
),
);

```

Alternatively, you may use the `$installdefs['copy']` index to copy the file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the `$installdefs['copy']` index and module-loadable packages, please refer to the Introduction to the Manifest page.

Download the module loadable example package [here](#).

Last Modified: 11/27/2017 12:24pm

Dependencies

Overview

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

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Module
Sugar Variable	<code>\$dependencies</code>
Extension Directory	<code>./custom/Extension/modules/<module>/Ext/Dependencies/</code>

Compiled Extension File	./custom/<module>/Ext/Dependencies/d eps.ext.php
Manifest Installdef	\$installdefs['dependencies']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in `./custom/Extension/modules/<module>/Ext/Dependencies/<file>.php` to map a new dependency in the system. The following example will create a new required field dependency on a module that is evaluated upon editing a record:

`./custom/Extension/modules/<module>/Ext/Dependencies/<file>.php`

```
<?php

$dependencies['<module>']['<unique name>'] = array(
    'hooks' => array("edit"),
    //Trigger formula for the dependency. Defaults to 'true'.
    'trigger' => 'true',
    'triggerFields' => array('<trigger field>'),
    'onload' => true,
    //Actions is a list of actions to fire when the trigger is true
    'actions' => array(
        array(
            'name' => 'SetRequired', //Action type
            //The parameters passed in depend on the action type
            'params' => array(
                'target' => '<field>',
                'label' => '<field label>', //normally <field>_label
                'value' => 'equal($<trigger field>, "Closed")',
            ),
        ),
    ),
);
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions, compiling your customization into `./custom/modules/<module>/Ext/Dependencies/deps.ext.php`, and the dependency will be in place.

Module Loadable Package

When building a module loadable package, you can use the `$installdefs['dependencies']` index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The base path of the file
to_module	String	The key for the module where the file will be installed

The example below demonstrates the proper install definition that should be used in the `./manifest.php` file in order to add the Dependency file to a specific module. You should note that when using this approach, Sugar will automatically execute Rebuild Extensions to reflect the new Dependency in the system.

`./manifest.php`

```
<?php
$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'dependency_example',
    'dependencies' => array(
        array(
            'from' =>
'<basepath>/Files/custom/Extension/modules/<module>/Ext/Dependencies/<
file>.php',
            'to_module' => '<module>',
        )
    )
)
```

);

Alternatively, you may use the `$installdefs['copy']` index for the Dependency Extension file. When using this approach, you may need to manually run a repair action such as a Quick Repair and Rebuild.

For more information on the `$installdefs['copy']` index and module-loadable packages, please refer to the Introduction to the Manifest page.

Last Modified: 01/04/2017 10:12am

EntryPointRegistry

Overview

The EntryPointRegistry extension maps additional entry points to the system. Please note that entry points will soon be deprecated. Developers should move custom logic to endpoints. For more information, please refer to the Entry Points page.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Application
Sugar Variable	<code>\$entry_point_registry</code>
Extension Directory	<code>./custom/Extension/application/Ext/EntryPointRegistry/</code>
Compiled Extension File	<code>./custom/application/Ext/EntryPointRegistry/entry_point_registry.ext.php</code>
Manifest Installdef	<code>\$installdefs['entrypoints']</code>

Implementation

The following sections illustrate the various ways to implement a customization to

a Sugar instance.

File System

When working directly with the filesystem, you can create a file in `./custom/Extension/application/Ext/EntryPointRegistry/` to map a new entry point in the system. The following example will create a new entry point called `exampleEntryPoint` that will display the text, "Hello World":

```
./custom/Extension/application/Ext/EntryPointRegistry/<file>.php
```

```
<?php

$entry_point_registry['exampleEntryPoint'] = array(
    'file' => 'custom/exampleEntryPoint.php',
    'auth' => true
);
```

Next, create the file that will contain the entry point logic. This file can be located anywhere you choose, but we recommend putting it in the custom directory. More information on custom entry points can be found on the [Creating Custom Entry Points](#) page.

```
./custom/exampleEntryPoint.php
```

```
<?php

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

echo "Hello World!";
```

Next, navigate to `Admin > Repair > Quick Repair and Rebuild`. The system will then rebuild the extensions and compile your customization into `./custom/application/Ext/EntryPointRegistry/entry_point_registry.ext.php`

Module Loadable Package

When building a module loadable package, you can use the `$installdefs['entrypoints']` index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The base path of the file to be installed.

The example below demonstrates the proper install definition that should be used in the `./manifest.php` file, in order to add the Entry Point Extension file to the system. You should note that when using this approach, you still need to use the `$installdefs['copy']` index for the entry point's logic file, however Sugar will automatically execute Rebuild Extensions to reflect the new Entry Point in the system.

`./manifest.php`

```
<?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'entryPoint_Example',
    'entrypoints' => array(
        array(
            'from' =>
'<basepath>/Files/custom/Extension/application/Ext/EntryPointRegistry/
<file>.php'
        )
    ),
    'copy' => array(
        array(
            'from' => '<basepath>/Files/custom/exampleEntryPoint.php',
            'to' => 'custom/exampleEntryPoint.php'
        )
    )
);
```

Alternatively, you may use the `$installdefs['copy']` index for the Entry Point Extension file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the `$installdefs['copy']` index and module loadable packages, please refer to the Introduction to the Manifest page.

Extensions

Overview

This extension allows for developers to create custom extensions within the framework. Custom extensions are used alongside the extensions found in `./ModuleInstaller/extensions.php`.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Application
Sugar Variable	<code>\$extensions</code>
Extension Directory	<code>./custom/Extension/application/Ext/Extensions/</code>
Compiled Extension File	<code>./custom/application/Ext/Extensions/extensions.ext.php</code>
Manifest Installdef	<code>\$installdefs['extensions']</code>

Parameters

- `section` : Section name in the manifest file
- `extdir` : The directory containing the extension files
- `file` : The name of the file where extension files are compiled into
- `module` : Determines how the framework will interpret the extension (optional)
 - `<Empty>` : Will enable the extension for all modules
 - `Ext` : `./custom/Extension/modules/<module>/Ext/<Custom Extension>/`
 - `Ext File` : `./custom/modules/<module>/Ext/<Extension Name>.ext.php`

-
- <Specific Module> : Will enable the extension for the specified module
 - Ext Directory : ./custom/Extension/modules/<Specific Module>/Ext/<Custom Extension>/
 - Ext File : ./custom/modules/<Specific Module>/Ext/<Extension Name>.ext.php
 - Application : enables the extension for application only
 - Ext Directory : ./custom/Extension/application/Ext/<Custom Extension>/
 - Ext File : ./custom/application/Ext/<Custom Extension>/<Extension Name>.ext.php

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in ./custom/Extension/application/Ext/Extensions/ to map a new extension in the system. The following example will create a new extension called "example", which is only enabled for "application":

```
./custom/Extension/application/Ext/Extensions/<file>.php
```

```
<?php
```

```
$extensions['example'] = array(  
    'section' => 'example',  
    'extdir' => 'Example',  
    'file' => 'example.ext.php',  
    'module' => 'application' //optional paramater  
);
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will rebuild the extensions and compile your customization into ./custom/application/Ext/Extensions/extensions.ext.php. Then you will be able to add your own extension files in ./custom/Extension/application/Ext/Example/.

Module Loadable Package

When building a module loadable package, you can use the `$installdefs['extensions']` index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The base path of the file to be installed

The example below will demonstrate the proper install definition that should be used in the `./manifest.php` file in order to add the Extension file to the system. You should note that when using this approach, Sugar will automatically execute `Rebuild Extensions` to reflect the new Extension in the system.

`./manifest.php`

```
<?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'customExtension_Example',
    'extensions' => array(
        array(
            'from' =>
'<basepath>/Files/custom/Extension/application/Ext/Extensions/<file>.p
hp'
        )
    )
);
```

Alternatively, you may use the `$installdefs['copy']` index for the Extension file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the `$installdefs['copy']` index and module-loadable packages, please refer to the Introduction to the Manifest page.

Last Modified: 01/04/2017 10:12am

FileAccessControlMap

Overview

The FileAccessControlMap extension restricts specific view actions from users of the system.

Note: This is only applicable to modules running in backward compatibility mode.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Module
Sugar Variable	<code>\$file_access_control_map</code>
Extension Directory	<code>./custom/Extension/modules/<module>/Ext/FileAccessControlMap/</code>
Compiled Extension File	<code>./custom/modules/<module>/Ext/FileAccessControlMap/file_access_control_map.ext.php</code>
Manifest Installdef	<code>\$installdefs['file_access']</code>

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in `./custom/Extension/modules/<module>/Ext/FileAccessControlMap/` to restrict a view of a module. The following example will create a new restriction for the detail view:

```
./custom/Extension/modules/<module>/Ext/FileAccessControlMap/<file>.php
```

```
<?php
```

```
$file_access_control_map['modules']['<lowercase module>']['actions'] =  
array(  
    'detailview',  
);
```

Navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and compile your customization into `./custom/modules/<module>/Ext/FileAccessControlMap/file_access_control_map.ext.php`

Module Loadable Package

When building a module loadable package, you can use the `$installdefs['file_access']` index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The base path of the file to be installed
to_module	String	The key of the module the file is to be installed to

The example below demonstrates the proper install definition that should be used in the `./manifest.php` file, in order to add the File Access Control Map file to a specific module. You should note that when using this approach, Sugar will automatically execute Rebuild Extensions to reflect the new Action in the system.

`./manifest.php`

```
<?php
```

```
$manifest = array(  
    ...  
);
```

```
$installdefs = array(  
    'id' => 'ActionRemap_Example',  
    'file_access' => array(  
        'from' => '...',  
        'to_module' => '...',  
    ),  
);
```

```

        array(
            'from' =>
'<basepath>/Files/custom/Extension/modules/<module>/Ext/FileAccessCont
rolMap/<file>.php',
            'to_module' => '<module>',
        )
    )
);

```

Alternatively, you may use the `$installdefs['copy']` index for the File Access Control Map Extension file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the `$installdefs['copy']` index and module-loadable packages, please refer to the Introduction to the Manifest page.

Last Modified: 01/04/2017 10:13am

Include

Overview

The Include extension maps additional modules in the system, typically when Module Builder deploys a module.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Application
Sugar Variable	<code>\$beanList</code> , <code>\$beanFiles</code> , <code>\$moduleList</code>
Extension Directory	<code>./custom/Extension/application/Ext/Include/</code>
Compiled Extension File	<code>./custom/application/Ext/Include/modules.ext.php</code>
Manifest Installdef	<code>\$installdefs['beans']</code>

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in `./custom/Extension/application/Ext/Include/` to register a new module in the system. This extension is normally used when deploying custom modules. The example below shows what this file will look like after a module is deployed:

```
./custom/Extension/application/Ext/Include/<file>.php
```

```
<?php
```

```
$beanList['cust_module'] = 'cust_module';  
$beanFiles['cust_module'] = 'modules/cust_module/cust_module.php';  
$moduleList[] = 'cust_module';
```

Next, navigate to **Admin > Repair > Quick Repair and Rebuild**. The system will then rebuild the extensions and compile your customization into `./custom/application/Ext/Include/modules.ext.php`.

Module Loadable Package

When building a module loadable package, you can use the `$installdefs['beans']` index to install the extension file.

Installdef Properties

Name	Type	Description
module	String	The key of the module to be installed
class	String	The class name of the module to be installed
path	String	The path to the module's class

tab	Boolean	Whether or not the module will have a navigation tab (defaults to false)
-----	---------	--

The example below demonstrates the proper install definition that should be used in the `./manifest.php` file, in order to add a custom module to the system. When using this approach, you still need to use the `$installdefs['copy']` index for Module directory, however, Sugar will automatically execute the database table creation process, relationship table creation process, as well as Rebuild Extensions to reflect the new Module in the system.

`./manifest.php`

```
<?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'Beans_Example',
    'beans' => array(
        array(
            'module' => 'cust_Module',
            'class' => 'cust_Module',
            'path' => 'modules/cust_Module/cust_Module.php',
            'tab' => true
        )
    ),
    'copy' => array(
        array(
            'from' => '<basepath>/Files/modules/cust_Module',
            'to' => 'modules/cust_Module'
        )
    )
);
```

Alternatively, you may use the `$installdefs['copy']` index for the Include Bean definition file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the `$installdefs['copy']` index and module loadable packages, please refer to the Introduction to the Manifest page.

Last Modified: 01/04/2017 10:13am

JSGroupings

Overview

The JSGroupings extension allows for additional JavaScript grouping files to be created or added to existing groupings within the system.

JSGroupings are file packages containing one or more minified JavaScript libraries. The groupings enhance system performance by reducing the number of JavaScript files that need to be downloaded for a given page. Some examples of JSGroupings in Sugar are `sugar_sidecar.min.js`, which contains the Sidecar JavaScript files, and `sugar_grp1.js`, which contains the base JavaScript files.

You can find all of the groups listed in `./jssource/JSGroupings.php`. Each group is loaded only when needed by a direct call (e.g., from a TPL file). For example, `sugar_grp1.js` is loaded for almost all Sugar functions, while `sugar_grp_yui_widgets.js` will usually be loaded for just record views.

To load a JSGroupings file for a custom module, simply add a new JSGrouping and then include the JavaScript file for your custom handlebars template. You can also

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Application
Sugar Variable	<code>\$js_groupings</code>
Extension Directory	<code>./custom/Extension/application/Ext/JSGroupings/</code>
Compiled Extension File	<code>./custom/application/Ext/JSGroupings/jsgroups.ext.php</code>
Manifest Installdef	<code>\$installdefs['jsgroups']</code>

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

Creating New JSGroupings

When working directly with the filesystem, you can create a file in `./custom/Extension/application/Ext/JSGroupings/` to add or append Javascript to JSGroupings in the system. The following example will add a new JSGrouping file to the system:

```
./custom/Extension/application/Ext/JSGroupings/<file>.php
```

```
<?php
```

```
//creates the file cache/include/javascript/newGrouping.js
$js_groupings[] = $newGrouping = array(
    'custom/file1.js' => 'include/javascript/newGrouping.js',
    'custom/file2.js' => 'include/javascript/newGrouping.js',
);
```

Next, create the Javascript files that will be grouped as specified in the JSGrouping definition above:

```
./custom/file1.js
```

```
function one(){
    //logic
}
```

```
./custom/file2.js
```

```
function two(){
    //logic
}
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and compile your customization into `./custom/application/Ext/JSGroupings/jsgroups.ext.php`.

Finally, navigate to Admin > Repair > Rebuild JS Grouping Files. This will create the grouping file in the cache directory as specified:

`./cache/include/javascript/newGrouping.js`

```
function one(){}/* End of File custom/file1.js */function two(){}/*
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. Similar to creating a new JSGrouping, to append to a core JSGrouping you can create a file in

`./custom/Extension/application/Ext/JSGroupings/`. The example below demonstrates how to add the file `./custom/file1.js` to `./cache/include/javascript/sugar_grp7.min.js`.

`./custom/Extension/application/Ext/JSGroupings/<file>.php`

```
<?php
```

```
//Loop through the groupings to find grouping file you want to append
to
```

```
foreach ($js_groupings as $key => $groupings)
{
    foreach ($groupings as $file => $target)
    {
        //if the target grouping is found
        if ($target == 'include/javascript/sugar_grp7.min.js')
        {
            //append the custom JavaScript file
            $js_groupings[$key]['custom/file1.js'] =
'include/javascript/sugar_grp7.min.js';
        }
        break;
    }
}
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and compile your customization into `./custom/application/Ext/JSGroupings/jsgroups.ext.php`.

Module Loadable Package

When building a module loadable package, you can use the `$installdefs['jsgroups']`

index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The basepath of the file to be installed
to_module	String	The key for the module where the file will be installed

The example below demonstrates the proper install definition that should be used in the `./manifest.php` file to add the JSGrouping Extension file to the system. When using this approach, you still need to use the `$installdefs['copy']` index for the custom JavaScript files you are adding to JSGroupings. Sugar will automatically execute Rebuild Extensions to reflect the new JSGrouping, however, you will still need to navigate to Admin > Repair > Rebuild JS Grouping Files, to create the grouping file in the cache directory.

`./manifest.php`

```
<?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'jsGroupings_Example',
    'jsgroups' => array(
        array(
            'from' =>
'<basepath>/Files/custom/Extension/application/Ext/JSGroupings/<file>.
php',
            'to_module' => 'application',
        )
    ),
    'copy' => array(
        array(
            'from' => '<basepath>/Files/custom/file1.js',
            'to' => 'custom/file1.js'
        )
    )
);
```

```
    array(  
        'from' => '<basepath>/Files/custom/file2.js',  
        'to' => 'custom/file2.js'  
    )  
);
```

Alternatively, you may use the `$installdefs['copy']` index for the JSGrouping Extension file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the `$installdefs['copy']` index and module-loadable packages, please refer to the Introduction to the Manifest page.

Considerations

- The grouping path you specify will be created in the cache directory.
- If you wish to add a grouping that contains a file that is part of another group already, add a `'.'` after `<file>.js` to make the element key unique.

Last Modified: 01/08/2017 01:34am

Language

Overview

The Language extension adds or overrides language strings.

This extension is applicable to both the application and module framework. For more information, please refer to the Language Framework page.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	All - Application & Module
Sugar Variables	<code>\$app_strings</code> / <code>\$app_list_strings</code> - when

	<p>adding language files to application</p> <p><code>\$mod_strings</code> - when adding language files to <module></p>
Extension Directory	<p>Application - ./custom/Extension/application/Ext/Language/</p> <p>Module - ./custom/Extension/modules/<module>/Ext/Language/</p>
Compiled Extension File	<p>Application - ./custom/application/Ext/Language/<language key>.lang.ext.php</p> <p>Module - ./custom/<module>/Ext/Language/<language key>.lang.ext.php</p>
Manifest Installdef	<code>\$installdefs['language']</code>

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

Creating New Application Label

When working directly with the filesystem, you can create a file in `./custom/Extension/application/Ext/Language/` to add Labels for languages to the system. The following example will add a new Label 'LBL_EXAMPLE_LABEL' to the system:

```
./custom/Extension/application/Ext/Language/<language>.<file>.php
```

```
<?php
    $app_strings['LBL_EXAMPLE_LABEL'] = 'Example Application Label';
```

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

into `./custom/application/Ext/Language/<language>.lang.ext.php`

Creating New Module Label

When working directly with the filesystem, you can create a file in `./custom/Extension/modules/<module>/Ext/Language/` to add Labels for languages to a particular module. The following example will add a new Label 'LBL_EXAMPLE_MODULE_LABEL' to a module:

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

```
<?php
    $mod_strings['LBL_EXAMPLE_MODULE_LABEL'] = 'Example Module Label';
```

Navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and compile your customization into `./custom/modules/<module>/Ext/Language/<language>.lang.ext.php`

Module Loadable Package

When building a module loadable package, you can use the `$installdefs['language']` index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The basepath of the file to be installed.
to_module	String	The key of the module the file is to be installed to.
language	String	The key of the language the file is to be installed to.

The example below will demonstrate the proper install definition that should be used in the `./manifest.php` file, in order to add the Language Extension file to the system. You should note that when using this approach Sugar will automatically execute Rebuild Extensions to reflect the new Labels in the system.

```
./manifest.php
```

```
<?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'language_Example',
    'language' => array(
        array(
            'from' =>
'<basepath>/Files/custom/Extension/application/Ext/Language/<language>
.lang.ext.php',
            'to_module' => 'application',
            'language' => '<language>'
        )
    )
);
```

Alternatively, you may use the `$installdefs['copy']` index to copy the file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the `$installdefs['copy']` index and module-loadable packages, please refer to the Introduction to the Manifest page.

Last Modified: 01/31/2018 10:27am

Layoutdefs

Overview

The Layoutdefs extension adds or overrides subpanel definitions.

Note: This extension is only applicable to modules running in backward compatibility mode.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Module
Sugar Variable	\$layout_defs
Extension Directory	./custom/Extension/modules/<module>/Ext/Layoutdefs/
Compiled Extension File	./custom/<module>/Ext/Layoutdefs/layoutdefs.ext.php
Manifest Installdef	\$installdefs['layoutdefs']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in `./custom/Extension/modules/<module>/Ext/Layoutdefs/` to add Layout Definition files to the system. The following example will add a subpanel definition for a module relationship:

`./custom/Extension/modules/<module>/Ext/Layoutdefs/<file>.php`

```
<?php
```

```

    $layout_defs["<module>"]["subpanel_setup"][ '<subpanel key>' ] =
array (
    'order' => 100,
    'module' => '<related module>',
    'subpanel_name' => 'default',
    'sort_order' => 'asc',
    'sort_by' => 'id',
    'title_key' => 'LBL_SUBPANEL_TITLE',
    'get_subpanel_data' => '<subpanel key>',
    'top_buttons' => array (
        array (
            'widget_class' => 'SubPanelTopButtonQuickCreate',
        ),
        array (
            'widget_class' => 'SubPanelTopSelectButton',

```

```

        'mode' => 'MultiSelect',
    ),
);

```

Please note that, if you are attempting to override parts of an existing subpanel definition, you should specify the exact index rather than redefining the entire array. An example of overriding the subpanel `top_buttons` index is shown below:

```

<?php

    $layout_defs["<module>"]["subpanel_setup"]["<subpanel
key>"]["top_buttons"] = array(
        array(
            'widget_class' => 'SubPanelTopButtonQuickCreate',
        ),
    );

```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and compile your customizations to `./custom/modules/<module>/Ext/Layoutdefs/layoutdefs.ext.php`.

Module Loadable Package

When building a module loadable package, you can use the `$installdefs['layoutdefs']` index to install the extension file.

Installdef Properties

Name	Type	Description
<code>from</code>	String	The base path of the file to be installed
<code>to_module</code>	String	The key for the module where the file will be installed

The example below demonstrates the proper install definition that should be used in the `./manifest.php` file in order to add the Layout Definition Extension file to the system. When using this approach, Sugar will automatically execute Rebuild Extensions to reflect the customizations added with the Layout definition.

./manifest.php

```
<?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'layoutdefs_Example',
    'layoutdefs' => array(
        array(
            'from' =>
'<basepath>/Files/custom/Extension/modules/<module>/Ext/Layoutdefs/<file>.php',
            'to_module' => '<module>',
        )
    )
);
```

Alternatively, you may use the `$installdefs['copy']` index to copy the file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the `$installdefs['copy']` index and module-loadable packages, please refer to the Introduction to the Manifest page.

Last Modified: 01/04/2017 10:14am

LogicHooks

Overview

The LogicHooks extension adds actions to specific events such as, for example, before saving a bean. For more information on logic hooks in Sugar, please refer to the Logic Hooks documentation.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	All - Application & Module
Sugar Variable	\$hook_array
Extension Directory	Application - ./custom/Extension/application/Ext/LogicHooks/ Module - ./custom/Extension/modules/<module>/Ext/LogicHooks/
Compiled Extension File	Application - ./custom/Extension/application/Ext/LogicHooks/ Module - ./custom/Extension/modules/<module>/Ext/LogicHooks/
Manifest Installdef	\$installdefs['hookdefs']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in `./custom/Extension/application/Ext/LogicHooks/` to add a new logic hook to the application. The following example will create a new `before_save` logic hook that executes for all modules:

```
./custom/Extension/application/Ext/LogicHooks/<file>.php
```

```
<?php
    $hook_array['before_save'][] = Array(
        1,
        'Custom Logic',
        'custom/application_hook.php',
        'ApplicationHookConsumer',
```

```
        'before_method'  
    );
```

Next, create the hook class:

`./custom/application_hook.php`

```
<?php  
  
    if (!defined('sugarEntry') || !sugarEntry) die('Not A Valid Entry  
Point');  
  
    class ApplicationHookConsumer  
    {  
        function before_method($bean, $event, $arguments)  
        {  
            //logic  
        }  
    }  
  
?>
```

Finally, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and the customizations will be compiled into `./custom/application/Ext/LogicHooks/logichooks.ext.php`. Your logic hook will run before saving records in any module.

Module Loadable Package

When building a module loadable package, you can use the `$installdefs['hookdefs']` index to install the extension file.

Installdef Properties

Name	Type	Description
<code>from</code>	String	The basepath of the file to be installed.
<code>to_module</code>	String	The key of the module the file is to be installed to.

The example below demonstrates the proper install definition that should be used

in the `./manifest.php` file in order to add the Action View Map file to a specific module. You should note that when using this approach, you still need to use the `$installdefs['copy']` index for the hook class file, however Sugar will automatically execute Rebuild Extensions to reflect the new logic hook in the system.

`./manifest.php`

```
<?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'actionView_example',
    'hookdefs' => array(
        array(
            'from' =>
'<basepath>/Files/custom/Extension/application/Ext/LogicHooks/<file>.p
hp',
            'to_module' => 'application',
        )
    ),
    'copy' => array(
        array(
            'from' => '<basepath>/Files/custom/application_hook.php',
            'to' => 'custom/application_hook.php',
        )
    )
);
```

Alternatively, you may use the `$installdefs['copy']` index to copy the file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the `$installdefs['copy']` index and module-loadable packages, please refer to the Introduction to the Manifest page.

Alternative Installdef

Although not recommended, you could utilize the `$installdefs['logic_hooks']` index to deploy a logic hook to the system. Please note that there are a couple caveats to this installation method:

- The `$installdefs['logic_hooks']` index method only works for module-based logic hooks.
- The `$installdefs['logic_hooks']` index method installs to `./custom/modules/<module>/logic_hooks.php`, which is not part of the Extension framework.

Properties

Name	Type	Description
module	String	The key of the module for the logic hook to be installed to.
hook	String	The type of logic hook to be installed.
order	Integer	The number in which the logic hook should run.
description	String	A description of the logic hook to be installed.
file	String	The file path which contains the logic hook class.
class	String	The class which houses the logic hook functionality.
function	String	The function the logic hook will execute.

The example below will demonstrate the `$installdefs['logic_hooks']` index in the `./manifest.php` file, in order to add an after save logic hook to a specific module. You should note that when using this approach, you still need to use the `$installdefs['copy']` index for the hook class file.

`./manifest.php`

```
<?php
$manifest = array(
    ...
);
```

```

$installdefs = array(
    'id' => 'actionView_example',
    'logic_hooks' => array(
        array(
            'module' => '<module>',
            'hook' => 'after_save',
            'order' => 1,
            'description' => 'Example After Save LogicHook',
            'file' => 'custom/modules/<module>/module_hook.php',
            'class' => '<module>HookConsumer'
            'function' => 'after'
        )
    ),
    'copy' => array(
        array(
            'from' =>
'<basepath>/Files/custom/modules/<module>/module_hook.php',
            'to' => 'custom/modules/<module>/module_hook.php',
        )
    )
);

```

Last Modified: 01/04/2017 10:14am

ScheduledTasks

Overview

The ScheduledTasks extension adds custom functions that can be used by scheduler jobs. For more information about schedulers in Sugar, please refer to the Schedulers documentation.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Module: Schedulers
Sugar Variable	\$job_strings

Extension Directory	./custom/Extension/modules/Schedulers/Ext/ScheduledTasks/
Compiled Extension File	./custom/Schedulers/Ext/ScheduledTasks/scheduledtasks.ext.php
Manifest Installdef	\$installdefs['scheduleddefs']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in `./custom/Extension/modules/Schedulers/Ext/ScheduledTasks/` to add a new Scheduler Task to the system. The following example will create a new Scheduler Task 'example_job':

```
./custom/Extension/modules/Schedulers/Ext/ScheduledTasks/<file>.php
```

```
<?php
$job_strings[] = 'exampleJob';

function exampleJob()
{
    //logic here

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

Next, create the Language file for the scheduler so that the job properly displays in Admin > Schedulers:

```
./custom/Extension/modules/Schedulers/Ext/Language/<language>.<file>.php
```

```
<?php
//Label will be LBL_[upper case function name]
$mod_strings['LBL_EXAMPLEJOB'] = 'Example Job';
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and the customizations will be compiled into `./custom/modules/Schedulers/Ext/ScheduledTasks/scheduledtasks.ext.php`

Module Loadable Package

When building a module loadable package, you can use the `$installdefs['scheduledefs']` index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The basepath of the file to be installed

The example below demonstrates the proper install definition that should be used in the `./manifest.php` file in order to add the custom Scheduler definition file to the system. When using this approach, Sugar will automatically execute Rebuild Extensions to reflect the new Scheduler in the system.

`./manifest.php`

```
<?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'actionView_example',
    'scheduledefs' => array(
        array(
            'from' =>
'<basepath>/Files/custom/Extension/modules/Scheduler/Ext/ScheduledTask
s/<file>.php',
        )
    ),
    'language' => array(
        array(
            'from'
=>'<basepath>/Files/custom/Extension/modules/Schedulers/Ext/Language/<
file>.php',
```

```

        'to_module' => 'Schedulers',
        'language' => 'en_us'
    )
)
);

```

Alternatively, you may use the `$installdefs['copy']` index to copy the file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the `$installdefs['copy']` index and module-loadable packages, please refer to the Introduction to the Manifest page.

Last Modified: 01/04/2017 10:15am

Sidecar

Overview

The Sidecar extension installs metadata files to their appropriate directories.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Module
Sugar Variable	<code>\$viewdefs</code>
Extension Directory	<code>./custom/Extension/modules/<module>/Ext/clients/<client>/<type>/<subtype>/</code>
Compiled Extension File	<code>./custom/<module>/Ext/clients/<client>/<type>/<subtype>/<subtype>.ext.php</code>
Manifest Installdef	<code>\$installdefs['sidecar']</code>

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in `./custom/Extension/modules/<module>/Ext/clients/<client>/<type>/<subtype>/` to append the metadata extensions. The example below demonstrates how to add a new subpanel to a specific module:

```
./custom/Extension/modules/<module>/Ext/clients/base/layouts/subpanels/<file>.php
```

```
<?php
```

```
$viewdefs['<module>']['base']['layout']['subpanels']['components'][] =  
array(  
    'layout' => 'subpanel',  
    'label' => 'LBL_RELATIONSHIP_TITLE',  
    'context' => array(  
        'link' => '<link_name>',  
    )  
);
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and compile your customization into `./custom/modules/<module>/Ext/clients/base/layouts/subpanels/subpanels.ext.php`

Module Loadable Package

When building a module loadable package, you can use the `$installdefs['sidecar']` index to install the metadata file.

Installdef Properties

Name	Type	Description
from	String	The base path of the file to be installed Note: When adding the file to a module loadable package, its 'from' path must be formatted as

		clients/<client>/<type>/<subtype>/<file>.php for Sugar to recognize the installation location.
to_module	String	<ul style="list-style-type: none"> • The key for the module where the file will be installed • If not populated, 'application' is used

The example below demonstrates the proper install definition that should be used in the `./manifest.php` file in order to add the metadata file to a specific module. When using this approach, Sugar will automatically execute Rebuild Extensions and Metadata Rebuild to reflect your changes in the system.

`./manifest.php`

```
<?php

$manifest = array(
    ...
);

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

Alternatively, you may use the `$installdefs['copy']` index to copy the file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the `$installdefs['copy']` index and module-loadable packages, please refer to the Introduction to the Manifest page.

Last Modified: 01/04/2017 10:15am

TinyMCE

Overview

The TinyMCE extension affects the TinyMCE WYSIWYG editor's configuration for backward compatible modules such as PDF Manager and Campaign Email Templates.

To review the default configuration for TinyMCE, please refer to the code in `./include/SugarTinyMCE.php`. Sidecar's TinyMCE configuration can be edited using the Sidecar Framework and editing the `htmleditable_tinymce` field.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Application
Sugar Variable	<code>\$defaultConfig</code> , <code>\$buttonConfigs</code> , <code>\$pluginsConfig</code>
Extension Directory	<code>./custom/Extension/application/Ext/TinyMCE/</code>
Compiled Extension File	<code>./custom/application/Ext/TinyMCE/tinymce.ext.php</code>
Manifest Installdef	<code>\$installdefs['tinymce']</code>

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in `./custom/Extension/application/Ext/TinyMCE/` to customize the TinyMCE configuration. The following example will increase the height of the TinyMCE

window, remove buttons, and remove plugins from the WYSIWYG Editor:

```
./custom/Extension/application/Ext/TinyMCE/<file>.php
```

```
<?php

$defaultConfig['height'] = '1000';

$buttonConfigs['default'] = array(
    'buttonConfig' =>
    "bold,italic,underline,strikethrough,separator,bullist,numlist",
    'buttonConfig2' =>
    "justifyleft,justifycenter,justifyright,justifyfull",
    'buttonConfig3' => "fontselect,fontsizeselect",
);

$pluginsConfig['default'] = 'advhr,preview,paste,directionality';
```

Navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and compile your customization into `./custom/application/Ext/TinyMCE/tinymce.ext.php`

Module Loadable Package

When building a module loadable package, you can use the `$installdefs['tinymce']` index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The base path of the file to be installed

The example below demonstrates the proper install definition that should be used in the `./manifest.php` file, in order to add the UserPage file to the system. You should note that when using this approach Sugar will automatically execute Rebuild Extensions to reflect the changes to TinyMCE in the system.

```
./manifest.php
```

```
<?php
```

```

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'tinyMCE_Example',
    'tinymce' => array(
        array(
            'from' =>
'<basepath>/Files/custom/Extension/application/Ext/TinyMCE/<file>.php'
,
            'to_module' => 'application'
        )
    )
);

```

Alternatively, you may use the `$installdefs['copy']` index to copy the file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the `$installdefs['copy']` index and module loadable packages, please refer to the Introduction to the Manifest page.

Last Modified: 01/04/2017 10:15am

UserPage

Overview

The UserPage extension adds sections to the User Management view.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Module: Users
Extension Directory	./custom/Extension/modules/Users/Ext/UserPage/
Compiled Extension File	./custom/Users/Ext/UserPage/userpage.

	ext.php
Manifest Installdef	\$installdefs['user_page']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in `./custom/Extension/modules/Users/Ext/UserPage/` to add custom elements to the User page. The following example will add a custom table to the Users module's detail view:

`./custom/Extension/modules/Users/Ext/UserPage/<file>.php`

```
<?php
$HTML=<<<HTML

    <table cellpadding="0" cellspacing="0" width="100%" border="0"
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 compile your customization into `./custom/modules/Users/Ext/UserPage/userpage.ext.php`

Module Loadable Package

When building a module loadable package, you can use the `$installdefs['user_page']` index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The base path of the file to be installed

The example below demonstrates the proper install definition that should be used in the `./manifest.php` file in order to add the UserPage file to the system. When using this approach, Sugar will automatically execute Rebuild Extensions to reflect the changes to the User view in the system.

`./manifest.php`

```
<?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'userPage_Example',
    'user_page' => array(
        array(
            'from' =>
'<basepath>/Files/custom/Extension/modules/Users/Ext/UserPage/<file>.p
hp',
        )
    )
);
```

Alternatively, you may use the `$installdefs['copy']` index to copy the file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the `$installdefs['copy']` index and module-loadable packages, please refer to the Introduction to the Manifest page.

Utils

Overview

The Utils extension adds functions to the global utility function list.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Application
Extension Directory	./custom/Extension/application/Ext/Utils/
Compiled Extension File	./custom/application/Ext/Utils/custom_utils.ext.php
Manifest Installdef	\$installdefs['utils']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in `./custom/Extension/application/Ext/Utils/` to map a new action in the system. The following example will create a new function called 'exampleUtilFunction' that can be used throughout the system:

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

```
<?php
```

```
function exampleUtilFunction()
{
    //logic
}
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and your customizations will be compiled into `./custom/application/Ext/Utils/custom_utils.ext.php`.

Alternatively, functions can also be added by creating `./custom/include/custom_utils.php`. This method of creating utils is still compatible but is not recommended from a best practices standpoint.

Module Loadable Package

When building a module loadable package, you can use the `$installdefs['utils']` index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The base path of the file to be installed

The example below demonstrates the proper install definition that should be used in the `./manifest.php` file in order to add the utils to the system. You should note that when using this approach, Sugar will automatically execute Rebuild Extensions to reflect the new utils in the system.

`./manifest.php`

```
<?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'utils_Example',
    'utils' => array(
        array(
            'from' =>
```

```
'<basepath>/Files/custom/Extension/application/Ext/Utils/<file>.php',  
    )  
);
```

Alternatively, you may use the `$installdefs['copy']` index to copy the file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the `$installdefs['copy']` index and module-loadable packages, please refer to the Introduction to the Manifest page.

Last Modified: 01/04/2017 10:16am

Vardefs

Overview

The Vardefs extension adds or overrides system vardefs, which provide the Sugar application with information about SugarBeans.

For more information on vardefs in Sugar, please refer to the Vardefs documentation .

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Module
Sugar Variable	\$dictionary
Extension Directory	./custom/Extension/modules/<module>/Ext/Vardefs/
Compiled Extension File	./custom/<module>/Ext/Vardefs/vardefs.ext.php
Manifest Installdef	\$installdefs['vardefs']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in `./custom/Extension/modules/<module>/Ext/Vardefs/` to edit or add vardefs to a module in the system.

The most common use of the Vardef extension is to alter the attributes of an existing vardef. To do this, avoid redefining the entire vardef and instead update the specific index you want to change. The following example updates the Required property on the Name field in a module:

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

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

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and your customizations will be compiled into `./custom/modules/<module>/Ext/Vardefs/vardefs.ext.php`.

Notice Never specify vardefs for a module under another module's extension path. For example, do not specify `$dictionary['Accounts']['fields']['name']['required'] = false` under `./custom/Extension/modules/Contacts/Ext/Vardefs/`. Doing so will result in unexpected behavior within the system.

Module Loadable Package

When building a module loadable package, you can use the `$installdefs['vardefs']` index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The base path of the file to be installed
to_module	String	The key for the module where the file will be installed

The example below demonstrates the proper install definition that should be used in the `./manifest.php` file in order to add the Vardefs file to a specific module. You should note that when using this approach Sugar will automatically execute `Rebuild Extensions` to reflect the vardef changes in the system.

`./manifest.php`

```
<?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'vardefs_Example',
    'vardefs' => array(
        array(
            'from' =>
'<basepath>/Files/custom/Extension/modules/<module>/Ext/Vardefs/<file>
.php',
            'to_module' => '<module>',
        )
    )
);
```

Alternatively, you may use the `$installdefs['copy']` index to copy the file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the `$installdefs['copy']` index and module-loadable packages, please refer to the [Introduction to the Manifest page](#).

Creating Custom Fields

If your goal is to manually create a custom field on an instance, you should be using the Module Installer to create the field. This can be used both for an installer package and programmatically. An example of creating a field from a module loadable package can be found under [Package Examples](#) in the article, [Creating an Installable Package that Creates New Fields](#). An example of programmatically creating a field can be found in the [Manually Creating Custom Fields](#) section of the [Module Vardefs](#) documentation.

Last Modified: 04/17/2017 11:04am

WirelessLayoutdefs

Overview

The WirelessLayoutdefs extension adds additional subpanels to wireless views. This extension is only applicable to modules running in backward compatibility mode.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Module
Sugar Variable	\$layout_defs
Extension Directory	./custom/Extension/modules/<module>/Ext/WirelessLayoutdefs/
Compiled Extension File	./custom/<module>/Ext/WirelessLayoutdefs/wireless.subpaneldefs.ext.php
Manifest Installdef	\$installdefs['wireless_subpanels']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in `./custom/Extension/modules/<module>/Ext/WirelessLayoutdefs/` to add a subpanel to a module in the system. The following example will add a new subpanel to a specified module:

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

```
<?php
```

```
$layout_defs['<module>']['subpanel_setup']['<subpanel module>'] =  
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 and compile your customization into `./custom/modules/<module>/Ext/WirelessLayoutdefs/wireless.subpaneldefs.ext.php`.

Module Loadable Package

When building a module loadable package, you can use the `$installdefs['wireless_subpanels']` index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The base path of the file to be installed
to_module	String	The key for the module where the file will be installed

The example below will demonstrate the proper install definition that should be used in the `./manifest.php` file in order to add the subpanel file to a specific module. You should note that when using this approach Sugar will automatically execute Rebuild Extensions to reflect the subpanel in the system.

`./manifest.php`

```
<?php
```

```
$manifest = array(  
    ...  
);
```

```

$installdefs = array(
    'id' => 'wirelessLayoutdefs_Example',
    'wireless_subpanels' => array(
        array(
            'from' =>
'<basepath>/Files/custom/Extension/modules/<module>/Ext/WirelessLayout
defs/<file>.php',
            'to_module' => '<module>',
        )
    )
);

```

Alternatively, you may use the `$installdefs['copy']` index to copy the file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the `$installdefs['copy']` index and module-loadable packages, please refer to the Introduction to the Manifest page.

Last Modified: 01/04/2017 10:16am

WirelessModuleRegistry

Overview

The `WirelessModuleRegistry` extension adds modules to the available modules for mobile.

Properties

The following extension properties are available. For more information, please refer to the Extension Property documentation.

Property	Value
Extension Scope	Application
Sugar Variable	<code>\$wireless_module_registry</code>
Extension Directory	<code>./custom/Extension/application/Ext/WirelessModuleRegistry/</code>
Compiled Extension File	<code>./custom/application/Ext/WirelessModuleRegistry/wireless_module_registry.ext.</code>

	php
Manifest Installdef	\$installdefs['wireless_modules']

Implementation

The following sections illustrate the various ways to implement a customization to a Sugar instance.

File System

When working directly with the filesystem, you can create a file in `./custom/Extension/application/Ext/WirelessModuleRegistry/` to add modules to the list of available modules for mobile. The following example will add a new module, 'cust_module', to the list of available modules for mobile:

`./custom/Extension/application/Ext/WirelessModuleRegistry/<file>.php`

```
<?php
$wireless_module_registry['cust_module'] = array(
    //enables/disables record creation
    'disable_create' => false,
);
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and compile your customization into `./custom/application/Ext/WirelessModuleRegistry/wireless_module_registry.ext.php`.

Module Loadable Package

When building a module loadable package, you can use the `$installdefs['wireless_modules']` index to install the extension file.

Installdef Properties

Name	Type	Description
from	String	The base path of the file to be installed.

to_module	String	The key of the module the file is to be installed to.
-----------	--------	---

The example below demonstrates the proper install definition that should be used in the `./manifest.php` file in order to add the Wireless Module Registry file to the system. You should note that when using this approach, Sugar will automatically execute Rebuild Extensions to reflect the new module in the mobile application.

`./manifest.php`

```
<?php

$manifest = array(
    ...
);

$installdefs = array(
    'id' => 'wirelessModules_Example',
    'wireless_modules' => array(
        array(
            'from' =>
'<basepath>/Files/custom/Extension/application/Ext/WirelessModuleRegis
try/<file>.php',
            'to_module' => 'application',
        )
    )
);
```

Alternatively, you may use the `$installdefs['copy']` index to copy the file. When using this approach, you may need to manually run repair actions such as a Quick Repair and Rebuild. For more information on the `$installdefs['copy']` index and module-loadable packages, please refer to the Introduction to the Manifest page.

Last Modified: 01/04/2017 10:16am

Filters

Overview

Filters are a way to predefine searches on views that render a list of records such as list views, pop-up searches, and lookups. This page explains how to implement the various types of filters for record list views.

Filters contain the following properties:

Property	Type	Description
id	String	A unique identifier for the filter
name	String	The label key for the display label of the filter
filter_definition	Array	The filter definition to apply to the results
editable	Boolean	Determines whether the user can edit the filter. Note: If you are creating a predefined filter for a user, this should be set to false
is_template	Boolean	Used with initial pop-up filters to determine if the filter is available as a template

Note: If a filter contains custom fields, those fields must be search-enabled in Studio > {Module Name} > Layouts > Search.

Operators

Operators, defined in `./clients/base/filters/operators/operators.php`, are expressions used by a filter to represent query operators. They are constructed in the `filter_definition` to help generate the appropriate query syntax that is sent to the database for a specific field type. Operators can be defined on a global or module level. The accepted paths are listed below:

- `./clients/base/filters/operators/operators.php`
- `./custom/clients/base/filters/operators/operators.php`
- `./modules/<module>/clients/base/filters/operators.php`
- `./custom/modules/<module>/clients/base/filters/operators.php`

The list of stock operators is shown below:

Operator	Label / Key	Description

\$contains	<ul style="list-style-type: none"> • is any of / LBL_OPERATOR_CONTAINS <ul style="list-style-type: none"> ◦ Used by multienum 	Matches anything that contains the value.
\$empty	<ul style="list-style-type: none"> • is empty / LBL_OPERATOR_EMPTY <ul style="list-style-type: none"> ◦ Used by enum and tag 	Matches on empty values.
\$not_contains	<ul style="list-style-type: none"> • is not any of / LBL_OPERATOR_NOT_CONTAINS ◦ Used by multienum 	Matches anything that does not contain the specified value.
\$not_empty	<ul style="list-style-type: none"> • is not empty / LBL_OPERATOR_NOT_EMPTY <ul style="list-style-type: none"> ◦ Used by enum and tag 	Matches on non-empty values.
\$in	<ul style="list-style-type: none"> • is any of / LBL_OPERATOR_CONTAINS <ul style="list-style-type: none"> ◦ Used by enum, int, relate, and tag 	Finds anything where field matches one of the values as specified as an array.
\$not_in	<ul style="list-style-type: none"> • is not any of / LBL_OPERATOR_NOT_CONTAINS ◦ Used by enum, relate, and tag 	Finds anything where the field does not match any of the values in the specified array of values.
\$equals	<ul style="list-style-type: none"> • exactly matches / LBL_OPERATOR_MATCHES <ul style="list-style-type: none"> ◦ Used by varchar, name, email, text, and textarea • is equal to / LBL_OPERATOR_EQUALS <ul style="list-style-type: none"> ◦ Used by 	Performs an exact match on that field.

	<p>currency, int, double, float, decimal, and date</p> <ul style="list-style-type: none"> • is / LBL_OPERATOR_IS <ul style="list-style-type: none"> ◦ Used by bool, teamset, phone, radioenum, and parent 	
\$starts	<ul style="list-style-type: none"> • starts with / LBL_OPERATOR_STARTS_WITH <ul style="list-style-type: none"> ◦ Used by varchar, name, email, text, textarea, and phone • is equal to / LBL_OPERATOR_EQUALS <ul style="list-style-type: none"> ◦ Used by datetime and datetimecombo 	Matches on anything that starts with the value.
\$not_equals	<ul style="list-style-type: none"> • is not equal to / LBL_OPERATOR_NOT_EQUALS <ul style="list-style-type: none"> ◦ Used by currency, int, double, float, and decimal • is not / LBL_OPERATOR_IS_NOT <ul style="list-style-type: none"> ◦ Used by teamset and radioenum 	Matches on non-matching values.
\$gt	<ul style="list-style-type: none"> • is greater than / LBL_OPERATOR_GREATER_THAN <ul style="list-style-type: none"> ◦ Used by currency, int, double, float, and decimal • after / LBL_OPERATOR_AFTER <ul style="list-style-type: none"> ◦ Used by date 	Matches when the field is greater than the value.
\$lt		Matches when the field is

	<ul style="list-style-type: none"> • is less than / LBL_OPERATOR_LESS_THAN <ul style="list-style-type: none"> ◦ Used by currency, int, double, float, and decimal • before / LBL_OPERATOR_BEFORE <ul style="list-style-type: none"> ◦ Used by date 	less than the value.
\$gte	<ul style="list-style-type: none"> • is greater than or equal to / LBL_OPERATOR_GREATER_TH AN_OR_EQUALS <ul style="list-style-type: none"> ◦ Used by currency, int, double, float, and decimal • after / LBL_OPERATOR_AFTER <ul style="list-style-type: none"> ◦ Used by datetime and datetimecombo 	Matches when the field is greater than or equal to the value
\$lte	<ul style="list-style-type: none"> • is less than or equal to / LBL_OPERATOR_LESS_THAN_O R_EQUALS <ul style="list-style-type: none"> ◦ Used by currency, int, double, float, and decimal • before / LBL_OPERATOR_BEFORE <ul style="list-style-type: none"> ◦ Used by datetime and datetimecombo 	Matches when the field is less than or equal to the value.
\$between	<ul style="list-style-type: none"> • is between / LBL_OPERATOR_BETWEEN <ul style="list-style-type: none"> ◦ Used by currency, int, double, float, and decimal 	Matches when a numerical value is between two other numerical values.

last_7_days	<ul style="list-style-type: none"> • last 7 days / LBL_OPERATOR_LAST_7_DAYS <ul style="list-style-type: none"> ◦ Used by date, datetime, and datetimecombo 	Matches date in the last 7 days relative to the current date.
next_7_days	<ul style="list-style-type: none"> • next 7 days / LBL_OPERATOR_NEXT_7_DAYS <ul style="list-style-type: none"> ◦ Used by date, datetime, and datetimecombo 	Matches dates in the next 7 days relative to the current date.
last_30_days	<ul style="list-style-type: none"> • last 30 days / LBL_OPERATOR_LAST_30_DAYS <ul style="list-style-type: none"> ◦ Used by date, datetime, and datetimecombo 	Matches dates in the last 30 days relative to the current date.
next_30_days	<ul style="list-style-type: none"> • next 30 days / LBL_OPERATOR_NEXT_30_DAYS <ul style="list-style-type: none"> ◦ Used by date, datetime, and datetimecombo 	Matches dates in the next 30 days relative to the current date.
last_month	<ul style="list-style-type: none"> • last month / LBL_OPERATOR_LAST_MONTH <ul style="list-style-type: none"> ◦ Used by date, datetime, and datetimecombo 	Matches dates in the previous month relative to the current month.
this_month	<ul style="list-style-type: none"> • this month / LBL_OPERATOR_THIS_MONTH <ul style="list-style-type: none"> ◦ Used by date, datetime, and datetimecombo 	Matches dates in the current month.
next_month	<ul style="list-style-type: none"> • next month / LBL_OPERATOR_NEXT_MONTH <ul style="list-style-type: none"> ◦ Used by date, datetime, and 	Matches dates in the next month relative to the current month.

	datetimecombo	
last_year	<ul style="list-style-type: none"> • last year / LBL_OPERATOR_LAST_YEAR <ul style="list-style-type: none"> ◦ Used by date, datetime, and datetimecombo 	Matches dates in the last year relative to the current year.
this_year	<ul style="list-style-type: none"> • this year / LBL_OPERATOR_THIS_YEAR <ul style="list-style-type: none"> ◦ Used by date, datetime, and datetimecombo 	Matches dates in the current year.
next_year	<ul style="list-style-type: none"> • next year / LBL_OPERATOR_NEXT_YEAR <ul style="list-style-type: none"> ◦ Used by date, datetime, and datetimecombo 	Matches dates in the next year relative to the current year.
\$dateBetween	<ul style="list-style-type: none"> • is between / LBL_OPERATOR_BETWEEN <ul style="list-style-type: none"> ◦ Used by date, datetime, and datetimecombo 	Matches dates between two given dates.
yesterday	<ul style="list-style-type: none"> • yesterday / LBL_OPERATOR_YESTERDAY <ul style="list-style-type: none"> ◦ Used by date, datetime, and datetimecombo 	Matches dates on yesterday relative to the current date.
today	<ul style="list-style-type: none"> • today / LBL_OPERATOR_TODAY <ul style="list-style-type: none"> ◦ Used by date, datetime, and datetimecombo 	Matches dates in the current date.
tomorrow	<ul style="list-style-type: none"> • tomorrow / LBL_OPERATOR_TOMORROW <ul style="list-style-type: none"> ◦ Used by date, datetime, and 	Matches dates on tomorrow relative to the current date.

Example

The example below defines a filter where the type field must contain the value Customer and the name field must start with the letter A.

```
$filters = array(
    array(
        'type' => array(
            '$in' => array(
                'Customer',
            ),
        ),
    ),
    array(
        'name' => array(
            '$starts' => 'A',
        ),
    ),
);
```

Sub-Expressions

Sub-expressions group filter expressions into groupings. By default, all expressions are bound by an \$and expression grouping.

Sub-Expression	Description
\$and	Joins the filters in an "and" expression
\$or	Joins the filters in an "or" expression

Note: Sub-Expressions are only applicable to predefined filters and cannot be used for initial filters.

The example below defines a filter where the name field must begin with the letters A or C.

```
$filters = array(
    '$or' => array (
        array(
            'name' => array(
                '$starts' => 'A',
            ),
        ),
    ),
);
```

```

    ),
    array(
        'name' => array(
            '$starts' => 'C',
        ),
    ),
)
);

```

Module Expressions

Module expressions operate on modules instead of specific fields. The current module can be specified by either using the module name `_this` or by leaving the module name as a blank string.

Module Expression	Description
<code>\$favorite</code>	Filters the records by the current users favorited items.
<code>\$owner</code>	Filters the records by the assigned user.

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

```

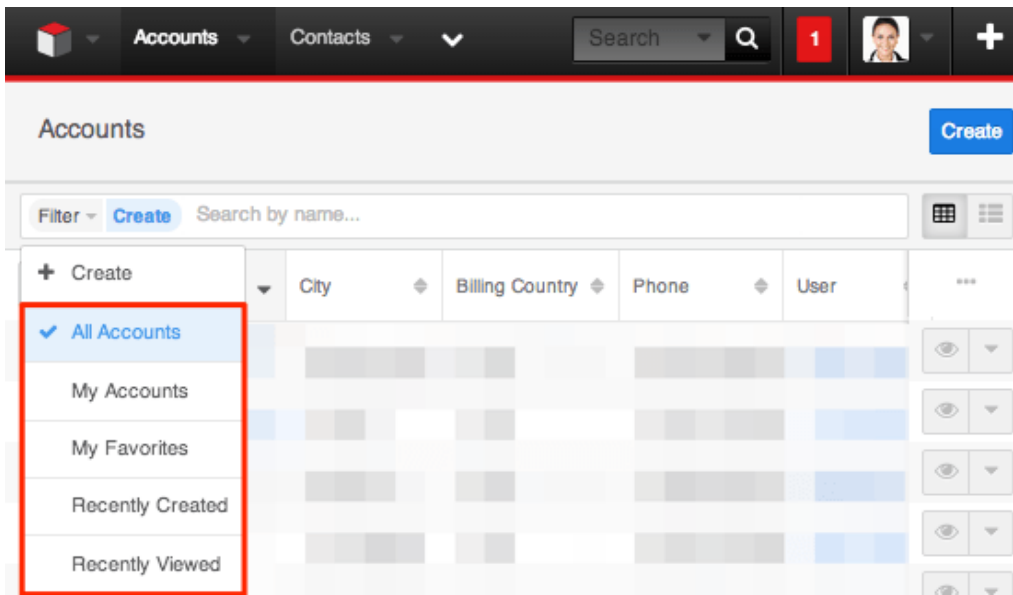
$filters = array(
    array(
        '$favorite' => '_this'
    ),
);

```

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, create:

```
./custom/Extension/modules/Accounts/Ext/clients/base/filters/basic/filterAccountTe  
mplate.php
```

```
<?php
```

```
$viewdefs['Accounts']['base']['filter']['basic']['filters'][] = array(  
    'id' => 'filterAccountTemplate',  
    'name' => 'LBL_FILTER_ACCOUNT_TEMPLATE',  
    'filter_definition' => array(  
        array(  
            'account_type' => array(  
                '$in' => array(),  
            ),  
        ),  
        array(  
            'assigned_user_id' => ''  
        )  
    ),  
    'editable' => true,  
    'is_template' => true,  
);
```

As you can see, the `filter_definition` contains arrays for `account_type` and `assigned_user_id`. These filter definitions will receive their values from the contact record view's metadata. You should also note that this filter has `is_template` and `editable` set to "true". This is required for initial filters.

Once the filter template is in place, modify the contact record view's metadata. To accomplish this, edit
./custom/modules/Contacts/clients/base/views/record/record.php to adjust the `account_name` field. If this file does not exist in your local Sugar installation, navigate to Admin > Studio > Contacts > Layouts > Record View and click "Save & Deploy" to generate it. In this file, identify the `panel_body` array as shown below:

```
1 =>
```

```

array (
  'name' => 'panel_body',
  'label' => 'LBL_RECORD_BODY',
  'columns' => 2,
  'labelsOnTop' => true,
  'placeholders' => true,
  'newTab' => false,
  'panelDefault' => 'expanded',
  'fields' =>
  array (
    0 => 'title',
    1 => 'phone_mobile',
    2 => 'department',
    3 => 'do_not_call',
    4 => 'account_name',
    5 => 'email',
  ),
),

```

Next, modify the `account_name` field to contain the initial filter parameters.

```

1 =>
array (
  'name' => 'panel_body',
  'label' => 'LBL_RECORD_BODY',
  'columns' => 2,
  'labelsOnTop' => true,
  'placeholders' => true,
  'newTab' => false,
  'panelDefault' => 'expanded',
  'fields' =>
  array (
    0 => 'title',
    1 => 'phone_mobile',
    2 => 'department',
    3 => 'do_not_call',
    4 => array (
      //field name
      'name' => 'account_name',

      //the name of the filter template
      'initial_filter' => 'filterAccountTemplate',

      //the display label for users

```

```

        'initial_filter_label' => 'LBL_FILTER_ACCOUNT_TEMPLATE',

        //the hardcoded filters to pass to the templates filter
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 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: 01/22/2018 02:32pm

Duplicate Check

Overview

The duplicate-check framework provides the capability to alter how the system searches for duplicate records in the database when creating records. For information on duplicate checking during imports, please refer to the index documentation. The duplicateCheck API endpoint utilizes this framework, which is called by default when saving a record from the user interface.

Default Strategy

The default duplicate-check strategy, located in `./data/duplicatecheck/FilterDuplicateCheck.php`, is referred to as `FilterDuplicateCheck`. This strategy utilizes the Filter API and a defined filter in the vardefs of the module to search for duplicates and rank them based on matching data.

Custom Filter

To alter a defined filter for a stock a module, you only need to update the Vardefs using the Extensions framework. If you are working with a custom module, you can modify the vardefs in `./modules/<module>/vardefs.php` directly. The `FilterDuplicateCheck` Strategy accepts two properties in its metadata:

Name	Type	Description
filter_template	array	An array containing the Filter Definition for which fields to search for duplicates on. Please consult the Filter API for further information on the filter syntax

ranking_fields	array	<p>A list of arrays with the following properties. The order in which you list the fields for ranking will determine the ranking score. The first ranks higher, than those after it.</p> <p>in_field_name: Name of field in vardefs</p> <p>dupe_field_name: Name of field returned by filter</p>
----------------	-------	--

Example

The following example will demonstrate how to manipulate the stock Accounts duplicate check to not filter on the shipping address city. The default Account duplicate_check defintion, located in ./modules/Accounts/vardefs.php, is shown below.

```

...
'duplicate_check' => array(
    'enabled' => true,
    'FilterDuplicateCheck' => array(
        'filter_template' => array(
            array(
                '$or' => array(
                    array('name' => array('$equals' => '$name')),
                    array('duns_num' => array('$equals' =>
'$duns_num'))),
                array(
                    '$and' => array(
                        array('name' => array('$starts' =>
'$name')),
                        array(
                            '$or' => array(
                                array('billing_address_city' =>
array('$starts' => '$billing_address_city')),
                                array('shipping_address_city' =>
array('$starts' => '$shipping_address_city')),
                            )
                        ),
                    )
                )
            )
        )
    )

```

```

        )
    ),
),
'ranking_fields' => array(
    array('in_field_name' => 'name', 'dupe_field_name' =>
'name'),
    array('in_field_name' => 'billing_address_city',
'dupe_field_name' => 'billing_address_city'),
    array('in_field_name' => 'shipping_address_city',
'dupe_field_name' => 'shipping_address_city'),
)
)
),
...

```

To add or remove fields from the check, you will need to manipulate `$dictionary['<module>']['duplicate_check']['FilterDuplicateCheck']['filter_template']` and `$dictionary['<module>']['duplicate_check']['FilterDuplicateCheck']['ranking_fields']`. For our example, we will simply remove the references to `shipping_address_city`. It is important to familiarize yourself with filter operators before making any changes.

`./custom/Extension/modules/Accounts/Ext/Vardefs/newFilterDuplicateCheck.php`

```

<?php

$dictionary['Account']['duplicate_check']['FilterDuplicateCheck'] =
array(
    'filter_template' => array(
        array(
            '$or' => array(
                array('name' => array('$equals' => '$name')),
                array('duns_num' => array('$equals' => '$duns_num')),
                array(
                    '$and' => array(
                        array('name' => array('$starts' => '$name')),
                        array(
                            '$or' => array(
                                array('billing_address_city' =>
array('$starts' => '$billing_address_city')),

```

Example

The following example will create a new duplicate-check class called "OneFieldDuplicateCheck" that will query the database based on the configured field for records that contain data similar to that one field:

`./custom/data/duplicatecheck/OneFieldDuplicateCheck.php`

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

class OneFieldDuplicateCheck extends DuplicateCheckStrategy
{
    protected $field;

    public function setMetadata($metadata)
    {
        if (isset($metadata['field'])) {
            $this->field = $metadata['field'];
        }
    }

    public function findDuplicates()
    {
        if (empty($this->field)){
            return null;
        }

        $Query = new SugarQuery();
        $Query->from($this->bean);

        $Query->where()->ends($this->field,$this->bean->{$this->field});
        $Query->limit(10);
        //Filter out the same Bean during Edits
        if (!empty($this->bean->id)) {
            $Query->where()->notEquals('id',$this->bean->id);
        }
        $results = $Query->execute();
        return array(
            'records' => $results
        );
    }
}
```

```

        );
    }
}

```

Vardef Settings

The duplicate-check vardef settings are configured on each module and can be altered using the Extension framework.

Name	Type	Description
enabled	boolean	Whether or not duplicate-check framework is enabled on the module
<class_name>	array	The class name that will provide duplicate checking, set to an array of metadata that gets passed to the duplicate-check class

If you want to enable the OneFieldDuplicateCheck strategy (shown above) for the Accounts module, you must create the following file:

```
./custom/Extension/modules/Accounts/Ext/Vardefs/newDuplicateCheck.php
```

```

<?php
$dictionary['Account']['duplicate_check'] = array(
    'enabled' => true,
    'OneFieldDuplicateCheck' => array(
        'field' => 'name'
    )
);

```

Finally, navigate to Admin > Repair > Quick Repair and Rebuild. The system will then enable the custom duplicate-check class.

Programmatic Usage

You can also use the duplicate-check framework in code such as in a logic hook or a scheduler. The following example shows how to use the module's defined duplicate-check strategy when utilizing a bean object by simply calling the

findDuplicates method on the bean object:

```
$account = BeanFactory::newBean('Accounts');
$account->name = 'Test';
$duplicates = $account->findDuplicates();

if (count($duplicates['records'])>0){
    $GLOBALS['log']->fatal("Duplicate records found for Account");
}
```

Last Modified: 04/25/2018 04:09am

Global Search

Overview

How to customize the global search results.

Configuring Search Results

The Search-List view, located in `./clients/base/views/search-list/`, is responsible for the global search results page. By default, it contains the row actions that are available for each result. The following sections will outline how to configure the primary and secondary fields.

Primary and Secondary Fields

Each row returned from the global search is split into two lines. The first line contains the primary fields and avatar. A primary field is a field considered to be important to the identification of a record and is usually composed of the records name value and a link to the record. The avatar is an icon specific to the module the results are coming from.

The second line contains the highlighted matches and any secondary fields specified. A highlight is a hit returned by elastic search. Highlights are driven from elastic and no definition is needed in the metadata to define them. Both primary and secondary fields can be highlighted. A secondary field is a field that has regularly accessed data. It is important to note that the initial type-ahead drop-down will not show specified secondary fields unless the user hits enter and

is taken to the search results page. Different modules have different secondary fields. For example, an account's secondary fields are email and phone number, while cases' secondary fields are case ID number and related account.

To leverage the metadata manager, the global search view reuses the same metadata format as other views. You can add any indexed field in elastic search to the search list view. An example of the stock metadata is shown below.

`./modules/{module}/clients/base/views/search-list/search-list.php`

```
<?php
```

```
$viewdefs[$moduleName]['base']['view']['search-list'] = array(
    'panels' => array(
        array(
            'name' => 'primary', // This is mandatory and the word
`primary` can not be changed.
            'fields' => array(
                /*
                * Put here the list of primary fields.
                * You can either define a field as a string and the
metadata
                * manager will load the field vardefs, or you can
define as an
                * array if you want to add properties or override
vardefs
                * properties just for the search results view.
                *
                * You most likely want to show the module icon on the
left of the
                * view. For this, you need to add the following
picture field as
                * a primary field.
                */
                array(
                    'name' => 'picture',
                    'type' => 'avatar',
                    'size' => 'medium',
                    'css_class' => 'pull-left',
                ),
                array(
                    'name' => 'name',
                    'type' => 'name',
                    'link' => true,
```

```

        'label' => 'LBL_SUBJECT',
    ),
),
),
/*
 * If you don't want secondary fields, you can remove the
following array.
 */
array(
    'name' => 'secondary', // This is mandatory and the word
`secondary` can not be changed.
    'fields' => array(
        /*
         * Put here the list of secondary fields
         */
        'status', // This field is defined as a string for
example.
        array(
            'name' => 'description',
            'label' => 'LBL_SEARCH_DESCRIPTION',
        ),
    ),
),
),
);

```

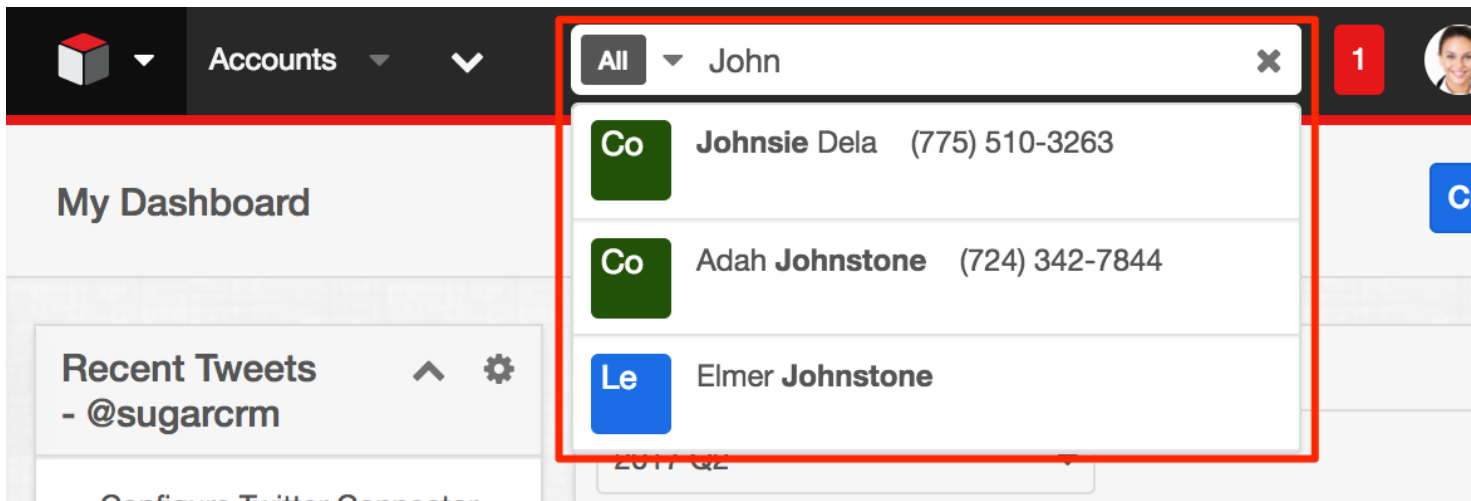
Note: You can technically override the row actions for the results of a module, however, the view currently only supports the preview action so it is not recommended to customize these actions.

Examples

The following examples demonstrate how to modify the primary and secondary rows of the global search.

Adding Fields to the Primary Row

This example will demonstrate how to add "Mobile" into the search-list view's primary row for Contact module results.



To accomplish this, you can duplicate `./modules/Contacts/clients/base/views/search-list/search-list.php` to `./custom/modules/Contacts/clients/base/views/search-list/search-list.php` if it doesn't exist. Once in place, add your `phone_mobile` definition to the `$viewdefs['Contacts']['base']['view']['search-list']['panels']` array definition with the name attribute equal to "primary". An example is shown below:

`./custom/modules/Contacts/clients/base/views/search-list/search-list.php`

```
<?php
```

```
$viewdefs['Contacts']['base']['view']['search-list'] = array(
    'panels' => array(
        array(
            'name' => 'primary',
            'fields' => array(
                array(
                    'name' => 'picture',
                    'type' => 'avatar',
                    'size' => 'medium',
                    'readonly' => true,
                    'css_class' => 'pull-left',
                ),
                array(
                    'name' => 'name',
                    'type' => 'name',
                    'link' => true,
                    'label' => 'LBL_SUBJECT',
                ),
            ),
            // Adding the mobile into first row of the results
            'phone_mobile',
        ),
    ),
);
```

```

        ),
    ),
    array(
        'name' => 'secondary',
        'fields' => array(
            array(
                'name' => 'email',
                'label' => 'LBL_PRIMARY_EMAIL',
            ),
        ),
    ),
),
);

```

Once in place, navigate to Admin > Repair > Quick Repair and Rebuild. Your changes will then be reflected in the system.

Adding Fields to the Secondary Row

This example will demonstrate how to add "Mobile" into the search-list view's secondary row for Contact module results. It is important to note that the initial type-ahead drop-down will not show specified secondary fields unless the user hits enter and is taken to the search results page.

The screenshot shows a search interface with a navigation bar containing 'Accounts' and a search bar with 'All john'. Below the search bar, the results are titled 'Search Results for: "john" (3)'. Two contact entries are listed:

- Johnsie Dela**: Primary Email: qa59@example.name, Mobile: (775) 510-3263
- Adah Johnstone**: Primary Email: beans.the.kid@example.cn, Mobile: (724) 342-7844

In both entries, the mobile phone numbers are enclosed in red rectangular boxes.

In this example, we are going to add Portal Name into search-list for Contact module results.

To accomplish this, you can duplicate

./modules/Contacts/clients/base/views/search-list/search-list.php to ./custom/modules/Contacts/clients/base/views/search-list/search-list.php if it doesn't exist. Once in place, add your phone_mobile definition to the \$viewdefs['Contacts']['base']['view']['search-list']['panels'] array definition with the name attribute equal to "secondary". An example is shown below:

```
<?php

$viewdefs['Contacts']['base']['view']['search-list'] = array(
    'panels' => array(
        array(
            'name' => 'primary',
            'fields' => array(
                array(
                    'name' => 'picture',
                    'type' => 'avatar',
                    'size' => 'medium',
                    'readonly' => true,
                    'css_class' => 'pull-left',
                ),
                array(
                    'name' => 'name',
                    'type' => 'name',
                    'link' => true,
                    'label' => 'LBL_SUBJECT',
                ),
            ),
        ),
        array(
            'name' => 'secondary',
            'fields' => array(
                array(
                    'name' => 'email',
                    'label' => 'LBL_PRIMARY_EMAIL',
                ),
                // Adding mobile to second row for search results in
                globalsearch
                'phone_mobile'
            ),
        ),
    ),
);
```

Once in place, navigate to Admin > Repair > Quick Repair and Rebuild. Your changes will then be reflected in the system.

Last Modified: 11/27/2017 12:24pm

Sugar Logic

Overview

Sugar Logic, a feature in all commercial editions of Sugar, enables custom business logic that is easy to create, manage, and reuse on both the server and client.

Sugar Logic is made up of multiple components that build on each other and is extensible at every step. The base component is the Sugar Formula Engine, which parses and evaluates human-readable formulas. Dependencies are units made up of triggers and actions that can express custom business logic. Each dependency defines a list of actions to be performed depending on the outcome of a trigger formula.

Terminology

- Formula : An expression that conforms to the Formula Engine syntax, consisting of nested functions and field variables
- Function : A method that can be called in a formula
- Trigger : A formula that evaluates to either true or false when a field in the equation is updated or when a record is retrieved/saved
- Action : A method that modifies the current record or layout in some way
- Dependency : A complete logical unit that includes a trigger and one or more actions

Sugar Formula Engine

Formulas

The fundamental object is called a formula. A formula can be evaluated for a given record using the Sugar Logic parser.

Some example formulas are:

Basic addition:

```
add(1, 2)
```

Boolean values:

```
not(equal($billing_state, "CA"))
```

Calculation:

```
multiply(number($employees), $seat_cost, 0.0833)
```

Types

Sugar Logic has several fundamental types: number, string, boolean, and enum (lists). Functions may take in any of these types or combinations thereof and return as output one of these types. Fields may also often have their value set to only a certain type.

Number Type

Number types essentially represent any real number (which includes positive, negative, and decimal numbers). They can be plainly typed as input to any function. For example, the operation $(10 + 10 + (15 - 5))$ can be performed as follows:

```
add(10, 10, subtract(15, 5))
```

String Type

A string type is very much like a string in most programming languages. However, it can only be declared within double quotes. For example, consider this function, which returns the length of the input string:

```
strlen("Hello World")
```

The function would appropriately return the value 11.

Boolean Type

A boolean type is simple. It can be one of two values: "true" or "false". This type mainly acts as a flag that indicates whether a condition has been met or not. For example, this function contains takes two strings as input and returns "true" if the first string contains the second string, or "false" otherwise:

```
and(contains("Hello World", "llo Wo"), true)
```

The function would appropriately return the value "true".

Enum Type (list)

An enum type is a collection of items. The items do not need to all be of the same type, they can be varied. An enum can be declared using the enum function as follows:

```
enum("hello world!", false, add(10, 15))
```

Alternatively, the createList() function (an alias to enum) can be used to create enums in a formula.

```
createList("hello world!", false, add(10, 15))
```

This function would appropriately return an enum type containing "hello world!", false, and 25 as its elements.

Link Type (relationship)

A link represents one side of a relationship and is used to access related records. For example, the accounts link field of Contacts is used to access the account_type field of the account related to a contact:

```
related($accounts, "account_type")
```

For most of the out-of-the-box relationships, links are named with the name of the related module, in lower case.

Follow these steps to find the name of the link fields for relationships that do not follow the convention above:

1. Open the vardef file for the module you are working on:
./cache/modules/{module}/{module}vardefs.php

2. Find the link field that matches the relationship you are looking for.

Functions

Functions are methods to be used in formulas. Each function has a function name, a parameter count, a parameter type requirement, and returns a value. Some functions such as `add()` can take any number of parameters. For example: `add(1)`, `add(1, 2)`, and `add(1, 2, 3)` are all valid formulas. Functions are designed to produce the same result whether executed on the server or client.

Triggers

A trigger is an object that listens for changes in field values and, after a change occurs, triggers the associated actions in a dependency.

Actions

Actions are functions that modify a target in some way and are fired when the trigger is `TRUE`. Most Actions require at least two parameters: a target and a formula. For example, a style action will change the style of a field based on a passed-in string formula. A value action will update a value of a field by evaluating a passed in formula.

notActions

`notActions` are functions that modify a target in some way and are fired when the trigger is `FALSE`. `notActions` are optional and if they are not defined then nothing will fire when the trigger is `FALSE`. Most `notActions` require at least two parameters: a target and a formula. For example, a style action will change the style of a field based on a passed-in string formula. A value action will update a value of a field by evaluating a passed in formula.

Dependencies

A Dependency describes a set of rules based on a trigger and a set of actions. Examples include a field whose properties must be updated dynamically or a panel which must be hidden when a drop down value is not selected. When a Dependency is triggered it will appropriately carry out the action it is designed to. A basic Dependency is when a field's value is dependent on the result of evaluating

a Expression. For example, consider a page with five fields with It's "a", "b", "c", "d", and "sum". A generic Dependency can be created between "sum" and the other four fields by using an Expression that links them together, in this case an add Expression. So we can define the Expression in this manner: 'add(\$a, \$b, \$c, \$d)' where each field id is prefixed with a dollar (\$) sign so that the value of the field is dynamically replaced at the time of the execution of the Expression.

An example of a more customized Dependency is when the field's style must be somehow updated to a certain value. For example, the DIV with id "temp" must be colored blue. In this case we need to change the background-color property of "temp". So we define a StyleAction in this case and pass it the field id and the style change that needs to be performed and when the StyleAction is triggered, it will change the style of the object as we have specified.

Sugar Logic Based Features

Calculated Fields

Fields with calculated values can now be created from within Studio and Module Builder. The values are calculated based on Sugar Logic formulas. These formulas are used to create a new dependency that are executed on the client side in edit views and the server side on save. The formulas are saved in the varies or vardef extensions and can be created and edited directly. For example, the metadata for a simple calculated commission field in opportunities might look like:

```
'commission_c' => array(
    'name' => 'commission_c',
    'type' => 'currency',
    'calculated' => true,
    'formula' => 'multiply($amount, 0.1)',
    //enforced causes the field to appear read-only on the layout
    'enforced' => true
),
```

Dependent fields

A dependent field will only appear on a layout when the associated formula evaluates to the Boolean value true. Currently these cannot be created through Studio and must be enabled manually with a custom vardef or vardef extension. The "dependency" property contains the expression that defines when this field should be visible. An example field that only appears when an account has an annual revenue greater than one million.

```
'dep_field'=> array(
    'name' => 'dep_field',
    'type' => 'varchar',
    'dependency' => 'greaterThan($annual_revenue, 1000000)',
),
```

Dependent dropdowns

Dependent dropdowns are dropdowns for which options change when the selected value in a trigger dropdown changes. The metadata is defined in the vardefs and contains two major components, a "trigger" id which is the name of the trigger dropdown field and a 'visibility grid' that defines the set of options available for each key in the trigger dropdown. For example, you could define a sub-industry field in accounts whose available values depend on the industry field.

```
'sub_industry_c' => array(
    'name' => 'sub_industry_c',
    'type' => 'enum',
    'options' => 'sub_industry_dom',
    'visibility_grid'=> array(
        'trigger' => 'industry',
        'values' => array(
            'Education' => array(
                'primary',
                'secondary',
                'college'
            ),
            'Engineering' => array(
                'mechanical',
                'electrical',
                'software'
            ),
        ),
    ),
),
```

Clearing the Sugar Logic Cache

The `./updatecache.php` script traverses the Expression directory for every file that ends with "Expression.php" (ignoring a small list of excepted files) and constructs both a PHP and a JavaScript functions cache file which resides in `./cache/Expressions/functions_cache.js` and `./cache/Expressions/functionmap.php`.

If you create your custom expressions, you will need to rebuild the sugar logic functions by navigating to:

Admin > Repair > Rebuild Sugar Logic Functions

Last Modified: 10/14/2016 04:12pm

Dependency Actions

Overview

The following sections outline the available SugarLogic dependency actions.

Dependency Parameters

Dependency definitions will generally contain most, if not all, of the parameters displayed in the table below. The following sections will outline each dependency action as well as dependency specific parameters.

Parameter	Type	Description
hooks	Array	The views to execute the trigger on. Possible values are: "edit", "view", "save" and "all". If you include 'save' or 'all' then SugarCRM will try to save the calculated value to the database. So if your dependency is display only then only include the views that it will show up on.
trigger	String	Optional. The trigger for the dependency. Defaults to 'true'.
triggerFields	Array	The list of fields to watch for change events. When changed, the trigger expressions will be

		recalculated.
onload	Boolean	Whether or not to trigger the dependencies when the page is loaded.
actions	Array	The list of dependencies to execute when the trigger expression is true.
notActions	Array	The list of dependencies to execute when the trigger expression is false.

Last Modified: 11/27/2017 12:24pm

ReadOnly

Overview

The SugarLogic ReadOnly action, located in `./include/Expressions/Actions/ReadOnlyAction.php`, is used to determine if a field is editable or not based on a formula.

Implementation

While the dependency metadata for your module can be defined in `./modules/<module>/metadata/dependencydefs.php` and `./custom/modules/<module>/metadata/dependencydef.php`, it is recommended to use the extension framework when customizing stock modules to prevent third party plugins from conflicting with your customizations. The following section will demonstrate how to implement a read-only dependency.

ReadOnly Parameters

Parameter	Type	Description
target	String	The name of the field to make read only.
value	String	This parameter can accept a boolean formula or true

	and false values. Normally you would put a SugarLogic formula here to set the boolean.
--	--

For more information on the various parameters in the dependency definitions, please refer to the dependency actions documentation.

Example

For our example, we will create a dependency on the Accounts module that makes the name field read-only when the lock_record_c field has been checked. The first step is to create the lock_record_c checkbox field in Studio and add it to your Record View layout. When this checkbox is checked, we will make the name field read-only. Our example extension definition is shown below:

```
./custom/Extension/modules/<module>/Ext/Dependencies/custom_name_read_only.php
```

```
<?php
```

```
    $dependencies['Accounts']['readonly_fields'] = array(
        'hooks' => array("edit"),
        'trigger' => 'true',
        //Optional, the trigger for the dependency. Defaults to
'true'.
        'triggerFields' => array('lock_record_c'),
        'onload' => true,
        //Actions is a list of actions to fire when the trigger is
true
        // You could list multiple fields here each in their own array
under 'actions'
        'actions' => array(
            array(
                'name' => 'ReadOnly',
                //The parameters passed in will depend on the action
type set in 'name'
                'params' => array(
                    'target' => 'name',
                    'value' => 'equal($lock_record_c,true)',
                ),
            ),
        ),
    );
```

Once you have all the files in place you will need to navigate to Admin > Repairs > and run a Quick Repair and Rebuild.

Note: It is important that the module name is plural ('Accounts' vs. 'Account') and that the name of the dependency, "readonly_fields" in this example, is unique.

Considerations

1. In some scenarios, you may want a specific field to always be read-only. To accomplish this, you can modify the 'value' attribute to always be "true". Given the above example, you would modify:

```
'value' => 'equal($lock_record_c,true)',
```

to be:

```
'value' => 'true',
```

Last Modified: 12/05/2017 06:53pm

SetOptions

Overview

The SugarLogic SetOptions action, located in `./include/Expressions/Actions/SetOptionsAction.php`, is used to set the options list of a dropdown field based on a formula.

Implementation

While the dependency metadata for your module can be defined in `./modules/<module>/metadata/dependencydefs.php` and `./custom/modules/<module>/metadata/dependencydef.php`, it is recommended to use the extension framework when customizing stock modules to prevent third party plugins from conflicting with your customizations. The following section will demonstrate how to implement a read-only dependency.

Setoptions Parameters

Parameter	Type	Description
target	String	The name of the dropdown field that you want to change the option list for
keys	String	A formula used to get the option list keys for the target field or a list name from which keys will be extracted.
labels	String	A formula used to get the option list labels for the target field or a list name from which labels will be extracted.

For more information on the various parameters in the dependency definitions, please refer to the dependency actions documentation.

Examples

The following sections outline the various ways this dependency can be implemented.

Using an Existing DropDown List

You can also set the options list to any current options list already in the system For example if you wanted to have the industry dropdown in Accounts show the 'bug_type_dom' list from Bugs you could do this

```
<?php
$dependencies['Leads']['setoptions_industry'] = array(
    'hooks' => array("edit","save"),
    'trigger' => 'true',
    'triggerFields' => array('industry'),
    'onload' => true,
    'actions' => array(
        array(
```

```
'name' => 'SetOptions',
'params' => array(
    'target' => 'industry',
    'keys' => 'getDropdownKeySet("bug_type_dom")',
    'labels' => 'getDropdownValueSet("bug_type_dom")'
),
),
),
);
```

This would grab the keys and label from the 'bug_type_dom' using the getDropdownKeySet() and getDropdownValueSet() JavaScript functions and display them instead of the normal list.

Once you have the file in place, you will need to navigate to Admin > Repairs > and run a Quick Repair and Rebuild.

Note: It is important that the module name is plural ('Accounts' vs. 'Account') and that the name of the dependency, "setoptions_industry" in this example, is unique.

Complex Dynamic Lists

For our first example, we will change a dropdown called fruits_c to include only fruits that are in season. This could be done with a dependent dropdown but that would require the user to pick the proper season. With this, we can have a function that returns only fruit that is in season right now. I added the dropdown fruits_c to Leads and created a new list for it that looks like this

```
$app_list_strings['fruits_list']=array (
    'Apples' => 'Apples',
    'Strawberries' => 'Strawberries',
    'Mangos' => 'Mangos',
    'Pineapples' => 'Pineapples',
    'Blackberries' => 'BlackBerries',
    'BlueBerries' => 'BlueBerries',
);
```

To keep it simple I made the labels and the keys the same.

Then I extended SugarLogic as outlined in Extending SugarLogic and made a new function called fruitInSeason() that returns a string reflecting what fruit is in season right now. To work for the createList() function it would return a list like "Apples","Mangos","BlueBerries".

./custom/Extension/modules/<module>/Ext/Dependencies/custom_phone_alternate.php

```
<?php
$dependencies['Leads']['setoptions_fruit'] = array(
    'hooks' => array("edit","save"),
    'trigger' => 'true',
    'triggerFields' => array('fruits_c'),
    'onload' => true,
    'actions' => array(
        array(
            'name' => 'SetOptions',
            'params' => array(
                'target' => 'fruits_c',
                'keys' => "createList(fruitInSeason())",
                'labels' => "createList(fruitInSeason())"
            ),
        ),
    ),
);
```

The createList() function is a JavaScript function from Sidecar. It requires a comma delimited quote enclosed list of options.

We only want this to affect EditViews and Saves, not normal record views since they need to be able to display all fruits and not a truncated list of them. So we make the 'hooks' array

```
'hooks' => array("edit","save"),
```

Once you have the file in place, you will need to navigate to Admin > Repairs > and run a Quick Repair and Rebuild.

Note: It is important that the module name is plural ('Leads' vs. 'Lead') and that the name of the dependency, "setoptions_fruit" in this example, is unique.

Last Modified: 11/27/2017 12:24pm

SetPanelVisibility

Overview

The SugarLogic SetPanelVisibility action, defined in `./include/Expressions/Actions/PanelVisibilityAction.php`, is used to determine the visibility of a record view panel based on a formula.

Implementation

While the dependency metadata for your module can be defined in `./modules/<module>/metadata/dependencydefs.php` and `./custom/modules/<module>/metadata/dependencydef.php`, it is recommended to use the extension framework when customizing stock modules to prevent third party plugins from conflicting with your customizations. The following section will demonstrate how to implement a read-only dependency.

SetPanelVisibility Parameters

Parameter	Type	Description
target	String	The id of the panel to hide
value	String	Formula used to determine if the panel should be visible.

For more information on the various parameters in the dependency definitions, please refer to the dependency actions documentation.

Example

For our example, we will create a dependency on the Cases module that will hide a specific panel if the status field on a case is set to "Closed". Our example extension definition is shown below:

```
./custom/Extension/modules/<module>/Ext/Dependencies/hide_panel_2_dep.php
```

```
<?php
```

```
$dependencies['Cases']['panel_2_visibility'] = array(
    'hooks' => array("edit", "view"),
    'trigger' => 'equal($status, "Closed")',
    'triggerFields' => array('status'),
    'onload' => true,
```

```

        //Actions is a list of actions to fire when the trigger is
true
        'actions' => array(
            array(
                'name' => 'SetPanelVisibility',
                'params' => array(
                    'target' => 'detailpanel_2',
                    'value' => 'true',
                ),
            ),
        ),
        //notActions is a list of actions to fire when the trigger is
false
        'notActions' => array(
            array(
                'name' => 'SetPanelVisibility',
                'params' => array(
                    'target' => 'detailpanel_2',
                    'value' => 'false',
                ),
            ),
        ),
    );

```

Once you have the file in place, you will need to navigate to Admin > Repairs > and run a Quick Repair and Rebuild.

Note: It is important that the module name is plural ('Cases' vs. 'Case') and that the name of the dependency, "panel_2_visibility" in this example, is unique.

Last Modified: 10/14/2016 04:12pm

SetRequired

Overview

The SugarLogic SetRequired action, located in `./include/Expressions/Actions/SetRequiredAction.php`, is used to determine if a field is required.

Implementation

While the dependency metadata for your module can be defined in `./modules/<module>/metadata/dependencydefs.php` and `./custom/modules/<module>/metadata/dependencydef.php`, it is recommended to use the extension framework when customizing stock modules to prevent third party plugins from conflicting with your customizations. The following section will demonstrate how to implement a read-only dependency.

SetRequired Parameters

Parameter	Type	Description
target	String	The name of the field to make required.
label	String	id of label element for this field
value	String	Formula used to determine if the field should be required.

For more information on the various parameters in the dependency definitions, please refer to the dependency actions documentation.

Example

For our example, we will create a dependency on the Cases module that will mark the resolution field as required when the status field is set to "Closed". Our example extension definition is shown below:

```
./custom/Extension/modules/<module>/Ext/Dependencies/required_resolution_dep.php
```

```
<?php
```

```
$dependencies['Cases']['required_resolution_dep'] = array(
    'hooks' => array("edit"),
    'trigger' => 'true',
    'triggerFields' => array('status'),
    'onload' => true,
    //Actions is a list of actions to fire when the trigger is
```

```
true
    'actions' => array(
        array(
            'name' => 'SetRequired',
            //The parameters passed in will depend on the action
type set in 'name'
            'params' => array(
                'target' => 'resolution',
                'label' => 'resolution_label',
                'value' => 'equal($status, "Closed")',
            ),
        ),
    ),
);
```

Once you have the file in place, you will need to navigate to Admin > Repairs > and run a Quick Repair and Rebuild.

Note: It is important that the module name is plural ('Cases' vs. 'Case') and that the name of the dependency, "required_resolution_dep" in this example, is unique.

Last Modified: 10/14/2016 04:12pm

SetValue

Overview

The SugarLogic SetValue action, located in `./include/Expressions/Actions/SetValueAction.php`, is used to set the value of a field based on a formula.

Implementation

While the dependency metadata for your module can be defined in `./modules/<module>/metadata/dependencydefs.php` and `./custom/modules/<module>/metadata/dependencydef.php`, it is recommended to use the extension framework when customizing stock modules to prevent third party plugins from conflicting with your customizations. The following section will demonstrate how to implement a read-only dependency.

SetValue Parameters

Parameter	Type	Description
target	String	The name of the field to target for visibility.
value	String	SugarLogic formula used to get the value for the target field.

For more information on the various parameters in the dependency definitions, please refer to the dependency actions documentation.

Examples

The follow sections outline the various ways this dependency can be implemented.

Dependency Extensions

For our example, we will create a dependency on the Leads module that will display the number of activities related to a Lead. Activities are composed of calls, meetings, tasks, notes, and emails. An example extension definition is shown below:

```
./custom/Extension/modules/<module>/Ext/Dependencies/custom_phone_alternate.php
```

```
<?php
```

```
    $dependencies['Leads']['activity_count_dep'] = array(
        'hooks' => array("edit", "view"), //not including save so that
the value isn't stored in the DB
        'trigger' => 'true', //Optional, the trigger for the
dependency. Defaults to 'true'.
        'onload' => true, //Whether or not to trigger the dependencies
when the page is loaded
        'actions' => array(
            array(
                'name' => 'SetValue',
                'params' => array(
                    'target' => 'activity_count_c',
```

```

        'value' => 'add(
            count($notes),
            count($calls),
            count($emails),
            count($meetings),
            count($tasks)
        )'
    )
)
);

```

Once you have the file in place, you will need to navigate to Admin > Repairs > and run a Quick Repair and Rebuild.

Note: It is important that the module name is plural ('Cases' vs. 'Case') and that the name of the dependency, "activity_count_dep" in this example, is unique.

Chaining Dependencies

You can also add as many actions as you need to the array. In the example below, we want to display our count value but prevent users from being able to edit the value. An example extension definition is shown below:

```

<?php

$dependencies['Leads']['number_of_cases_dep'] = array(
    'hooks' => array("edit", "view"), //not including save so that
the value isn't stored in the DB
    'trigger' => 'true', //Optional, the trigger for the
dependency. Defaults to 'true'.
    //'triggerFields' => array('status'), //unneeded for this
example as its not field triggered
    'onload' => true,
    'actions' => array(
        array(
            'name' => 'SetValue',
            'params' => array(
                'target' => 'activity_count_c',
                'value' => 'add(
                    count($notes),
                    count($calls),
                    count($emails),

```

```

        count($meetings),
        count($tasks)
    )'
    )
),
array(
    'name' => 'ReadOnly',
    'params' => array(
        'target' => 'activity_count_c',
        'value' => 'true', //Set to true instead of a
formula because its always read-only
    ),
)
);

```

Once you have the file in place, you will need to navigate to Admin > Repairs > and run a Quick Repair and Rebuild.

Note: It is important that the module name is plural ('Cases' vs. 'Case') and that the name of the dependency, "number_of_cases_dep" in this example, is unique.

Dependencies in Field Definitions

Unlike several of the other dependencies, SetValue is built into Studio. So this dependency can be set as a custom vardef value or in the vardefs file of a custom module. If you wanted to add this dependency to an existing field then you can create a vardef extension such as `./custom/Extension/modules/<module>/Ext/Vardefs/`. An example extension definition is shown below:

`./custom/Extension/modules/Accounts/Ext/Vardefs/activity_count_c.php`

```

<?php

    $dictionary['Lead']['fields']['activity_count_c']['options'] =
'numeric_range_search_dom';
    $dictionary['Lead']['fields']['activity_count_2_c']['calculated']
= 'true';
    $dictionary['Lead']['fields']['activity_count_2_c']['formula'] =
'add(
    count($calls),
    count($emails),

```

```

        count($meetings),
        count($notes),
        count($tasks)
    )';
    $dictionary['Lead']['fields']['activity_count_2_c']['enforced'] =
'true';

$dictionary['Lead']['fields']['activity_count_2_c']['enable_range_search'] = '1';

```

Once you have the file in place, you will need to navigate to Admin > Repairs > and run a Quick Repair and Rebuild.

Last Modified: 11/27/2017 12:24pm

SetVisibility

Overview

The SugarLogic SetVisibility action, located in `./include/Expressions/Actions/VisibilityAction.php`, is used to determine the visibility logic of a field based on a formula.

Implementation

While the dependency metadata for your module can be defined in `./modules/<module>/metadata/dependencydefs.php` and `./custom/modules/<module>/metadata/dependencydef.php`, it is recommended to use the extension framework when customizing stock modules to prevent third party plugins from conflicting with your customizations. The following section will demonstrate how to implement a read-only dependency.

SetVisibility Parameters

Parameter	Type	Description
target	String	The name of the field to target for visibility.
value	String	This parameter can accept

	a boolean formula or true and false values. Normally you would put a SugarLogic formula here to set the boolean.
--	--

For more information on the various parameters in the dependency definitions, please refer to the dependency actions documentation.

Examples

The follow sections outline the various ways this dependency can be implemented.

SetVisibility Dependency Extensions

For our example, we will create a dependency on the Accounts module that shows the phone_alternate field when the phone_office field has been populated. An example is shown below.

```
./custom/Extension/modules/<module>/Ext/Dependencies/custom_phone_alternate.php
```

```
<?php
```

```
$dependencies['Accounts']['phone_alternate_hide'] = array(
    'hooks' => array("edit"),
    'triggerFields' => array('phone_office'),
    'onload' => true,
    //Actions is a list of actions to fire when the trigger is
true
    'actions' => array(
        array(
            'name' => 'SetVisibility',
            'params' => array(
                'target' => 'phone_alternate',
                'value' => 'not(equal($phone_office,""))',
            ),
        ),
    ),
);
```

Once you have the file in place, you will need to navigate to Admin > Repairs >

and run a Quick Repair and Rebuild.

Note: It is important that the module name is plural ('Accounts' vs. 'Account') and that the name of the dependency, "phone_alternate_hide" in this example, is unique.

Visibility Dependencies in Field Definitions

Unlike several of the other dependencies, SetVisibility is built into Studio. So this dependency can be set as a custom vardef value or in the varefs file for a new module. If you wanted to add this dependency to an existing field then you could add a file to ./custom/Extension/modules/<module>/Ext/Vardefs/. An example is shown below.

To accomplish this, we will create an extension in ./custom/Extension/modules/Accounts/Ext/Vardefs/.

./custom/Extension/modules/Accounts/Ext/Vardefs/phone_alternate.php

```
<?php
```

```
$dictionary['Account']['fields']['phone_alternate']['dependency']='not  
(equal($phone_office,""))
```

Next, you will need to navigate to Admin > Repairs > and run a Quick Repair and Rebuild. Once that is done, you can enter a value into phone_office and the phone_alternate field will show up once you tab out of the phone_office field. If you were coding a custom module with new fields, then you would just include it in the modules vardefs.php file as shown below

```
<?php
```

```
$dictionary['myModule'] = array(  
    ...  
    'fields' => array(  
        ...  
        'phone_alternate' => array(  
            'name' => 'phone_alternate',  
            'vname' => 'LBL_PHONE_ALTERNATE',  
            'type' => 'varchar',  
            'len' => 10,  
            'dependency' => 'not(equal($phone_office,""))',  
            'comment' => 'Other Phone Number',
```

```
        'merge_filter' => 'enabled',
        ),
        ...
    )
    ...
);
```

Last Modified: 11/27/2017 12:24pm

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 positive";
// 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 equivalent 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 execution 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: 10/14/2016 04:12pm

Using Sugar Logic Directly

How to use Sugar Logic

Last Modified: 10/14/2016 04:12pm

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: 04/02/2018 03:17pm

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: 10/14/2016 04:12pm

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: 10/14/2016 04:12pm

Using Dependencies in Logic Hooks

Overview

Dependencies can not only be executed on the server side but can be useful entirely on the server. For example, you could have a dependency that sets a rating based on a formula defined in a language file.

```
<?php

require_once("include/Expressions/Dependency.php");
require_once("include/Expressions/Trigger.php");
require_once("include/Expressions/Expression/Parser/Parser.php");
require_once("include/Expressions/Actions/ActionFactory.php");

class Update_Account_Hook
{
    function updateAccount($bean, $event, $args)
    {
        $formula = translate('RATING_FORMULA', 'Accounts');
        $triggerFields = Parser::getFieldsFromExpression($formula);
        $dep = new Dependency('updateRating');
        $dep->setTrigger(new Trigger('true', $triggerFields));
        $dep->addAction(ActionFactory::getNewAction('SetValue', array(
            'target' => 'rating',
            'value' => $formula
        )));
        $dep->fire($bean);
    }
}
```

Last Modified: 10/14/2016 04:12pm

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 Administration `saveSetting()` function as shown here:

```
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 a filter parameter. If no value is passed to `retrieveSettings()`, all settings will be returned. An example is shown here:

```
require_once('modules/Administration/Administration.php');

$administrationObj = new Administration();

//Retrieve all settings in the category of 'MyCategory'.
//This parameter can be left empty to retrieve all settings.
$administrationObj->retrieveSettings('MyCategory');

//Use a specific setting
$MySetting = $administrationObj->settings['MyCategory_MySetting'];
```

Considerations

The Administration class will store the settings in the config table. Alternatively, you can use the Configurator class located in `./modules/Configurator/Configurator.php` to store the settings in the `config_override.php` file.

Configurator

Overview

The Configurator class, located in `./modules/Configurator/Configurator.php`, handles the config settings found in `./config.php` and `./config_override.php`.

Retrieving Settings

You can access the Sugar config settings by using the global variable `$GLOBALS['sugar_config']` as shown below:

```
global $sugar_config;

//Use a specific setting
$MySetting = $sugar_config['MySetting'];
```

If you should need to reload the config settings, this is an example of how to retrieve a specific setting using the configurator:

```
require_once 'modules/Configurator/Configurator.php';

$configuratorObj = new Configurator();

//Load config
$configuratorObj->loadConfig();

//Use a specific setting
$MySetting = $configuratorObj->config['MySetting'];
```

Creating / Updating Settings

To create or update a specific setting, you can specify the new value using the configurator as shown below:

```
require_once 'modules/Configurator/Configurator.php';
```

```
$configuratorObj = new Configurator();

//Load config
$configuratorObj->loadConfig();

//Update a specific setting
$configuratorObj->config['MySetting'] = "MySettingsValue";

//Save the new setting
$configuratorObj->saveConfig();
```

Considerations

When looking to store custom settings, the Configurator class will store the settings in the `./config_override.php` file. 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: 10/14/2016 04:12pm

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 <code>./cache/config.js</code> 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, Enterprise, Ultimate
Override Example	<code>\$sugar_config['additional_js_config'] = array();</code>

additional_js_config.alertAutoCloseDelay

Description	Defines the default auto close delay for system alerts. It is important to note that after changing this setting in your configuration, you must navigate to Admin > Repairs > Quick Repair & Rebuild.
Type	Integer : Milliseconds
Versions	7.8.1.0+
Editions	Professional, Enterprise, Ultimate
Default Value	9000
Override Example	<code>\$sugar_config['additional_js_config']['alertAutoCloseDelay'] = 3000;</code>

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, Enterprise, Ultimate
Default Value	cache
Override Example	<code>\$sugar_config['additional_js_config']['authStore'] = 'cookie';</code>

additional_js_config.disableOmnibarTypeahead

Description	Disables the typeahead feature in the listview Omnibar and force users to hit Enter when filtering. From a technical perspective, this will reduce the number of hits to the server.
Type	Boolean
Range of values	true and false
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Override Example	<code>\$sugar_config['additional_js_config']['disableOmnibarTypeahead'] = true;</code>

additional_js_config.skipTutorial (Deprecated in future release)

Description	Disables the tutorial system shown
-------------	------------------------------------

	when users log in for the first time.
Type	Boolean
Range of values	true and false
Versions	7.0.0 - 7.8.2.2
Editions	Professional, Enterprise, Ultimate
Default Value	false
Override Example	<code>\$sugar_config['additional_js_config']['skipTutorial'] = true;</code>

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 <code>./include/MVC/Controller/file_access_control_map.php</code> and overridden by creating <code>./custom/include/MVC/Controller/file_access_control_map.php</code> .
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Override Example	<code>\$sugar_config['admin_access_control'] = true;</code>

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	Professional, Enterprise, Ultimate
Default Value	false
Override Example	<code>\$sugar_config['admin_export_only'] = true;</code>

allow_oauth_via_get

Description	As of 7.8, Sugar does not support auth tokens being passed in from GET query string parameters by default. While allowing this functionality is not a recommended practice from a security standpoint, administrators may enable the setting at their own risk.
Type	Boolean
Range of values	true and false
Versions	7.8.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Override Example	<code>\$sugar_config['allow_oauth_via_get'] = true;</code>

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 enable 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	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['allow_pop_inbound'] = true;</code>

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 Sugar's cloud environment will have this setting enforced as false.
Type	Boolean
Range of values	true and false
Versions	5.5.1+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Sugar Cloud Value	false
Override Example	<code>\$sugar_config['allow_sendmail_outbound'] = true;</code>

analytics

Description	An array defining properties for an analytics connector. Instances running on Sugar's cloud environment will have this setting enforced as a predetermined array.
-------------	---

Type	Array
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Sugar Cloud Value	a predetermined array
Override Example	<code>\$sugar_config['analytics'] = array();</code>

analytics.connector

Description	The name of the connector to use for gathering analytics. Instances running on Sugar's cloud environment will have this setting enforced as a predetermined value.
Type	String : Connector name
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Sugar Cloud Value	a predetermined value
Override Example	<code>\$sugar_config['analytics']['connector'] = 'GoogleAnalytics';</code>

analytics.enabled

Description	Determines if the analytics are enabled. Instances running on Sugar's cloud environment will have this setting enforced as true.
Type	Boolean
Range of values	true and false
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Sugar Cloud Value	true
Override Example	<code>\$sugar_config['analytics']['enable</code>

```
d'] = true;
```

analytics.id

Description	The tracking id for the analytics connector. Instances running on Sugar's cloud environment will have this setting enforced as a predetermined value.
Type	String : Tracking ID
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Sugar Cloud Value	a predetermined value
Override Example	<pre>\$sugar_config['analytics']['id'] = 'UA-XXXXXXX-X';</pre>

api

Description	API specific configurations.
Type	Integer
Versions	7.5.0.0+
Editions	Professional, Enterprise, Ultimate
Override Example	<pre>\$sugar_config['api']=array();</pre>

api.timeout

Description	The timeout in seconds when uploading files through the REST API.
Type	Integer : Seconds
Versions	7.5.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	180
Override Example	<pre>\$sugar_config['api']['timeout'] =</pre>

240;

authenticationClass

Description	The class to be used for login authentication.
Type	String : Sugar Authentication Class
Range of values	SAMLAuthenticate
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	SugarAuthenticate
Override Example	<pre>\$sugar_config['authenticationClass'] = 'SAMLAuthenticate';</pre>

aws

Description	The Amazon AWS configuration. Used for storing uploads on S3. The <code>upload_wrapper_class</code> setting must also be updated to <code>SugarUploadS3</code> to enable this configuration. You can do this by placing <pre>\$sugar_config['upload_wrapper_class'] = 'SugarUploadS3';</pre> in your <code>config_override.php</code> .
Type	Array
Versions	6.7.0+
Editions	Professional, Enterprise, Ultimate
Override Example	<pre>\$sugar_config['aws'] = array();</pre>

aws.aws_key

Description	Amazon AWS public key.
-------------	------------------------

Type	String : Key
Versions	6.7.0+
Editions	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['aws']['aws_key'] = 'key';</code>

aws.aws_secret

Description	Amazon AWS secret key.
Type	String : Key
Versions	6.7.0+
Editions	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['aws']['aws_secret'] = 'secret';</code>

aws.upload_bucket

Description	The Amazon S3 bucket name for uploads.
Type	String : S3 bucket name
Versions	6.7.0+
Editions	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['aws']['upload_bucket'] = 'bucket';</code>

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	Professional, Enterprise, Ultimate
Default Value	cache/
Override Example	<code>\$sugar_config['cache_dir'] = 'cache/';</code>

cache_expire_timeout

Description	The length of time cached items should be expired after.
Type	Integer : Seconds
Versions	6.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	300
Override Example	<code>\$sugar_config['cache_expire_timeout'] = 400;</code>

calendar

Description	An array that defines all of the various settings for the Calendar module.
Type	Array
Versions	6.4.0+
Editions	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['calendar'] = array();</code>

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	Professional, Enterprise, Ultimate
Default Value	15
Override Example	<code>\$sugar_config['calendar']['day_time_step'] = 15;</code>

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	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['calendar']['default_view'] = 'week';</code>

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	Professional, Enterprise, Ultimate
Default Value	true
Override Example	<code>\$sugar_config['calendar']['items_draggable'] = true;</code>

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	Professional, Enterprise, Ultimate
Default Value	true
Override Example	<code>\$sugar_config['calendar']['items_resizable'] = true;</code>

calendar.show_calls_by_default

Description	Display/Hide calls by default.
Type	Boolean
Range of values	true and false
Versions	6.4.0+
Editions	Professional, Enterprise, Ultimate
Default Value	true
Override Example	<code>\$sugar_config['calendar']['show_calls_by_default'] = true;</code>

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	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['calendar']['week_timestep'] = 30;</code>

check_query

Description	Validates queries when adding limits.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	true
Override Example	<code>\$sugar_config['check_query'] = true;</code>

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	<code>\$sugar_config['check_query_cost'] = 10;</code>

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, Enterprise, Ultimate
Default Value	true
Override Example	<code>\$sugar_config['collapse_subpanels'] = true;</code>

cron

Description	Array that defines all of the cron parameters.
Type	Array
Versions	6.5.0+
Editions	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['cron'] = array();</code>

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, Enterprise, Ultimate
Default Value	false
Override Example	<code>\$sugar_config['cron']['enforce_runtime'] = true;</code>

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 <code>max_jobs</code> to set this value due to bug #62936 (https://web.sugarcrm.com/support/issues/62936).
Type	Integer
Versions	6.5.0+
Editions	Professional, Enterprise, Ultimate
Default Value	6.5.x: 10 7.6.x: 25
Override Example	<code>\$sugar_config['cron']['max_cron_jobs'] = 10;</code>

`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, <code>cron.php</code> 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 <code>cron.enforce_runtime</code> is set to true.
Type	Integer : Seconds
Versions	6.5.0+
Editions	Professional, Enterprise, Ultimate
Default Value	180
Override Example	<code>\$sugar_config['cron']['max_cron_runtime'] = 60;</code>

`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	Professional, Enterprise, Ultimate
Default Value	30
Override Example	<code>\$sugar_config['cron']['min_cron_interval'] = 30;</code>

csrf

Description	An array defining attributes for CSRF token protection.
Type	Array
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Override Example	<code>\$sugar_config['csrf'] = array();</code>

csrf.soft_fail_form

Description	When opted in for CSRF form authentication, any failures will result in a CSRF message to the user indicating a potential CSRF attack and an aborted action. If you are unsure whether the CSRF tokens are properly implemented in your backward compatible modules, this configuration parameter can be set to true. Setting this to true will avoid any exceptions being thrown to the end user. By default, any failures will result in a fatal log message indicating the issue. If you are running in "soft failure", a second fatal message will be logged such as "CSRF: attack vector NOT mitigated, soft failure mode enabled".
-------------	--

	Be careful enabling this configuration as it will only log CSRF authentication failures and will not protect your system.
Type	Boolean
Range of values	true and false
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Override Example	<code>\$sugar_config['csrf']['soft_fail_form'] = true;</code>

csrf.token_size

Description	The size in bytes of the CSRF token to generate.
Type	Integer : Bytes
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	32
Override Example	<code>\$sugar_config['csrf']['token_size'] = 16;</code>

custom_help_base_url

Description	Allows an instance to specify a custom help url for their user
Type	String : URL
Versions	6.4.3+
Editions	Professional, Enterprise, Ultimate
Default Value	<code>http://www.sugarcrm.com/crm/product_doc.php</code>
Override Example	<code>\$sugar_config['custom_help_base_ur</code>

	l'] = 'http://www.custom_url/index.php';
--	---

custom_help_url

Description	Designate the URL used to redirect the user to help documentation.
Type	String : Website address
Versions	6.4.3+
Editions	Professional, Enterprise, Ultimate
Default Value	http://www.sugarcrm.com/crm/product_doc.php
Override Example	<code>\$sugar_config['custom_help_url'] = 'http://www.sugarcrm.com/crm/product_doc.php';</code>

dbconfig

Description	Defines all of the connection parameters for the database server.
Type	Array
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['dbconfig'] = array();</code>

dbconfig.db_host_instance

Description	Defines the host instance for MSSQL connections.
Type	String : Database Host Instance Name
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate

Default Value	empty
Override Example	<code>\$sugar_config['dbconfig']['db_host_instance'] = 'SQLEXPRESS';</code>

dbconfig.db_host_name

Description	Part of the 'dbconfig' array. Defines the host name of the database server.
Type	String : Host name
Range of values	host name of database server
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	empty
Override Example	<code>\$sugar_config['dbconfig']['db_host_name'] = 'localhost';</code>

dbconfig.db_manager

Description	Part of the 'dbconfig' array. Defines the specific library used to connect with your database. Instances running on Sugar's cloud environment will have this setting enforced as MySQLiManager.
Type	String : Database Manager
Range of values	The class name of the database driver, Possible values are: 'MySQLManager', 'MySQLiManager', 'FreeTDSManager', 'MssqlManager', and 'SqlsrvManager'.
Versions	6.4.0+
Editions	Professional, Enterprise, Ultimate
Default Value	Determined by install: 'MySQLManager', 'MySQLiManager', 'FreeTDSManager', 'MssqlManager', or 'SqlsrvManager'
Sugar Cloud Value	MySQLiManager

Override Example	<code>\$sugar_config['dbconfig']['db_manager'] = 'MysqliManager';</code>
------------------	--

dbconfig.db_name

Description	Part of the 'dbconfig' array. Defines the database name to connect to on the database server.
Type	String : Database Name
Range of values	database name
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	empty
Override Example	<code>\$sugar_config['dbconfig']['db_name'] = 'sugar_db';</code>

dbconfig.db_password

Description	Part of the 'dbconfig' array. Defines the password that correlates to the db_user_name parameter.
Type	String : Password
Range of values	password of the user defined in db_user_name
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	empty
Override Example	<code>\$sugar_config['dbconfig']['db_password'] = 'sql_password';</code>

dbconfig.db_port

Description	Part of the 'dbconfig' array. Defines the
-------------	---

	port number on the server to connect to for authentication and transactions.
Type	String : Network Port
Range of values	port number to connect to on the database server
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	3306
Override Example	<code>\$sugar_config['dbconfig']['db_port'] = '3306';</code>

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 Engine
Range of values	'mysql', 'mssql', 'db2', and 'oracle'
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['dbconfig']['db_type'] = 'mysql';</code>

dbconfig.db_user_name

Description	Defines the user to connect to the Sugar database as.
Type	String : Database User
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['dbconfig']['db_user_name'] = 'sql_user';</code>

dbconfigoption.autofree

Description	Automatically frees the database reference when it closes the reference.
Type	Boolean
Range of values	true and false
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Override Example	<code>\$sugar_config['dbconfigoption']['autofree'] = true;</code>

dbconfigoption.collation

Description	The set of rules to be used for comparing characters in a character set.
Type	String : Collation type
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	utf8_general_ci
Override Example	<code>\$sugar_config['dbconfigoption']['collation'] = 'utf8_general_ci';</code>

dbconfigoption.debug

Description	Enables debugging on the database connection.
Type	Boolean
Range of values	true and false
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	false

Override Example	<code>\$sugar_config['dbconfigoption']['debug'] = true;</code>
------------------	--

dbconfigoption.persistent

Description	Determines whether Sugar should use persistent connection when possible to connecting to the database.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Override Example	<code>\$sugar_config['dbconfigoption']['persistent'] = true;</code>

dbconfigoption.ssl

Description	Enables SSL on the database connection.
Type	Boolean
Range of values	true and false
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Override Example	<code>\$sugar_config['dbconfigoption']['ssl'] = true;</code>

dbconfigoption.ssl_options

Description	An array detailing the SSL database connection options.

Type	Array
Range of values	true and false
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	empty
Override Example	<code>\$sugar_config['dbconfigoption']['ssl_options'] = array();</code>

dbconfigoption.ssl_options.ssl_cpath

Description	The path the trusted SSL certificate authority file in PEM format for SSL connection to the database.
Type	String : File path
Range of values	true and false
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	empty
Override Example	<code>\$sugar_config['dbconfigoption']['ssl_options']['ssl_cpath'] = 'path/to/ca-cert';</code>

dbconfigoption.ssl_options.ssl_cert

Description	The path the SSL certificate file for SSL connection to the database.
Type	String : File path
Range of values	true and false
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	empty
Override Example	<code>\$sugar_config['dbconfigoption']['ssl_options']['ssl_cert'] = 'path/to/cert';</code>

dbconfigoption.ssl_options.ssl_cipher

Description	The SSL cipher to be used for encryption.
Type	String : Cipher
Range of values	true and false
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	empty
Override Example	<pre>\$sugar_config['dbconfigoption']['ssl_options']['ssl_cipher'] = 'cipher';</pre>

dbconfigoption.ssl_options.ssl_key

Description	The path the SSL key for SSL connection to the database.
Type	String : File path
Range of values	true and false
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	empty
Override Example	<pre>\$sugar_config['dbconfigoption']['ssl_options']['ssl_key'] = 'path/to/key';</pre>

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	Professional, Enterprise, Ultimate
Default Value	2
Override Example	<code>\$sugar_config['default_currency_significant_digits'] = 2;</code>

default_date_format

Description	Modifies the default date format for all users.
Type	String : Date format
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	m/d/Y
Override Example	<code>\$sugar_config['default_date_format'] = 'm/d/Y';</code>

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	Professional, Enterprise, Ultimate
Default Value	.
Override Example	<code>\$sugar_config['default_decimal_seperator'] = '.';</code>

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	Professional, Enterprise, Ultimate
Default Value	sugar
Override Example	<code>\$sugar_config['default_email_client'] = 'sugar';</code>

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	Professional, Enterprise, Ultimate
Default Value	en_us
Override Example	<code>\$sugar_config['default_language'] = 'en_us';</code>

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	Professional, Enterprise, Ultimate
Default Value	,
Override Example	<code>\$sugar_config['default_number_grouping_seperator'] = ',';</code>

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	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['default_permissions'] = array();</code>

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 Sugar's cloud environment will have this setting enforced as 1528.
Type	Integer : Octal Value
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Sugar Cloud Value	1528

Override Example	<code>\$sugar_config['default_permissions']['dir_mode'] = 1528;</code>
------------------	--

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 Sugar's cloud environment will have this setting enforced as 432.
Type	Integer : Octal value
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Sugar Cloud Value	432
Override Example	<code>\$sugar_config['default_permissions']['file_mode'] = 432;</code>

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 Sugar's cloud environment will have this setting enforced as empty.
Type	String : Web group
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Sugar Cloud Value	empty

Override Example	<code>\$sugar_config['default_permissions']['group'] = 'apache';</code>
------------------	---

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 Sugar's cloud environment will have this setting enforced as empty
Type	String : Web user
Range of values	Apache user
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Sugar Cloud Value	empty
Override Example	<code>\$sugar_config['default_permissions']['user'] = 'apache';</code>

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	Professional, Enterprise, Ultimate
Default Value	false
Override Example	<code>\$sugar_config['default_user_is_admin'] = true;</code>

developerMode

Description	Rebuilds various cached files when a page is accessed. Can be set by an admin in Admin > System Settings. Instances running on Sugar's cloud environment will have this setting enforced as false.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Sugar Cloud Value	false
Override Example	<code>\$sugar_config['developerMode'] = true;</code>

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, Enterprise, Ultimate
Default Value	604800
Override Example	<code>\$sugar_config['diagnostic_file_max_lifetime'] = 604800;</code>

disabled_languages

Description	Allows an admin to select languages that are disabled for the instance.

Type	String : CSV Language Keys
Versions	6.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	empty
Override Example	<code>\$sugar_config['disabled_languages'] = 'bg_BG,da_DK,de_DE';</code>

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	Professional, Enterprise, Ultimate
Default Value	false
Override Example	<code>\$sugar_config['disable_count_query'] = true;</code>

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	Professional, Enterprise, Ultimate

Default Value	false
Override Example	<code>\$sugar_config['disable_export'] = true;</code>

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, Enterprise, Ultimate
Override Example	<code>\$sugar_config['disable_related_calc_fields'] = true;</code>

disable_unknown_platforms

Description	Controls whether or not unregistered platforms are allowed to be used when
-------------	--

	logging in using v10 REST API. Custom platforms can be registered using the Platform extension. Used to prevent excessive metadata generation when invalid or unrecognized platform types are specified in an API call.
Type	Boolean
Range of values	true and false
Versions	7.6.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	7.6 - 7.10: false 7.11: true
Override Example	<code>\$sugar_config['disable_unknown_platforms'] = true;</code>

disable_uw_upload

Description	Disables the upgrade wizard from being accessible through the Sugar admin interface. Instances running on Sugar's cloud environment will have this setting enforced as true.
Type	Boolean
Range of values	true and false
Versions	5.2.0.j+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Sugar Cloud Value	true
Override Example	<code>\$sugar_config['disable_uw_upload'] = true;</code>

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	Professional, Enterprise, Ultimate
Default Value	false
Override Example	<code>\$sugar_config['disable_vcr'] = true;</code>

dump_slow_queries

Description	Logs slow queries to the sugar log file. Instances running on Sugar's cloud environment will have this setting enforced as false.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Sugar Cloud Value	false
Override Example	<code>\$sugar_config['dump_slow_queries'] = true;</code>

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	Professional, Enterprise, Ultimate
Default Value	,
Override Example	<code>\$sugar_config['email_address_separator'] = ',';</code>

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	Professional, Enterprise, Ultimate
Default Value	sugar
Override Example	<code>\$sugar_config['email_default_client'] = 'sugar';</code>

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	Professional, Enterprise, Ultimate
Default Value	true
Override Example	<code>\$sugar_config['email_default_delete_attachments'] = false;</code>

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	Professional, Enterprise, Ultimate
Default Value	html
Override Example	<pre>\$sugar_config['email_default_editor'] = 'plain';</pre>

email_mailer_timeout

Description	The connection timeout period when sending an email.
Type	Integer : Seconds
Versions	7.8.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	10
Override Example	<pre>\$sugar_config['email_mailer_timeout'] = 30;</pre>

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, Enterprise, Ultimate
Default Value	false
Override Example	<code>\$sugar_config['enable_inline_reports_edit'] = true;</code>

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, Enterprise, Ultimate
Default Value	true
Override Example	<code>\$sugar_config['enable_mobile_redirect'] = false;</code>

external_cache.memcache.host

Description	The host url for memcache.
Type	String : Host URL
Versions	6.2.0+
Editions	Professional, Enterprise
Default Value	127.0.0.1
Override Example	<code>\$sugar_config['external_cache']['memcache']['host'] = '192.168.1.1';</code>

external_cache.memcache.port

Description	The host port for memcache.
Type	Integer : Port number
Versions	6.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	11211
Override Example	<code>\$sugar_config['external_cache']['memcache']['port'] = 11212;</code>

external_cache_db_gc_probability

Description	Probability factor to determine when garbage collection on stale keys from the DB backend will happen.
Type	Integer : Probability factor
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	0.0001
Override Example	<code>\$sugar_config['external_cache_db_gc_probability'] = 0.0005;</code>

external_cache_db_gc_threshold

Description	The threshold in milliseconds to flag garbage collection queries as [SLOW] in sugarcrm.log.
Type	Integer : Milliseconds
Range of values	true and false
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	200
Override Example	<code>\$sugar_config['external_cache_db_gc_threshold'] = 500;</code>

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 Sugar's cloud environment will have this setting enforced as true.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Sugar Cloud Value	true
Override Example	<pre>\$sugar_config['external_cache_disabled'] = true;</pre>

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 Sugar's cloud environment will have this setting enforced as true.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Sugar Cloud Value	true
Override Example	<pre>\$sugar_config['external_cache_disabled_apc'] = true;</pre>

```
bled_apc'] = false;
```

external_cache_disabled_db

Description	Disables the database key/value cache from being eligible as cache backend.
Type	Boolean
Range of values	true and false
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Override Example	<pre>\$sugar_config['external_cache_disabled_db'] = true;</pre>

external_cache_disabled_memcache

Description	Disables Memcache caching in Sugar. Recommended setting is false. This setting is normally normally only enabled to determine if there is a conflict with PHP caching. Instances running on Sugar's cloud environment will have this setting enforced as true.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Sugar Cloud Value	true
Override Example	<pre>\$sugar_config['external_cache_disabled_memcache'] = true;</pre>

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 Sugar's cloud environment will have this setting enforced as true.
Type	Boolean
Range of values	true and false
Versions	6.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Sugar Cloud Value	true
Override Example	<code>\$sugar_config['external_cache_disabled_memcached'] = false;</code>

external_cache_disabled_mongo

Description	Disables Mongo caching in Sugar. Recommended setting is false. This setting is normally normal only enabled to determine if there is a conflict with PHP caching. Instances running on Sugar's cloud environment will have this setting enforced as true.
Type	Boolean
Range of values	true and false
Versions	6.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Sugar Cloud Value	true
Override Example	<code>\$sugar_config['external_cache_disabled_mongo'] = true;</code>

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	Professional, Enterprise, Ultimate
Override Example	<pre>\$sugar_config['external_cache_disabled_smash'] = false;</pre>

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 Sugar's cloud environment will have this setting enforced as true.
Type	Boolean
Range of values	true and false
Versions	6.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Sugar Cloud Value	true
Override Example	<pre>\$sugar_config['external_cache_disabled_wincache'] = false;</pre>

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 Sugar's cloud environment will have this setting enforced as true.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Sugar Cloud Value	true
Override Example	<code>\$sugar_config['external_cache_disabled zend'] = false;</code>

external_cache_force_backend

Description	Force given external key/value cache backend. Make sure that the requirements are met and any other cache backend specific configuration is applied.
Type	String : Backend type
Range of values	apc, db, file, memcache, memcached, memory, redis, smash, wincache, and zend
Versions	6.2.0+
Editions	Professional, Enterprise
Default Value	empty
Override Example	<code>\$sugar_config['external_cache_force_backend'] = 'db';</code>

forms

Description	An array defining form requirements.
Type	Array
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['forms'] = array();</code>

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	Professional, Enterprise, Ultimate
Default Value	false
Override Example	<code>\$sugar_config['forms']['requireFirst'] = true;</code>

freebusy_use_vcal_cache

Description	Prior to Sugar version 7.6, FreeBusy Calendar searches used the vcal table to cache user meeting and call activity for the purpose of determining user availability in for future free time search. As of 7.6, a new design allowed for more accurate free/busy searching that no longer used this cache, but the cache was still being written to, by default, when calls/meetings were created/updated for backward compatibility reasons. As of Sugar 7.9, in order to enhance performance, the cache is no longer being written to by default. Instead, the override variable
-------------	---

	'FreeBusyCache_Enabled' must be set to true for the Cache to be written. Since this cache is no longer used inside the Sugar application, this option should be used with caution as the cache itself is expected to be removed from the Sugar product in a future release.
Type	Boolean
Range of values	true and false
Versions	6.1.0RC1+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Override Example	<code>\$sugar_config['freebusy_use_vcal_cache'] = true;</code>

hide_admin_backup

Description	Removes the Backups option in the admin menu and also prevents direct access to the feature. Instances running on Sugar's cloud environment will have this setting enforced as true.
Type	Boolean
Range of values	true and false
Versions	6.5.1+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Sugar Cloud Value	true
Override Example	<code>\$sugar_config['hide_admin_backup'] = true;</code>

hide_admin_licensing

Description	Hides the License settings subpanel in the administrative panel. Instances
-------------	--

	running on Sugar's cloud environment will have this setting enforced as false.
Type	Boolean
Range of values	true and false
Versions	6.5.0+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Sugar Cloud Value	false
Override Example	<code>\$sugar_config['hide_admin_licensing'] = true;</code>

hide_full_text_engine_config

Description	Determines if the FTS settings are present in the admin search page in Admin > Search. Instances running on Sugar's cloud environment will have this setting enforced as true.
Type	Boolean
Range of values	true and false
Versions	6.5.15+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Sugar Cloud Value	true
Override Example	<code>\$sugar_config['hide_full_text_engine_config'] = true;</code>

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	Professional, Enterprise, Ultimate
Default Value	false
Override Example	<code>\$sugar_config['hide_subpanels'] = true;</code>

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 expanded, 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	Professional, Enterprise, Ultimate
Default Value	false
Override Example	<code>\$sugar_config['hide_subpanels_on_login'] = true;</code>

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, Enterprise, Ultimate
Default Value	50
Override Example	<code>\$GLOBALS['sugar_config']['history_max_viewed'] = 25;</code>

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 Sugar's cloud environment will have this setting enforced as true.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	true
Sugar Cloud Value	true
Override Example	<code>\$sugar_config['installer_locked'] = false;</code>

jobs

Description	Job Queue configurations.
Type	Array
Versions	6.5.0+
Editions	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['jobs'] = array();</code>

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	Professional, Enterprise, Ultimate
Default Value	21
Override Example	<code>\$sugar_config['jobs']['hard_lifetime'] = 21;</code>

jobs.max_retries

Description	Maximum number of failures for job. Default is 5.
Type	Integer : Number of failures
Versions	6.5.0+
Editions	Professional, Enterprise, Ultimate
Default Value	5
Override Example	<code>\$sugar_config['jobs']['max_retries'] = 5;</code>

jobs.min_retry_interval

Description	Minimal interval between job reruns. Default is 30 seconds.
Type	Integer : Seconds
Versions	6.5.0+
Editions	Professional, Enterprise, Ultimate
Default Value	30
Override Example	<code>\$sugar_config['jobs']['min_retry_interval'] = 30;</code>

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	Professional, Enterprise, Ultimate
Default Value	7
Override Example	<code>\$sugar_config['jobs']['soft_lifetime'] = 7;</code>

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	Professional, Enterprise, Ultimate
Default Value	3600
Override Example	<code>\$sugar_config['jobs']['timeout'] = 86400;</code>

list_max_entries_per_page

Description	Listview items per page.
Type	String : Records per page
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	20

Override Example	<code>\$sugar_config['list_max_entries_per_page'] = '20';</code>
------------------	--

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, Enterprise, Ultimate
Default Value	100
Override Example	<code>\$sugar_config['list_report_max_per_page'] = 100;</code>

logger

Description	An array that defines all of the logging settings.
Type	Array
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['logger'] = array();</code>

logger.file.dateFormat

Description	The date format for the log file is any value that is acceptable to the PHP <code>strftime()</code> function. The default is '%c'. For a complete list of available date formats, please see the <code>strftime()</code> 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	Professional, Enterprise, Ultimate
Default Value	%c
Override Example	<code>\$sugar_config['logger']['file']['dateFormat'] = '%c';</code>

logger.file.ext

Description	The extension of the log file. The default value is '.log'. Instances running on Sugar's cloud environment will have this setting enforced as .log.
Type	String : File extension
Range of values	Extension for the log
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	.log
Sugar Cloud Value	.log
Override Example	<code>\$sugar_config['logger']['file']['ext'] = '.log';</code>

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	Professional, Enterprise, Ultimate
Default Value	10
Override Example	<code>\$sugar_config['logger']['file']['maxLogs'] = 10;</code>

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 Sugar's cloud environment will have this setting enforced as 10MB.
Type	String : Size
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	10MB
Sugar Cloud Value	10MB
Override Example	<code>\$sugar_config['logger']['file']['maxSize'] = '10MB';</code>

logger.file.name

Description	The name of the log file to be written to. Instances running on Sugar's cloud environment will have this setting enforced as sugarcrm.
Type	String : Filename
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	sugarcrm

Sugar Cloud Value	sugarcrm
Override Example	<code>\$sugar_config['logger']['file']['name'] = 'sugarcrm';</code>

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 <code>strftime()</code> PHP documentation at http://php.net/manual/en/function.strftime.php . Instances running on Sugar's cloud environment will have this setting enforced as empty.
Type	String : Suffix pattern
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	empty
Sugar Cloud Value	empty
Override Example	<code>\$sugar_config['logger']['file']['suffix'] = '%m_%Y';</code>

logger.level

Description	Determines the logging level of the system. The recommended setting is 'fatal'. Instances running on Sugar's cloud 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	Professional, Enterprise, Ultimate
Default Value	fatal
Sugar Cloud Value	fatal
Override Example	<code>\$sugar_config['logger']['level'] = 'fatal';</code>

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 <code>App.config.logger.writeToServer = true;</code> in their browsers console. To simulate logging actions through the console, developers can use: <code>App.logger.trace('message');</code> <code>App.logger.debug('message');</code> <code>App.logger.info('message');</code> <code>App.logger.warn('message');</code> <code>App.logger.fatal('message');</code>
Type	Boolean
Range of values	true and false
Versions	7.5.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Override Example	<code>\$sugar_config['logger']['write_to_server'] = true;</code>

logger_visible

Description	Determines whether the Logger Settings panel is visible to administrators in Admin > System Settings. Instances running on Sugar's cloud environment will have this setting
-------------	---

	enforced as false.
Type	Boolean
Range of values	true and false
Versions	6.7.0+
Editions	Professional, Enterprise, Ultimate
Default Value	true
Sugar Cloud Value	false
Override Example	<code>\$sugar_config['logger_visible'] = false;</code>

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 Sugar's cloud environment will have this setting enforced as ..
Type	String : Directory Path
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	.
Sugar Cloud Value	.
Override Example	<code>\$sugar_config['log_dir'] = '..';</code>

log_file

Description	Designates the file name where the instance's logs will be stored. Instances running on Sugar's cloud environment will have this setting enforced as sugarcrm.log.
Type	String : Name of the log file

Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	sugarcrm.log
Sugar Cloud Value	sugarcrm.log
Override Example	<code>\$sugar_config['log_file'] = 'new_sugarcrm.log'</code>

log_memory_usage

Description	Logs the memory usage. Instances running on Sugar's cloud environment will have this setting enforced as false.
Type	Boolean
Range of values	true and false
Versions	5.5.0+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Sugar Cloud Value	false
Override Example	<code>\$sugar_config['log_memory_usage'] = true;</code>

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, Enterprise, Ultimate
Default Value	false
Override Example	<code>\$sugar_config['maintenanceMode'] = false;</code>

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, Enterprise, Ultimate
Default Value	false
Override Example	<pre>\$sugar_config['mark_emails_seen'] = true;</pre>

mass_actions

Description	Array that defines mass action behaviors.
Type	Array
Versions	7.0.0+
Editions	Professional, Enterprise, Ultimate
Override Example	<pre>\$sugar_config['mass_actions'] = array();</pre>

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, Enterprise, Ultimate
Default Value	20
Override Example	<code>\$sugar_config['mass_actions']['mass_link_chunk_size'] = 20;</code>

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 Sugar's cloud environment will have this setting enforced as 1.
Type	Integer : Seconds
Versions	6.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	null
Sugar Cloud Value	1
Override Example	<code>\$sugar_config['max_session_time'] = 1;</code>

minify_resources

Description	Determines whether minification and compression are applied to javascript resources
Type	Boolean
Range of values	true and false
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate

Default Value	true
Override Example	<code>\$sugar_config['minify_resources'] = false;</code>

moduleInstaller

Description	Array that defines restrictions on module installations via the Module Loader utility.
Type	Array
Versions	5.2.0.j+
Editions	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['moduleInstaller'] = array();</code>

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	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['moduleInstaller']['disableFileScan'] = true;</code>

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 Sugar's cloud environment will have this setting enforced as true.
Type	Boolean
Range of values	true and false
Versions	5.5.0+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Sugar Cloud Value	true
Override Example	<code>\$sugar_config['moduleInstaller']['packageScan'] = true;</code>

moduleInstaller.validExt

Description	Part of the moduleInstaller array. When moduleInstaller.packageScan is set to true, Sugar will not allow certain file extensions to be present in an installable package. By default, Sugar allows the following extensions: 'png', 'gif', 'jpg', 'css', 'js', 'php', 'txt', 'html', 'htm', 'tpl', 'pdf', 'md5', 'xml', 'hbs', 'less', and 'wsdl'. This parameter allows you to define additional extensions deemed safe to install on your instance of Sugar. Instances running on Sugar's cloud environment will have this setting enforced as <code>array('eot','svg','tff','woff','woff2','xml')</code> .
Type	Array : Extensions
Range of values	File extensions to allow

Versions	6.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	array()
Sugar Cloud Value	array('eot','svg','tff','woff','woff2','xml')
Override Example	<code>\$sugar_config['moduleInstaller']['validExt'] = array('swf', 'log');</code>

mso_fixup_paragraph_tags

Description	Determines whether email HTML is scrubbed for empty paragraph tags when displayed in the application. Setting this value to true will enable the HTML scrubbing. Enabling this setting does not affect the HTML stored in the database from the initial import. Created as a result of bug 66022 (https://web.sugarcrm.com/support/issues/66022).
Type	Boolean
Range of values	true and false
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Override Example	<code>\$sugar_config['mso_fixup_paragraph_tags'] = true;</code>

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, Enterprise, Ultimate
Default Value	false
Override Example	<code>\$sugar_config['noPrivateTeamUpdate'] = true;</code>

oauth_token_expiry

Description	Sets whether OAuth tokens will expire.
Type	String
Range of values	true and false
Versions	7.5.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	False
Override Example	<code>\$sugar_config['oauth_token_expiry'] = '0';</code>

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, Enterprise, Ultimate
Default Value	86400
Override Example	<code>\$sugar_config['oauth_token_life'] = '86400';</code>

passwordHash

Description	Array that defines the password hashing behaviors.
Type	Array
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['passwordHash'] = array();</code>

passwordHash.algo

Description	The specific algorithm to be used by the hashing backend. The available values depend on the selected passwordHash.backend. See http://php.net/manual/en/password.constants.php for more information when using the native backend.
Type	String : Algorithm Type
Range of values	For native backend: "PASSWORD_DEFAULT" and "PASSWORD_BCRYPT". For sha2 backend: "CRYPT_SHA256" and "CRYPT_SHA512"
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	"PASSWORD_DEFAULT" for native. "CRYPT_SHA256" for sha2 backend.
Override Example	<code>\$sugar_config['passwordHash']['algo'] = 'PASSWORD_BCRYPT';</code>

passwordHash.allowLegacy

Description	Allow logins of users who have their password stored using the insecure legacy MD5 hash. During the transition
-------------	--

	period for the 7.7 series, this will be allowed out of the box. Versions past 7.7 will no longer allow by default authentication against insecure hashes. This configuration parameter can be used to change the default behavior.
Type	Boolean
Range of values	true and false
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	7.7.x: true 7.8.x+:false
Override Example	<code>\$sugar_config['passwordHash']['allowLegacy'] = false;</code>

passwordHash.backend

Description	The password hash backend class to use. By default, the "native" backend is used which uses Blowfish to hash the passwords in the database. An alternative is using the "sha2" backend which makes use of SHA-2 hashing instead. Depending on the backend, different configuration options are available for passwordHash.algo and passwordHash.options.
Type	String
Range of values	'sha2' and 'native'
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	native
Override Example	<code>\$sugar_config['passwordHash']['backend'] = 'sha2';</code>

passwordHash.options

Description	The available configuration values depend on the selected passwordHash.backend. See http://php.net/manual/en/function.password-hash.php when using the native backend and http://php.net/manual/en/function.crypt.php for the sha2 backend. Note that only the following specified options are allowed. For native backend: passwordHash.options.cost. For sha2 backend: passwordHash.options.rounds.
Type	Array
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Override Example	<pre>\$sugar_config['passwordHash']['options'] = array();</pre>

passwordHash.options.cost

Description	An available option when the passwordHash.backend configuration is set to "native". More information on the native backend can be found at http://php.net/manual/en/function.password-hash.php
Type	Integer
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	10
Override Example	<pre>\$sugar_config['passwordHash']['options']['cost'] = 15;</pre>

passwordHash.options.rounds

Description	An available option when the passwordHash.backend configuration is set to "sha2". More information on the sha2 backend can be found at http://php.net/manual/en/function.crypt.php
Type	Integer
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	5000
Override Example	<code>\$sugar_config['passwordHash']['options']['rounds'] = 4000;</code>

passwordHash.rehash

Description	When enabled, the system will automatically rehash the user's password when successfully authenticated. This allows adoption to the newly configured password hash backend without resetting user's passwords.
Type	Boolean
Range of values	true and false
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	true
Override Example	<code>\$sugar_config['passwordHash']['rehash'] = false;</code>

passwordsetting

Description	Defines all of the password requirements for the instance.
Type	Array

Versions	5.5.0+
Editions	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['passwordsetting'] = array();</code>

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	Professional, Enterprise, Ultimate
Default Value	1
Override Example	<code>\$sugar_config['passwordsetting']['forgotpasswordON'] = '0';</code>

passwordsetting.linkexpiration

Description	Determines whether the password reset link expires.
Type	String
Range of values	0, 1
Versions	6.0.0+
Editions	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['passwordsetting']['linkexpiration'] = '0';</code>

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	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['passwordsetting']['onespecial'] = '0';</code>

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	Professional, Enterprise, Ultimate
Default Value	1
Override Example	<code>\$sugar_config['passwordsetting']['onenumber'] = '1';</code>

passwordsetting.oneupper

Description	Configures whether at least one upper-case letter is required in users' passwords.
Type	Boolean
Range of values	0, 1
Versions	6.0.0+
Editions	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['passwordsetting']['oneupper'] = 1;</code>

passwordsetting.systexpirationtype

Description	Specifies the unit of measurement for passwordsetting.systexpirationtime. 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	Professional, Enterprise, Ultimate
Default Value	1
Override Example	<code>\$sugar_config['passwordsetting']['systexpirationtype'] = '7';</code>

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, Enterprise, Ultimate
Default Value	86400
Override Example	<code>\$sugar_config['pdf_file_max_lifetime'] = 86400;</code>

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, Enterprise, Ultimate
Override Example	<code>\$sugar_config['perfProfile'] = array();</code>

perfProfile.TeamSecurity

Description	Determines whether the team security filtering is applied.
Type	Array
Versions	7.2.2.0+
Editions	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['perfProfile']['Team Security'] = array();</code>

perfProfile.TeamSecurity.default

Description	This setting is used to enable or disable Team Security denormalization feature
-------------	---

	by default across the application.
Type	Array
Range of values	true and false
Versions	7.7.2.0+
Editions	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['perfProfile']['Team Security']['default'] = array();</code>

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: <code>\$sugar_config['perfProfile']['Team Security']['{module}']['teamset_prefetch'] = true;</code>
Type	Boolean
Range of values	true and false
Versions	7.2.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Override Example	<code>\$sugar_config['perfProfile']['Team Security']['default']['teamset_prefetch'] = true;</code>

perfProfile.TeamSecurity.default.teamset_prefetch_max

Description	The maximum amount of team set ID's to include in the team security clause. If
-------------	--

	<p>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:</p> <pre>\$sugar_config['perfProfile']['TeamSecurity']['{module}']['teamset_prefetch_max'] = true;</pre>
Type	Integer : Records
Versions	7.2.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	empty
Override Example	<pre>\$sugar_config['perfProfile']['TeamSecurity']['default']['teamset_prefetch_max'] = 500;</pre>

perfProfile.TeamSecurity.default.where_condition

Description	<p>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:</p> <pre>\$sugar_config['perfProfile']['TeamSecurity']['{module}']['where_condition'] = true;</pre>
Type	Boolean
Range of values	true and false
Versions	7.2.2.0+
Editions	Professional, Enterprise, Ultimate

Default Value	false
Override Example	<code>\$sugar_config['perfProfile']['Team Security']['default']['where_condition'] = true;</code>

pmse_settings_default

Description	Settings for troubleshooting and debugging the advanced workflow.
Type	Array
Versions	7.6.0.0+
Editions	Enterprise, Ultimate
Override Example	<code>\$sugar_config['pmse_settings_default'] = array();</code>

pmse_settings_default.error_number_of_cycles

Description	Number of cycles before triggering an error in advanced workflow.
Type	String : Cycles
Versions	7.6.0.0+
Editions	Enterprise, Ultimate
Default Value	10
Override Example	<code>\$sugar_config['pmse_settings_default']['error_number_of_cycles'] = 15;</code>

pmse_settings_default.error_timeout

Description	Time in seconds of timeout before triggering an error in advanced workflow.
Type	String : Seconds

Versions	7.6.0.0+
Editions	Enterprise, Ultimate
Default Value	40
Override Example	<code>\$sugar_config['pmse_settings_default']['error_timeout'] = 45;</code>

pmse_settings_default.logger_level

Description	The default logger level in advanced workflow.
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	<code>\$sugar_config['pmse_settings_default']['logger_level'] = 'emergency';</code>

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	Professional, Enterprise, Ultimate
Default Value	true
Override Example	<code>\$sugar_config['require_accounts'] = false;</code>

rest_response_etag_cache_age

Description	Controls how long the browser should cache rest responses.
Type	Integer : Seconds
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	10
Override Example	<code>\$sugar_config['rest_response_etag_cache_age'] = 15;</code>

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	<code>\$sugar_config['roleBasedViews'] = true;</code>

SAML_provisionUser

Description	Determines whether or not a new user is auto-provisioned when authenticating through SAML. Setting this setting to false will prevent a new user from being created.
Type	Boolean
Range of values	true and false

Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	true
Override Example	<code>\$sugar_config['SAML_provisionUser'] = false;</code>

SAML_X509Cert

Description	The SAML Certificate Key.
Type	String : SAML Certificate Key
Versions	6.1.0+
Editions	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['SAML_X509Cert'] = '-----BEGIN CERTIFICATE-----CERTIFICATE KEY-----END CERTIFICATE-----';</code>

search_engine

Description	Array that defines the search engine behaviors.
Type	Array
Versions	7.2.0+
Editions	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['search_engine'] = array();</code>

search_engine.max_bulk_delete_threshold

Description	The maximum number of records that can be deleted at a time.
Type	Integer : Number of records

Versions	7.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	3000
Override Example	<code>\$sugar_config['search_engine']['max_bulk_delete_threshold'] = 3000;</code>

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, Enterprise, Ultimate
Default Value	15000
Override Example	<code>\$sugar_config['search_engine']['max_bulk_query_threshold'] = 20000;</code>

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, Enterprise, Ultimate
Default Value	5000
Override Example	<code>\$sugar_config['search_engine']['search_engine.max_bulk_threshold'] = 10000;</code>

search_engine.postpone_job_time

Description	Amount of time to postpone a job by so that it's not executed twice during the same request.
Type	Integer : Seconds
Versions	7.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	120
Override Example	<code>\$sugar_config['search_engine']['search_engine.postpone_job_time'] = 70;</code>

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	Professional, Enterprise, Ultimate
Default Value	false
Override Example	<code>\$sugar_config['search_wildcard_infront'] = true;</code>

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 Sugar's cloud environment will have this setting enforced as empty.
Type	String
Range of values	Directory path
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	empty
Sugar Cloud Value	empty
Override Example	<code>\$sugar_config['session_dir'] = '/tmp/SugarSession/';</code>

showThemePicker

Description	Removes the theme selection drop down.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	true
Override Example	<code>\$sugar_config['showThemePicker'] = false;</code>

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	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['show_download_tab'] = true;</code>

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	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['site_url'] = 'http://my.sugarinstance.com';</code>

slow_query_time_msec

Description	Slow query time threshold. Instances running on Sugar's cloud environment will have this setting enforced as 5000.
Type	String : Milliseconds
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	5000
Sugar Cloud Value	5000
Override Example	<code>\$sugar_config['slow_query_time_msec'] = '1000';</code>

stack_trace_errors

Description	Displays stack trace of errors.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Override Example	<code>\$sugar_config['stack_trace_errors'] = true;</code>

studio_max_history

Description	When layout changes are made in Studio, a history of those changes are recorded with each save & deploy under <code>./custom/history/modules/</code> . The <code>studio_max_history</code> 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 Sugar's cloud 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	Professional, Enterprise, Ultimate
Default Value	50
Sugar Cloud Value	50
Override Example	<code>\$sugar_config['studio_max_history'] = 100;</code>

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	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['sugar_version'] = '6.7.3';</code>

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	Professional, Enterprise, Ultimate
Default Value	<code>cache/xml/</code>
Override Example	<code>\$sugar_config['tmp_dir'] = 'cache/xml/';</code>

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, Enterprise, Ultimate
Default Value	86400

Override Example	<code>\$sugar_config['tmp_file_max_lifetime'] = 86400;</code>
------------------	---

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	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['tracker_max_display_length'] = 45;</code>

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	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['unique_key'] = 'c0b5475f3f5b26ddb2976edc8865b5f6';</code>

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	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['upload_badext'][] = 'swf';</code>

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	Professional, Enterprise, Ultimate
Default Value	upload/
Override Example	<code>\$sugar_config['upload_dir'] = 'upload/';</code>

upload_maxsize

Description	The maximum individual file size that users can upload to modules that support file uploads. Restricts the upload limit from within Sugar without affecting any other applications that may be running on your server. This
-------------	---

	limit can be easily adjusted in Sugar via Admin > System Settings but is only useful if it is set to a size smaller than the php.ini limits. For more information, refer to the Uploads documentation.
Type	Integer : Filesize in bytes
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['upload_maxsize'] = 40000000;</code>

upload_wrapper_class

Description	The name of the class used as upload wrapper.
Type	String : Upload wrapper class
Versions	6.7.0+
Editions	Professional, Enterprise, Ultimate
Default Value	UploadStream
Override Example	<code>\$sugar_config['upload_wrapper_class'] = 'SugarUploadS3';</code>

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	Professional, Enterprise, Ultimate
Default Value	false
Override Example	<code>\$sugar_config['use_common_ml_dir'] = true;</code>

use_php_code_json

Description	Determines if the environment has a valid version of PHP-JSON. This should be determined by the function <code>returnPhpJsonStatus()</code> and shouldn't be overridden.
Type	Boolean
Range of values	true and false
Versions	5.2.0+
Editions	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['use_php_code_json'] = true;</code>

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	Professional, Enterprise, Ultimate
Default Value	true
Override Example	<code>\$sugar_config['use_real_names'] = true;</code>

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	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['use_sprites'] = false;</code>

validation

Description	An array that defines settings for user input validation behaviors.
Type	Array
Versions	7.7.1.0+
Editions	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['validation'] = array();</code>

validation.compat_mode

Description	Determines compatibility mode for superglobals in the input validation framework. When enabled, setting, unsetting, and modifying <code>\$_GET</code> , <code>\$_POST</code> , or <code>\$_REQUEST</code> values through PHP code is supported. Using PHP code to manipulate superglobals is discouraged. User input parameters should be considered read-only. As a transition period, this is currently
-------------	---

	allowed out-of-box, but may change in a future release. It is highly recommended for developers to verify their customizations work with this parameter set to false in preparation for the 7.9 release.
Type	Boolean
Range of values	true and false
Versions	7.7.1.0+
Editions	Professional, Enterprise, Ultimate
Default Value	7.7.x: true 7.8.x: true 7.9.x: false
Override Example	<code>\$sugar_config['validation']['compat_mode'] = false;</code>

validation.soft_fail

Description	Determines whether soft failure mode for the input validation framework is enabled. When this mode is enabled, any input validation violations will only be reported as a warning in the sugarcrm.log without having any negative impact on the request. When disabled, violations are reported as fatal in the sugarcrm.log and actual exceptions are being thrown resulting in an HTTP 500 response. It is highly recommended for developers to verify their customizations work with this parameter set to false in preparation for the 8.0 release.
Type	Boolean
Range of values	true and false
Versions	7.7.1.0+
Editions	Professional, Enterprise, Ultimate
Default Value	7.7.x: true

	7.8.x: true 7.9.x: true 8.0.x: false
Override Example	<code>\$sugar_config['validation']['soft_fail'] = false;</code>

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	Professional, Enterprise, Ultimate
Default Value	2
Override Example	<code>\$sugar_config['vcal_time'] = '5';</code>

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	Professional, Enterprise, Ultimate
Default Value	true

Override Example	<code>\$sugar_config['verify_client_ip'] = false;</code>
------------------	--

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	Professional, Enterprise, Ultimate
Default Value	10
Override Example	<code>\$sugar_config['wl_list_max_entries_per_page'] = 10;</code>

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	Professional, Enterprise, Ultimate
Default Value	3
Override Example	<code>\$sugar_config['wl_list_max_entries_per_subpanel'] = 3;</code>

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	Professional, Enterprise, Ultimate
Override Example	<code>\$sugar_config['xhprof_config'] = array();</code>

xhprof_config.enable

Description	Enables the xhprof profiler.
Type	Boolean
Range of values	true and false
Versions	6.5.10+
Editions	Professional, Enterprise, Ultimate
Default Value	false
Override Example	<code>\$sugar_config['xhprof_config']['enable'] = true;</code>

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, Enterprise, Ultimate
Override Example	<code>\$sugar_config['xhprof_config']['filter_wt'] = 2;</code>

xhprof_config.flags

Description	The flags for xhprof profiler.
Type	String : xhprof flags
Versions	6.5.10+
Editions	Professional, Enterprise, Ultimate

Override Example	<pre>\$sugar_config['xhprof_config']['flags'] = XHPROF_FLAGS_CPU + XHPROF_FLAGS_MEMORY;</pre>
------------------	---

xhprof_config.ignored_functions

Description	An array of function names to ignore from the profile.
Type	Array : Function names
Versions	6.5.10+
Editions	Professional, Enterprise, Ultimate
Override Example	<pre>\$sugar_config['xhprof_config']['ignored_functions'] = array("function_name");</pre>

xhprof_config.log_to

Description	The path to log the xhprof profiler output to.
Type	String : Directory path
Versions	6.5.10+
Editions	Professional, Enterprise, Ultimate
Override Example	<pre>\$sugar_config['xhprof_config']['log_to'] = '{instance server path}/cache/xhprof';</pre>

xhprof_config.manager

Description	The xhprof manager class to use. Prior to 7.7, specifying values <code>xhprof_config.save_to</code> , <code>xhprof_config.mongoddb_uri</code> , <code>xhprof_config.mongoddb_db</code> , <code>xhprof_config.mongoddb_collection</code> ,
-------------	---

	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 <code>./custom/include/SugarXHprof/</code> and create a file with your custom 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	Professional, Enterprise, Ultimate
Default Value	SugarXHprof
Override Example	<code>\$sugar_config['xhprof_config']['manager'] = 'CustomSugarXHprof';</code>

xhprof_config.memory_limit

Description	The memory limit to set while saving profile data to storage.
Type	String : Memory Usage
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	2048M
Override Example	<code>\$sugar_config['xhprof_config']['memory_limit'] = '1024M';</code>

xhprof_config.mongodb_collection

Description	The name of the mongo db collections.
Type	String : Mongo collection name

Versions	7.6.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	results
Override Example	<code>\$sugar_config['xhprof_config']['mongodb_db'] = 'results_new';</code>

xhprof_config.mongodb_db

Description	The name of the mongo database.
Type	String : Database name
Versions	7.6.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	xhprof
Override Example	<code>\$sugar_config['xhprof_config']['mongodb_db'] = 'xhprof2';</code>

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, Enterprise, Ultimate
Default Value	results
Override Example	<code>\$sugar_config['xhprof_config']['mongodb_options'] = array();</code>

xhprof_config.mongodb_uri

Description	The mongo server URL.
Type	String : URL
Versions	7.6.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	mongodb://localhost:27017
Override Example	<code>\$sugar_config['xhprof_config']['mongodb_uri'] = 'mongodb://localhost:27018';</code>

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	Professional, Enterprise, Ultimate
Default Value	10
Override Example	<code>\$sugar_config['xhprof_config']['sample_rate'] = 1;</code>

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, Enterprise, Ultimate
Default Value	mongodb
Override Example	<code>\$sugar_config['xhprof_config']['save_to'] = 'mongodb';</code>

xhprof_config.track_elastic

Description	Whether or not to track elastic data.
Type	Boolean
Range of values	true or false
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	true
Override Example	<code>\$sugar_config['xhprof_config']['track_elastic'] = false;</code>

xhprof_config.track_elastic_backtrace

Description	Whether or not to store the elastic backtrace data.
Type	Boolean
Range of values	true or false
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	true
Override Example	<code>\$sugar_config['xhprof_config']['track_elastic_backtrace'] = false;</code>

xhprof_config.track_sql

Description	Whether or not to track SQL queries.
Type	Boolean
Range of values	true or false
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate

Default Value	true
Override Example	<code>\$sugar_config['xhprof_config']['track_sql'] = false;</code>

xhprof_config.track_sql_backtrace

Description	Whether or not to store the SQL backtrace data.
Type	Boolean
Range of values	true or false
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	true
Override Example	<code>\$sugar_config['xhprof_config']['track_sql_backtrace'] = false;</code>

xhprof_config.track_sql_backtrace

Description	Whether or not to store the SQL backtrace data.
Type	Boolean
Range of values	true or false
Versions	7.7.0.0+
Editions	Professional, Enterprise, Ultimate
Default Value	true
Override Example	<code>\$sugar_config['xhprof_config']['track_sql_backtrace'] = false;</code>

Last Modified: 04/27/2018 01:05am

Module Loader

Module Loader is used to install custom file packages such as patches, integrations, and other customizations into a Sugar instance in the form of a Module Loadable Package. This documentation covers the basics and best practices for creating module loadable packages for a Sugar installation.

Last Modified: 10/14/2016 04:12pm

Introduction to the Manifest

Overview

Module loadable packages rely on a manifest.php file to define the basic properties and installation steps for the package. This documentation explains the various components that make up the manifest file.

Manifest Definitions

Inside of the manifest.php file, there is a \$manifest variable that defines the basic properties of the module loadable package. The various manifest properties are outlined below:

Name	Type	Displayed in Module Loader	Description
key	String		
name	String		
description	String		

built_in_version
String

version
String

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

Array

Manifest Example

An example of a 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' => '',
);
```

Installation Definitions

The following section outlines the indexes specified in the `$installdefs` array contained in the `./manifest.php` file. The `$installdefs` array indexes are used by the Module Loader to determine the actual installation steps that need to be taken to install the package.

`$installdef` Actions

Name	Type	Description
<code>action_file_map</code>	Array	
<code>action_remap</code>	Array	
<code>action_view_map</code>	Array	
<code>administration</code>	Array	
<code>appscheduledefs</code>	Array	

beans
Array

connectors
Array

copy
Array

•

•

custom_fields
Array

console
Array

dashlets
Array

dependencies
Array

entrypoints
Array

extensions
Array

file_access
Array

hookdefs
Array

id
String

image_dir
String

jsgroups
Array

language
Array

layoutdefs
Array

layoutfields
Array

logic_hooks
Array

pre_execute

Array

post_execute
Array

pre_uninstall
Array

post_uninstall

Array

relationships

Array

scheduledefs

Array

sidecar

Array

tinymce

Array

user_page

Array

utils

Array

vardefs

Array

wireless_modules

Array

wireless_subpanels

Array

Note: Anything printed to the screen in the `pre_execute`, `post_execute`, `pre_uninstall`, or `post_uninstall` scripts will be displayed when clicking on the

Display Log link.

Last Modified: 09/29/2017 09:49am

Module Loader Restrictions

Overview

SugarCRM's hosting objective is to maintain the integrity of the standard Sugar functionality when we upgrade a customer instance and limit any negative impact our upgrade has on the customer's modifications. All instances hosted on Sugar's cloud service have package scanner enabled by default. This setting is not configurable and all packages must pass the package scan for installation on Sugar's cloud environment.

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 Sugar's cloud environment.

To enable Package Scan and its associated scans, add this setting to `config_override.php`:

```
$sugar_config['moduleInstaller']['packageScan'] = true;
```

There are two categories of access control in the Package Scan:

- File Scan
- Module Loader Actions

Enabling File Scan

By enabling Package Scan, File Scan will be performed on all files in the package uploaded through Module Loader. File Scan will be performed when a Sugar administrator attempts to install the package. Please note that these settings can only be managed on an on-site deployment. These settings are not permitted to be modified when hosted on Sugar's cloud service.

File Scan performs three checks:

- File extensions must be in the approved list of valid extension types.
- Files must not contain any suspicious classes.
- Files must not contain any suspicious function calls.

Please refer to the next three sections which outline the default requirements for the File Scan checks.

Valid Extension Types

File extensions must be in the approved list of valid extension types. The following extension types are valid by default:

- css
- gif
- hbs
- htm
- html
- jpg
- js
- md5
- pdf
- php

-
- png
 - tpl
 - txt
 - xml

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()`) 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
 - chown
 - 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
 - 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`
 - `timestamponcehandler`
 - `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()) are prohibited by default.
 - SugarLogger::setLevel
 - SugarAutoLoader::put
 - SugarAutoLoader::unlink

Disabling File Scan

Note: Disabling File Scan is prohibited for instances on Sugar's cloud service.

To disable File Scan, add the following configuration setting to `config_override.php`:

```
$sugar_config['moduleInstaller']['disableFileScan'] = false;
```

Extending the List of Valid Extension Types

Note: Modifying the valid extensions list is prohibited for instances on Sugar's cloud service.

To add more file extensions to the approved list of valid extension types, add the file extensions to the `validExt` array. The example below adds a `.log` file extension

and .htaccess to the valid extension type list in config_override.php:

```
$sugar_config['moduleInstaller']['validExt'] = array(
    'log',
    'htaccess'
);
```

Blacklisting Additional Function Calls

Note: Blacklist modifications are prohibited for instances on Sugar's cloud service.

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'
);
```

Overriding Blacklisted Function Calls

Note: Blacklist modifications are prohibited for instances on Sugar's cloud service.

To override the blacklist and allow a specific function to be included in packages, add the function call to the `blackListExempt` array. The example below removes the restriction for the `file_put_contents()` function, allowing it to be included in the package:

```
$sugar_config['moduleInstaller']['blackListExempt'] = array(
    'file_put_contents'
);
```

Disabling Restricted Copy

To ensure upgrade-safe customizations, System Administrators must restrict the copy action to prevent modifying the existing Sugar source code files. New files may be added anywhere (to allow new modules to be added), but core Sugar source code files must not be overwritten. This is enabled by default when you enable Package Scan.

Note: Disabling Restricted Copy is prohibited for instances on Sugar's cloud service.

To disable Restricted Copy, use this configuration setting:

```
$sugar_config['moduleInstaller']['disableRestrictedCopy'] = true;
```

Module Loader Actions

Module loader actions, defined in `./ModuleInstall/ModuleScanner.php`, are identifiers that map to the installation definitions used in the `$installdefs` of a manifest.

Action	\$installdef Actions	Description
<code>install_administration</code>	<code>administration</code>	Installs an administration section into the Admin page
<code>install_connectors</code>	<code>connectors</code>	Installs Sugar Cloud Connectors
<code>install_copy</code>	<code>copy</code>	Installs files or directories
<code>install_dashlets</code>	<code>dashlets</code>	Installs dashlets into the Sugar application
<code>install_images</code>	<code>image_dir</code>	Install images into the custom directory
<code>install_languages</code>	<code>language</code>	Installs language files
<code>install_layoutdefs</code>	<code>layoutdefs</code>	Installs layouts
<code>install_layoutfields</code>	<code>layoutfields</code>	Installs custom fields
<code>install_logichooks</code>	<code>logic_hooks</code>	Installs logic hooks
<code>install_relationships</code>	<code>relationships</code>	Installs relationships
<code>install_userpage</code>	<code>user_page</code>	Installs a section to the User record page
<code>install_vardefs</code>	<code>vardefs</code>	Installs vardefs
<code>post_execute</code>	<code>post_execute</code>	Called after a package is installed
<code>pre_execute</code>	<code>pre_execute</code>	Called before a package is installed

Disabling Module Loader Actions

Certain Module Loader actions may be considered less desirable than others by a System Administrator. A System Administrator may want to allow some Module Loader actions, but disable specific actions that could impact the upgrade-safe integrity of the Sugar instance.

Note: Disabling Module Loader actions is prohibited for instances on Sugar's cloud service.

By default, all Module Loader actions are allowed. Enabling Package Scan does not affect the Module Loader actions. To disable specific Module Loader actions, add the action to the `disableActions` array. The example below restricts the `pre_execute` and `post_execute` actions:

```
$sugar_config['moduleInstaller']['disableActions'] = array(
    'pre_execute',
    'post_execute'
);
```

Disabling Upgrade Wizard

If you are hosting Sugar and wish to lock down the upgrade wizard, you can set `disable_uw_upload` to `'true'` in the `config_override`. This is intended for hosting providers to prevent unwanted upgrades.

```
$sugar_config['disable_uw_upload'] = true;
```

Last Modified: 03/16/2018 03:06pm

Module Loader Restriction Alternatives

Overview

This article provides workarounds for commonly used functions that are blacklisted by Sugar for Sugar's cloud 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 use on Sugar's cloud service due to its ability to call other restricted functions.

Restricted use:

```
/**
 * Returns whether the input integer is odd
 * @param $var
 * @return int
 */
function odd($var) {
    return($var & 1);
}
```

```
}
```

```
$myArray = array("a"=>1, "b"=>2, "c"=>3, "d"=>4, "e"=>5);  
$filteredArray = array_filter($myArray, "odd");
```

An alternative to using `array_filter` is to use a `foreach` loop.

```
$filteredArray = array();  
$myArray = array("a"=>1, "b"=>2, "c"=>3, "d"=>4, "e"=>5);  
  
foreach ($myArray as $key => $value) {  
    // check whether the input integer is odd  
    if($value & 1) {  
        $filteredArray[$key] = $value;  
    }  
}
```

copy()

The `copy` method is sometimes used by developers when duplicating files in the uploads directory.

Restricted use:

```
$result = copy($oldFile, $newFile);
```

An alternative to using `copy` is the `duplicate_file` method found in the `UploadFile` class.

```
require_once('include/upload_file.php');  
  
$uploadFile = new UploadFile();  
$result = $uploadFile->duplicate_file($oldFileId, $newFileId);
```

file_exists()

The `file_exists` method is used by developers to determine if a file exists.

Restricted use:

```
if(file_exists($file_path)) {  
    require_once($file);  
}
```

```
}
```

An alternative to using `file_exists` is the `fileExists` method found in the `SugarAutoLoader` class.

```
$file = 'include/utils.php';  
if (SugarAutoloader::fileExists($file)) {  
    require_once($file);  
}
```

`file_get_contents()`

The `file_get_contents` method is used to retrieve the contents of a file.

Restricted use:

```
$file_contents = file_get_contents('file.txt');
```

An alternative to using `file_get_contents` and `sugar_file_get_contents` is the `get_file_contents` method found in the `UploadFile` class.

```
require_once('include/upload_file.php');  
  
$uploadFile = new UploadFile();  
  
//get the file location  
$uploadFile->temp_file_location =  
UploadFile::get_upload_path($file_id);  
$file_contents = $uploadFile->get_file_contents();
```

`fwrite()`

The `fwrite` method is a function used to write content to a file. As there isn't currently a direct alternative for this function, you may find one of the following a good solution to what you are trying to achieve.

Adding/Removing Logic Hooks

When working with logic hooks, it is very common for a developer to need to modify `./custom/modules/<module>/logic_hooks.php`. When creating module loadable packages, developers will sometimes use `fwrite` to modify this file upon

installation to include their additional hooks. As of Sugar 6.3, Logic Hook Extensions were implemented to allow a developer to append custom hooks. If you would prefer to edit the `logic_hooks.php` file, you will need to use the `check_logic_hook_file` method as described below:

```
//Adding a logic hook
require_once("include/utils.php");

$my_hook = Array(
    999,
    'Example Logic Hook',
    'custom/modules/<module>/my_hook.php',
    'my_hook_class',
    'my_hook_function'
);

check_logic_hook_file("Accounts", "before_save", $my_hook);
```

Removing a logic hook can be done by using `remove_logic_hook`:

```
//Removing a logic hook
require_once("include/utils.php");

$my_hook = Array(
    999,
    'Example Logic Hook',
    'custom/modules/<module>/my_hook.php',
    'my_hook_class',
    'my_hook_function'
);

remove_logic_hook("Accounts", "before_save", $my_hook);
```

getimagesize()

The `getimagesize` method is used to retrieve information about an image file.

Restricted use:

```
$img_size = getimagesize($path);
```

If you are looking to verify an image is `.png` or `.jpeg`, you can use the `verify_uploaded_image` method:

```
require_once('include/utils.php');
```

```
if (verify_uploaded_image($path)) {  
    //logic  
}
```

If you are looking to get the mime type of an image, you can use the `get_file_mime_type` method:

```
$mime_type = get_file_mime_type($path);
```

Last Modified: 03/16/2018 02:44pm

Sugar Exchange Package Guidelines

Overview

The Sugar platform is open and flexible. Developers can customize any feature or integrate any system with Sugar using a Sugar Module Loadable Package. This flexibility requires guidelines so that SugarExchange customers can rest assured that all package offerings adhere to consistent quality standards and will not interfere with existing customizations or prevent software upgrades.

This guide outlines the minimum standards we expect from any Sugar Developer writing code for the Sugar platform. From the development of Sugar packages to security, user interface, encapsulation, and performance considerations, SugarCRM takes the safety and integrity of the Sugar application and its community very seriously. Compliance with these guidelines is a requirement for any package installed into Sugar's cloud service. Failure to follow these guidelines is grounds for having your package removed from Sugar's cloud service and de-listed from SugarExchange.

SugarCRM reserves the right to change these guidelines at any time. Please send any questions or feedback on these guidelines to developers@sugarcrm.com.

Best Practices

These best practice guidelines have been curated based on years of collective experience working with Sugar Packages that get used by Sugar customers every day.

Use a consistent coding style

For all PHP code, the style standard that SugarCRM uses is PSR-2. For JavaScript code, we use the applicable PSR-2 conventions as well. For JavaScript code, we emphasize readability which means we make use of utilities like Underscore.js and avoid nested callbacks. All functions, methods, classes in our code are required to have PHPDoc and JSDoc. While not all the code in the Sugar code base currently complies with these standards, we do enforce it as the standard on new code. Using a consistent code style increases the readability and maintainability of application code for those that come after you.

Invest in unit testing during development

For all JavaScript and PHP code, we strongly recommend the creation of unit tests. For PHP, we use PHPUnit and have developed a framework (to be shared) for testing Sugar server-side code using this framework. For JavaScript code, we use Jasmine and have also developed a framework (to be shared) there to bootstrap Sidecar metadata so that Jasmine tests can run without dependency on a Sugar server.

We recommend running unit tests frequently during the development process. We suggest developers run tests before each commit and as part of an automated continuous integration process. Running tests often means you catch failures sooner which makes them easier and cheaper to fix.

Use Sugar REST APIs

The easiest way to integrate with a Sugar instance is not to install anything into Sugar at all. For Sugar 7, we have a full client REST API that is used to drive our web interface, our mobile client, and our plug-ins. If you can do it from our user interface, you can use our REST API separately to do it as well. If the REST API doesn't do everything you need it to do, it is very easily extensible. You can easily write code that adds your custom API endpoints in a minimally invasive way.

Use Module Builder when possible

Many integrations can be accomplished with the help of Module Builder and the Sugar REST API. Module Builder allows you to quickly design new modules for Sugar that match concepts that you want to add to a Sugar instance. These custom modules can then be installed and populated via the REST API, making it a

powerful integration mechanism that doesn't require writing a line of Sugar code. Using custom modules will prevent conflicts with other customizations, as well. For more information, please refer to the Module Builder documentation.

Use Extension framework and Dashlets for Sugar code customizations

The Extension framework is the way to add server-side changes to a Sugar instance in a loosely coupled way. Dashlets are the best way to add new, custom user interface components to a Sugar instance that gives users maximum flexibility in how they use your app. These mechanisms also avoid conflicts with other customizations since Extensions are additive and do not replace core files.

Security Guidelines

Protecting and controlling access to CRM data is of paramount importance. It is important that Sugar Packages are good stewards of CRM data and access control.

Use TLS/SSL for web services calls

Just as we don't recommend running Sugar in production without TLS/SSL, all web-services calls that are initiated from a Sugar Package should also use SSL. The concern is the exposure of user credentials, OAuth/session tokens, or sensitive CRM data via plaintext transmission that would otherwise be handled securely.

Do not hardcode sensitive information

You also should not hardcode any credentials, API keys, tokens, etc, within a Sugar Package. Sugar Packages and Sugar application code is never encrypted so there is always a risk that an attacker could discover these things and abuse them. Usernames and passwords, OAuth tokens, and similar credentials for accessing 3rd party systems should be stored in the database (for example, on the config table). The Sugar platform also provides encryption utilities that allow information to be stored in an encrypted form. These settings could then be changeable by the Administrator via the Administration panel or some other end-user input.

User Interface Guidelines

Sugar packages can be used to add new user interface components or front-end

customizations to Sugar. It is important to consider the impact that these changes have on the user experience and the look and feel of the Sugar application.

Use the Sugar 7 Styleguide

In Sugar 7, we have doubled down on our emphasis on creating the best possible user experience. While other applications may use different usage patterns than the ones used in Sugar 7, it is important to think about how new functionality or integrations that you build in Sugar 7 fits within the overall Sugar 7 user experience. Users do not tolerate an inconsistent experience within a given tool - it makes it harder to learn to use, which lowers adoption of Sugar as well as your application.

We want to make it easier for you to build applications within Sugar that use a consistent theme and user experience patterns, so we have included a styleguide for the Sugar application. You can find a link to the styleguide from the Sugar Administration page. There you can find all the information you need to leverage Sugar 7's styling including our CSS framework.

Don't use incompatible UI frameworks or external CSS libraries

Mixing outside user interface frameworks into the Sugar application via a Sugar Package can easily break many different parts of the application. Please only include as much CSS as you need and make sure that it is properly formed so that it doesn't affect other parts of the Sugar application. At the very least, using different frameworks or themes within your application will create a disjointed experience for the end user. While your package may be installed into Sugar, the user experience will not be seamless.

Encapsulation Guidelines

An important aspect of the quality of a Sugar Package is how well encapsulated and loosely coupled it is. A well encapsulated and loosely coupled package will encounter the fewest issues during upgrades, fewer breakages due to core code changes or due to interactions with other installed packages, as well minimizing bugs or problems that end users encounter.

Use Extensions framework and Custom Modules as much as possible

A well-encapsulated package prefers the use of Custom Modules over customizing

and repurposing existing Sugar Modules. A loosely coupled package uses Extensions framework for customizing and connecting with the core Sugar application. Only override the behavior of core Sugar application files as a last resort.

Avoid customizations to core Sugar application files

Developers are strongly discouraged from overriding core Sugar Modules or Sugar framework code. In many cases, a cleaner approach for accomplishing the same goal exists. For example, using a logic hook to extend the behavior of a SugarBean instead of overriding the SugarBean itself. Every core customization is a barrier to successful upgrades that creates recurring development costs over time. This is exacerbated in heavily customized Sugar instances as other customizations may exist on these files. Anytime there is a conflict then manual intervention by a Sugar Developer is required which is not only inconvenient but costly for everyone involved. If you do make core Sugar customizations then keeping track of such changes is very important to get in front of potential conflicts with other packages and upgrades.

Use Package Scanner to ensure your Sugar Package is ready for Sugar Cloud

Sugar Packages should be designed with Sugar's cloud service in mind, as many Sugar customers choose this hosting option over hosting on-site or through a Sugar partner. Code that gets loaded into Sugar's cloud service must pass the Package Scanner utility and must not adversely impact the SugarCRM infrastructure. This stipulation is outlined in the Sugar Cloud Policy Guide. Package Scanner can be enabled on any Sugar instance via the Sugar Administration panel which allows this to be tested easily. You should also educate yourself in some alternatives to 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 APM, which can monitor browser and server performance simultaneously, as well as XHprof for Sugar PHP profiling. Slow query logging can also be enabled via the Sugar Administration

panel under System Settings. For more information, refer to the System documentation. The Sugar Engineering team typically uses Chrome DevTools for JavaScript profiling.

Index for large frequently used queries

The most common performance bottleneck for Sugar is the database. Using slow query logging makes it possible to identify bottlenecks and correct them. One way to address query bottlenecks is to extend vardefs to add indices during package install to improve the performance of those queries.

Add scheduled jobs to prune large database tables

If your Package adds database tables that can tend to grow very large over time, it is a best practice to include scheduled jobs in your package that can be used to prune the size of this database over time. For Sugar, these background tasks can be created and managed using the Job Queue framework. At the very least, you'll want to create a Custom Job that Sugar Administrators can then run as needed. This will allow the package to remain in tip-top shape for your users over time. Especially if they are running on Sugar's cloud service because direct access to the underlying SQL database to do manual tuning and cleanup is not permitted.

Ensure your application does not block user interface during long running processes

If your application prevents the user from getting feedback or using the interface while it is running a long process, this will impact the perceived performance of the application. Users typically expect some UI feedback within half a second. If you have transactions that will take longer than that, it is best to use the Job Queue framework to defer them for later or move them into a scheduled Custom Job.

Last Modified: 03/19/2018 02:52pm

Examples

Module Loader Examples.

Creating an Installable Package for a Logic Hook

Overview

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

The installer package is available here for your reference.

Package Example

The following files are relative to the base of the .zip archive.

./manifest.php

```
<?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',
            ),
        ),
    );

```

?>

Note: For more details on the \$manifest or \$installdef can be found in the Introduction to the Manifest .

./custom/modules/Accounts/accounts_save.php

<?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: 11/27/2017 12:24pm

Creating an Installable Package That Copies Files

Overview

This is an overview of how to create a module loadable package that will copy files into your instance of Sugar. This is most helpful when your instance is hosted in Sugar's cloud environment or by a third party. For more details on the `$manifest` or `$installdefs` 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: 03/15/2018 05:51pm

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
list
'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
list
'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: 10/14/2016 04:12pm

Removing Files with an Installable Package

Overview

This is an overview of how to create a module loadable package that will remove files from the custom directory that may have been left over from a previous package installation or legacy versions of your code customization.

Note: For Sugar cloud customers, this method will not work as removing files is prohibited by Package Scanner. To have files removed you can submit a case to Sugar Support with a list of files to be removed or request a package approval and provide your module loadable package.

This example will install a custom file, and use a pre-execute script to remove files from a previous customization. Typically, uninstalling a package would remove the files of the customization, however in some scenarios, it is desired to install a new package on top of a previous version, and removing files with the new package might be necessary. The full package is downloadable here for your reference.

For more details on the `$manifest` or `$installdefs` options, you can visit the [Introduction to the Manifest documentation](#).

Manifest Example

```
<basepath>/manifest.php
```

```
<?php
```

```
    $manifest = array(
        'acceptable_sugar_flavors' =>
array('CE', 'PRO', 'CORP', 'ENT', 'ULT'),
        'acceptable_sugar_versions' => array(
            'exact_matches' => array(),
            'regex_matches' => array('[6-7]\\.[0-9]\\.[0-9]$'),
        ),
        'author' => 'SugarCRM',
        'description' => 'Installs a custom class file',
        'icon' => '',
        'is_uninstallable' => true,
        'name' => 'Example Removing File Installer',
        'published_date' => '2017-12-06 12:00:00',
        'type' => 'module',
        'version' => 'v1.2'
    );
```

```
    $installdefs = array(
        'id' => 'package_1341607504',
        'copy' => array(
            'from' =>
'<basepath>/Files/custom/src/CustomClass.php',
```

```
        'to' => 'custom/src/CustomClass.php',
    ),
),
'pre_execute' => array(
    '<base_path>/removeLegacyFiles.php',
),
'remove_files' => array(
    'custom/include/CustomClass.php'
)
);
```

```
?>
```

<basepath>/removeLegacyFiles.php

```
<?php
if (isset($this->installdefs['remove_files'])) {
    foreach($this->installdefs['remove_files'] as $relpath){
        if (SugarAutoloader::fileExists($relpath)) {
            SugarAutoloader::unlink($relpath);
        }
    }
}
?>
```

The full package is downloadable [here](#) for your reference.

Last Modified: 03/15/2018 07:57pm

Module Builder

Overview

The Module Builder tool allows programmers to create custom modules without writing code and to create relationships between new and existing CRM modules. To illustrate how to use Module Builder, this article will show how to create and deploy a custom module.

For this example, a custom module to track media inquiries will be created to track public relations requests within a CRM system. This use case is an often requested enhancement for CRM systems that 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_media request/

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: 10/14/2016 04:12pm

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.

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 with 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 re-deploy a package in a production environment

Re-deploying 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

Only create one module per 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.

Wait to create relationships until after deploying the module

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 after it is deployed

Once the package is deployed, delete it from Module Builder so that it will not accidentally be redeployed. The only exception to this rule is in a development environment as you may want to continue working and testing until you are ready to move the module to your production environment. If you ever want to uninstall the module at a later date, you can do so under Admin > Module Loader.

Last Modified: 11/27/2017 12:24pm

Entry Points

Overview

Entry points, defined in `./include/MVC/Controller/entry_point_registry.php`, were used to ensure that proper security and authentication steps are applied consistently across the entire application. While they are still used in some areas of Sugar, any developers using custom entry points should adjust their customizations to use the latest REST v10 API endpoints instead.

Accessing Entry Points

Available custom entry points can be accessed using a URL pattern as follows:

```
http://{sugar url}/index.php?entryPoint={entry point name}
```

The entry point name will be the specified index in the `$entry_point_registry` array. Access to this entry point outside of sugar will be dependent on the `auth` parameter defined in the `$entry_point_registry` array as well. More information on

creating custom entry points can be found [here](#) and [here](#).

Last Modified: 10/14/2016 04:12pm

Creating Custom Entry Points

Overview

As of 6.3.x, entry points can be created using the extension framework. The entry point extension directory, located at `./custom/Extension/application/Ext/EntryPointRegistry/`, is compiled into `./custom/application/Ext/EntryPointRegistry/entry_point_registry.ext.php` after a Quick Repair and Rebuild. Additional information can be found in the extensions `EntryPointRegistry` section.

Custom Entry Points

Prior to 6.3.x, an entry point could be added by creating the file `./custom/include/MVC/Controller/entry_point_registry.php`. This method of creating entry points is still compatible but is not recommended from a best practices standpoint as duplicating `./include/MVC/Controller/entry_point_registry.php` to `./custom/include/MVC/Controller/entry_point_registry.php` will prevent any upgrader updates to `./include/MVC/Controller/entry_point_registry.php` from being reflected in the system. Entry point registries contain two properties:

- `file` - The path to the entry point.
- `auth` - A Boolean value that determines whether or not the user must be authenticated in order to access the entry point.

```
$entry_point_registry['customEntryPoint'] = array(  
    'file' => 'path/to/customEntryPoint.php',  
    'auth' => true  
);
```

Example

The first step is to create the actual entry point. This is where all of the logic for your entry point will be located. This file can be located anywhere you choose. For

my example, I will create:

```
./custom/customEntryPoint.php
```

```
<?php

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

    echo "Hello World!";
```

Next, we will need to create our extension in the application extensions. This will be located at:

```
./custom/Extension/application/Ext/EntryPointRegistry/customEntryPoint.php
```

```
<?php

    $entry_point_registry['customEntryPoint'] = array(
        'file' => 'custom/customEntryPoint.php',
        'auth' => true
    );
```

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

```
./custom/application/Ext/EntryPointRegistry/entry_point_registry.ext.php
```

containing your registry entry. We are now able to access our entry point by navigating to:

```
http://{sugar url}/index.php?entryPoint=customEntryPoint
```

Last Modified: 11/28/2017 05:48pm

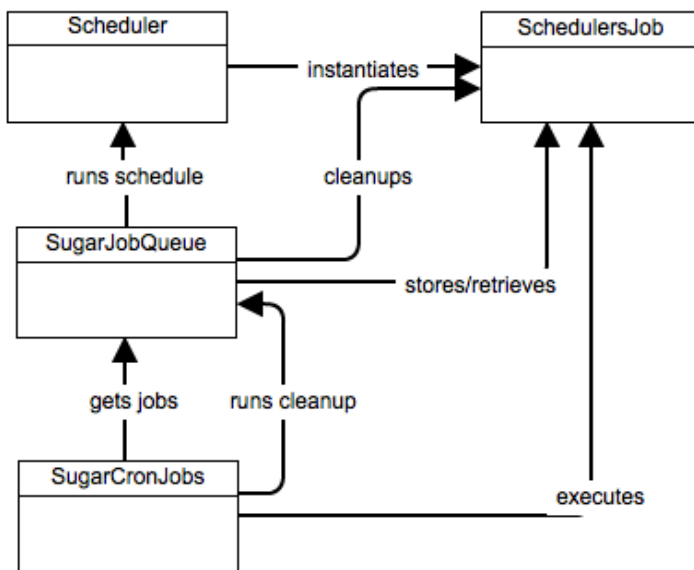
Job Queue

Overview

The Job Queue executes automated tasks in Sugar through a scheduler, which integrates with external UNIX systems and Windows systems to run jobs that are scheduled through those systems. Jobs are the individual runs of the specified function from a scheduler.

The Job Queue is composed of the following parts:

- SugarJobQueue : Implements the queue functionality. The queue contains the various jobs.
- SchedulersJob : A single instance of a job. This represents a single executable task and is held in the SugarJobQueue.
- Scheduler : This is a periodically occurring job.
- SugarCronJobs : The cron process that uses SugarJobQueue to run jobs. It runs periodically and does not support parallel execution.



Stages

Schedule Stage

On the scheduling stage (checkPendingJobs in Scheduler class), the queue checks if any schedules are qualified to run at this time and do not have job instance already in the queue. If such schedules exist, a job instance will immediately be created for each.

Execution Stage

The SQL queue table is checked for any jobs in the "Pending" status. These will be set to "Running" and then executed in accordance to its target and settings.

Cleanup Stage

The queue is checked for jobs that are in the "Running" state longer than the defined timeout. Such jobs are considered "Failed" jobs (they may be re-queued if their definition includes re-queuing on failure).

Last Modified: 10/14/2016 04:12pm

Schedulers

Overview

Sugar provides a Scheduler service that can execute predefined functions asynchronously on a periodic basis. The Scheduler integrates with external UNIX systems and Windows systems to run jobs that are scheduled through those systems. The typical configuration is to have a UNIX cron job or a Windows scheduled job execute the Sugar Scheduler service every couple of minutes. The Scheduler service checks the list of Schedulers defined in the Scheduler Admin screen and executes any that are currently due.

A series of schedulers are defined by default with every Sugar installation. For detailed information on these stock schedulers, please refer to the Schedulers documentation.

Schedulers

| Job Name | | Search | Clear |
|--------------------------|---|----------------------------------|-------|
| <input type="checkbox"/> | Delete | | |
| Scheduler | Interval | | |
| <input type="checkbox"/> | Create Future TimePeriods | On the hour; 23:00 | |
| <input type="checkbox"/> | Clean Jobs Queue | On the hour; 05:00 | |
| <input type="checkbox"/> | Run Email Reminder Notifications | As often as possible. | |
| <input type="checkbox"/> | Process Workflow Tasks | As often as possible. | |
| <input type="checkbox"/> | Run Report Generation Scheduled Tasks | On the hour; 06:00 | |
| <input type="checkbox"/> | Prune tracker tables | On the hour; 02:00; 1st | |
| <input type="checkbox"/> | Check Inbound Mailboxes | As often as possible. | |
| <input type="checkbox"/> | Run Nightly Process Bounced Campaign Emails | On the hour; From 02:00 to 06:00 | |
| <input type="checkbox"/> | Run Nightly Mass Email Campaigns | On the hour; From 02:00 to 06:00 | |
| <input type="checkbox"/> | Prune Database on 1st of Month | On the hour; 04:00; 1st | |
| <input type="checkbox"/> | Update tracker_sessions table | As often as possible. | |
| <input type="checkbox"/> | Delete | | |

Config Settings

- `cron.max_cron_jobs` - Determines the maximum number of jobs executed per cron run
- `cron.max_cron_runtime` - Determines the maximum amount of time a job can run before forcing a failure
- `cron.min_cron_interval` - Specified the minimum amount of time between cron runs

Considerations

- Schedulers execute the next time the cron runs after the interval of time has passed, which may not be at the exact time specified on the scheduler.

-
- If you see the message "Job runs too frequently, throttled to protect the system" in the Sugar log, the cron is running too frequently.
 - If you would prefer the cron to run more frequently, set the `cron.min_cron_interval` setting to 0 and disable throttling completely.

Last Modified: 10/14/2016 04:12pm

Creating Custom Schedulers

Overview

In addition to the default schedulers that are packaged with Sugar, developers can create custom scheduler jobs.

Defining the Job Label

The first step to create a custom scheduler is creating a label extension file. This will add the display text for the scheduler job when creating a new scheduler in Admin > Scheduler. The file path of our file will be in the format of `./custom/Extension/modules/Schedulers/Ext/Language/<language key>.<name>.php`. For our example, name the file `en_us.custom_job.php`.

`./custom/Extension/modules/Schedulers/Ext/Language/en_us.custom_job.php`

```
<?php
```

```
    $mod_strings['LBL_CUSTOM_JOB'] = 'Custom Job';
```

Defining the Job Function

Next, define the custom job's function using the extension framework. The file path of the file will be in the format of `./custom/Extension/modules/Schedulers/Ext/ScheduledTasks/<function_name>.php`. For this example, name the file `custom_job.php`. Prior to 6.3.x, job functions were added by creating the file `./custom/modules/Schedulers/_AddJobsHere.php`. This method of creating functions is still compatible but is not recommended from a best practices standpoint.

`./custom/Extension/modules/Schedulers/Ext/ScheduledTasks/custom_job.php`

```
<?php
```

```
    array_push($job_strings, 'custom_job');
    function custom_job()
    {
        //logic here
        //return true for completed
        return true;
    }
```

Using the New Job

Once the files are in place, navigate to Admin > Repair > Quick Repair and Rebuild. This will rebuild the extension directories with our additions. Next, navigate to Admin > Scheduler > Create Scheduler. In the Jobs dropdown, there will be a new custom job in the list.

Last Modified: 10/14/2016 04:12pm

Scheduler Jobs

Overview

Jobs are the individual runs of the specified function from a scheduler. This article will outline the various parts of a Scheduler Job.

Properties

- name : Name of the job
- scheduler_id : ID of the scheduler that created the job. This may be empty as schedulers are not required to create jobs
- execute_time : Time when job is ready to be executed
- status : Status of the job
- resolution : Notes whether or not the job was successful
- message : Contains messages produced by the job, including errors
- target : Function or URL called by the job
- data : Data attached to the job
- requeue : Boolean to determine whether the job will be replaced in the queue upon failure

-
- `retry_count` : Determines how many times the system will retry the job before failure
 - `failure_count` : The number `f` times the job has failed
 - `job_delay` : The delay (in seconds) between each job run
 - `assigned_user_id` : User ID of which the job will be executed as
 - `client` : The name of the client owning the job
 - `percent_complete` : For postponed jobs, this can be used to determine how much of the job has been completed

Creating a Job

To create job, you must first create an instance of `SchedulesJobs` class and use `submitJob` in `SugarJobQueue`. An example is shown below:

```
<?php

    $jq = new SugarJobQueue();
    $job = new SchedulersJob();
    $job->name = "My Job";
    $job->target = "function::myjob";
    $jobid = $jq->submitJob($job);

    echo "Created job $jobid!";
```

Job Targets

Job target contains two components, type and name, separated by "::". Type can be:

- `function` : Name or static method name (using `::` syntax). This function will be passed the job object as the first parameter and if data is not empty, it will be passed as the second parameter.
- `url` : External URL to call when running the job

Running the Job

The job is run via the `runJob()` function in `SchedulersJob`. This function will return a boolean success value according to the value returned by the target function. For URL targets, the HTTP error code being less than 400 will return success.

If the function updated the job status from 'running', the return value of a function is ignored. Currently, the postponing of a URL job is not supported.

Job status

Jobs can be in following statuses:

- `queued` : The job is waiting in the queue to be executed
- `running` : The job is currently being executed
- `done` : The job has executed and completed

Job Resolution

Job resolution is set when the job is finished and can be:

- `queued` : The job is still not finished
- `success` : The job has completed successfully
- `failure` : The job has failed
- `partial` : The job is partially done but needs to be completed

Job Logic Hooks

The scheduler jobs module has two additional logic hooks that can be used to monitor jobs. The additional hooks that can be used are shown below:

- `job_failure_retry` : Executed when a job fails but will be retried
- `job_failure` : Executed when the job fails for the final time

You can find more information on these hooks in the Job Queue Hooks section.

Last Modified: 10/14/2016 04:12pm

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: 10/14/2016 04:12pm

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 ./custom/modules/Schedulers/Ext/ScheduledTasks/scheduledtasks.ext.php containing our new function. We are now able to queue and run the scheduler job from a logic hook.

Last Modified: 10/14/2016 04:12pm

Access Control Lists

Overview

Access Control Lists (ACLs) are used to restrict access to the modules, data, and actions available to users in Sugar. The administrator defines ACLs in the Roles module via Admin > Role Management.

Checking Access

You can verify role access to content by using the `checkAccess()` method found in the `ACLController`.

Parameters

- `$category` : Corresponds to the module directory where the bean resides (e.g., Accounts)
- `$action` : The action you want to check against (e.g., Edit)
 - These actions correspond to actions in the `acl_actions` table as well as actions performed by the user within the application.
- `$is_owner` : Verifies if the owner of the record is attempting an action
 - Defaults to false
 - This parameter is relevant when the access level is set to `ALLOW_OWNER`
- `$type` : Defaults to "module"

Example

```
if (ACLController::checkAccess($category, $action, $is_owner, $type))
{
    //Code
}
```

See the Role Management documentation for a list of actions and their possible values.

Last Modified: 02/07/2018 04:24pm

Teams

Overview

Teams provide the ability to limit access at the record level, allowing sharing flexibility across functional groups. Developers can manipulate teams programmatically provided they understand Sugar's visibility framework.

For more information on teams, please refer to the Team Management documentation.

Note: Teams are exclusive to commercial editions of Sugar (i.e. Professional, Enterprise, Ultimate).

Database Tables

| Table | Description |
|-------------------|---|
| teams | Each record in this table represents a team in the system. |
| team_sets | Each record in this table represents a unique team combination. For example, each user's private team will have a corresponding team set entry in this table. A team set may also be comprised of one or more teams. |
| team_sets_teams | The team_sets_teams table maintains the relationships to determine which teams belong to a team set. Each table that previously used the team_id column to maintain team security now uses the team_set_id column's value to associate the record to team(s). |
| team_sets_modules | This table is used to manage team sets and keep track of which modules have records that are or were associated to a particular team set. |

Never modify these tables without going through the PHP object methods. Manipulating the team sets with direct SQL may cause undesired side effects in the system due to how the validations and security methods work.

Module Team Fields

In addition to the team tables, each module table also contains a `team_id` and `team_set_id` column. The `team_set_id` contains the id of a record in the `team_sets` table that contains the unique combination of teams associated with the record. The `team_id` will contain the id of the primary team designated for a record.

Team Sets (`team_set_id`)

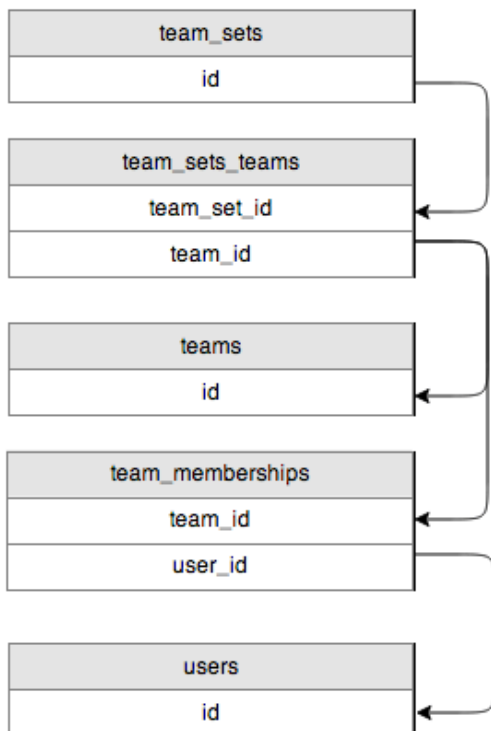
As mentioned above, Sugar implemented this feature not as a many-to-many relationship but as a one-to-many relationship. On each table that had a `team_id` field we added a `'team_set_id'` field. We have also added a `team_sets` table, which maintains the `team_set_id`, and a `team_sets_teams` table, which relates a team set to the teams. When performing a team based query we use the `'team_set_id'` field on the module table to join to `team_sets_teams.team_set_id` and then join all of the teams associated with that set. Given the list of teams from `team_memberships` we can then decide if the user has access to the record.

Primary Team (`team_id`)

The `team_id` is still being used, not only to support backwards compatibility with workflow and reports, but also to provide some additional features. When displaying a list, we use the team set to determine whether the user has access to the record. When displaying the data, we show the team from team id in the list. When the user performs a mouseover on that team, Sugar performs an Ajax call to display all of the teams associated with the record. This `team_id` field is designated as the Primary Team because it is the first team shown in the list, and for sales territory management purposes, can designate the team that actually owns the record and can report on it.

Team Security

The `team_sets_teams` table allows the system to check for permissions on multiple teams. The following diagram illustrates table relationships in SugarBean's `add_team_security_where_clause` method.



Using the `team_sets_teams` table the system will determine which teams are associated with the `team_set_id` and then look in the `team_memberships` table for users that belong to the team(s).

TeamSetLink

Typically, any relationship in a class is handled by the `data/Link2.php` class. As a part of dynamic teams, we introduced the ability to provide your own custom `Link` class to handle some of the functionality related to managing relationships. The `team_security` parent vardefs in the Sugar objects contain the following in the 'teams' field definition:

```

'link_class' => 'TeamSetLink',
'link_file' => 'modules/Teams/TeamSetLink.php',

```

The `link_class` entry defines the class name we are using, and the `link_file` tells us where that class file is located. This class extends the legacy `Link.php` and overrides some of the methods used to handle relationships such as 'add' and 'delete'.

Last Modified: 11/27/2017 12:17pm

Manipulating Teams Programmatically

Overview

How to manipulate team relationships.

Fetching Teams

To fetch teams related to a bean, you will need to retrieve an instance of a `TeamSet` object and use the `getTeams()` method to retrieve the teams using the `team_set_id`. An example is shown below:

```
//Create a TeamSet bean - no BeanFactory
require_once('modules/Teams/TeamSet.php');
$teamSetBean = new TeamSet();

//Retrieve the bean
$bean = BeanFactory::getBean($module, $record_id);

//Retrieve the teams from the team_set_id
$teams = $teamSetBean->getTeams($bean->team_set_id);
```

Adding Teams

To add a team to a bean, you will need to load the 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: 10/14/2016 04:12pm

Visibility Framework

Overview

The visibility framework provides the capability to alter the queries Sugar uses to retrieve records from the database. This framework can allow for additional restrictions or specific logic to meet business requirements of hiding or showing only specific records. Visibility classes only apply to certain aspects of Sugar

record retrieval, e.g. List Views, Dashlets, and Filter Lookups.

Custom Row Visibility

Custom visibility class files are stored under `./custom/data/visibility/`. The files in this directory can be enabled or disabled by modifying a module's visibility property located in `./custom/Extension/modules/<module>/Ext/Vardefs/`. Every enabled visibility class is merged into the module's definition, allowing multiple layers of logic to be added to a module.

Visibility Class

To add custom row visibility, you must create a visibility class that will extend the core `SugarVisibility` class `./data/SugarVisibility.php`. The visibility class has the ability to override the following methods:

| Name | Description |
|--------------------------------------|---|
| <code>addVisibilityQuery</code> | Add visibility clauses to a <code>SugarQuery</code> object. |
| <code>addVisibilityFrom</code> | [Deprecated] Add visibility clauses to the FROM part of the query. This method should still be implemented, as not all objects have been switched over to use <code>addVisibilityQuery()</code> method. |
| <code>addVisibilityFromQuery</code> | [Deprecated] Add visibility clauses to the FROM part of <code>SugarQuery</code> . This method should still be implemented, as not all objects have been switched over to use <code>addVisibilityQuery()</code> method. |
| <code>addVisibilityWhere</code> | [Deprecated] Add visibility clauses to the WHERE part of the query. This method should still be implemented, as not all objects have been switched over to use <code>addVisibilityQuery()</code> method. |
| <code>addVisibilityWhereQuery</code> | [Deprecated] Add visibility clauses to the WHERE part of <code>SugarQuery</code> . This method should still be implemented, as not all objects have been switched over to use <code>addVisibilityQuery()</code> method. |

The visibility class should also implement

Sugarcrm\Sugarcrm\Elasticsearch\Provider\Visibility\StrategyInterface so that the visibility rules are also applied to the global search. StrategyInterface has the following functions that should be implemented:

| Name | Description |
|--------------------------------|---|
| elasticBuildAnalysis | Build Elasticsearch analysis settings. This function can be empty if you do not need any special analyzers. |
| elasticBuildMapping | Build Elasticsearch mapping. This function should contain a mapping of fields that should be analyzed. |
| elasticProcessDocumentPreIndex | Process document before it's being indexed. This function should perform any actions to the document that needs to be completed before it is indexed. |
| elasticGetBeanIndexFields | Bean index fields to be indexed. This function should return an array of the fields that need to be indexed as part of your custom visibility. |
| elasticAddFilters | Add visibility filters. This function should apply the Elastic filters. |

Example

The following example creates a custom visibility filter that determines whether Opportunity records should be displayed based on their Sales Stage. Opportunity records with Sales Stage set to Closed Won or Closed Lost will not be displayed in the Opportunities module or global search for users with the Demo Visibility role.

`./custom/data/visibility/FilterOpportunities.php:`

```
<?php
use
Sugarcrm\Sugarcrm\Elasticsearch\Provider\Visibility\StrategyInterface
as ElasticStrategyInterface;
use Sugarcrm\Sugarcrm\Elasticsearch\Provider\Visibility\Visibility;
use Sugarcrm\Sugarcrm\Elasticsearch\Analysis\AnalysisBuilder;
use Sugarcrm\Sugarcrm\Elasticsearch\Mapping\Mapping;
use Sugarcrm\Sugarcrm\Elasticsearch\Adapter\Document;
```

```

/**
 *
 * Custom visibility class for Opportunities module:
 *
 * This demo allows to restrict access to opportunity records based on
the
 * user's role and configured filtered sales stages.
 *
 * The following $sugar_config parameters are available:
 *
 *
$sugar_config['custom']['visibility']['opportunities']['target_role']
 * This parameter takes a string containing the role name for which
 * the filtering should apply.
 *
 *
$sugar_config['custom']['visibility']['opportunities']['filter_sales_s
tages']
 * This parameters takes an array of filtered sales stages. If
current user is
 * member of the above configured role, then the opportunities with
the sale
 * stages as configured in this array will be inaccessible.
 *
 *
 * Example configuration given that 'Demo Visibility' role exists
(config_override.php):
 *
 *
$sugar_config['custom']['visibility']['opportunities']['target_role']
= 'Demo Visibility';
 *
$sugar_config['custom']['visibility']['opportunities']['filter_sales_s
tages'] = array('Closed Won', 'Closed Lost');
 *
 */
class FilterOpportunities extends SugarVisibility implements
ElasticStrategyInterface
{
    /**
     * The target role name
     * @var string
     */
    protected $targetRole = '';

```

```

/**
 * Filtered sales stages
 * @var array
 */
protected $filterSalesStages = array();

/**
 * {@inheritdoc}
 */
public function __construct(SugarBean $bean, $params = null)
{
    parent::__construct($bean, $params);

    $config = SugarConfig::getInstance();
    $this->targetRole = $config->get(
        'custom.visibility.opportunities.target_role',
        $this->targetRole
    );
    $this->filterSalesStages = $config->get(
        'custom.visibility.opportunities.filter_sales_stages',
        $this->filterSalesStages
    );
}

/**
 * {@inheritdoc}
 */
public function addVisibilityWhere(&$query)
{
    if (!$this->isSecurityApplicable()) {
        return $query;
    }
    $whereClause = sprintf(
        "%s.sales_stage NOT IN (%s)",
        $this->getTableAlias(),
        implode(',', array_map(array($this->bean->db, 'quoted'),
    $this->filterSalesStages))
    );

    if (!empty($query)) {
        $query .= " AND $whereClause ";
    } else {
        $query = $whereClause;
    }
}

```

```

        }
        return $query;
    }

/**
 * {@inheritdoc}
 */
public function addVisibilityWhereQuery(SugarQuery $sugarQuery,
$options = array())
{
    $where = null;
    $this->addVisibilityWhere($where, $options);
    if (!empty($where)) {
        $sugarQuery->where()->addRaw($where);
    }
    return $sugarQuery;
}

/**
 * Check if we can apply our security model
 * @param User $user
 * @return false|User
 */
protected function isSecurityApplicable(User $user = null)
{
    $user = $user ?: $this->getCurrentUser();

    if (!$user) {
        return false;
    }
    if (empty($this->targetRole) ||
empty($this->filterSalesStages)) {
        return false;
    }
    if (!is_string($this->targetRole) ||
!is_array($this->filterSalesStages)) {
        return false;
    }
    if (!$this->isUserMemberOfRole($this->targetRole, $user)) {
        return false;
    }
    if ($user->isAdminForModule("Opportunities")) {
        return false;
    }
}

```

```

        return $user;
    }

/**
 * Get current user
 * @return false|User
 */
protected function getCurrentUser()
{
    return empty($GLOBALS['current_user']) ? false :
$GLOBALS['current_user'];
}

/**
 * Check if given user has a given role assigned
 * @param string $targetRole Name of the role
 * @param User $user
 * @return boolean
 */
protected function isUserMemberOfRole($targetRole, User $user)
{
    $roles = ACLRole::getUserRoleNames($user->id);
    return in_array($targetRole, $roles) ? true : false;
}

/**
 * Get table alias
 * @return string
 */
protected function getTableAlias()
{
    $tableAlias = $this->getOption('table_alias');
    if (empty($tableAlias)) {
        $tableAlias = $this->bean->table_name;
    }
    return $tableAlias;
}

/**
 * {@inheritdoc}
 */
public function elasticBuildAnalysis(AnalysisBuilder
$analysisBuilder, Visibility $provider)
{

```

```

        // no special analyzers needed
    }

    /**
     * {@inheritdoc}
     */
    public function elasticBuildMapping(Mapping $mapping, Visibility
$provider)
    {
        $mapping->addNotAnalyzedField('visibility_sales_stage');
    }

    /**
     * {@inheritdoc}
     */
    public function elasticProcessDocumentPreIndex(Document $document,
\SugarBean $bean, Visibility $provider)
    {
        // populate the sales_stage into our explicit filter field
        $sales_stage = isset($bean->sales_stage) ? $bean->sales_stage
: '';
        $document->setDataField('visibility_sales_stage',
$sales_stage);
    }

    /**
     * {@inheritdoc}
     */
    public function elasticGetBeanIndexFields($module, Visibility
$provider)
    {
        // make sure to pull sales_stage regardless of search
        return array('sales_stage');
    }

    /**
     * {@inheritdoc}
     */
    public function elasticAddFilters(User $user,
\Elastica\Footer\Bool $filter, Visibility $provider)
    {
        if (!$this->isSecurityApplicable($user)) {
            return;
        }
    }

```

```

        // apply elastic filter to exclude the given sales stages
        $filter->addMustNot($provider->createFilter(
            'OpportunitySalesStages',
            array(
                'filter_sales_stages' => $this->filterSalesStages,
            )
        ));
    }
}

```

./custom/Extension/modules/Opportunities/Ext/Vardefs/opp_visibility.php

```

<?php

if (!defined('sugarEntry') || !sugarEntry) {
    die('Not A Valid Entry Point');
}
$dictionary['Opportunity']['visibility']['FilterOpportunities'] =
true;

```

./custom/src/Elasticsearch/Provider/Visibility/Filter/OpportunitySalesStagesFilter.php

```

<?php

namespace
Sugarcrm\Sugarcrm\custom\Elasticsearch\Provider\Visibility\Filter;

use
Sugarcrm\Sugarcrm\Elasticsearch\Provider\Visibility\Filter\FilterInterface;
use Sugarcrm\Sugarcrm\Elasticsearch\Provider\Visibility\Visibility;

/**
 *
 * Custom opportunity filter by sales_stage
 *
 * This logic can exist directly in the FilterOpportunities visibility
class.
 * However by abstracting the (custom) filters makes it possible to
re-use
 * them in other places as well.
 */
class OpportunitySalesStagesFilter implements FilterInterface

```

```

{
    /**
     * @var Visibility
     */
    protected $provider;

    /**
     * {@inheritdoc}
     */
    public function setProvider(Visibility $provider)
    {
        $this->provider = $provider;
    }

    /**
     * {@inheritdoc}
     */
    public function buildFilter(array $options = array())
    {
        return new \Elastica\Footer\Terms(
            'visibility_sales_stage',
            $options['filter_sales_stages']
        );
    }
}

```

After creating the above files, log in to your Sugar instance as an administrator and navigate to Administration > Repair > Quick Repair and Rebuild.

Next, perform a full reindex by navigating to Administration > Search and selecting the "delete existing data" option.

Execute a cron to process all of the queued records into Elasticsearch by doing the following:

1. Open a command line client and navigate to your Sugar directory.
2. Execute `chmod +x bin/sugarcrm` to ensure `bin/sugarcrm` is executable.
3. Execute `php cron.php` to consume the queue.
4. Execute `bin/sugarcrm elastic:queue` to see if the queue has finished.

Repeat steps 3 and 4 until the queue has 0 records.

This example requires the Sales Stage field to be part of the Opportunities module.

Navigate to Administration > Opportunities and ensure the Opportunities radio button is selected.

Create a new role named Demo Visibility and assign a user to this role. Note: if you are using the sample data, do NOT use Jim as he has admin permission on the Opportunities module and will be able to view all records. We recommend using Max.

Configure your instance to filter opportunities for a given sales stages for this role by adding the following to ./config_override.php:

```
<?php

$sugar_config['custom']['visibility']['opportunities']['target_role']
= 'Demo Visibility';
$sugar_config['custom']['visibility']['opportunities']['filter_sales_s
tages'] = array('Closed Won', 'Closed Lost');
```

Log in as the user to whom you assigned the Demo Visibility role. Observe that opportunity records in the sales stages "Closed Won" and "Closed Lost" are no longer accessible.

You can download a module loadable package of this example here. You can browse the code on GitHub.

Last Modified: 01/18/2018 12:23pm

TinyMCE

The following sections detail the use and customization of TinyMCE.

Last Modified: 10/14/2016 04:12pm

Modifying the TinyMCE Editor

Overview

This article is a brief overview on how to modify the default settings for the TinyMCE editor by creating an override file. There are two different overrides that can be applied to buttons and default settings.

Overriding Buttons

The first override file is for the toolbar buttons. To do this, you must create `./custom/include/tinyButtonConfig.php`. Within this file, you will be able to override the button layout for the TinyMCE editor.

There are 3 different layouts you can change:

- `default` : Used when creating an email template
- `email_compose` : Used when composing an email from the full form under the Emails module
- `email_compose_light` : Used when doing a quick compose from a listview or subpanel

Example File

`./custom/include/tinyButtonConfig.php`

```
<?php

    //create email template
    $buttonConfigs['default'] = array(
        'buttonConfig' =>
"code,help,separator,bold,italic,underline,strikethrough,separator,justifyleft,justifycenter,justifyright,justifyfull,separator,forecolor,backgroundcolor,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,inserttime,separator,preview"
    );

    //Standard email compose
    $buttonConfigs['email_compose'] = array(
        'buttonConfig' =>
```

```

"code,help,separator,bold,italic,underline,striketrough,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,striketrough,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 on the TinyMCE website.

Last Modified: 10/14/2016 04:12pm

SugarPDF

Overview

As of version 6.7.x, Sugar includes a Smarty template engine called PDF Manager. This allows administrators to create and manage templates for generated PDF files for any deployed module, custom or standard. The PDF Manager is available to only administrative users and can be accessed through Admin > PDF Manager. For more information on the PDF Manager, please refer to the PDF Manager documentation in the Administration guide.

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 | Overridden to allow for the loading of custom fonts. The custom fonts direction is defined by the <code>K_PATH_CUSTOM_FONTS</code> var. By default, this value is set to <code>./custom/vendor/tcpdf/fonts/</code> |
| Cell | Overridden to apply the <code>prepare_string()</code> 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 |
|-----------------------------|---|
| <code>predisplay</code> | Preprocessing before the display method is called. Is intended to setup general PDF document properties like margin, footer, header, etc. |
| <code>display</code> | Performs the actual PDF content generation. This is where the logic to display output to the PDF should be placed. |
| <code>process</code> | Calls <code>predisplay</code> and <code>display</code> . |
| <code>writeCellTable</code> | Method to print a table using the cell print method of TCPDF |
| <code>writeHTMLTable</code> | Method to print a table using the <code>writeHTML</code> print method of TCPDF |

Custom PDF classes that extend Sugarpdf can be located in the following directories:

-
- ./include/Sugarpdf/sugarpdf/sugarpdf.<pdf view>.php
 - ./modules/<module>/sugarpdf/sugarpdf.<pdf view>.php
 - ./custom/modules/<module>/sugarpdf/sugarpdf.<pdf view>.php

PDF Manager classes that extend Sugarpdf are located in the following directories:

- ./include/Sugarpdf/sugarpdf/sugarpdf.smarty.php
- ./include/Sugarpdf/sugarpdf/sugarpdf.pdfmanager.php
- ./custom/include/Sugarpdf/sugarpdf/sugarpdf.pdfmanager.php

Each extended class will define a specific PDF view that is accessible with the following URL parameters:

- module=<module>
- action=sugarpdf
- sugarpdf=<pdf view>

Within each custom PDF class, the display method will need to be redefined. If you would like, it is also possible to override other methods like Header. The process method of this class will be called from ViewSugarpdf. When a PDF is being generated, the most relevant sugarpdf.<pdf action>.pdf class is determined by the SugarpdfFactory.

ViewSugarpdf

The ViewSugarpdf class, located in ./include/MVC/View/views/view.sugarpdf.php, is used to create the SugarViews that generate PDFs. These views can be found and/or created in one of the following directory paths:

- ./modules/<module>/views/view.sugarpdf.php
- ./custom/modules/<module>/views/view.sugarpdf.php

SugarViews can be called by navigating to a URL in the format of:

```
http://<url>/index.php?module=<module>&action=sugarpdf&sugarpdf=<pdf 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 ViewSugarpdf class uses the SugarpdfFactory class, located in `./include/Sugarpdf/SugarpdfFactory.php`, to find the most relevant `sugarpdf.<pdf action>.pdf` class which generates the PDF document for a given PDF view and module. If one is not found, then the core Sugarpdf class is used.

The SugarpdfFactory class loads the first class found for the specified PDF action as determined by the following order:

- `./custom/modules/<module>/sugarpdf/sugarpdf.<pdf view>.php`
- `./modules/<module>/sugarpdf/sugarpdf.<pdf view>.php`
- `./custom/include/Sugarpdf/sugarpdf/sugarpdf.<pdf view>.php`
- `./include/Sugarpdf/sugarpdf/sugarpdf.<pdf view>.php`
- `./include/Sugarpdf/sugarpdf.php`

SugarpdfHelper

The SugarpdfHelper, located in `./include/Sugarpdf/SugarpdfHelper.php`, is included by Sugarpdf. This is a utility file that contains many functions we use to generate PDFs.

Available functions are:

- `wrapTag`, `wrapTD`, `wrapTable`, etc. : These functions help to create an HTML code.
- `prepare_string` : This function prepare a string to be ready for the PDF printing.
- `format_number_sugarpdf` : This function is a copy of `format_number()` from currency with a fix for sugarpdf.

PdfManagerHelper

The PdfManagerHelper, located in `./modules/PdfManager/PdfManagerHelper.php`,

is primarily utilized by the pdfmanager Sugarpdf view. This class file contains methods useful for accessing PDF Manager templates. Some of the primary methods are:

- `getAvailableModules` : Returns an array of available modules for use with PdfManager.
- `getFields` : Takes an module name and returns a list of fields and links available for this module in PdfManager.
- `getPublishedTemplatesForModule` : Returns an array of the available templates for a specific module.

FontManager

The FontManagerclass, located in `./include/Sugarpdf/FontManager.php`, is a stand-alone class that manages all the fonts for TCPDF.

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: 10/14/2016 04:12pm

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: 10/14/2016 04:12pm

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: 10/14/2016 04:12pm

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: 10/14/2016 04:12pm

DateTime

Overview

The `SugarDateTime` class, located in, `./include/SugarDateTime.php`, extends PHP's built in `DateTime` class and can be used to perform common operations on date and time values.

Setting

With existing `SugarDateTime` objects, it is recommended to clone the object before modifying or performing operations on it.

```
$new_datetime = clone $datetime;
```

Another option is to instantiate a new `SugarDateTime` object.

```
// defaults to now
```

```
$due_date_time = new SugarDateTime();
```

```
$due_date_time->setDate($dueYear, $dueMonth, $dueDay);  
$due_date_time->setTime($dueHour, $dueMin);  
$due_date_time->setTimezone($dueTZ);
```

Note : When creating a new SugarDateTime object, the date and time will default to the current date and time in the timezone configured in PHP.

SugarDateTime Objects can also be modified by passing in common English textual date/time strings to manipulate the value.

```
$due_date = clone $date;  
$end_of_the_month->modify("last day of this month");  
$end_of_next_month->modify("last day of next month");  
$thirty_days_later->modify("+30 days");
```

Formatting

When formatting SugarDateTime objects into a string for display or logging purposes, there are a few options you can utilize through the formatDateTime() method.

Return the date/time formatted for the database:

```
$db_datetime_string = formatDateTime("datetime", "db");
```

Return the date/time formatted using a user's preference:

```
global $current_user;  
$user_datetime_string = formatDateTime("datetime", "user",  
$current_user);
```

Return the date/time formatted following ISO-8601 standards:

```
$iso_datetime_string = formatDateTime("datetime", "iso");
```

There are times when it is needed to return only certain parts of a date or time. The format() method accepts a string parameter to define what parts or information about the date/time should be returned.

```
// 28 Dec 2016  
echo $due_date_time->format("j M Y");
```

```
// 2016-12-28 19:01:09
echo $due_date_time->asDb();

// The date/time 2016-12-28T11:01:09-08:00 is set in the timezone of
America/Los_Angeles
echo "The date/time " . $due_date_time->format("c") . " is set in the
timezone of " . $due_date_time->format("e");

// The time when this date/time object was created is 11:01:09 AM
-0800
echo "The time when this date/time object was created is " .
$due_date_time->format("H:i:s A O");

// There are 31 days in the month of December
echo "There are " . $due_date_time->format("t") . " days in the month
of " . $due_date_time->format("F");

// There are 366 days in the year of 2016
echo "There are " . $due_date_time->__get("days_in_year") . " days in
the year of " . $due_date_time->format("Y");

// Between Wed, 28 Dec 2016 11:01:09 -0800 and the end of the year,
there are 4 days.
echo "Between " . $due_date_time . " and the end of the year, there
are " . ($due_date_time->__get("days_in_year") -
$due_date_time->format("z")) . " days.";
```

For more information on the available options, please refer to
<http://php.net/manual/en/function.date.php>

TimeDate Class

The TimeDate class, located in `./include/TimeDate.php`, utilizes the SugarDateTime class to provide an extended toolset of methods for date/time usage.

Setting

With existing TimeDate objects, it is recommended to clone the object before modifying or performing operations on it.

```
$new_timedate = clone $timedate;
```

Another option is to instantiate a new TimeDate object.

```
// current time in UTC
$timedate = new TimeDate();
$now_utcTZ = $timedate->getNow();

// current time in user's timezone
$timedate = new TimeDate($current_user);
$now_userTZ = $timedate->getNow(true);
```

Note : When creating a new TimeDate object, the date and time will default to the current date and time in the UTC timezone unless a user object is passed in.

Formatting

TimeDate objects can return the Date/Time in either a string format or in a SugarDateTime object. The TimeDate object has many formatting options; some of the most common ones are :

| | |
|-------------------------|---|
| getNow | Get the current date/time as a SugarDateTime object |
| fromUser | Get SugarDateTime object from user's date/time string |
| asUser | Format DateTime object as user's date/time string |
| fromDb | Get SugarDateTime object from a DB date/time string |
| fromString | Create a SugarDateTime object from a string |
| get_db_date_time_format | Gets the current database date and time format |
| to_display_date_time | Format a string as user's display date/time |

getNow

returns SugarDateTime object

Parameters

| Name | Data Type | Default | Description |
|-----------------------|-----------|---------|---|
| <code>\$userTz</code> | bool | false | Whether to use the user's timezone or not. False will use UTC |

Returns a `SugarDateTime` object set to the current date/time. If there is a user object associate to the `TimeDate` object, passing true to the `$userTz` parameter will return the object in user's timezone. If no user is associated or false is passed, the timezone returned will be in UTC.

Example

```
$timedate = new TimeDate($current_user);  
  
// returns current date/time in UTC  
$now_utcTZ = $timedate->getNow();  
  
// returns current date/time in the user's timezone  
$now_userTZ = $timedate->getNow(true);
```

fromUser

returns `SugarDateTime` object

Parameters

| Name | Data Type | Default | Description |
|---------------------|-----------|---------|---|
| <code>\$date</code> | string | n/a | Date/Time string to convert to an object |
| <code>\$user</code> | User | null | User to utilize formatting and timezone info from |

Returns a `SugarDateTime` object converted from the passed in string. If there is a user object passed in as a parameter will return the object in user's timezone. If no user is passed in, the timezone returned will be in UTC.

Example

```
$date = "2016-12-28 13:09";  
$timedate = new TimeDate();  
$datetime = $timedate->fromUser($date, $current_user);
```

asUser

returns string

Parameters

| Name | Data Type | Default | Description |
|--------|-----------|---------|---|
| \$date | string | n/a | Date/Time string to convert to an object |
| \$user | User | null | User to utilize formatting and timezone info from |

Returns a string of the date and time formatted based on the user's profile settings. If there is a user object passed in as a parameter it will return the object in user's timezone. If no user is passed in, the timezone returned will be in UTC.

Example

```
$datetime = new datetime("2016-12-28 13:09");  
$timedate = new TimeDate();  
$formattedDate = $timedate->asUser($datetime, $current_user);
```

fromDb

returns SugarDateTime

Parameters

| Name | Data Type | Default | Description |
|--------|-----------|---------|---|
| \$date | string | n/a | Date/Time string in database format to convert to an object |

Returns a SugarDateTime object converted from the passed in database formatted string.

Note : If the string does not match the database format exactly, this function will return boolean false.

Example

```
// Y-m-d H:i:s
$db_datetime = "2016-12-28 13:09:01";
$time_date = new TimeDate();
$date_time = $time_date->fromDb($db_datetime);

// returns bool(false)
$wrong_datetime = "2016-12-28 13:09";
$time_date = new TimeDate();
$date_time = $time_date->fromDb($time_date);
```

fromString

returns SugarDateTime

Parameters

| Name | Data Type | Default | Description |
|--------|-----------|---------|--|
| \$date | string | n/a | Date/Time string to convert to an object |
| \$user | User | null | User to utilize timezone info from |

Returns a SugarDateTime object converted from the passed in database formatted string. If there is a user object passed in as a parameter it will return the object in user's timezone. If no user is passed in, the timezone returned will be in UTC.

Example

```
$datetime_str = "December 28th 2016 13:09";
$time_date = new TimeDate();
$date_time = $time_date->fromString($datetime_str);
```

get_db_date_time_format

returns string

Parameters

N/a

Returns a string of the current database date and time format settings.

Note : For just the date format use get_db_date_format() and for just the time format use get_db_time_format().

Example

```
$timedate = new TimeDate();  
  
// Y-m-d H:i:s  
$db_format = $timedate->get_db_date_time_format();
```

to_display_date_time

returns string

Parameters

| Name | Data Type | Default | Description |
|--------------|-----------|---------|--|
| \$date | string | n/a | Date/Time string in database format |
| \$meridiem | boolean | true | Deprecated -- Remains for backwards compatibility only |
| \$convert_tz | boolean | true | Convert to user's timezone |
| \$user | User | null | User to utilize formatting and timezone info from |

Returns a string of the date and time formatted based on the user's profile

settings. If no user is passed in, the current user's default will be used. If `$convert_tz` is true the string returned will be in the passed in user's timezone. If no user is passed in, the timezone returned will be in UTC.

Note : If the string does not match the database format exactly, this function will return boolean false.

Example

```
$datetime_str = "2016-12-28 13:09:01";

// 12-28-2016 07:09
$timedate = new TimeDate();
$datetime = $timedate->to_display_date_time($datetime_str);

// 2016-12-28 13:09
$timedate = new TimeDate();
$datetime = $timedate->to_display_date_time($datetime_str, true,
false, $diff_user);
```

Parsing

In addition to formatting, TimeDate objects can also return Date/Time values based on special parameters. The TimeDate object has many date parsing options; some of the most common ones are :

| | |
|-----------------------------|---|
| <code>parseDateRange</code> | Gets the start and end date/time from a range keyword |
|-----------------------------|---|

`parseDateRange`

returns array

Parameters

| Name | Data Type | Default | Description |
|----------------------|-----------|---------|--|
| <code>\$range</code> | string | n/a | String from a predetermined list of available ranges |
| <code>\$user</code> | User | null | User to utilize |

| | | | |
|---------------------|---------|------|--------------------------------------|
| | | | formatting and
timezone info from |
| \$adjustForTimezone | boolean | true | Convert to user's
timezone |

Returns an array of SugarDateTime objects containing the start and Date/Time values calculated based on the entered parameter. If no user is passed in, the current user's default will be used. If \$adjustForTimezone is true the string returned will be in the passed in user's timezone. If NULL is passed in as the user, the timezone returned will be in UTC. The available values for \$range are :

| Range String | Description |
|--------------|---|
| yesterday | Yesterday's start and end date/time |
| today | Today's start and end date/time |
| tomorrow | Tomorrows's start and end date/time |
| last_7_days | Start and end date/time of the last seven days |
| next_7_days | Start and end date/time of the next seven days |
| last_30_days | Start and end date/time of the last thirty days |
| next_30_days | Start and end date/time of the next thirty days |
| next_month | Start and end date/time of next month |
| last_month | Start and end date/time of last month |
| this_month | Start and end date/time of the current month |
| last_year | Start and end date/time of last year |
| this_year | Start and end date/time of the current year |
| next_year | Start and end date/time of next year |

Example

```
$timedate = new TimeDate($current_user);

// returns today's start and end date/time in UTC
$today_utcTZ = $timedate->parseDateRange("today", NULL, false);
```

```
// returns today's start and end date/time in the user's timezone
$today_userTZ = $timedate->parseDateRange("today", $current_user,
true);
```

Last Modified: 10/14/2016 04:12pm

Shortcuts

Overview

Shortcuts is a framework to add shortcut keys to the application. When shortcut keys are registered, they are registered to a particular shortcut session. Shortcut sessions group shortcut keys, and they can be activated, deactivated, saved, and restored. Global shortcut keys can be registered, but they are not tied to a particular shortcut session. They are rather applied everywhere in the application and can only be applied once.

The Shortcut framework is implemented on top of Mousetrap library.

Shortcut Sessions

In order to register a shortcut in Sugar, a shortcut session must first be created by adding `ShortcutSession` plugin to the top-level layout JavaScript controller.

```
plugins: ['ShortcutSession']
```

Then, the top-level layout needs to list which shortcuts are allowed in the shortcut session. Shortcuts can be listed in the top-level layout JavaScript controller :

```
./custom/clients/layout/my-layout/my-layout.js
```

```
shortcuts: [
  'Sidebar:Toggle',
  'Record:Save',
  'Record:Cancel',
  'Record:Action:More'
]
```

Shortcuts can also be listed in the metadata :

`./custom/clients/layout/my-layout/my-layout.php`

```
'shortcuts' => array(
    'Sidebar:Toggle',
    'Record:Save',
    'Record:Cancel',
    'Record:Action:More'
),
```

Register

Once a shortcut session is created, shortcut keys can be registered by adding the following in your JavaScript code :

```
app.shortcuts.register('<unique shortcut ID>', '<shortcut keys>',
<callback function>, <current component>, <call on focus?>);
```

Since shortcut keys should only be registered once in your component, they should be registered inside `initialize()` or someplace where re-rendering the component would not register more than once.

Shortcut IDs

Shortcut IDs should be unique across the application. They should be namespaced by convention, for example `Record:Next`, `Sidebar:Toggle`, `List:Edit`. If the namespace starts with `Global:`, it assumes that the shortcut is global.

Global shortcuts

Global shortcuts are shortcut keys that are applied once and are available everywhere in the application.

```
app.shortcuts.register(app.shortcuts.GLOBAL + 'Footer:Help', '?',
function() {}, this, false);
```

Shortcut Keys

There are only certain keys that are supported under the Shortcut framework. The following keyboard keys can be used :

-
- All alphanumeric characters and symbols
 - shift, ctrl, alt, option, meta, command
 - backspace, tab, enter, return, capslock, esc, escape, space, pageup, pagedown, end, home, ins, del
 - left, up, right, down

Additional Features

In addition to standard keys, the Shortcut framework also supports some additional features :

- Key combinations : 'ctrl+s'
- Multiple keys : ['ctrl+a', 'command+a']
- Key sequences : 'f a'

Input Focus

The fifth parameter in the register method specifies whether or not the shortcut key should be fired when the user is focused in an input field. It is false by default.

```
app.shortcuts.register('Record:Save', ['ctrl+s', 'command+s'],  
function() {}, this, true);
```

Limitations

Shortcut keys do not work on sidepanels, like dashboards and previews.

Shortcuts Help

Anytime a new shortcut is created, a help text should be provided in `./clients/base/layouts/shortcuts/shortcuts.php` with a shortcut ID and a language string.

```
'List:Favorite' => 'LBL_SHORTCUT_FAVORITE_RECORD',
```

Shortcuts help is displayed when Shortcuts button is clicked.

Advanced Features

Enable/disable dynamically

Shortcuts feature can be enabled and disabled dynamically via code by calling `app.shortcuts.enable()` and `app.shortcuts.disable()`.

Dynamically saving, creating new, and restoring sessions

A new shortcut session is created when a user visits a top-level layout with `ShortcutSession` plugin. A new session can be created dynamically:

```
app.shortcuts.createSession(<array of shortcut IDs>, <component>);
```

But before creating a new session, the current session should be saved first.

```
app.shortcuts.saveSession();
app.shortcuts.createSession(<array of shortcut IDs>, <component>);
```

Once a new session is created, shortcut keys can be registered to it. When the need for the session is done, the previous shortcut session can be restored.

```
app.shortcuts.restoreSession();
```

Example

The following example will be to add some custom shortcuts onto a custom layout. For more information on how to create a custom layout, please refer to the [Creating Layouts](#) documentation.

First, on our custom JavaScript controller for our layout, we must enable the `ShortcutSession` plugin as well as list the shortcuts we will be enabling :

```
./custom/clients/base/layouts/my-layout/my-layout.js
```

```
(( {
  plugins: ['ShortcutSession'],
  shortcuts: [
    'Global:MyLayout:MyCallback',
  ],
}))
```

Next, on the custom view being rendered on our layout, we will register the new shortcuts as well as define a callback method :

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

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

    // call myCallback method when command + a is pressed
    app.shortcuts.register(app.shortcuts.GLOBAL +
'MyLayout:MyCallback', 'command+a', this.myCallback, this, false);

    app.shortcuts.saveSession();
    app.shortcuts.createSession([
        'MyLayout:InlineCall'
    ], this);

    // call inline code when control + m or command + m is pressed
    app.shortcuts.register('MyLayout:InlineCall',
['ctrl+m','command+m'], function() {
    console.log("Ctrl or Command m has been pressed");
    }, this, false);
},
myCallback: function(){
    console.log("MyCallback called from Command a");
},
...
```

The last step will be to create a new label for our shortcut help text and register the help so it displays when "Shortcuts" is click in the footer of the page :

`./custom/Extension/application/Ext/Language/en_us.LBL_MYLAYOUT_SHORTCUT_HELP.php`

```
<?php
```

```
$app_strings['LBL_MYLAYOUT_MYCALLBACK_HELP'] = "Activates my Callback Method";
$app_strings['LBL_MYLAYOUT_MYINLINECALL_HELP'] = "Activates Inline Code";
```

`./custom/Extension/application/Ext/clients/base/layouts/shortcuts/shortcuts.php`

```
<?php
```

```
$viewdefs['base']['layout']['shortcuts']['help']['Global:MyLayout:MyCallback'] = 'LBL_MYLAYOUT_MYCALLBACK_HELP';  
$viewdefs['base']['layout']['shortcuts']['help']['MyLayout:InlineCall'] = 'LBL_MYLAYOUT_MYINLINECALL_HELP';
```

Navigate to Admin > Repair > Quick Repair and Rebuild. The system will then rebuild the extensions and the shortcuts should be available from the custom layout.

Last Modified: 04/25/2018 04:09am

Web Accessibility

Overview

Learn about the Sugar Accessibility Plugin for Sidecar.

Introduction

Making your application accessible -- per the standards defined by the W3C's WAI specifications -- is a hard thing to do and even harder to maintain. The goal of the Sugar Accessibility Plugin for Sidecar is to automatically apply rules to your rendered HTML, so that you don't have to be concerned with all of the intricacies of accessibility.

With respect to programmatically applying accessibility rules, you can generally assume that the rules fall into one of three categories:

1. Rules that are dependent on the context of the element's use and cannot be applied programmatically because the context is never clear.
2. Rules that can be applied programmatically, but only when the context is clear.
3. Rules that can always be applied programmatically.

We plan to continue to develop this plugin to address more and more accessibility concerns in an abstract way, with the intention of completely covering the latter two cases. In the meantime, the plugin handles two very specific cases: and (2)

One from the third category.

How It Works

The plugin listens to the render event for all `View.Component` objects that are created. Anytime a component is rendered, the plugin runs its own plugins (hereinafter referred to as "helpers") on the component. Each of these helpers is responsible for determining if any modifications are necessary in order for the component's HTML to meet accessibility standards and then carrying out those changes in the DOM. This behavior is done automatically as a part of the sidecar framework, so you do not need to do anything to start using it.

Sometimes, a component that you write will modify the HTML after it has been rendered. The plugin has no means of becoming aware of these changes to the HTML and you will have to tell it to look for rules to apply. One example is found in `View.Fields.Base.ActionmenuField`.

When the user selects all records in a list view, an alert is flashed indicating that all visible records are now selected. Within this alert, a link can be clicked to select all records, even those that are not currently visible. A new onclick event listener is registered for this link. And because this link is added to the DOM after the component is rendered, the author of the component must make sure that the new HTML meets accessibility requirements. Here is how that is done:

```
var $el = $('#select-all');
$el.on('click', function() {...});
app.accessiblity.run($el, 'click');
```

This will only run the click helper. If you want to run all helpers, then call `app.accessiblity.run($el)` (without the second parameter). But be aware that some helpers only support `View.Component` objects, while others support either `View.Component` objects or jQuery DOM elements. So running all helpers on a jQuery DOM element may fail, or at least fail to apply some accessibility rules as expected.

When the logger is configured for debug mode, messages are logged indicating which helpers are being run and which helpers could not be run when intended.

Plugins

A plugin (or helper) is a module that applies an accessibility rule to a `View.Component` (or jQuery DOM element). At runtime, these helpers can be found

in `app.accessibility.helpers` and implement a `run` method. The `run` method takes a `View.Component` (or jQuery DOM element) and then checks its HTML to determine if anything needs to be done in order to make the HTML compliant with accessibility standards related to the rule or task with which the helper is concerned. If any changes are necessary, the helper then modifies the HTML to comply.

Click

The click helper is responsible for making an element compliant with accessibility standards when click events are bound to said element. Since this helper only deals with `onclick` events, it only inspects elements within the component that include `onclick` event listeners.

If the tag name cannot be determined for an element being inspected, then there is no way of knowing whether or not the element is accessible. Thus, the element is assumed to be compliant.

Inherently focusable elements are those elements that require no intervention. These elements include:

- `button`
- `input`
- `select`
- `textarea`

Conditionally focusable elements are those elements that require intervention under certain circumstances. In the case of `<a>` and `<area>` tags, these elements are compliant as long as they contain an `href` attribute. These elements include:

- `a`
- `area`

All other elements are not inherently focusable and require a `tabindex` attribute of `-1` if a `tabindex` attribute does not already exist. This helper adds `tabindex="-1"` to any elements within the component that are not compliant.

When the logger is configured for debug mode, messages are logged...

1. In the event that no `onclick` events were found within the component. Thus,

no action is taken.

2. In the event that an element being inspected has no tag name.
3. To report the type of element being made compliant.
4. To report the type of element that is already found to be compliant.

Label

The label helper adds an aria-label to the form element found within the component. This helper only inspects elements that can be found via the component's fieldTag selector and is considered a "best effort" approach.

This helper will only work on View.Field components since it is extremely unlikely for the fieldTag selector for View.Layout or View.View components to match form elements.

A form element is considered to be compliant if the aria-label attribute is already present or if its tag is not one that requires a label. These elements include:

- button
- input
- select
- textarea

This helper adds `aria-label="{label}"` to the element that needs to be made compliant. `View.Field.label` is the label that is assigned to the attribute. The component must be a `View.Component`. Plain jQuery DOM elements are not sufficient since they do not include `alabel` property.

API

`SUGAR.accessibility.init()`

Initializes the accessibility module to execute all accessibility helpers on components as they are rendered. This is called by the application during bootstrapping.

`SUGAR.accessibility.run()`

Loads the accessibility helpers that are to be run and executes them on the

component.

Arguments

| Name | Type | Required | Description |
|-----------|-----------------------|----------|---|
| component | View.Component/jQuery | true | The element to test for accessibility compliance. |
| helper | String/Array | false | One or more names of specified helpers to run. All registered helpers will be run if undefined. |

Returns

Void

SUGAR.accessibility.whichHelpers()

Get the helpers registered on a specific element.

| Name | Type | Required | Description |
|--------|--------------|----------|--|
| helper | String/Array | true | One or more names of specified helpers to run. |

Returns

Array - The accessibility helpers that were requested. Filters out any named helpers that are not registered. All registered helpers are returned if no helper names are provided as a parameter.

SUGAR.accessibility.getElementTag()

Generates a human-readable string for identifying an element. For example, `a[name="link"][class="btn btn-link"][href="http://www.sugarcrm.com/"]`. Primarily used for logging purposes, this method is useful for debugging.

Arguments

| Name | Type | Required | Description |
|-------------------|--------|----------|--|
| <code>\$el</code> | jQuery | true | The element for which the tag should be generated. |

Returns

String - A string representing an element's tag, with all attributes. The element's selector, if one exists, is returned when a representation cannot be reasonably generated.

Last Modified: 10/14/2016 04:12pm

CLI

Overview

As of Sugar 7.7.1, Sugar includes a beta command line interface tool built using the Symfony Console Framework. Sugar's CLI is intended to be an administrator or developer level power tool to execute PHP code in the context of Sugar's code base. These commands are version specific and can be executed on a preinstalled Sugar instance or on an installed Sugar instance. Sugar's CLI is not intended to be used as a tool to interact remotely with an instance nor is it designed to interact with multiple instances.

Commands

Sugar Commands are an implementation of Console Commands. They extend the base console classes to execute Sugar specific actions. Each instance of Sugar is shipped with a list of predefined commands. The current list of commands is shown

below.

| Command | Description |
|-----------------------|---|
| help | Displays help for a command |
| list | Lists commands |
| elastic:explain | Execute global search explain queries for debugging purposes. As this will generate a lot of output in JSON format, it is recommended to redirect the output to a file for later processing |
| elastic:indices | Show Elasticsearch index statistics |
| elastic:queue | Show Elasticsearch queue statistics |
| elastic:queue_cleanup | Cleanup records from Elasticsearch queue |
| elastic:routing | Show Elasticsearch index routing |
| password:config | Show password hash configuration |
| password:reset | Reset user password for local user authentication |
| password:weak | Show users having weak or non-compliant password hashes |
| search:fields | Show search engine enabled fields |
| search:module | Enable/disable given module for search |
| search:reindex | Schedule SearchEngine reindex |
| search:status | Show search engine availability and enabled modules |

Note : For advanced users, a bash autocompletion script for CLI is located at `./src/Console/Resources/sugarcrm-bash-completion` for inclusion in `~/.bashrc`. More details on using the script can be found in the file's header comments.

Usage

The command executable, located at `./bin/sugarcrm`, is executed from the root of the Sugar instance. The command signature is shown below:

```
php bin/sugarcrm <command> [options] [arguments]
```

Help

The command console has built-in help documentation. If you pass in a command that isn't found, you will be shown the default help documentation.

SugarCRM Console version <version>

Usage:

command [options] [arguments]

Options:

| | |
|----------------------|--|
| -h, --help | Display this help message |
| -q, --quiet | Do not output any message |
| -V, --version | Display this application version |
| --ansi | Force ANSI output |
| --no-ansi | Disable ANSI output |
| -n, --no-interaction | Do not ask any interactive question |
| --profile | Display timing and memory usage information |
| -v vv vvv, --verbose | Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug |

Available commands:

| | |
|-----------------|---|
| help | Displays help for a command |
| list | Lists commands |
| elastic | |
| elastic:indices | Show Elasticsearch index statistics |
| elastic:queue | Show Elasticsearch queue statistics |
| elastic:routing | Show Elasticsearch index routing |
| search | |
| search:fields | Show search engine enabled fields |
| search:reindex | Schedule SearchEngine reindex |
| search:status | Show search engine availability and enabled modules |

Additional help documentation is also available for individual commands. Some examples of accessing the help are shown below.

Passing the word "help" before a command name:

```
php bin/sugarcrm help <command>
```

Passing the "-h" option:

```
php bin/sugarcrm <command> -h
```

An example of the list help documentation is shown below.

Usage:

```
list [options] [--] []
```

Arguments:

```
namespace          The namespace name
```

Options:

```
--xml              To output list as XML  
--raw              To output raw command list  
--format=FORMAT    The output format (txt, xml, json, or md)
```

```
[default: "txt"]
```

Help:

The list command lists all commands:

```
php bin/sugarcrm list
```

You can also display the commands for a specific namespace:

```
php bin/sugarcrm list test
```

You can also output the information in other formats by using the `--format` option:

```
php bin/sugarcrm list --format=xml
```

It's also possible to get raw list of commands (useful for embedding command runner):

```
php bin/sugarcrm list --raw
```

Custom Commands

Sugar allows developers the ability to create custom commands. These commands are registered using the Extension Framework. Each custom command will extend the stock Command class and implement a specific mode interface. The file system location of the command code can exist anywhere in the instance's file system including a separate composer loaded repository.

Best Practices

The following are some common best practices developers should keep in mind

when adding new commands :

- Do not extend from existing commands
- Do not duplicate an already existing command
- Pick the correct mode for the command
- Limit the logic in the command
- Do not put reusable components in the command itself

Each command requires specific sections of code in order to properly create the command. These requirements are listed in the sections below.

Command Namespace

It is recommended to name any custom commands using a proper "command namespace". These namespaces are different than PHP namespaces and reduce the chances of collisions occurring between vendors. An example of this would be to prefix any commands with a namespace format of vendor:category:command
E.g. MyDevShop:repairs:fixMyModule

Note : The CLI framework does not allow overriding of existing commands.

PHP Namespace

The header of the PHP command file will need to define the following namespace:

```
namespace Sugarcrm\Sugarcrm\custom\Console\Command;
```

In addition to the namespace, the following use commands are required for different operations :

| Name | Description | Example |
|---------|---|--|
| Command | Every command will extend the Command class | <pre>use
Symfony\Component\Console\Command\Command
;</pre> |
| Mode | Whether the command is an Instance or Standalone Mode command | <pre>use
Sugarcrm\Sugarcrm\Console\CommandRegistry\Mode\InstanceModeInterface;</pre> |

| | | |
|--------|-----------------------------------|---|
| | | use
Sugarcrm\\Sugarcrm\\Con
sole\\CommandRegistry\\
Mode\\StandaloneModeInt
erface; |
| Input | Accepts Input from the
command | use
Symfony\\Component\\Con
sole\\Input\\InputInterface
; |
| Output | Sends Output from the
command | use
Symfony\\Component\\Con
sole\\Output\\OutputInterf
ace; |

Mode

When creating Sugar commands, developers will need to specify the mode or set of modes for the command. These modes help prepare the appropriate resources for execution of the command.

| Mode | Description |
|-----------------|---|
| Instance Mode | Commands that require an installed Sugar instance. |
| Standalone Mode | Commands that do not require an installed Sugar instance. |

Note : Multiple modes can be selected.

Class Definition

The custom command class definition should extend the stock command class as well as implement a mode's interface.

```
// Instance Mode Class Definition
class <classname> extends Command implements InstanceModeInterface
{
    //protected methods
}
```

```
// Standalone Mode Class Definition
class <classname> extends Command implements StandaloneModeInterface
```

```

{
    //protected methods
}

// Implementing both Modes
class <classname> extends Command implements InstanceModeInterface,
StandaloneModeInterface
{
    //protected methods
}

```

Methods

Each command class implements two protected methods :

| Method | Description |
|---|--|
| configure() | Runs on the creation of the command. Useful to set the name, description, and help on the command. |
| execute(InputInterface \$input, OutputInterface \$output) | The code to run for executing the command. Accepts an input and output parameter. |

An example of implementing the protected methods is shown below:

```

protected function configure()
{
    $this->setName('sugardev:helloworld')
        ->setDescription('Hello World')
        ->setHelp('This command accepts no paramters and returns
"Hello World".')
    ;
}

protected function execute(InputInterface $input, OutputInterface
$output)
{
    $output->writeln("Hello world -> " .
$this->getApplication()->getMode());
}

```

Registering Command

After creating the custom command file, register it with the Extension Framework. This file will be located in `./custom/Extension/application/Ext/Console/`. An example of registering a command is below:

```
<?php

Sugarcrm\Sugarcrm\Console\CommandRegistry\CommandRegistry::getInstance
()
    ->addCommand(new Sugarcrm\Sugarcrm\custom\Console\Command\<Command
Class Name>());
```

Next, navigate to Admin > Repair > Quick Repair and Rebuild. The custom command will now be available.

Example

The following sections demonstrate how to create a "Hello World" example. This command does not alter the Sugar system and will only output display text. First, create the command class under the appropriate namespace.

`./custom/src/Console/Command/HelloWorldCommand.php`

```
<?php

namespace Sugarcrm\Sugarcrm\custom\Console\Command;

use
Sugarcrm\Sugarcrm\Console\CommandRegistry\Mode\InstanceModeInterface;
use Symfony\Component\Console\Command\Command;
use Symfony\Component\Console\Input\InputInterface;
use Symfony\Component\Console\Output\OutputInterface;

/**
 *
 * Hello World Example
 *
 */
class HelloWorldCommand extends Command implements
InstanceModeInterface
{

    protected function configure()
    {
```

```

        $this
            ->setName('sugardev:helloworld')
            ->setDescription('Hello World')
            ->setHelp('This command accepts no paramters and returns
"Hello World".')
        ;
    }

    protected function execute(InputInterface $input, OutputInterface
$output)
    {
        $output->writeln("Hello world -> " .
$this->getApplication()->getMode());
    }
}

```

Next, register the new command:

```
./custom/Extension/application/Ext/Console/RegisterHelloWorldCommand.php
```

```

<?php

// Register HelloWorldCommand
Sugarcrm\Sugarcrm\Console\CommandRegistry\CommandRegistry::getInstance
()
    ->addCommand(new
Sugarcrm\Sugarcrm\custom\Console\Command\HelloWorldCommand());

```

Navigate to Admin > Repair > Quick Repair and Rebuild. The new command will be executable from the root of the Sugar instance. It can be executed by running the following command:

```
php bin/sugarcrm sugardev:helloworld
```

Result:

```
Hello world -> InstanceMode
```

Help text can be displayed by executing the following command:

```
php bin/sugarcrm help sugardev:helloworld
```

Result:

Usage:

sugardev:helloworld

Options:

| | |
|----------------------|--|
| -h, --help | Display this help message |
| -q, --quiet | Do not output any message |
| -V, --version | Display this application version |
| --ansi | Force ANSI output |
| --no-ansi | Disable ANSI output |
| -n, --no-interaction | Do not ask any interactive question |
| --profile | Display timing and memory usage information |
| -v vv vvv, --verbose | Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug |

Help:

This command accepts no parameters and returns "Hello World".

Download the module loadable example package [here](#).

Last Modified: 11/27/2017 12:17pm

Performance Tuning

The following sections detail ways to enhance the performance of your Sugar instance.

Last Modified: 10/14/2016 04:12pm

Sugar Performance

Overview

As your company uses Sugar over time, the size of your database will naturally grow and, without the proper maintenance, performance will inevitably begin to degrade. The purpose of this article is to review some of the most common recommendations for increasing the performance in Sugar to help increase the system's efficiency for your users.

Note: This guide is intended for on-site installations. Customers hosted on Sugar's cloud service should file a support case for any performance issues.

General Settings in Sugar

The following recommendations can be modified in the Sugar admin interface

- Do Not Set Listview and Subpanel Items Per Page to Excessive Settings. Under Admin > System Settings, there are two settings 'Listview items per page' and 'Subpanel items per page'. The defaults for these settings are 20 and 10 respectively. When increasing these values, it should be expected that general system wide performance will be impacted. We generally recommend 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

General Settings

Disable client IP verification : Eliminates the system checking to see if the user is accessing Sugar from the IP address of their last page load.

```
$sugar_config['verify_client_ip'] = false;
```

BWC Modules

For modules running in Backward Compatibility mode, the following settings can be used to speed up performance:

Drop the absolute totals from listviews : Eliminates performing expensive count queries on the database when populating listviews and subpanels.

```
$sugar_config['disable_count_query'] = true;
```

Disable automatic searches on listviews : Forces a user to perform a search when they access a listview rather than loading the results from their last search.

```
$sugar_config['save_query'] = 'populate_only';
```

Hide all subpanels : Increases performance by collapsing all subpanels when accessing a detailview every time and not querying for data until a user explicitly expands a subpanel

```
$sugar_config['hide_subpanels'] = true;
```

Hide subpanels per session : Increases performance by collapsing all subpanels when accessing a detailview when the user logs in but any subpanels expanded during the user's session will remain expanded until the user logs out

```
$sugar_config['hide_subpanels_on_login'] = true;
```

General Environment Checks

Depending on your environment and version of Sugar, there may be additional changes your system administrator can make to improve performance.

- If your instance is running Sugar 6.3.x or lower, we highly recommend upgrading to 6.4.x or 6.5.x. Beginning in 6.4.x, we made a number of query improvement to address overall application performance. With 6.5.x, we drastically improved the UI to improve the speed of page loads.
- If your instance of Sugar is version 6.3.x or higher with a MySQL database, we highly recommend upgrading the MySQL 5.5. MySQL 5.5 offers performance improvements in a number of areas over 5.1 such as subselects in queries.
- If you are using MySQL as your database, we strongly recommend using

InnoDB. InnoDB is tested to be better performing than MyISAM and should be the default configuration for using MySQL with Sugar.

- If you are running PHP 5.2.x, we strongly recommend upgrading to a supported version of PHP 5.3. Our current list of supported PHP versions can be found on our Supported Platforms page.
- If you are using a single server setup (web server and database on the same server), we have the following recommendations.
 - The server should have a minimum of 8 GB of RAM and roughly follow the 60/40 rule (60% for database / 40% for web server). On a 8 GB server, this would mean 4.8 GB for database and 3.2 GB for web server.
 - Make sure the following parameters are set for MySQL (assumption is that the engine is InnoDB)
 - `innodb_buffer_pool_size` = 4294967296 (4 GB in size)
 - `innodb_log_buffer_size` = anywhere from 10485760 (10 MB - Minimal writes) to 104857600 (100 MB - Lots of writes)
- If you are using IE 8 or lower, we recommend upgrading to IE 9 or using Google Chrome. Earlier versions of IE 8 exhibit poor performance with our application and we recommend updating your browser to IE 9 or changing to Chrome.

PHP Caching

Whether your instance of Sugar is deployed on a Linux or Windows server, you should utilize opcode caching to ensure optimal performance. For Linux servers, APC is the recommended opcode cache for PHP with the following guidelines and settings:

- Use the latest stable version.
- `apc.shm_size` should be close to your program size. For Sugar, that's at least 150 MB (default for `apc.shm_size` 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: 03/15/2018 07:47pm

PHP Profiling

Overview

As of the 6.6.2 release, Sugar introduced the ability to profile via XHProf, which is an easy-to-use, hierarchical profiler for PHP. This allows developers to better manage and understand customer performance issues introduced by their customizations. This tool enables quick and accurate identification of the sources of performance sinks within the code by generating profiling logs. Profiling gives you the ability to see the call stack for the entire page load with timing details around function and method calls as well as statistics on call frequency.

Assuming XHProf is installed and enabled in your PHP configuration (which you can learn how to do in the PHP Manual), you can enable profiling in Sugar by adding the following parameters to the `./config_override.php` file:

```
$sugar_config['xhprof_config']['enable'] = true;
$sugar_config['xhprof_config']['log_to'] = '{instance server
path}/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: 10/14/2016 04:12pm

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. Sugar cloud 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):

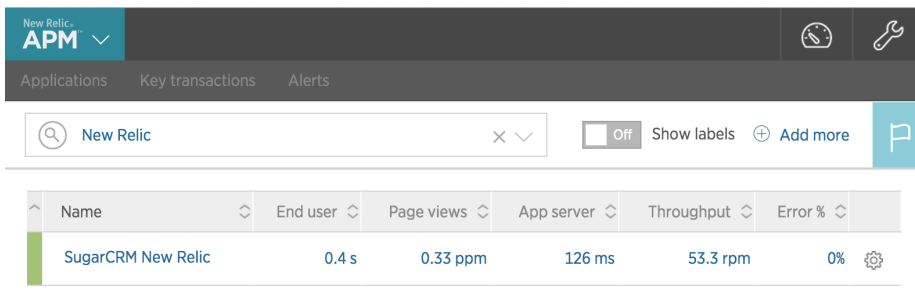
```
$sugar_config['metrics_enabled'] = 1;
$sugar_config['metric_providers']['SugarMetric_Provider_Newrelic'] =
'include/SugarMetric/Provider/Newrelic.php';
$sugar_config['metric_settings']['SugarMetric_Provider_Newrelic']['app
licationname'] = "SugarCRM New Relic";
```

- 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

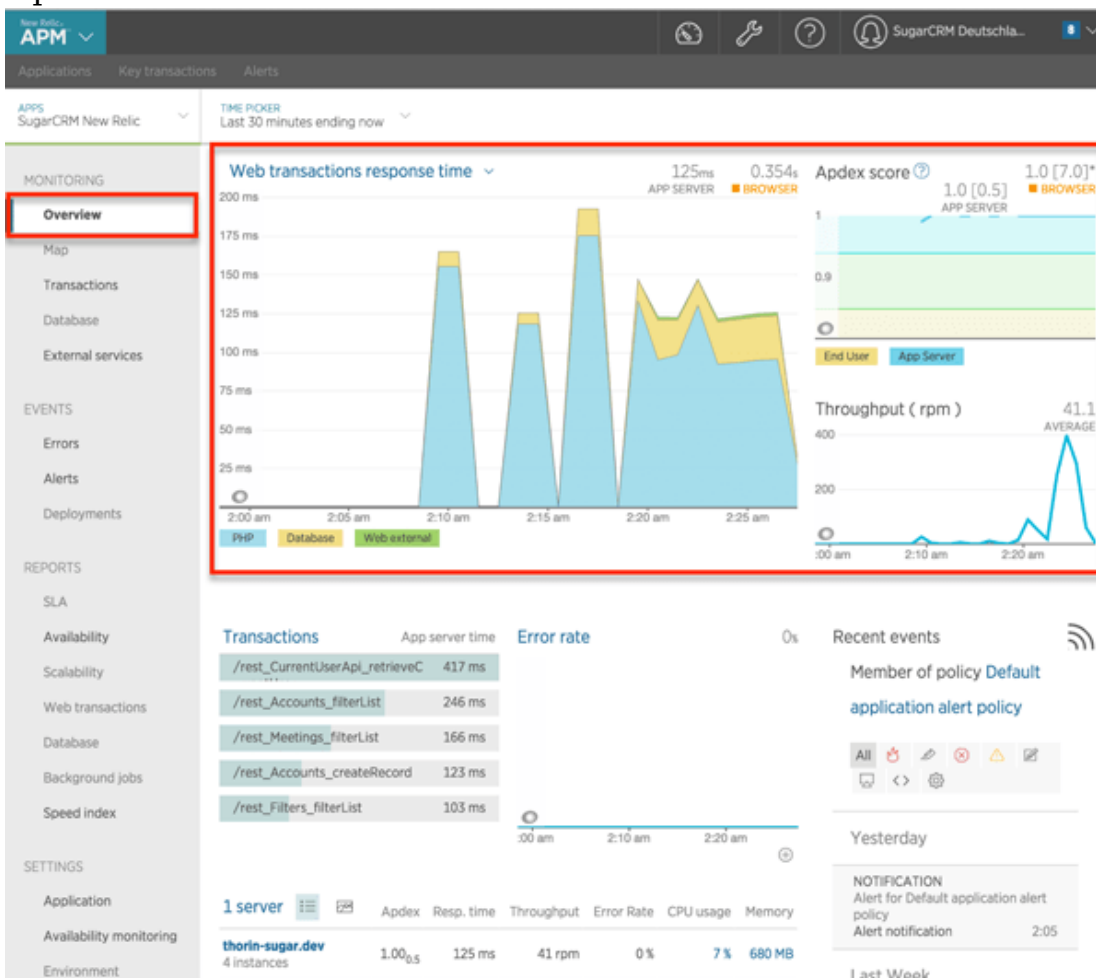
New Relic integrated with Sugar enables you to view the exact functions that cause unusual performance behavior in your instance, such as unexpected triggering of logic hooks, database queries that should be optimized, or customizations that are responding slower than expected.

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.



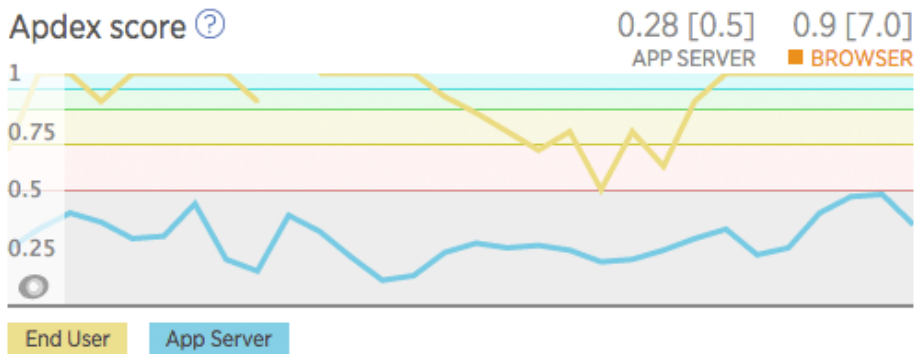
Overview Dashboard

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

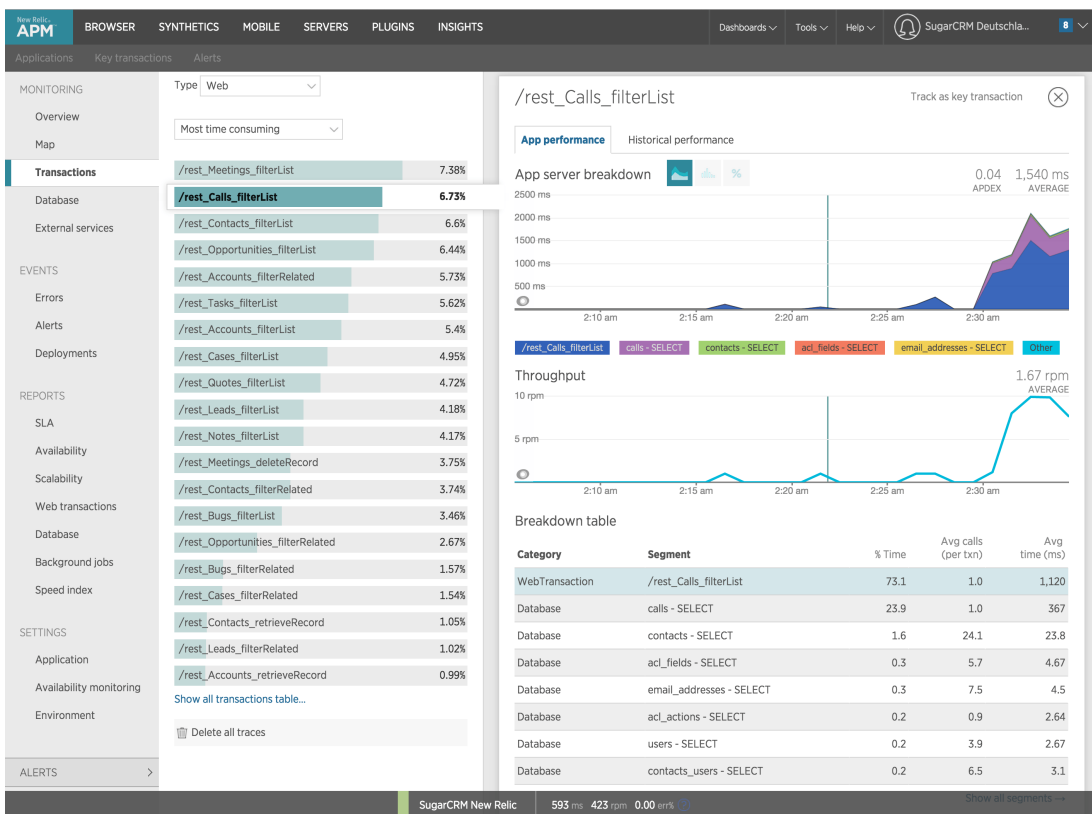
the users, a response time between 0.5 seconds and 2.0 seconds is tolerable, and any value higher than 2.0 seconds becomes frustrating.



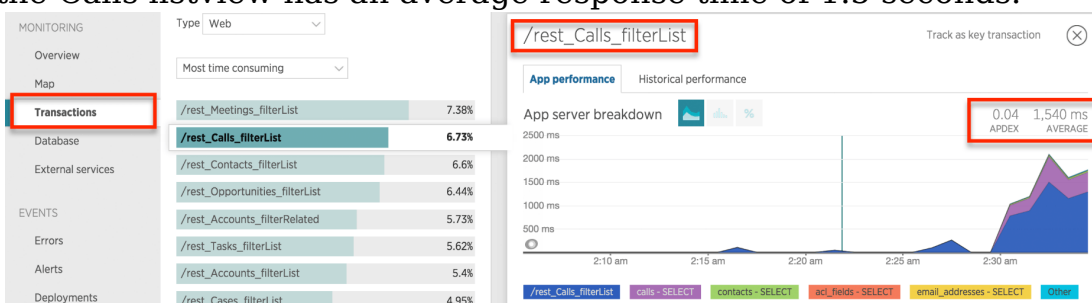
The default Apdex T-value for New Relic is 0.5 seconds but it can be configured to match your current environment and user expectations. For information on changing the Apdex T-value, please refer to the [Change Your Apdex Settings](#) article in the New Relic documentation.

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 Apex or slower. Click on the transaction trace to learn what is having the negative impact on the performance for this activity.

Transaction traces

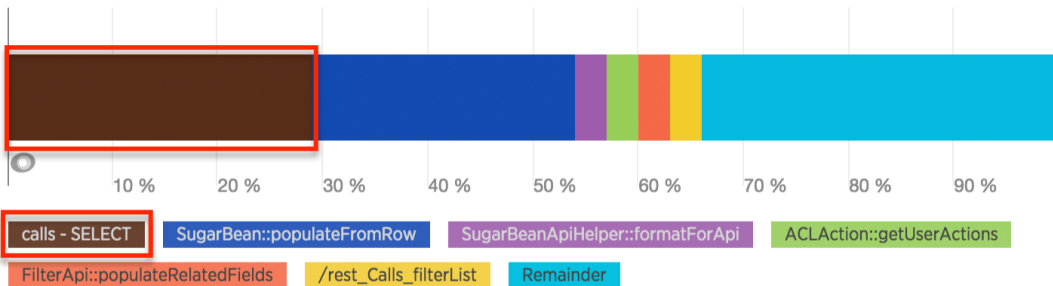
Sample performance details

| | App server | Browser |
|-----------------------|------------|---------|
| 02:32 – 4 minutes ago | 3,491 ms | |
| 02:27 – 9 minutes ago | 98 ms | |

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.

2015-02-19 02:32:54 3,490ms **1,920ms (54.9%)** 29ms (0.83%)
 TRACE TIME RESP. TIME USER CPU BURN SYSTEM CPU BURN

Summary Trace details SQL statements



| Slowest components | Count | Duration | % |
|----------------------------|-------|----------|-----|
| calls - SELECT | 1 | 1,000 ms | 29% |
| SugarBean::populateFromRow | 102 | 860 ms | 25% |

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.

Summary **Trace details** SQL statements

Expand performance problems Collapse all

| Duration (ms) | Duration (%) | Segment | Drilldown | Timestamp |
|---------------|--------------|------------------------|-----------|-----------|
| 3,490 | 100.00% | /rest_Calls_filterList | | 0.000 s |
| 3,400 | 97.39% | RestService::execute | | 0.001 s |

Scroll down to find the SELECT query on calls took 1 second to load.

| | | | | |
|--------------|---------------|----------------------------|--|---------|
| 1,000 | 28.76% | MysqlManager::queryMulti | | 0.431 s |
| 1,000 | 28.70% | calls - SELECT | | 0.432 s |
| 17.0 | 0.49% | SugarBean::populateFromRow | | 1.437 s |

Click on the database icon next to the action to reveal the SQL query called by Sugar.

1,000 28.70%

calls - SELECT



0.432 s

SQL query

```
SELECT case when jt?_favorite_link.id IS NOT NULL then ? else ? end my_favorite, case when jt?_following_link.id IS NOT NULL then ? else ? end following, calls.name name, calls.status status, jt?_contacts.salutation contact_name__salutation, jt?_contacts.first_name contact_name__first_name, jt?_contacts.last_name contact_name__last_name, calls.parent_type parent_type, calls.parent_id parent_id, calls.date_start date_start, calls.date_end date_end, jt?_assigned_user_link.first_name assigned_user_name__first_name, jt?_assigned_user_link.last_name assigned_user_name__last_name, jt?_contacts.id contact_id, calls.assigned_user_id assigned_user_id, calls.id id, calls.date_modified date_modified, calls.created_by created_by FROM calls INNER JOIN (select tst.team_set_id from team_sets_teams tst INNER JOIN team_memberships team_memberships ON tst.team_id = team_memberships.team_id AND team_memberships.user_id = ? AND team_memberships.deleted=? group by tst.team_set_id) calls_tf on calls_tf.team_set_id = calls.team_set_id LEFT JOIN sugarfavorites sf_calls ON (calls.id = sf_calls.record_id AND sf_calls.deleted = ? AND sf_calls.module = ? AND sf_calls.created_by = ?) LEFT JOIN subscriptions jt?_subscriptions ON (calls.id = jt?_subscriptions.parent_id AND jt?_subscriptions.deleted = ? AND jt?_subscriptions.parent_type = ? AND jt?_subscriptions.created_by = ?) LEFT JOIN users jt?_following_link ON (jt?_following_link.id = jt?_subscriptions.created_by AND jt?_following_link.deleted = ?) LEFT JOIN calls_contacts jt?_calls_contacts ON (calls.id = jt?_calls_contacts.call_id AND jt?_calls_contacts.deleted = ?) LEFT JOIN contacts jt?_contacts ON (jt?_contacts.id = jt?_calls_contacts.contact_id AND jt?_contacts.deleted = ?) LEFT JOIN users jt?_assigned_user_link ON (calls.assigned_user_id = jt?_assigned_user_link.id AND jt?_assigned_user_link.deleted = ?) LEFT JOIN users jt?_favorite_link ON (jt?_favorite_link.id = sf_calls.modified_user_id AND jt?_favorite_link.deleted = ?) WHERE calls.deleted = ? ORDER BY calls.date_modified DESC LIMIT ?, ?
```

To view all SQL queries at once, click on the SQL Statements tab.

Summary

For critical business applications like CRM, an APM tool can help you 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: 03/15/2018 06:31pm

Backward Compatibility

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
- 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 Sidecar 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: 05/02/2017 01:04pm

Enabling Backward Compatibility

Overview

How to enable backward compatibility for a module.

Enabling Backward Compatibility

Backward Compatibility Mode is not a permanent solution for modules with legacy customizations. If you should need to temporarily get a module working due to legacy customizations, you can follow the steps below to enable the legacy MVC framework. Please note that switching stock Sugar modules from the Sidecar framework to backward compatibility mode is not supported and may result in unexpected behaviors in the application.

Enabling BWC

To enable backward compatibility, you must first create a file in `./custom/Extension/application/Ext/Include/` for the module. If the module is custom, there will already be an existing file in this folder pertaining to the module that you can edit.

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

```
<?php  
  
    $bwcModules[] = '<module key>';
```

Once the file is in place, you will need to navigate to Admin > Repair > Quick Repair and Rebuild. The Quick Repair can wait until you have completed the following sections for this customization.

Updating the MegaMenu Module Link

Once you have enabled backward compatibility for a module, you will then need to manually update the module's link in the MegaMenu. This will control the navigation when a user clicks the actual module name on the MegaMenu.

./custom/modules/<module>/clients/base/layouts/records/records.php

```
<?php  
  
    $viewdefs['<module key>']['base']['layout']['records'] = array(  
        'name' => 'bwc',  
        'type' => 'bwc',  
        'components' =>  
        array(  
            array(  
                'view' => 'bwc',  
            ),  
        ),  
    );
```

Once the file is in place, you will need to navigate to Admin > Repair > Quick Repair and Rebuild. The Quick Repair can wait until you have completed the following section for this customization.

Updating the MegaMenu Sub-Navigation Links

Once you have updated the MegaMenu module link, you will then need to manually update the module's MegaMenu action links.

The module's deployed sub-navigation links for a module will be similar to the file shown below:

./modules/<module>/clients/base/menus/header/header.php

```
<?php

$moduleName = '<module key>';

$viewdefs[$moduleName]['base']['menu']['header'] = array(
    array(
        'route' => "#$moduleName/create",
        'label' => 'LNK_NEW_RECORD',
        'acl_action' => 'create',
        'acl_module' => $moduleName,
        'icon' => 'icon-plus',
    ),
    array(
        'route' => "#$moduleName",
        'label' => 'LNK_LIST',
        'acl_action' => 'list',
        'acl_module' => $moduleName,
        'icon' => 'icon-reorder',
    ),
    array(
        'route' =>
"#bwc/index.php?module=Import&action=Step1&import_module=$moduleName&
return_module=$moduleName&return_action=index",
        'label' => 'LBL_IMPORT',
        'acl_action' => 'import',
        'acl_module' => $moduleName,
        'icon' => '',
    ),
);
```

This file should be duplicated to the custom directory and edited to adjust the URLs to the BWC format. The example below demonstrates the changes needed to the duplicated file.

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

```
<?php

$moduleName = '<module key>';

$viewdefs[$moduleName]['base']['menu']['header'] = array(
    array(
```

```

        'route' =>
"#bwc/index.php?module=$moduleName&action=EditView",
        'label' => 'LNK_NEW_RECORD',
        'acl_action' => 'create',
        'acl_module' => $moduleName,
        'icon' => 'icon-plus',
    ),
    array(
        'route' =>
"#bwc/index.php?module=$moduleName&action=ListView",
        'label' => 'LNK_LIST',
        'acl_action' => 'list',
        'acl_module' => $moduleName,
        'icon' => 'icon-reorder',
    ),
    array(
        'route' =>
"#bwc/index.php?module=Import&action=Step1&import_module=$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.get().modules` after running a Quick Repair and Rebuild in your browser's console window:

Using Developer Tools in Google Chrome

1. Open Developer Tools
 - This can be done in the following ways:
 1. Command + Option + i

2. View > Developer > Developer Tools

3. Right-click on the web page and selecting "Inspect Element"

2. Select the "Console" tab

3. Run the following command, replacing <module key> and verify that it returns true:

```
App.metadata.get().modules.<module key>.isBwcEnabled
```

Using Firebug in Google Chrome or Mozilla Firefox

1. Open Firebug

2. Select the "Console" tab

3. Run the following command, replacing <module key>, and verify that it returns true:

```
App.metadata.get().modules.<module key>.isBwcEnabled
```

Last Modified: 12/06/2016 04:11pm

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 will 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: 10/14/2016 04:12pm

Advanced Workflow

Introduction

Advanced Workflow is the next generation workflow tool for SugarCRM. Introduced in 7.6, Advanced Workflow is a Business Process Management (BPM) and a Business Process Modeling Notation (BPMN) tool. Advanced Workflow provides a simple drag-and-drop user interface along with an intelligent BPM flow designer to allow for the creation of easy yet powerful process development through standard BPM/BPMN diagrams.

Advanced Workflow enables administrators to streamline common business processes by managing approvals, sales processes, call triaging, and more. Advanced Workflow is an easy-to-use workflow tool that adds advanced BPM functionality to the core Sugar software stack.

A business process is a set of logically related tasks that are performed in order to achieve a specific organizational goal. It presents all of the tasks that must be completed in a simplified and streamlined format. Advanced Workflow empowers Sugar administrators, allowing for the automation of vital business processes for their organization.

The Advanced Workflow suite features an extensive toolbox of modules that provide the ability to easily create digital forms and map out fully functioning workflows.

Advanced Workflow Modules

Advanced Workflow is broken down into four distinct modules, each with its own area of responsibility. Three of the four Advanced Workflow modules have access control settings that restricts its visibility to System Administrators, Module Administrators, and Module Developers.

Process Definitions

A process definition defines the steps in an overall business process. Process definitions are created by either a Sugar administrator, a module Administrator or a module Developer. The process definition consists of a collection of activities and their relationships, criteria to indicate the start and end of the process, and information about the individual activities (e.g., participants) contained within the business process.

Business Rules

A business rule is a reusable set of conditions and outcomes that can be embedded in a process definition. The set of rules may enforce business policy, make a decision, or infer new data from existing data. For example, if Sally manages all business opportunities of \$10,000 or more, and Chris manages all business opportunities under \$10,000, a business rule can be created and used by all relevant process definitions based on the same target module to ensure that the assignment policy is respected. In the case of an eventual personnel change, only the business rule will need to be edited to affect all related processes.

Email Templates

A process email template is required in order to include a Send Message event in a process definition. The SugarCRM core product includes several places where email templates can be created for different purposes, but Advanced Workflow requires all sent messages to be created via the Process Email Templates module.

Processes

A process is a running instance of a process definition. A single process begins every time a process definition meeting certain criteria is executed. For example, a single process definition could be created to automate quote approvals, but because users may engage in several quote approvals per day, each approval will be represented by a separate process instance, all governed by the single process definition.

Database Tables

Information surrounding various portions of Advanced Workflow can be found in the following Advanced Workflow database tables:

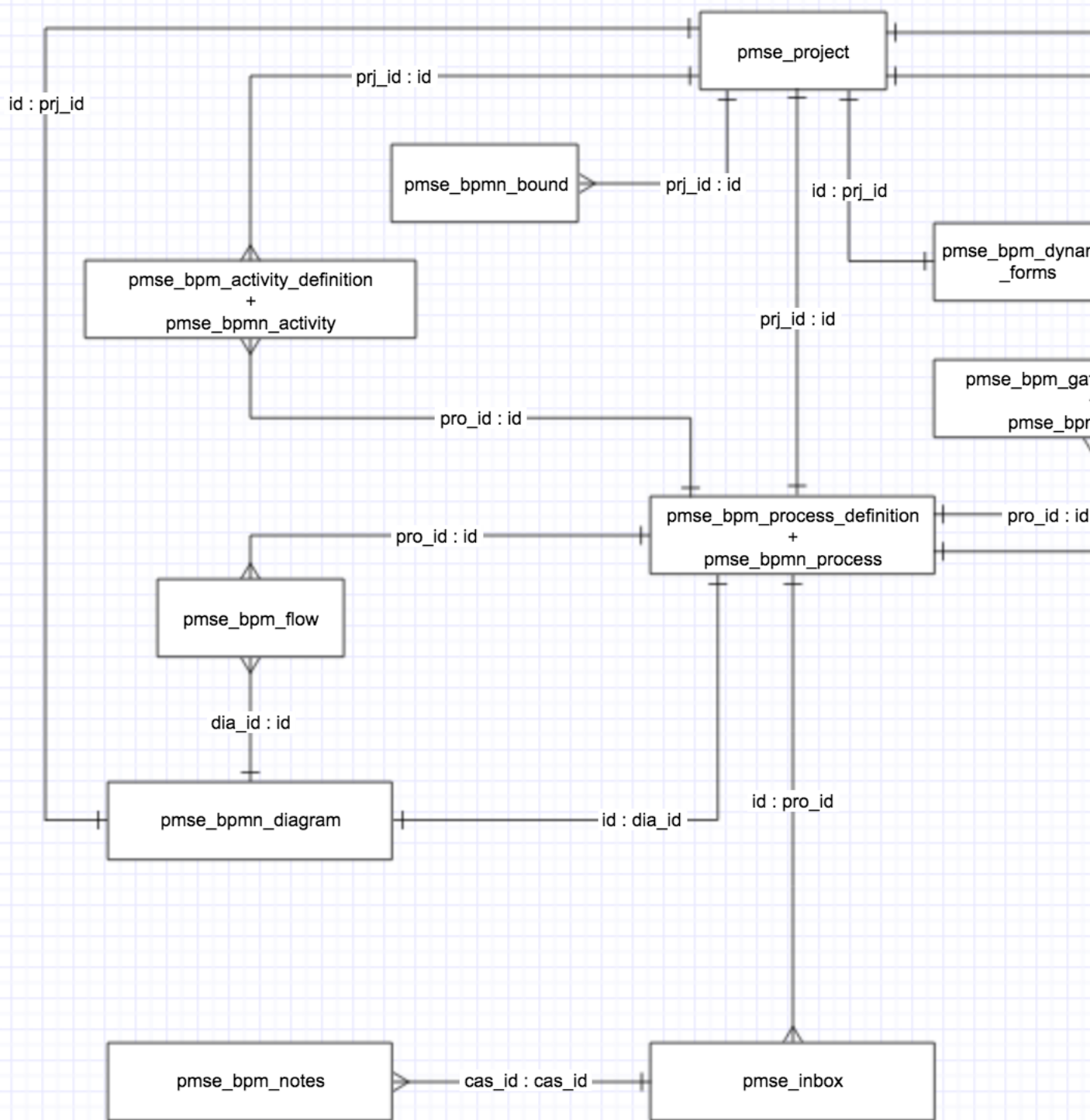
| Table name | Description |
|------------------------------|---|
| pmse_bpm_access_management | This table is not used. |
| pmse_bpm_activity_definition | Holds definition data for an activity. Maps directly to the pmse_bpmn_activity table and may, in the future, be merged with the pmse_bpmn_activity table to prevent forked data backends. |
| pmse_bpm_activity_step | This table is not used. |
| pmse_bpm_activity_user | This table is not used. |
| pmse_bpm_config | This table is not used. |
| pmse_bpm_dynamic_forms | Holds module specific form information - basically viewdefs for a module - for a process. |
| pmse_bpm_event_definition | Holds definition data for an event. Maps directly to the pmse_bpmn_event table and may, in the future, be merged with |

| Table name | Description |
|-----------------------------|--|
| | the pmse_bpmn_event table to prevent forked data backends. |
| pmse_bpm_flow | Holds information about triggered process flows as well as state of any process that is currently running. |
| pmse_bpm_form_action | Holds information about a process that has been acted upon. |
| pmse_bpm_gateway_definition | Holds definition data for a gateway. Maps directly to the pmse_bpmn_event table and may, in the future, be merged with the pmse_bpmn_gateway table to prevent forked data backends. |
| pmse_bpm_group | This table is not used. |
| pmse_bpm_group_user | This table is not used. |
| pmse_bpm_notes | Holds notes saved for a process record. |
| pmse_bpm_process_definition | Holds definition data for a process definition. Maps directly to the pmse_bpmn_process table and may, in the future, be merged with the pmse_bpmn_process table to prevent forked data backends. |
| pmse_bpm_related_dependency | Holds information about dependencies related to an event |
| pmse_bpm_thread | Holds information related to process threads for running processes. |
| pmse_bpmn_activity | Please see pmse_bpm_activity_definition |
| pmse_bpmn_artifact | Holds BPM artifact information, like Comments on a process diagram. At present, Advanced Workflow only supports the text annotation artifact. |
| pmse_bpmn_bound | Data related to element boundaries on the diagram. |
| pmse_bpmn_data | This table is not used. |
| pmse_bpmn_diagram | Data related to a process definition's diagram. |
| pmse_bpmn_documentation | This table is not used. |
| pmse_bpmn_event | Please see pmse_bpm_event_definition |

| | |
|-----------------------|--|
| pmse_bpmn_extension | This table is not used. |
| pmse_bpmn_flow | Holds information about the connecting flows of a diagram. |
| pmse_bpmn_gateway | Please see pmse_bpm_gateway_definition |
| pmse_bpmn_lane | This table is not used. |
| pmse_bpmn_laneset | This table is not used. |
| pmse_bpmn_participant | This table is not used. |
| pmse_bpmn_process | Please see pmse_bpm_process_definition |
| pmse_business_rules | Holds business rule record data. |
| pmse_emails_templates | Holds email template record data. |
| pmse_inbox | Holds information about processes that are currently running or that have completed. |
| pmse_project | Holds process definition record information. |

Relationships

The following Entity Relationship Diagram highlights the relationships between the various Advanced Workflow tables.



Flows

When a process definition is created the following 5 tables get populated:

1. pmse_bpm_dynamic_forms contains dyn_view_defs which is the JSON encoded string of module specific form information
2. pmse_bpm_process_definition contains the module associated with and the status of a process definition
3. pmse_bpmn_diagram contains process definition name as well and a diagram uid
4. pmse_bpmn_process contains process definition name. The same Id is shared between pmse_bpm_process_definition and pmse_bpmn_process dia_id is a foreign key related to pmse_bpmn_diagram
5. pmse_project contains process definition name, project id, module, status. Other tables have a foreign key of prj_id in them.

Upon adding a start/end/send message event:

1. pmse_bpmn_event contains event name
2. pmse_bpm_event_definition Shares id with pmse_bpmn_event
3. pmse_bpmn_bound

When Settings is changed to "New Records Only"

1. pmse_bpm_related_dependency

Upon adding an action or activity:

1. pmse_bpm_activity_definition contains the action name
 - The act_fields column contains the JSON encoded list of fields which will be changed
2. pmse_bpm_activity contains action name. Shares id with pmse_bpm_activity_definition. act_script_type contains the type of action e.g. CHANGE_FIELD
3. pmse_bpmn_bound

Upon adding a connector between start event and action:

1. pmse_bpmn_flow contains origin and destination info

Upon adding a Gateway:

-
1. pmse_bpm_gateway_definition
 2. pmse_bpm_gateway shares id with pmse_bpm_gateway_definition
 3. pmse_bpmn_flowflo_condition contains JSON encoded string of conditions required to proceed ahead with an action or activity
 4. pmse_bpmn_bound

When a process runs:

1. pmse_inboxcas_status indicates the status of a process
2. pmse_bpm_thread
3. pmse_bpm_flow Stores the entire flow that the process elements went through (so if 3 elements and 2 connectors are in the process definition then 5 records will exist)

After Process has run and appears in Process Management:

1. pmse_bpm_case_data
2. pmse_bpm_form_action contains frm_action, cas_pre_data , and cas_data

Add a comment:

1. pmse_bpmn_artifact contains the comment
2. pmse_bpm_bound

Add a business rule to a process definition:

1. pmse_business_rulesrst_source_definition contains the conditions of the business rule
2. pmse_bpm_activity_definitionact_fields contains the business rule id

Add a email template to a process definition:

1. pmse_email_templates
2. pmse_bpm_event_definitionevn_criteria contains the email template id

Extension and Customization

In 7.8 Sugar introduced the ProcessManager library to support extending Advanced Workflow in an upgrade safe way. For more information on extending Advanced Workflow, or on using the Process Manager library, please read the Process Manager documentation.

Caveats

In Sugar 7.8 the Process Manager library brought about the Registry Object to maintain the state of Advanced Workflows during a PHP Process. This change introduced a limitation in Advanced Workflow that prevents the same Process Definition from running on multiple records inside the same PHP process. For certain customizations in Sugar, if you are updating multiple records in a module using SugarBean, you may want them to trigger the defined Process Definitions for each record. The following code can be added to your customization to allow for that functionality:

```
use Sugarcrm\Sugarcrm\ProcessManager\Registry;

//custom code
Registry\Registry::getInstance()->drop('triggered_starts');
$bean->save();
```

The 'triggered_starts' registry contains the Start Event IDs that have been triggered previously in the PHP process, thus allowing the SugarBean::save() method to trigger Advanced Workflow and go through the same Start Event again.

Last Modified: 10/03/2017 05:15pm

Extending Advanced Workflow (Process Manager)

Introduction

From its earliest version, Sugar has always been a tool that allows for customizing and extending in a way that allows developers to shape it and mold it to a specific business need. Most components of Sugar are customizable or extensible through the use of custom classes (for custom functionality) or extensions (for custom metadata). However, a recent addition to Sugar did not fit this paradigm.

Process Manager

The Process Manager Library was created to give developers the ability to customize any portion of Advanced Workflow where object instantiation was previously handled using the 'new' operator. By way of ProcessManager\Factory we can now develop custom versions of most any PMSE* class and have that class used in place of the core PMSE* class.

Limitations

The current implementation of the Process Manager Library only handles server-side customizations by way of the Factory class. These customizations can be applied to any instantiable pmse_* object, however, static classes and singleton classes, such as the following, are not affected by the library:

- PMSE
- PMSEEngineUtils
- PMSELogger

While some effort has been made to allow for the creation of custom actions in the Process Definition designer, as of Sugar 7.8, that is the only client customization allowance that will be implemented.

Library Structure

The Process Manager Library is placed under the Sugarcrm\Sugarcrm\ProcessManager namespace and contains the following directories and classes:

- Factory
- Exception/
 - BaseException
 - DateTimeException
 - ExceptionInterface
 - ExecutionException
 - InvalidDataException
 - NotAuthorizedException
 - RuntimeException
- Field/
 - Evaluator/

-
- AbstractEvaluator
 - Base
 - Currency
 - Datetime
 - Decimal
 - EvaluatorInterface
 - Int
 - Multienum
 - Relate
 - Registry/
 - Registry
 - RegistryInterface

Examples

Getting an Advanced Workflow object

Almost all objects in Advanced Workflow can be instantiated through the Process Manager Factory. When loading a class, the Factory getPMSEObject method will look in the following directories as well as the custom versions of these directories for files that match the object name being loaded:

- modules/pmse_Business_Rules/
- modules/pmse_Business_Rules/clients/base/api/
- modules/pmse_Emails_Templates/
- modules/pmse_Emails_Templates/clients/base/api/
- modules/pmse_Inbox/clients/base/api/
- modules/pmse_Inbox/engine/
- modules/pmse_Inbox/engine/parser/
- modules/pmse_Inbox/engine/PMSEElements/
- modules/pmse_Inbox/engine/PMSEHandlers/
- modules/pmse_Inbox/engine/PMSEPreProcessor/
- modules/pmse_Inbox/engine/wrappers/
- modules/pmse_Project/clients/base/api/
- modules/pmse_Project/clients/base/api/wrappers/
- modules/pmse_Project/clients/base/api/wrappers/PMSEObservers/

What this means is that virtually any class that is found by name in these directories can be instantiated and returned via the getPMSEObject() method. This also means that customizations to any of these classes can be easily implemented by creating a Custom version of the class under the appropriate ./custom directory path.

For example, to create a custom version of PMSEProjectWrapper, you would create a CustomPMSEProjectWrapper class at `./custom/modules/pmse_Project/clients/base/api/wrappers/CustomPMSEProjectWrapper.php`. This allows you to have full control of your instance of Advanced Workflow while giving you the flexibility to add any classes you want and consuming them through the factory.

```
<?php

// Save this file at
./custom/modules/pmse_Project/clients/base/api/wrappers/CustomPMSEProjectWrapper.php
require_once
'modules/pmse_Project/clients/base/api/wrappers/PMSEProjectWrapper.php
';
class CustomPMSEProjectWrapper extends PMSEProjectWrapper
{
}
```

This can now be consumed by simply calling the `getPMSEObject()` method:

```
<?php

// Set the process manager library namespace
use \Sugarcrm\Sugarcrm\ProcessManager\Factory;

// Get our wrapper
$wrapper =
ProcessManager\Factory::getPMSEObject('PMSEProjectWrapper');
```

Getting an Advanced Workflow Element Object

Element objects in Advanced Workflow generally represent some part of the Process engine, and often time map to design elements. All Element objects can be found in the `modules/pmse_Inbox/engine/PMSEElements/` directory.

Getting an Advanced Workflow Element object can be done easily by calling the `getElement()` method of the Process Manager Factory:

```
<?php

use \Sugarcrm\Sugarcrm\ProcessManager\Factory;
```

```
// Get a default PMSEElement object
$element = ProcessManager\Factory::getElement();
```

To get a specific object, you can pass the object name to the getElement() method:

```
$activity = ProcessManager\Factory::getElement('PMSEActivity');
```

Alternatively, you can remove the PMSE prefix and call the method with just the element name:

```
$gateway = ProcessManager\Factory::getElement('Gateway');
```

To create a custom Advanced Workflow Element you can simply create a new file at ./custom/modules/pmse_Inbox/engine/PMSEElements/ where the file name is prefixed with Custom.

```
<?php

// Save this file at
./custom/modules/pmse_Inbox/engine/PMSEElements/CustomPMSEScriptTask.php
require_once
'modules/pmse_Inbox/engine/PMSEElements/PMSEScriptTask.php';
use Sugarcrm\Sugarcrm\ProcessManager;
class CustomPMSEScriptTask extends PMSEScriptTask
{
}
```

Note: All Advanced Workflow Element classes must implement the PMSERunnable interface. Failure to implement this interface will result in an exception when using the Factory to get an Element object.

To get a custom ScriptTask object via the Factory, just call getElement():

```
$obj = ProcessManager\Factory::getElement('ScriptTask');
```

To make your own custom widget element, you can implement the PMSERunnable interface:

```
<?php

// Save this file at
./custom/modules/pmse_Inbox/engine/PMSEElements/CustomPMSEWidget.php
require_once
```

```
'modules/pmse_Inbox/engine/PMSEElements/PMSERunnable.php';
```

```
use Sugarcrm\Sugarcrm\ProcessManager;
class CustomPMSEWidget implements PMSERunnable
{
}
```

To get a Widget object via the Factory, just call getElement():

```
$obj = ProcessManager\Factory::getElement('Widget');
```

Getting and Using a Process Manager Field Evaluator Object

Process Manager Field Evaluator objects determine if a field on a record has changed and if a field on a record is empty according to the field type. The purpose of these classes can grow to include more functionality in the future.

Getting a field evaluator object is also done through the Factory:

```
<?php
use\Sugarcrm\Sugarcrm\ProcessManager\Factory;

// Get a Datetime Evaluator. Type can be date, time, datetime or
datetimecombo$eval =ProcessManager\Factory::getFieldEvaluator(['type'
=> 'date']);
```

A more common way of getting a field evaluator object is to pass the field definitions for a field on a SugarBean into the Factory:

```
$eval =ProcessManager\Factory::getFieldEvaluator($bean ->field_defs[
$field]);
```

Once the evaluator object is fetched, you may initialize it with properties:

```
/* This is commonly used for field change evaluations */// $bean is a
SugarBean object// $field is the name of the field being evaluated//
$data is an array of data that is used in the evaluation$eval ->init(
$bean, $field, $data);

// Has the field changed?$changed = $eval->hasChanged();
```

Or you can set properties onto the evaluator:

```
/* This is commonly used for empty evaluations */ // Set the bean onto
the evaluator $eval->setBean($bean);

// And set the name onto the evaluator
$eval->setName($field);

// Are we empty? $isEmpty = $eval->isEmpty();
```

Creating and consuming a custom field evaluator is as easy as creating a custom class and extending the Base class:

```
<?php
// Save this file at ./
custom/src/ProcessManager/Field/Evaluator/CustomWidget.phpnamespace
Sugarcrm\Sugarcrm\ProcessManager\Field\Evaluator;

/** * Custom widget field evaluator * @package ProcessManager */class
```

Alternatively, you can create a custom evaluator that extends the `AbstractEvaluator` parent class if you implement the `EvaluatorInterface` interface.

```
<?php
// Save this file at ./
custom/src/ProcessManager/Field/Evaluator/CustomWidget.phpnamespace
Sugarcrm\Sugarcrm\ProcessManager\Field\Evaluator;

/** * Custom widget field evaluator * @package ProcessManager */class
```

And at this point, you can use the Factory to get your widget object:

```
// Get a custom widget evaluator object $eval = ProcessManager\Factory
::getFieldEvaluator(['type' => 'widget']);
```

Getting an Exception Object with Logging

Getting an Advanced Workflow exception with logging is as easy as calling a single Factory method. Currently, the Process Manager library supports the following exception types:

- DateTime
- Execution
- InvalidData

-
- NotAuthorized
 - Runtime

As a fallback, the Base exception can be used. All exceptions log their message to the PMSE log when created, and all exceptions implement ExceptionInterface.

The most basic way to get a ProcessManager exception is to call the `getException()` method on the Factory with no arguments:

```
<?php
use\Sugarcrm\Sugarcrm\ProcessManager\Factory;

// Get a BaseException object, with a default message// of 'An unknown
Process Manager exception had occurred'$exception =ProcessManager\
Factory::getException();
```

To get a particular type of exception, or to set your message, you can add the first two arguments:

```
if ($error) {
    throwProcessManager\Factory::getException('InvalidData', $error);
}
```

It is possible to set an exception code, as well as a previous exception, when creating an Exception object:

```
// Assume $previous is an ExecutionExceptionthrowProcessManager\
Factory ::getException('Runtime', 'Cannot carry out the action', 255,
```

Finally, to create your own custom exception classes, you must add a new custom file in the following path with the file named appropriately:

```
<?php
// Save this file at ./c
ustom/src/ProcessManager/Exception/CustomEvaluatorException.php
namespaceSugarcrm\Sugarcrm\ProcessManager\Exception;

/** * Custom Evaluator exception * @package ProcessManager */class
CustomEvaluatorException extends BaseException implements
ExceptionInterface{}
```

And to consume this custom exception, call the Factory:

```
$exception =ProcessManager\Factory::getException('Evaluator', 'Could
```

```
not evaluate the expression');
```

Maintaining State Throughout a Process

The Registry class implements the RegistryInterface which exposes the following public methods:

- set(\$key, \$value, \$override = false)
- get(\$key, \$default = null)
- has(\$key)
- drop(\$key)
- getChanges(\$key)
- reset()

To use the Registry class, simply use the namespace to get the singleton:

```
<?php
use\Sugarcrm\Sugarcrm\ProcessManager\Registry;

$registry =Registry\Registry::getInstance();
```

To set a value into the registry, simply add it along with a key:

```
$registry->set('reg_key', 'Hold on to this');
```

NOTE: To prevent named index collision, it is advisable to write indexes to contain a namespace indicator, such as a prefix.

```
$registry->set('mykeyindex:reg_key', 'Hold on to this');
```

Once a value is set in the registry, it is immutable unless the set(\$key, \$value, \$override = false) method is called with the override argument set to true:

```
// Sets the reg_key value
$registry->set('reg_key', 'Hold on to this');

// Since the registry key reg_key is already set, this will do nothing

// To forcefully set it, add a true for the third argument
$registry->set('reg_key', 'No wait!', true);
```

When a key is changed on the registry, these changes are logged and can be inspected using getChanges():

```
// Set and reset a value on the registry
$registry->set('key', 'Foo');
$registry->set('key', 'Bar', true);
$registry->set('key', 'Baz', true);

// Get the changes for this key
$changes = $registry->getChanges('key')

/*$changes will be: array(
    0 => array(
        'from' => 'Foo',
        'to' => 'Bar',
    ),
    1 => array(
        'from' => 'Bar',
        'to' => 'Baz',
    ), */
```

To get the value of the key you just set, call the `get()` method:

```
$value = $registry->get('reg_key');
```

To get a value of a key with a default, you can pass the default value as the second argument to `get()`:

```
// Will return either the value for attendee_count or 0
$count = $registry->get('attendee_count', 0);
```

To see if the registry currently holds a value, call the `has()` method:

```
if ($registry->has('field_list')) {
    // Do something here
}
```

To remove a value from the registry, use the `drop()` method. This will add an entry to change log, provided the key was already set.

```
$registry->drop('key');
```

Finally, to clear the entire registry of all values and change log entries, use the `reset()` method.

```
$registry->reset();
```

Note: This is very destructive. Please use with caution because you can purge settings that you didn't own.

Last Modified: 11/27/2017 12:17pm

Integration

Overview

How to integrate with Sugar APIs

Last Modified: 11/30/2016 01:29pm

Best Practices

Overview

Best practices when integrating and migrating Sugar.

Latency When Posting Data To Sugar

When integrating with Sugar, it is best to avoid long-running web requests. While performance-draining requests are not always obvious, once an issue has been identified, it is best to move the processing to the backend.

By default, we expect a request to an endpoint such as POST /Accounts to be straightforward, however, adding Workflows, Logic Hooks, and Sugar Logic may stress the otherwise simple process and cause issues with webheads and other resources being used in the process. If you are experiencing these symptoms, the following sections may help you.

Disabling Related Calculation Fields

When a calculated field in Sugar uses the related function in the Sugar Logic, this will cause the calculated field to be executed when the related module is updated. This can cause a cascading effect through the system to update related calculated fields. When this happens you may receive a 502 Gateway Error. You can disable the related calculation field updates temporarily or permanently by adding the following line to the config_override.php file:

```
$sugar_config['disable_related_calc_fields'] = true;
```

Note : This is a global setting that will affect all modules. If you have a calculated field in Accounts that sums up all Opportunities for the account, setting this value to true will no longer update the opportunity account sum in Accounts until the account record itself is modified. However, if this setting is left disabled, the sum

would update any time a related opportunity or the account is modified.

More information on this setting can be found in the core configuration settings.

Disabling Logic Hooks

When data is being migrated into Sugar, logic hooks may be adding unnecessary time to your API requests. It is highly recommended for you to disable any unnecessary logic hooks from your system during an initial import. Logic hook definitions may be located in the following files and/or directories:

- ./custom/modules/logic_hooks.php
- ./custom/modules/<module>/logic_hooks.php
- ./custom/Extension/application/Ext/LogicHooks/
- ./custom/Extension/modules/<module>/Ext/LogicHooks/

Disabling Workflows

When data is being migrated into Sugar, workflows may be adding unnecessary time to your API requests. It is highly recommended for you to disable any unnecessary workflows from your system during an initial import in Admin > Workflows.

Queuing Data in the Job Queue

One solution for long running requests is to send the data to the Job Queue to be processed by the cron. To accomplish this, make a file customization to add to the target processing function. For this article's use case, create ./custom/Extension/modules/Schedulers/Ext/ScheduledTasks/CustomCreateAccountJob.php and send any stored data to it for processing. Enter the following code in the php file:

```
<?php
```

```
function CustomCreateAccountJob($job)
{
    if (!empty($job->data)) {
        $account = BeanFactory::newBean('Accounts');
        $fields=json_decode($job->data, true);
```

```
    foreach($fields as $field => $value) {
        $account->$field = $value;
    }

    $account->save();

    if (!empty($account->id)) {
        return true;
    }
}

return false;
}
```

Next, modify the request to pass the new account data to the SchedulerJobs module in the data field and specify the new function in the target field, as follows:

```
curl -v -H "oauth-token: 4afd4aea-df99-7cec-4d94-560a97cda9f8" -H
"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: 11/27/2017 12:17pm

Web Services

Overview

Web Services allow for communication between different applications and platforms. Sugar currently supports REST and SOAP APIs. The following sections will outline how to interact with the APIs and what versions of the API we recommend for use.

Versioning

API versioning is the process of creating a new set of API endpoints for new functionality while leaving 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 |
|---------|--------------|------------------------|--------------|------------------------|
| 7.8.x | v10 | /rest/v10/ | v4.1 | /service/v4_1/soap.php |
| 7.7.x | v10 | /rest/v10/ | v4.1 | /service/v4_1/soap.php |
| 6.5.x | v4.1 | /service/v4_1/rest.php | v4.1 | /service/v4_1/soap.php |

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: 05/17/2017 06:02pm

v10

Overview

v10 API documentation.

What Is REST?

REST stands for 'Representational State Transfer'. As of 7.x, REST is a core component of Sugar that defines how all information is exchanged within the application. 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.

Getting Started

How to Access the v10 REST Service

The base endpoint for the v10 REST service can be found at:

```
http://<site_url>/rest/v10/
```

For your reference, all v10 endpoints can be found by navigating to http://<site_url>/rest/v10/help. Once you have identified your instance's base endpoint, we can begin by authenticating.

Authentication

Sugar 7 uses two-legged OAuth2 for authentication. You simply need to do a POST to </rest/v10/oauth2/token> with the following parameters:

| | | |
|------------|--------|--|
| grant_type | String | Type of request. Available grant types are "password" and "refresh_token". |
| client_id | String | The client_id of "sugar" will automatically create an OAuth Key in the system and can be used for "password" authentication. The client_id of "support_portal" will create an OAuth Key if the portal system is enabled and will allow for portal authentication. Other client_id's can be created by the administrator in the OAuthKeys section in the Administration section and can be used in the future for additional grant types, if the client secret is filled in, it will be checked to validate use of the client |

| | | id. |
|---------------|--------|--|
| client_secret | String | The client's secret key. |
| username | String | The username of the user authenticating to the system. |
| password | String | The plaintext password the user authenticating to the system. |
| platform | String | Defaults to "base" allows you to have custom meta-data per platform. This parameter should be an identifiable string. Do not use random GUIDS or changing variables. Doing so may result in large ./cache directories. |

First, we are going to login using a grant_type of "password" and a platform of "custom". Normally, when logging into Sugar, users login with a platform type of "base". We are using "custom" to avoid any potential login conflicts.

```
curl -X POST -H Cache-Control:no-cache -d '{
  "grant_type":"password",
  "client_id":"sugar",
  "client_secret":"",
  "username":"<username>",
  "password":"<password>",
  "platform":"custom"
}' http://<site_url>/rest/v10/oauth2/token
```

Once you get the response you'll need to hold onto the access_token and the refresh_token. Anytime the access_token is invalidated, you'll want to make another request to the token endpoint with a grant_type of "refresh_token". Store just the refresh_token in long term storage - not the username and password. The response from the server will be as follows:

```
{
  "access_token": "5ee48ec7-023e-ecff-5184-530bd0358868",
  "expires_in": 3600,
  "token_type": "bearer",
  "scope": null,
  "refresh_token": "5f197357-0167-f7a6-7912-530bd03275b6",
  "refresh_expires_in": 1209600,
```

```
  "download_token": "5f531625-e301-e3ea-1b11-530bd098be41"
}
```

Avoiding Login Conflicts

Login conflicts often occur when building integrations or running data migrations with the platform of "base" or any other client type that is in use. This is due to the fact that Sugar uses the same REST API to power all the various clients such as Sugar, Portal, Mobile, and even the Outlook Plugin. Due to this, you need to let the API know you aren't conflicting with another client that may be in use. The way to accomplish this is the `/rest/v10/oauth2/token` call by changing the platform parameter to something other than "base", "mobile", or "portal". It is best to name it something that describes and identifies your current integration.

Input / Output Data Types

The default input / output datatype for REST is JSON.

Date Handling

Date and date time inputs should be formatted following the ISO 8601 format. If the time zone is not included in a request, Sugar will assume the time zone of the user making the request.

Filter on a specific date:

```
{"date_start": "2015-08-12"}
```

Filter on a date keyword using `$dateRange`:

```
{"date_start": {"$dateRange": "today"}}
```

Filter on date range using manual time zones:

```
{"date_start": {"$dateBetween":  
["2015-09-10T00:00:00+10:00", "2015-09-10T23:59:59+10:00"]}}
```

Last Modified: 07/31/2017 10:44pm

Endpoints

The following sections contain the in-app help documentation for endpoints.

Last Modified: 11/30/2016 09:02pm

/Accounts/:record/link/:link_name/filter GET

Overview

Lists related filtered records.

Summary

This endpoint will return a set of related records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Care will need to be taken to make sure this filter has appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

| Name | Type | Description | Required |
|---------|---------|---|----------|
| filter | String | The filter expression. Filter expressions are explained below. | False |
| max_num | Integer | A maximum number of records to return. Default is 20. | False |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |

| Name | Type | Description | Required |
|----------|--------|--|----------|
| fields | String | Comma delimited list of fields to return. The field date_modified will always be returned. This argument can be combined with the view argument. Example:
name,account_type,description | False |
| view | String | Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list". Example: record | False |
| order_by | String | How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon. Example:
name:DESC,account_type:DESC,date_modified:ASC | False |

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
  "filter": [
    {
      "name": "Nelson Inc"
    }
  ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
  "filter": [
    {
      "name": {
        "$starts": "Nelson"
      }
    }
  ]
}
```

Below is a list of operation types:

| Operation | Description |
|---------------------------|--|
| <code>\$equals</code> | Performs an exact match on that field. |
| <code>\$not_equals</code> | Performs an exact match on that field. |

| Operation | Description |
|--------------|---|
| \$not_equals | Matches on non-matching values. |
| \$starts | Matches on anything that starts with the value. |
| \$ends | Matches anything that ends with the value. |
| \$contains | Matches anything that contains the value |
| \$in | Finds anything where field matches one of the values as specified as an array. |
| \$not_in | Finds anything where field does not matches any of the values as specified as an array. |
| \$is_null | Checks if the field is null. This operation does not need a value specified. |
| \$not_null | Checks if the field is not null. This operation does not need a value specified. |
| \$lt | Matches when the field is less than the value. |
| \$lte | Matches when the field is less than or equal to the value. |
| \$gt | Matches when the field is greater than the value. |
| \$gte | Matches when the field is greater than or equal to the value. |

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "\$and" and "\$or".

Example

```
{
```

```

"filter":[
  {
    "$or":[
      {
        "name":"Nelson Inc"
      },
      {
        "name":"Nelson LLC"
      }
    ]
  }
]
}

```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "\$favorite" and "\$owner".

Example

```

{
  "filter":[
    {
      "$favorite":"_this"
    }
  ]
}

```

Response Arguments

| Name | Type | Description |
|-------------|---------|--|
| next_offset | Integer | Displays the next offset for retrieval of additional results. -1 will be returned when there are no more |

| Name | Type | Description |
|---------|-------|---|
| | | records. |
| records | Array | An array of results containing matched records. |

Reponse

```
{
  "next_offset":-1,
  "records":[
    {
      "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
      "name":"Dale Spivey",
      "date_entered":"2013-02-26T19:12:00+00:00",
      "date_modified":"2013-02-28T05:03:00+00:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "description":"",
      "img":"",
      "deleted":false,
      "assigned_user_id":"seed_sally_id",
      "assigned_user_name":"Sally Bronsen",
      "team_name":[
        {
          "id":"East",
          "name":"East",
          "name_2":"",
          "primary":false
        },
        {
          "id":1,
          "name":"Global",
          "name_2":"",
          "primary":false
        },
        {
          "id":"West",
          "name":"West",
          "name_2":"",

```

```
        "primary":true
    }
],
"salutation": "",
"first_name": "Dale",
"last_name": "Spivey",
"full_name": "Dale Spivey",
"title": "VP Operations",
"linkedin": "",
"facebook": "",
"twitter": "",
"googleplus": "",
"department": "",
"do_not_call": false,
"phone_home": "(523) 825-4311",
"email": [
    {
        "email_address": "sugar.dev.sugar@example.co.jp",
        "opt_out": "0",
        "invalid_email": "0",
        "primary_address": "1"
    },
    {
        "email_address": "the.support@example.biz",
        "opt_out": "0",
        "invalid_email": "0",
        "primary_address": "0"
    }
],
"phone_mobile": "(373) 861-0757",
"phone_work": "(212) 542-9596",
"phone_other": "",
"phone_fax": "",
"email1": "sugar.dev.sugar@example.co.jp",
"email2": "the.support@example.biz",
"invalid_email": false,
"email_opt_out": false,
"primary_address_street": "345 Sugar Blvd.",
"primary_address_street_2": "",
"primary_address_street_3": "",
"primary_address_city": "Denver",
"primary_address_state": "CA",
"primary_address_postalcode": "87261",
"primary_address_country": "USA",
```

```
"alt_address_street":"","  
"alt_address_street_2":"","  
"alt_address_street_3":"","  
"alt_address_city":"","  
"alt_address_state":"","  
"alt_address_postalcode":"","  
"alt_address_country":"","  
"assistant":"","  
"assistant_phone":"","  
"picture":"","  
"email_and_name1":"","  
"lead_source":"Campaign",  
"account_name":"Smallville Resources Inc",  
"account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",  
"opportunity_role_fields":"","  
"opportunity_role_id":"","  
"opportunity_role":"","  
"reports_to_id":"","  
"report_to_name":"","  
"portal_name":"DaleSpivey97",  
"portal_active":true,  
"portal_password":"$1$yKMAONHM$Y5S.8CY.WZCZCwfGD1a1Q\/",  
"portal_password1":"","  
"portal_app":"","  
"preferred_language":"en_us",  
"campaign_id":"","  
"campaign_name":"","  
"c_accept_status_fields":"","  
"m_accept_status_fields":"","  
"accept_status_id":"","  
"accept_status_name":"","  
"sync_contact":"","  
"my_favorite":false,  
"_acl":{  
  "fields":{  
  
  }  
}  
},  
{  
  "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",  
  "name":"Florence Haddock",  
  "date_entered":"2013-02-26T19:12:00+00:00",  
  "date_modified":"2013-02-26T19:12:00+00:00",
```

```
"modified_user_id":"1",
"modified_by_name":"Administrator",
"created_by":"1",
"created_by_name":"Administrator",
"description":"",
"img":"",
"deleted":false,
"assigned_user_id":"seed_sally_id",
"assigned_user_name":"Sally Bronsen",
"team_name":[
  {
    "id":"East",
    "name":"East",
    "name_2":"",
    "primary":false
  },
  {
    "id":1,
    "name":"Global",
    "name_2":"",
    "primary":false
  },
  {
    "id":"West",
    "name":"West",
    "name_2":"",
    "primary":true
  }
],
"salutation":"",
"first_name":"Florence",
"last_name":"Haddock",
"full_name":"Florence Haddock",
"title":"Director Sales",
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(729) 845-3137",
"email":[
  {
    "email_address":"dev.vegan@example.de",
```

```
        "opt_out": "1",
        "invalid_email": "0",
        "primary_address": "0"
    },
    {
        "email_address": "section71@example.it",
        "opt_out": "0",
        "invalid_email": "0",
        "primary_address": "1"
    }
],
"phone_mobile": "(246) 233-1382",
"phone_work": "(565) 696-6981",
"phone_other": "",
"phone_fax": "",
"email1": "section71@example.it",
"email2": "dev.vegan@example.de",
"invalid_email": false,
"email_opt_out": false,
"primary_address_street": "111 Silicon Valley Road",
"primary_address_street_2": "",
"primary_address_street_3": "",
"primary_address_city": "Denver",
"primary_address_state": "CA",
"primary_address_postalcode": "79900",
"primary_address_country": "USA",
"alt_address_street": "",
"alt_address_street_2": "",
"alt_address_street_3": "",
"alt_address_city": "",
"alt_address_state": "",
"alt_address_postalcode": "",
"alt_address_country": "",
"assistant": "",
"assistant_phone": "",
"picture": "",
"email_and_name1": "",
"lead_source": "Support Portal User Registration",
"account_name": "Smallville Resources Inc",
"account_id": "d5db6292-5c24-eb61-e202-512d09f0134e",
"opportunity_role_fields": "",
"opportunity_role_id": "",
"opportunity_role": "",
"reports_to_id": "",
```

```

"report_to_name": "",
"portal_name": "FlorenceHaddock169",
"portal_active": true,
"portal_password": "$1$WFhTbK6$JF9BCGSqL\NCRbhueX5ia0",
"portal_password1": "",
"portal_app": "",
"preferred_language": "en_us",
"campaign_id": "",
"campaign_name": "",
"c_accept_status_fields": "",
"m_accept_status_fields": "",
"accept_status_id": "",
"accept_status_name": "",
"sync_contact": "",
"my_favorite": false,
"_acl": {
  "fields": {
    }
  }
}
]
}

```

Change Log

| Version | Change |
|---------|--|
| v10 | Added
/<module>/:record/link/:link_name/filter
GET endpoint. |

Last Modified: 05/31/2017 05:25pm

/Activities GET

Activities on the home page

Summary:

This endpoint lists activities across the entire system. It uses a subscription model, and can be queried like a normal bean collection, but without search, ordering or filtering.

Query Parameters:

| Param | Description | Optional |
|---------|---|----------|
| max_num | A maximum number of records to return | Optional |
| offset | How many records to skip over before records are returned | Optional |

Input Example:

This endpoint does not accept any input

Output Example:

```
{
  "next_offset": 20, This will be set to -1 when there are no more
records after this "page".
  "records": [{
    "id": "22fb8b16-de1d-f1dc-b15b-51240efde177",
    "date_entered": "2013-02-19T23:47:11+00:00",
    "date_modified": "2013-02-19T23:47:11+00:00",
    "created_by": "1",
    "deleted": "0",
    "parent_id": "f5bb0331-2c0f-5c7c-b4db-5123caac0056",
    "parent_type": "Contacts",
    "activity_type": "post", This will be the type of activity
performed.
    "data": {
      "value": "This is a test post on a contact I'm subscribed
to."
    },
    "comment_count": 0, This will be set to the total number of
comments on the post.
    "last_comment": { This will be the last comment on the post,
```

which can be used to create a comment model on the frontend.

```
        "deleted": 0,
        "data": []
    },
    "fields": [],
    "first_name": null,
    "last_name": "Administrator",
    "created_by_name": " Administrator" This will be the
locale-formatted full name of the user.
    }, ... ]
}
```

Last Modified: 11/30/2016 09:16pm

/Activities/filter GET

Activities on the home page

Summary:

This endpoint lists activities across the entire system. It uses a subscription model, and can be queried like a normal bean collection, but without search, ordering or filtering.

Query Parameters:

| Param | Description | Optional |
|---------|---|----------|
| max_num | A maximum number of records to return | Optional |
| offset | How many records to skip over before records are returned | Optional |

Input Example:

This endpoint does not accept any input

Output Example:

```
{
  "next_offset": 20, This will be set to -1 when there are no more
  records after this "page".
  "records": [{
    "id": "22fb8b16-de1d-f1dc-b15b-51240efde177",
    "date_entered": "2013-02-19T23:47:11+00:00",
    "date_modified": "2013-02-19T23:47:11+00:00",
    "created_by": "1",
    "deleted": "0",
    "parent_id": "f5bb0331-2c0f-5c7c-b4db-5123caac0056",
    "parent_type": "Contacts",
    "activity_type": "post", This will be the type of activity
    performed.
    "data": {
      "value": "This is a test post on a contact I'm subscribed
    to."
    },
    "comment_count": 0, This will be set to the total number of
    comments on the post.
    "last_comment": { This will be the last comment on the post,
    which can be used to create a comment model on the frontend.
      "deleted": 0,
      "data": []
    },
    "fields": [],
    "first_name": null,
    "last_name": "Administrator",
    "created_by_name": " Administrator" This will be the
    locale-formatted full name of the user.
  }, ... ]
}
```

Last Modified: 11/30/2016 09:17pm

/Administration/elasticsearch/indices GET

Overview

[ADMIN] Elasticsearch index statistics

Summary

This endpoint returns the index statistics for the Elasticsearch backend. This endpoint is only available to administrators.

Response

```
{
  "autobr2583_shared": {
    "_shards": {
      "total": 1,
      "successful": 1,
      "failed": 0
    },
    "indices": {
      "autobr2583_shared": {
        "index": {
          "primary_size_in_bytes": 1134148,
          "size_in_bytes": 1134148
        },
        "translog": {
          "operations": 1100
        },
        "docs": {
          "num_docs": 1100,
          "max_doc": 1100,
          "deleted_docs": 0
        },
        "merges": {
          "current": 0,
          "current_docs": 0,
          "current_size_in_bytes": 0,
          "total": 1,
          "total_time_in_millis": 103,
          "total_docs": 1000,
          "total_size_in_bytes": 1138070
        },
        "refresh": {
          "total": 13,
          "total_time_in_millis": 177
        }
      }
    }
  }
}
```

```
    },
    "flush":{
      "total":0,
      "total_time_in_millis":0
    },
    "shards":[[{
      "routing":{
        "state":"STARTED",
        "primary":true,
        "node":"4rjVEVuYQQqKl4sT1FUxNA",
        "relocating_node":null,
        "shard":0,
        "index":"autobr2583_shared"
      },
      "state":"STARTED",
      "index":{
        "size_in_bytes":1134148
      },
      "translog":{
        "id":1427003304006,
        "operations":1100
      },
      "docs":{
        "num_docs":1100,
        "max_doc":1100,
        "deleted_docs":0
      },
      "merges":{
        "current":0,
        "current_docs":0,
        "current_size_in_bytes":0,
        "total":1,
        "total_time_in_millis":103,
        "total_docs":1000,
        "total_size_in_bytes":1138070
      },
      "refresh":{
        "total":13,
        "total_time_in_millis":177
      },
      "flush":{
        "total":0,
        "total_time_in_millis":0
      }
    }
  ]
}
```

```
    }
  }
}
}]]
```

Change Log

| Version | Change |
|---------|---|
| v10 | Added
/Administration/elasticsearch/indices
GET endpoint. |

Last Modified: 05/31/2017 05:27pm

/Administration/elasticsearch/mapping GET

Overview

[ADMIN] Elasticsearch mapping

Summary

This endpoint returns the mapping for every available index. This endpoint is only available to administrators.

Response

```
{
  "autobr2583_shared": {
    "KBDocuments": {
      "dynamic": "false",
      "include_in_all": false,
      "_all": {
        "enabled": false
      },
      "properties": {
```

```

    "kbedocument_name":{
      "type":"string",
      "index":"not_analyzed",
      "fields":{
        "gs_string_default":{
          "type":"string",
          "analyzer":"gs_analyzer_default"
        },
        "gs_string_ngram":{
          "type":"string",
          "index_analyzer":"gs_analyzer_ngram",
          "search_analyzer":"gs_analyzer_default"
        }
      }
    },
    "include_in_all":false
  }
},
"RevenueLineItems":{
  "dynamic":"false",
  "include_in_all":false,
  "_all":{
    "enabled":false
  },
  "properties":{
    "date_modified":{
      "type":"string",
      "index":"not_analyzed",
      "fields":{
        "gs_datetime":{
          "type":"date",
          "index":"no",
          "format":"YYYY-MM-dd HH:mm:ss"
        }
      }
    },
    "include_in_all":false
  },
  "description":{
    "type":"string",
    "index":"not_analyzed",
    "fields":{
      "gs_string_default":{
        "type":"string",
        "analyzer":"gs_analyzer_default"
      }
    }
  }
}

```

```
    },
    "gs_string_ngram":{
      "type":"string",
      "index_analyzer":"gs_analyzer_ngram",
      "search_analyzer":"gs_analyzer_default"
    }
  },
  "include_in_all":false
}
}
}
}
}
```

Change Log

| Version | Change |
|---------|---|
| v10 | Added
/Administration/elasticsearch/mapping
GET endpoint. |

Last Modified: 05/31/2017 05:27pm

/Administration/elasticsearch/queue GET

Overview

[ADMIN] Elasticsearch queue statistics

Summary

This endpoint returns the queue statistics for the Elasticsearch backend. This endpoint is only available to administrators.

Response

```
{
```

```
"total":2238,
"queued":{
  "Cases":"250",
  "KBDocuments":"5",
  "Notes":"50",
  "Quotes":"2",
  "Accounts":"51",
  "Contacts":"201",
  "Leads":"201",
  "Opportunities":"150",
  "RevenueLineItems":"578",
  "Bugs":"50",
  "Contracts":"2",
  "Manufacturers":"2",
  "ProductCategories":"43",
  "Tasks":"200",
  "Calls":"50",
  "Emails":"200",
  "Meetings":"200",
  "Products":"3"
}
```

Change Log

| Version | Change |
|---------|---|
| v10 | Added
/Administration/elasticsearch/queue
GET endpoint. |

Last Modified: 05/31/2017 05:27pm

/Administration/elasticsearch/routing GET

Overview

[ADMIN] Elasticsearch index routing

Summary

This endpoint returns an overview of the current read and write indices per module. This routing information is based on the configured index routing strategies and the index configuration per module. Note that this output shows the default routing only as no context is available to determine dynamic routing strategies. This endpoint is only available to administrators.

Response

```
{
  "Contacts": {
    "strategy": "static",
    "routing": {
      "write_index": "autobr2583_contacts",
      "read_indices": [
        "autobr2583_contacts"
      ]
    }
  },
  "Accounts": {
    "strategy": "static",
    "routing": {
      "write_index": "autobr2583_shared",
      "read_indices": [
        "autobr2583_shared"
      ]
    }
  },
  "Emails": {
    "strategy": "rolling",
    "routing": {
      "write_index": "autobr2583_emails_201503",
      "read_indices": [
        "autobr2583_emails_201503",
        "autobr2583_emails_201502",
        "autobr2583_emails_201501"
      ]
    }
  }
}
```

Change Log

| Version | Change |
|---------|---|
| v10 | Added
/Administration/elasticsearch/routing
GET endpoint. |

Last Modified: 05/31/2017 05:27pm

/Administration/search/fields GET

Overview

[ADMIN] Search field configuration

Summary

This endpoint lists the full text search configuration of all fields for the full text search enabled modules. This endpoint is only available to administrators.

Request Arguments

| Name | Type | Description | Required |
|----------------|---------|--|----------|
| module_list | String | Comma delimited list of modules to return. If omitted, all search enabled modules will be returned.
Example:
Accounts,Contacts | False |
| search_only | Boolean | When set, only searchable fields are returned.
Defaults to false. | False |
| order_by_boost | Boolean | When set, a flat list of searchable fields | False |

| Name | Type | Description | Required |
|------|------|-------------------------------------|----------|
| | | is returned ordered by boost value. | |

Response

```
{
  "Accounts": {
    "name": {
      "name": "name",
      "type": "name",
      "searchable": true,
      "boost": 1
    },
    "date_modified": {
      "name": "date_modified",
      "type": "datetime",
      "searchable": false
    }
  }
}
```

Response using order_by_boost

```
{
  "Quotes.quote_num": 1.5,
  "Manufacturers.name": 1,
  "Bugs.name": 0.9,
  "ProjectTask.name": 0.5,
}
```

Change Log

| Version | Change |
|---------|---|
| v10 | Added /Administration/search/fields GET endpoint. |

/Administration/search/reindex POST

Overview

[ADMIN] Search schedule reindex

Summary

This endpoint schedules a reindex. This endpoint is only available to administrators.

Request Arguments

| Name | Type | Description | Required |
|-------------|---------|---|----------|
| module_list | String | Comma delimited list of modules to be reindexed. If omitted, all search enabled modules will be reindexed. Example: Accounts,Contacts | False |
| clear_data | Boolean | If set the records of the selected modules will be removed from the search backend and the schema will be recreated. All records from given modules will be queued to be reindexed which is orchestrated by the job queue. If not set, nothing is | False |

| Name | Type | Description | Required |
|------|------|--|----------|
| | | dropped from the search backend and all records of the selected modules are queued for reindexing. This is the default behavior.
Example:
Accounts, Contacts | |

Response

```
{
  "success": true
}
```

Change Log

| Version | Change |
|---------|---|
| v10 | Added /Administration/search/reindex POST endpoint. |

Last Modified: 05/31/2017 05:27pm

/Administration/search/status GET

Overview

[ADMIN] Search status

Summary

This endpoint returns the status of the search backend including a list of the activated full text search modules. This endpoint is only available to

administrators.

Response

```
{
  "available":true,
  "enabled_modules":
  [
    "Accounts",
    "Bugs",
    "Calls",
    "Campaigns",
  ]
}
```

Change Log

| Version | Change |
|---------|---|
| v10 | Added /Administration/search/status GET endpoint. |

Last Modified: 05/31/2017 05:27pm

/Calendar/invitee_search GET

Overview

Temporary API - Do Not Use! This endpoint will be removed in an upcoming release. Use /search endpoint instead. Searches for people to invite to an event (meeting or call).

Request Arguments

| Name | Type | Description | Required |
|------|--------|--------------------|----------|
| q | String | The search text to | True |

| Name | Type | Description | Required |
|---------------|--------|---|----------|
| | | match records on. | |
| module_list | String | Comma delimited list of modules to search. | True |
| search_fields | String | Comma delimited list of fields to search. | True |
| fields | String | Comma delimited list of fields to return. Search fields will automatically be added to return list. | True |

Response Arguments

| Name | Type | Description |
|-------------|---------|---|
| next_offset | Integer | Currently only returns -1 as paging is not supported yet. |
| records | Array | An array of results containing matched records. |

Response

```
{
  "next_offset": -1,
  "records": [
    {
      "id": "17283aad-8d15-c545-fc5a-5422fe3d8ef8",
      "date_modified": "2014-09-24T13:25:28-04:00",
      "email": [
        {
          "email_address": "hr.phone.support@example.cn",
          "invalid_email": false,

```

```

        "opt_out": true,
        "primary_address": false,
        "reply_to_address": false
    },
    {
        "email_address":
"section.section.phone@example.edu",
        "invalid_email": false,
        "opt_out": false,
        "primary_address": true,
        "reply_to_address": true
    }
],
"full_name": "Renaldo Cuffee",
"account_name": "EEE Endowments LTD",
"_acl": {
    "fields": {}
},
"_module": "Contacts",
"_search": {
    "highlighted": {
        "account_name": {
            "text": "EEE Endowments LTD",
            "module": "Contacts",
            "label": "LBL_ACCOUNT_NAME"
        }
    }
}
},
{
    "id": "19c0fa4a-a6c0-940f-7ad6-5422fe62a00a",
    "date_modified": "2014-09-24T13:25:28-04:00",
    "email": [
        {
            "email_address": "qa.beans@example.com",
            "invalid_email": false,
            "opt_out": true,
            "primary_address": false,
            "reply_to_address": false
        },
        {
            "email_address": "the.phone.sales@example.de",
            "invalid_email": false,
            "opt_out": false,

```

```

        "primary_address": true,
        "reply_to_address": true
    }
],
"full_name": "Noble Canto",
"account_name": "EEE Endowments LTD",
"_acl": {
    "fields": {}
},
"_module": "Contacts",
"_search": {
    "highlighted": {
        "account_name": {
            "text": "EEE Endowments LTD",
            "module": "Contacts",
            "label": "LBL_ACCOUNT_NAME"
        }
    }
}
}
]
}

```

Change Log

| Version | Change |
|---------|--|
| v10 | Added /<module>/send_invites GET endpoint. |

Last Modified: 05/31/2017 05:30pm

/Calls POST

Overview

Create a single event or a series of event records.

Request Arguments

| Name | Type | Description | Required |
|----------------|---------------------|--|----------|
| <record field> | <record field type> | The name value list of fields to populate. | True |

Request

```
{
  "name": "Department Meeting",
  "description": "Weekly Department Meeting",
  "date_start": "yyyy-mm-ddThh:mm:ss-00:00",
  "date_end": "yyyy-mm-ddThh:mm:ss-00:00",
  "repeat_type": "Weekly",
  "repeat_interval": "1", /* Every Week */
  "repeat_dow": "25", /* Tue and Fri */
  "repeat_count": "4", /* 4 Recurrences */
  "repeat_until": "",
}
```

Response Arguments

| Name | Description | Start Date | End Date |
|----------------|--|------------|----------|
| <record field> | Returns the fields for the newly created record. | | |

Response

```
{
}
```

Change Log

| Version | Change |
|---------|--------------------------------|
| v10 | Added /<module> POST endpoint. |

Last Modified: 05/31/2017 05:31pm

/Calls/:record DELETE

Overview

Deletes either a single event record or a series of event records

Request Arguments

| Name | Type | Description | Required |
|----------------|---------|--|----------|
| all_reurrences | Boolean | Flag to delete all events in a series. | False |

Request

`http://{site_url}/rest/v10/Meetings/2c9e5c09-6824-0d14-f5cb-5130321ac3cf?all_reurrences=true`

Response Arguments

| Name | Type | Description |
|------|--------|---------------------------------------|
| id | String | Returns the ID of the deleted record. |

Response

{

```
    "id": "11cf0d0a-40af-8cb1-9da0-5057a5f511f9"
  }
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added /<module>/:record DELETE endpoint. |

Last Modified: 05/31/2017 05:28pm

/Calls/:record PUT

Overview

Update a calendar event record of the specified type.

Request Arguments

| Name | Type | Description | Required |
|----------------|---------|--|----------|
| all_reurrences | Boolean | Flag to update all events in a series. | False |

Request

```
http://{site_url}/rest/v10/Meetings/2c9e5c09-6824-0d14-f5cb-5130321ac3
cf?all_reurrences=true
{
  "name": "Department Meeting",
  "description": "Weekly Department Meeting",
  "date_start": "yyyy-mm-ddThh:mm:ss-00:00",
  "date_end": "yyyy-mm-ddThh:mm:ss-00:00",
  "repeat_type": "Weekly",
  "repeat_interval": "1", /* Every Week */
  "repeat_dow": "25", /* Tue and Fri */
}
```

```
"repeat_count": "4",          /* 4 Recurrences */
"repeat_until": "",
}
```

Response Arguments

| Name | Description | Start Date | End Date |
|----------------|--|------------|----------|
| <record field> | Returns the fields for the updated record. | | |

Response

```
{
}
```

Change Log

| Version | Change |
|---------|---------------------------------------|
| v10 | Added /<module>/:record PUT endpoint. |

Last Modified: 05/31/2017 05:28pm

/Contacts/:record/freebusy GET

Get a contact's FreeBusy schedule

Summary:

This endpoint returns a list of time slots for which the specified person is busy.

Request

GET /Contacts/:id/freebusy

Response

```
{
  "id": "foo"
  "module": "Users",
  "freebusy": [
    {
      "start": "2014-08-24T08:45:00-04:00",
      "end": "2014-08-24T09:15:00-04:00"
    },
    {
      "start": "2014-08-30T05:45:00-04:00",
      "end": "2014-08-30T06:15:00-04:00"
    },
    {
      "start": "2014-09-12T15:45:00-04:00",
      "end": "2014-09-12T16:15:00-04:00"
    }
  ]
}
```

Last Modified: 05/31/2017 05:54pm

/Dashboards/Activities GET

Gets current user's dashboards.

Summary:

This endpoint returns a list of dashboards.

Query Parameters:

| Param | Type | Description | Required |
|-----------|---------|--|----------|
| filter | String | The filter expression. See the | False |
| filter_id | String | Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND. | False |
| max_num | Integer | Max number of records that can be returned. Default is 20, unless overruled by your Sugar configuration. | False |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |
| fields | String | Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters (applicable to link and collection fields). The fields id and date_modified will always be returned. For more details on additional field parameters, see Relate API and Collection API. | False |
| order_by | String | See Filter API. | False |
| deleted | Boolean | Boolean to show | False |

| | |
|--|------------------------------------|
| | deleted records in the result set. |
|--|------------------------------------|

Request

GET Dashboards/: "offset":0, "limit":3, "order_by":"date_entered desc"

Response

```
{
  "next_offset" : 3
  "records" : [
    {
      id: "1848af4f-bdde-810d-c121-511e7da454c6"
      name: "Dashboard1"
      url: "http://localhost/ult/sugarcrm/rest/v10/Dashboards/1848af4f-bdde-810d-c121-511e7da454c6"
    },
    {
      id: "2c17875e-57f1-a2dd-f9cb-511d7192c65e"
      name: "Dashboard2"
      url: "http://localhost/ult/sugarcrm/rest/v10/Dashboards/2c17875e-57f1-a2dd-f9cb-511d7192c65e"
    },
    {
      id: "bb0e2c83-660a-1fd7-fb69-511e7d6b05bd"
      name: "Dashboard3"
      url: "http://localhost/ult/sugarcrm/rest/v10/Dashboards/bb0e2c83-660a-1fd7-fb69-511e7d6b05bd"
    }
  ]
}
```

Last Modified: 05/31/2017 05:54pm

/Dashboards GET

Gets current user's dashboards.

Summary:

This endpoint returns a list of dashboards.

Query Parameters:

| Param | Type | Description | Required |
|--------|--------|--------------------------------|----------|
| filter | String | The filter expression. See the | False |

| | | | |
|-----------|---------|--|-------|
| filter_id | String | Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND. | False |
| max_num | Integer | Max number of records that can be returned. Default is 20, unless overruled by your Sugar configuration. | False |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |
| fields | String | Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters (applicable to link and collection fields). The fields id and date_modified will always be returned. For more details on additional field parameters, see Relate API and Collection API. | False |
| order_by | String | See Filter API. | False |
| deleted | Boolean | Boolean to show deleted records in the result set. | False |

Request

GET Dashboards/: "offset":0, "limit":3, "order_by":"date_entered desc"

Response

```
{  "next_offset" : 3  "records" :      [  {  id:
"1848af4f-bdde-810d-c121-511e7da454c6"      name: "Dashboard1"
url:
"http://localhost/ult/sugarcrm/rest/v10/Dashboards/1848af4f-bdde-810d-c121-511e
7da454c6"      },  {  id:
"2c17875e-57f1-a2dd-f9cb-511d7192c65e"      name: "Dashboard2"
url:
"http://localhost/ult/sugarcrm/rest/v10/Dashboards/2c17875e-57f1-a2dd-f9cb-511d
7192c65e"      },  {  id:
"bb0e2c83-660a-1fd7-fb69-511e7d6b05bd"      name: "Dashboard3"
url:
"http://localhost/ult/sugarcrm/rest/v10/Dashboards/bb0e2c83-660a-1fd7-fb69-511e
7d6b05bd"      }  ]  }
```

Last Modified: 05/31/2017 05:54pm

/Dashboards POST

Create a new dashboard.

Summary:

This endpoint creates a dashboard.

Request Arguments

| Param | Description | Optional |
|---|-------------|----------|
| This endpoint does not accept any parameters. | | |

Request

```
{  "name":"Test Dashboard",  "metadata":{ }, }
```

Response

```
{  "id": "5526df55-d3b5-ccd6-59b1-5058a5b5492c",  "name": "Test Dashboard",  "date_entered": "2012-09-18T09:46:00-0700",  "date_modified": "2012-09-18T09:46:00-0700",  "modified_user_id": "1",  "modified_by_name": "Administrator",  "created_by": "1",  "created_by_name": "Administrator",  "description": "",  "deleted": "0",  "assigned_user_id": "1",  "assigned_user_name": "Administrator",  "metadata":{ },  "module": "Home", }
```

Last Modified: 11/30/2016 09:16pm

/Dashboards/<module> GET

Gets current user's dashboards.

Summary:

This endpoint returns a list of dashboards.

Query Parameters:

| Param | Type | Description | Required |
|-----------|---------|--|----------|
| filter | String | The filter expression. See the | False |
| filter_id | String | Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND. | False |
| max_num | Integer | Max number of records that can be returned. Default is 20, unless overruled by your Sugar configuration. | False |

| | | | |
|----------|---------|--|-------|
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |
| fields | String | Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters (applicable to link and collection fields). The fields id and date_modified will always be returned. For more details on additional field parameters, see Relate API and Collection API. | False |
| order_by | String | See Filter API. | False |
| deleted | Boolean | Boolean to show deleted records in the result set. | False |

Request

GET Dashboards/: "offset":0, "limit":3, "order_by":"date_entered desc"

Response

```
{
  "next_offset" : 3
  "records" : [
    {
      id:
      "1848af4f-bdde-810d-c121-511e7da454c6"
      name: "Dashboard1"
      url:
      "http://localhost/ult/sugarcrm/rest/v10/Dashboards/1848af4f-bdde-810d-c121-511e7da454c6"
    },
    {
      id:
```

```
"2c17875e-57f1-a2dd-f9cb-511d7192c65e"           name: "Dashboard2"
url:
"http://localhost/ult/sugarcrm/rest/v10/Dashboards/2c17875e-57f1-a2dd-f9cb-511d
7192c65e"           },           {           id:
"bb0e2c83-660a-1fd7-fb69-511e7d6b05bd"           name: "Dashboard3"
url:
"http://localhost/ult/sugarcrm/rest/v10/Dashboards/bb0e2c83-660a-1fd7-fb69-511e
7d6b05bd"           }           ]           }
```

Last Modified: 05/31/2017 05:54pm

/Dashboards/<module> POST

Create a new dashboard.

Summary:

This endpoint creates a dashboard.

Request Arguments

| Param | Description | Optional |
|---|-------------|----------|
| This endpoint does not accept any parameters. | | |

Request

```
{  "name":"Test Dashboard",  "metadata":{ }, }
```

Response

```
{  "id": "5526df55-d3b5-ccd6-59b1-5058a5b5492c",  "name": "Test Dashboard",  "date_entered": "2012-09-18T09:46:00-0700",  "date_modified": "2012-09-18T09:46:00-0700",  "modified_user_id": "1",  "modified_by_name": "Administrator",  "created_by": "1",  "created_by_name": "Administrator",  "description": "",  "deleted": "0",  "assigned_user_id": "1",  "assigned_user_name": "Administrator",  "metadata":{ },  "module": "Home", }
```

Last Modified: 11/30/2016 09:18pm

/Documents/:record/file/:field POST

Overview

Attaches a file to a field on a record.

Request Arguments

| Name | Type | Description | Required |
|--------------------|---------|--|--|
| format | String | The data format. Currently accepts 'sugar-html-json'. | True |
| delete_if_fails | Boolean | Indicates whether the API is to mark related record deleted if the file upload fails. | False |
| oauth_token | String | The oauth_token value. | False - Required if only if delete_if_fails is true. |
| <attachment field> | String | The field and file to populate.
Example:
{": "@\\path\\to\\ExampleDocument.txt"} | True |

Request

```
{  
  "format": "sugar-html-json",  
  "delete_if_fails": true,  
  "oauth_token": "43b6b327-cc70-c301-3299-512ffb99ad97",  
  "<attachment field>": "@\\path\\to\\ExampleDocument.txt"  
}
```

Response Arguments

| Name | Type | Description |
|----------------|---------|---------------------------|
| content-type | String | The files content type. |
| content-length | Integer | The files content length. |
| name | String | The files name. |
| uri | String | The URI of the file. |

Response

```
{
  "content-type": "text/plain",
  "content-length": 16,
  "name": "ExampleDocument.txt",

  "uri": "http://sugarcrm/rest/v10/Notes/ca66c92f-5a8b-28b4-4fc8-512d099b790b/file/<attachmentfield>?format=sugar-html-json&delete_if_fails=1&oauth_token=6ec91cf3-1f97-25b9-e0b1-512f8971b2d4"
}
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added /<module>/:record/file/:field POST endpoint. |

Last Modified: 05/31/2017 05:25pm

/Documents/:record/file/:field PUT

Overview

This endpoint takes a file or image and saves it to a record that already contains an attachment in the specified field. The PUT method is very similar to the POST method but differs slightly in how the request is constructed. PUT requests should send the file data as the body of the request. Optionally, a filename query parameter can be sent with the request to assign a name. Additionally, the PUT method can accept base64 encoded file data which will be decoded by the endpoint if the `content_transfer_encoding` parameter is set to 'base64'.

NOTE: In lieu of the `content_transfer_encoding` parameter, a request header of `X-Content-Transfer-Encoding` can also be sent with a value of 'base64'. In the event both the content transfer encoding header and request parameter are sent, the header will be used.

Request Arguments

| Name | Type | Description | Required |
|---------------------------|--------|---|----------|
| filename | String | Filename to save the document as | False |
| content_transfer_encoding | String | When set to 'base64', indicates the file contents are base64 encoded and will result in the file contents being base64 decoded before being saved | False |

Request

PUT /rest/v10/Notes/abcd-1234/file/field/ HTTP/1.1 Host: localhost Connection: keep-alive Content-Length: 23456 Content-Type: application/document-doc ...This is where the bin data would be

Response Arguments

| Name | Type | Description |
|-------|-------|----------------------------|
| field | Array | The fields containing file |

| Name | Type | Description |
|----------------------|---------|---------------------------|
| | | properties. |
| field.content-type | String | The files content type. |
| field.content-length | Integer | The files content length. |
| field.name | String | The files name. |
| field.width | Integer | The width of the image. |
| field.height | Integer | The height of the image. |
| field.uri | String | The URI of the file. |

Response

```
{
  "picture": {
    "content-type": "image/png",
    "content-length": 72512,
    "name": "1a7b8f5c-b11c-0094-c8d8-512e9daaa983",
    "width": 933,
    "height": 519,

"uri": "http://sugarcrm/rest/v10/Contacts/fa300a0e-0ad1-b322-9601-512d0983c19a/file/picture"
  }
  "attachment": {
    "content-type": "application/document-doc",
    "content-length": "873921",
    "name": "myFile.doc",
    "uri":
"http://sugarcrm/rest/v10/Contacts/f2f9aa4d-99a8-e86e-f4d5-512d0986effa/file/attachment"
  }
}
```

Change Log

| Version | Change |
|---------|---|
| v10 | Added /<module>/:record/file/:field PUT |

endpoint.

Last Modified: 05/31/2017 05:25pm

/EmbeddedFiles/:record/file/:field GET

Overview

Retrieves an attached file for a specific field on a record.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

The response from this request is an HTTP response with a forced download. This can be used in rendering images through the API or in offering immediate file downloads.

Response

Cache-Control: no-cache, must-revalidate Connection: keep-alive
Content-Encoding: gzip Content-Type: application/octet-stream Date:
Mon, 30 Jan 2012 17:00:46 GMT Expires: Mon, 30 Jan 2012 17:00:45 GMT
Last-Modified: Thu, 10 Nov 2011 19:01:31 GMT Transfer-Encoding:
chunked Vary: Accept-Encoding...

Change Log

| Version | Change |
|---------|---|
| v10 | Added /<module>/:record/file/:field GET endpoint. |

Last Modified: 05/31/2017 05:26pm

/EmbeddedFiles/:record/file/:field POST

Overview

Attaches a file to a field on a record.

Request Arguments

| Name | Type | Description | Required |
|--------------------|---------|--|--|
| format | String | The data format. Currently accepts 'sugar-html-json'. | True |
| delete_if_fails | Boolean | Indicates whether the API is to mark related record deleted if the file upload fails. | False |
| oauth_token | String | The oauth_token value. | False - Required if only if delete_if_fails is true. |
| <attachment field> | String | The field and file to populate.
Example:
{":": "@\\path\\to\\ExampleDocument.txt"} | True |

Request

```
{  
  "format": "sugar-html-json",  
  "delete_if_fails": true,  
  "oauth_token": "43b6b327-cc70-c301-3299-512ffb99ad97",  
  "<attachment field>": "@\\path\\to\\ExampleDocument.txt"  
}
```

Response Arguments

| Name | Type | Description |
|----------------|---------|---------------------------|
| content-type | String | The files content type. |
| content-length | Integer | The files content length. |
| name | String | The files name. |
| uri | String | The URI of the file. |

Response

```
{
  "content-type": "text/plain",
  "content-length": 16,
  "name": "ExampleDocument.txt",

  "uri": "http://sugarcrm/rest/v10/Notes/ca66c92f-5a8b-28b4-4fc8-512d099b790b/file/<attachment
field>?format=sugar-html-json&delete_if_fails=1&oauth_token=6ec91cf3-1f97-25b9-e0b1-512f8971b2d4"
}
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added /<module>/:record/file/:field POST endpoint. |

Last Modified: 05/31/2017 05:25pm

/EmbeddedFiles/:record/file/:field PUT

Overview

This endpoint takes a file or image and saves it to a record that already contains an attachment in the specified field. The PUT method is very similar to the POST method but differs slightly in how the request is constructed. PUT requests should send the file data as the body of the request. Optionally, a filename query parameter can be sent with the request to assign a name. Additionally, the PUT method can accept base64 encoded file data which will be decoded by the endpoint if the `content_transfer_encoding` parameter is set to 'base64'.

NOTE: In lieu of the `content_transfer_encoding` parameter, a request header of `X-Content-Transfer-Encoding` can also be sent with a value of 'base64'. In the event both the content transfer encoding header and request parameter are sent, the header will be used.

Request Arguments

| Name | Type | Description | Required |
|---------------------------|--------|---|----------|
| filename | String | Filename to save the document as | False |
| content_transfer_encoding | String | When set to 'base64', indicates the file contents are base64 encoded and will result in the file contents being base64 decoded before being saved | False |

Request

PUT /rest/v10/Notes/abcd-1234/file/field/ HTTP/1.1 Host: localhost Connection: keep-alive Content-Length: 23456 Content-Type: application/document-doc ...This is where the bin data would be

Response Arguments

| Name | Type | Description |
|-------|-------|----------------------------|
| field | Array | The fields containing file |

| Name | Type | Description |
|----------------------|---------|---------------------------|
| | | properties. |
| field.content-type | String | The files content type. |
| field.content-length | Integer | The files content length. |
| field.name | String | The files name. |
| field.width | Integer | The width of the image. |
| field.height | Integer | The height of the image. |
| field.uri | String | The URI of the file. |

Response

```
{
  "picture": {
    "content-type": "image/png",
    "content-length": 72512,
    "name": "1a7b8f5c-b11c-0094-c8d8-512e9daaa983",
    "width": 933,
    "height": 519,

"uri": "http://sugarcrm/rest/v10/Contacts/fa300a0e-0ad1-b322-9601-512d0983c19a/file/picture"
  }
  "attachment": {
    "content-type": "application/document-doc",
    "content-length": "873921",
    "name": "myFile.doc",
    "uri":
"http://sugarcrm/rest/v10/Contacts/f2f9aa4d-99a8-e86e-f4d5-512d0986effa/file/attachment"
  }
}
```

Change Log

| Version | Change |
|---------|---|
| v10 | Added /<module>/:record/file/:field PUT |

endpoint.

Last Modified: 05/31/2017 05:25pm

/Employees GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

| Name | Type | Description | Required |
|--------|--------|--|----------|
| filter | String | The filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query parameters in one of two ways for GET requests: <ol style="list-style-type: none">1. By specifying individual filter | False |

| Name | Type | Description | Required |
|-----------|---------|--|----------|
| | | <p>arguments as distinct parameters. Example: filter[0][id]=1. By specifying the whole filter as a single JSON-encoded string. Note that this syntax is currently not supported on certain modules. Example: filter=[{"id": "1"}].</p> | |
| filter_id | String | Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND. | False |
| max_num | Integer | A maximum number of records to return. Default is 20. | False |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |
| fields | String | Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and additional field | False |

| | | | |
|----------|--------|---|-------|
| | | <p>parameters (applicable to link and collection fields). The fields id and date_modified will always be returned.</p> <p>Example:
 name,account_type,description,{ "name": "opportunities", "fields": ["id", "name", "sales_status"], "order_by": "date_closed: desc" }</p> <p>For more details on additional field parameters, see Relate API and Collection API.</p> | |
| view | String | <p>Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list".</p> <p>Example: record</p> | False |
| order_by | String | <p>How to sort the returned records, in a comma delimited list with the direction appended to the column name</p> | False |

| | | | |
|---------|---------|--|-------|
| | | after a colon.
Example:
name:DESC,account_type:DESC,date_modified:ASC | |
| q | String | A search expression, will search on this module. Cannot be used at the same time as a filter expression or id. | False |
| deleted | Boolean | Boolean to show deleted records in the result set. | False |

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
  "filter":[
    {
      "name":"Nelson Inc"
    }
  ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
  "filter": [
    {
      "name": {
        "$starts": "Nelson"
      }
    }
  ]
}
```

Below is a list of operation types:

| Operation | Description |
|--------------|---|
| \$equals | Performs an exact match on that field. |
| \$not_equals | Performs an exact match on that field. |
| \$not_equals | Matches on non-matching values. |
| \$starts | Matches on anything that starts with the value. |
| \$ends | Matches anything that ends with the value. |
| \$contains | Matches anything that contains the value |
| \$in | Finds anything where field matches one of the values as specified as an array. |
| \$not_in | Finds anything where field does not matches any of the values as specified as an array. |
| \$is_null | Checks if the field is null. This operation does not need a value specified. |
| \$not_null | Checks if the field is not null. This operation does not need a value specified. |
| \$lt | Matches when the field is less than the value. |
| \$lte | Matches when the field is less than or equal to the value. |
| \$gt | Matches when the field is greater than the value. |

| Operation | Description |
|-----------|---|
| \$gte | Matches when the field is greater than or equal to the value. |

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "\$and" and "\$or".

Example

```
{
  "filter": [
    {
      "$or": [
        {
          "name": "Nelson Inc"
        },
        {
          "name": "Nelson LLC"
        }
      ]
    }
  ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "\$favorite" and "\$owner".

Example

```
{
```

```

"filter":[
  {
    "$favorite": "_this"
  }
]
}

```

Response Arguments

| Name | Type | Description |
|-------------|---------|---|
| next_offset | Integer | Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records. |
| records | Array | An array of results containing matched records. |

Response

```

{
  "next_offset":-1,
  "records":[
    {
      "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
      "name":"Dale Spivey",
      "date_modified":"2013-02-28T05:03:00+00:00",
      "description":"",
      "opportunities": [
        {
          _module: "Opportunities",
          "id": "b0701501-1fab-8ae7-3942-540da93f5017",
          "name": "360 Vacations - 228 Units",
          "date_modified": "2014-09-08T16:05:00+03:00",
          "sales_status": "New"
        },
      ],
      "_acl": {

```

```

        "fields": {
        }
    },
    {
        "id": "95e17367-9b3d-0e26-22dc-512d0961fedf",
        "name": "Florence Haddock",
        "date_modified": "2013-02-26T19:12:00+00:00",
        "description": "",
        "opportunities": [
            {
                _module: "Opportunities"
                date_modified: "2014-09-08T16:05:00+03:00"
                id: "9ce7c088-8ee4-7cd3-18f1-540da944d4c0"
                name: "360 Vacations - 312 Units"
                sales_status: "New"
            },
        ],
        "_acl": {
            "fields": {
            }
        }
    }
]
}

```

Change Log

| Version | Change |
|---------|--------------------------------------|
| v10 | Added /<module>/filter GET endpoint. |

Last Modified: 05/31/2017 05:31pm

/ExpressionEngine/:record/related GET

Retrieve a Forecasting Information In SugarChart Format

Summary:

This endpoint is used to return the json Data for SugarCharts to use to display the needed chart.

Query Parameters:

| Param | Description | Optional |
|-----------------|--|----------|
| timeperiod_id | Show for a specific time period | |
| user_id | Show for a specific user | |
| display_manager | Pipeline or Committed are valid values. | |
| dataset | Which Forecast dataset to show, valid values are likely, best, worst. Defaults to likely if one is not specified | Optional |
| group_by | Show Which fields the y-axis shows on the chart. Can be any field in the opportunity module, defaults to Sales Stage | Optional |
| ranges | Pipeline or Committed are valid values. | Optional |

Input Example:

```
{  'user_id':'seed_max_id',
'timeperiod_id':'36f7085a-5889-ea75-84c8-50f42bd1a5ba',
'display_manager':'false',  'group_by': 'forecast',  'dataset': 'likely',  'ranges':
'include', }
```

Output Example:

```
{ "color" : [ "#8c2b2b",          "#468c2b",          "#2b5d8c",
"#cd5200",          "#e6bf00",          "#7f3acd",          "#00a9b8",
"#572323",          "#004d00",          "#000087",          "#e48d30",
"#9fba09",          "#560066",          "#009f92",          "#b36262",
"#38795c",          "#3D3D99",          "#99623d",          "#998a3d",
"#994e78",          "#3d6899",          "#CC0000",          "#00CC00",
```

"link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

| Name | Type | Description | Required |
|-----------|--------|---|----------|
| filter | String | The filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query parameters in one of two ways for GET requests: <ol style="list-style-type: none">1. By specifying individual filter arguments as distinct parameters. Example: filter[0][id]=1.2. By specifying the whole filter as a single JSON-encoded string. Note that this syntax is currently not supported on certain modules. Example: filter=[{"id": "1"}]. | False |
| filter_id | String | Identifier for a preexisting filter. If | False |

| Name | Type | Description | Required |
|---------|---------|---|----------|
| | | filter is also set, the two filters are joined with an AND. | |
| max_num | Integer | A maximum number of records to return. Default is 20. | False |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |
| fields | String | Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters (applicable to link and collection fields). The fields id and date_modified will always be returned.
Example:
name,account_type,description,{"name":"opportunities","fields":["id","name","sales_status"],"order_by":"date_closed:desc"}
For more details on additional field parameters, see Relate API and Collection API. | False |
| view | String | Instead of defining the fields | False |

| | | | |
|----------|---------|--|-------|
| | | argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list".
Example: record | |
| order_by | String | How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon.
Example:
name:DESC,account_type:DESC,date_modified:ASC | False |
| q | String | A search expression, will search on this module. Cannot be used at the same time as a filter expression or id. | False |
| deleted | Boolean | Boolean to show deleted records in the result set. | False |

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
  "filter":[
    {
      "name":"Nelson Inc"
    }
  ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
  "filter":[
    {
      "name":{
        "$starts":"Nelson"
      }
    }
  ]
}
```

Below is a list of operation types:

| Operation | Description |
|---------------------------|---|
| <code>\$equals</code> | Performs an exact match on that field. |
| <code>\$not_equals</code> | Performs an exact match on that field. |
| <code>\$not_equals</code> | Matches on non-matching values. |
| <code>\$starts</code> | Matches on anything that starts with the value. |

| Operation | Description |
|------------|---|
| \$ends | Matches anything that ends with the value. |
| \$contains | Matches anything that contains the value |
| \$in | Finds anything where field matches one of the values as specified as an array. |
| \$not_in | Finds anything where field does not matches any of the values as specified as an array. |
| \$is_null | Checks if the field is null. This operation does not need a value specified. |
| \$not_null | Checks if the field is not null. This operation does not need a value specified. |
| \$lt | Matches when the field is less than the value. |
| \$lte | Matches when the field is less than or equal to the value. |
| \$gt | Matches when the field is greater than the value. |
| \$gte | Matches when the field is greater than or equal to the value. |

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "\$and" and "\$or".

Example

```
{
  "filter": [
    {
      "$or": [
        {
```

```

        "name": "Nelson Inc"
      },
      {
        "name": "Nelson LLC"
      }
    ]
  }
]
}

```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "\$favorite" and "\$owner".

Example

```

{
  "filter": [
    {
      "$favorite": "_this"
    }
  ]
}

```

Response Arguments

| Name | Type | Description |
|-------------|---------|---|
| next_offset | Integer | Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records. |
| records | Array | An array of results containing matched |

| Name | Type | Description |
|------|------|-------------|
| | | records. |

Response

```
{
  "next_offset": -1,
  "records": [
    {
      "id": "fa300a0e-0ad1-b322-9601-512d0983c19a",
      "name": "Dale Spivey",
      "date_modified": "2013-02-28T05:03:00+00:00",
      "description": "",
      "opportunities": [
        {
          _module: "Opportunities",
          "id": "b0701501-1fab-8ae7-3942-540da93f5017",
          "name": "360 Vacations - 228 Units",
          "date_modified": "2014-09-08T16:05:00+03:00",
          "sales_status": "New"
        }
      ],
      "_acl": {
        "fields": {
        }
      }
    }
  ],
  {
    "id": "95e17367-9b3d-0e26-22dc-512d0961fedf",
    "name": "Florence Haddock",
    "date_modified": "2013-02-26T19:12:00+00:00",
    "description": "",
    "opportunities": [
      {
        _module: "Opportunities"
        date_modified: "2014-09-08T16:05:00+03:00"
        id: "9ce7c088-8ee4-7cd3-18f1-540da944d4c0"
        name: "360 Vacations - 312 Units"
        sales_status: "New"
      }
    ],
    "_acl": {
  }
```

```
        "fields": {
          }
        }
      ]
    }
```

Change Log

| Version | Change |
|---------|--------------------------------------|
| v10 | Added /<module>/filter GET endpoint. |

Last Modified: 05/31/2017 05:31pm

/ForecastManagerWorksheets GET

Returns a collection of ForecastManagerWorksheet models

Summary:

This end point is used to return the ManagerWorksheets for a user at a given timeperiod. If no timeperiod is provide it will use the default timeperiod, likewise if no user is provided, it will default to the logged in user.

Url Parameters:

| Param | Description | Optional |
|---------------|---|----------|
| timeperiod_id | Show for a specific time period, defaults to the current time period if one is not passed | Optional |
| user_id | Show for a specific user, defaults to current user if not defined | Optional |

Possible Errors

| Error | Description |
|-------------------------|--|
| 412 - Invalid Parameter | When the passed in timeperiod and/or user can not be found |
| 403 - Not Authorized | If you are not an administrator, but you are trying to view another managers forecast manager worksheet, you will receive a 403 Not Authorized Error |

Url Example:

/rest/v10/ForecastManagerWorksheets/:timeperiod_id/:user_id

Output Example:

```
{  "next_offset":-1,  "records":[  {    "id":"401594f5-5b46-fb66-1627-55771fe8723e",    "name":"Sally Bronsen",    "date_modified":"2015-06-09T13:13:26-04:00",    "created_by":"1",    "quota":"1932.444445",    "best_case":"29848.000000",    "best_case_adjusted":"37310.000000",    "likely_case":"28694.444445",    "likely_case_adjusted":"35868.055556",    "worst_case":"27540.888889",    "worst_case_adjusted":"34426.111111",    "timeperiod_id":"adb78e81-3fbd-b4e0-287f-55771fd04a06",    "draft":true,    "is_manager":false,    "user_id":"seed_sally_id",    "opp_count":10,    "pipeline_opp_count":3,    "pipeline_amount":"2415.555556",    "closed_amount":"26278.888889",    "manager_saved":true,    "show_history_log":0,    "following":false,    "assigned_user_id":"seed_sarah_id",    "assigned_user_name":"Sarah Smith",    "team_name":[      {        "id":"1",        "name":"Global",        "name_2":"","primary":true      }    ],    "currency_id":"-99",    "base_rate":"1.000000",    "_acl":{"fields":{"_module":"ForecastManagerWorksheets"}},    {      "id":"28226f12-a3c9-bd84-041e-55771f064c4e",      "name":"Max Jensen",      "date_modified":"2015-06-09T13:13:26-04:00",      "created_by":"1",      "quota":"3141.333332",      "best_case":"15848.222223",      "best_case_adjusted":"13206.851853",      "likely_case":"14928.222222",      "likely_case_adjusted":"12440.185185",      "worst_case":"14008.222223",      "worst_case_adjusted":"11673.518519",      "timeperiod_id":"adb78e81-3fbd-b4e0-287f-55771fd04a06",      "draft":true,      "is_manager":false,      "user_id":"seed_max_id",      "opp_count":12,
```



```

"pipeline_opp_count":3,          "pipeline_amount":"2617.777777",
"closed_amount":"12310.444445", "manager_saved":true,
"show_history_log":0,          "following":false,
"assigned_user_id":"seed_sarah_id", "assigned_user_name":"Sarah Smith",
"team_name":[                  { "id":"1", "name":"Global",
"name_2":""," "primary":true } ],
"currency_id":"-99",          "base_rate":"1.000000", "_acl":{
"fields":{                    } }, "_module":"ForecastManagerWorksheets"
}, { "id":"1aca66af-f069-bba4-28a1-55771fe27506",
"name":""," "date_modified":"2015-06-09T13:13:26-04:00",
"created_by":"1", "quota":"10142.333333",
"best_case":"37625.000000", "best_case_adjusted":"37625.000000",
"likely_case":"34241.333333", "likely_case_adjusted":"34241.333333",
"worst_case":"30857.666667", "worst_case_adjusted":"30857.666667",
"timeperiod_id":"adb78e81-3fbd-b4e0-287f-55771fd04a06", "draft":true,
"is_manager":true, "user_id":"seed_sarah_id", "opp_count":13,
"pipeline_opp_count":6, "pipeline_amount":"10142.333333",
"closed_amount":"24099.000000", "manager_saved":true,
"show_history_log":0, "following":false,
"assigned_user_id":"seed_sarah_id", "assigned_user_name":"Sarah Smith",
"team_name":[                  { "id":"1", "name":"Global",
"name_2":""," "primary":true } ],
"currency_id":"-99",          "base_rate":"1.000000", "_acl":{
"fields":{                    } }, "_module":"ForecastManagerWorksheets"
} ] ]

```

Last Modified: 05/31/2017 05:54pm

/ForecastManagerWorksheets/assignQuota POST

Assign Quotas to All Reporting Users for a given timeperiod

Summary:

This endpoint is used to assign out the quota to the reporting users with out having to commit a forecast.

Parameters:

| Param | Description | Optional |
|-------|-------------|----------|
|-------|-------------|----------|

| | | |
|---------------|--|-------|
| user_id | The Manager's user id for who is assigning out the quota | false |
| timeperiod_id | Which timeperiod are we assigning quota for | false |

Input Example:

```
{ "user_id" : "seed_sarah_id",      "timeperiod_id" :
"36f7085a-5889-ea75-84c8-50f42bd1a5ba" }
```

Output Example:

```
{success: true}
```

Last Modified: 05/31/2017 05:54pm

/ForecastManagerWorksheets/export GET

Overview

Returns a record set in CSV format along with HTTP headers to indicate content type.

Request Arguments

| Name | Type | Description | Required |
|--------|-------|---|----------|
| uid | Array | A list of bean ids. | False |
| filter | Array | The filter expression. More information on filter expressions can be found in /<module>/filter. | False |
| sample | Array | Indicates whether to download sample data. | False |

| Name | Type | Description | Required |
|------|------|-------------|----------|
|------|------|-------------|----------|

Request

Exporting Records by Specific IDs

```
{
  "uid": "d43243c6-9b8e-2973-ae2-512d09bc34b4"
}
```

Exporting Records by a List of IDs

```
{
  "uid": [
    "d43243c6-9b8e-2973-ae2-512d09bc34b4",
    "b3e87a3f-cd8f-7b86-467a-512d09e8d240"
  ]
}
```

Exporting Records Using a Filter

```
{
  "filter": [
    {
      "name": "airline"
    }
  ]
}
```

Exporting a Sample Result Set

```
{
  "sample": true
}
```

Response Arguments

| Name | Type | Description |
|--------------|--------|--|
| <csv export> | String | Returns the selected records in CSV format with the content headers. |

Response

result: HTTP/1.1 200 OK
Date: Fri, 01 Mar 2013 04:05:55 GMT
Server: Apache/2.2.22 (Unix) DAV/2 PHP/5.3.17
X-Powered-By: PHP/5.3.17
Expires: Mon, 26 Jul 1997 05:00:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: cache
Content-Disposition: attachment; filename=Accounts.csv
Content-transfer-encoding: binary
Last-Modified: Fri, 01 Mar 2013 04:05:55 GMT
Cache-Control: post-check=0, pre-check=0
Content-Length: 1080
Connection: close
Content-Type: application/octet-stream; charset=UTF-8

"Name","ID","Website","Email Address","Office Phone","Alternate Phone","Fax","Billing Street","Billing City","Billing State","Billing Postal Code","Billing Country","Shipping Street","Shipping City","Shipping State","Shipping Postal Code","Shipping Country","Description","Type","Industry","Annual Revenue","Employees","SIC Code","Ticker Symbol","Parent Account ID","Ownership","Campaign ID","Rating","Assigned User Name","Assigned To","Team ID","Teams","Team Set ID","Date Created","Date Modified","Modified By","Created By","Deleted","Image","last_activity_date","Linkedin Company ID","Facebook Account","Twitter Account","Google Plus ID"
"Arts & Crafts
Inc","d43243c6-9b8e-2973-ae2-512d09bc34b4","","","(052)
034-1853","","","777 West Filmore Ln","Santa
Monica","CA","35354","USA","777 West Filmore Ln","Santa

Monica", "CA", "35354", "USA", "", "Customer", "Transportation", "", "", "", "",
"", "", "", "", "sally", "seed_sally_id", "West", "West", "West", "02/26/2013
07:12 pm", "02/26/2013 07:12 pm", "1", "1", "0", "", "02/26/2013 07:12
pm", "", "", "", ""

Change Log

| Version | Change |
|---------|--------------------------------------|
| v10 | Added /<module>/export GET endpoint. |

Last Modified: 05/31/2017 05:30pm

/ForecastManagerWorksheets/filter GET

Returns a collection of ForecastManagerWorksheet models

Summary:

This end point is used to return selective ManagerWorksheets with the optional filter parameter.

Query Parameters:

| Param | Description | Optional |
|--------|----------------------------|----------|
| filter | What you want to filter on | Optional |

Possible Errors

| Error | Description |
|-------------------------|--|
| 412 - Invalid Parameter | When the passed in timeperiod and/or user can not be found |
| 403 - Not Authorized | If you are not an administrator, but you |

are trying to view another managers forecast manager worksheet, you will receive a 403 Not Authorized Error

Url Example:

```
/rest/v10/ForecastManagerWorksheets/filter { "filter":[ {"assigned_user_id" : "seed_jim_id"}, {"timeperiod_id":"e546185a-5889-ea75-84c8-511178d1a5ba"} ], }
```

Output Example:

```
{ "next_offset": -1, "records": [ { "id": "ad3908c7-83a3-f360-c223-51117844c208", "name": "Grow-Fast Inc - 1000 units", "date_entered": "2013-02-05T21:22:00+00:00", "date_modified": "2013-02-05T21:22:00+00:00", "modified_user_id": "1", "modified_by_name": "Administrator", "created_by": "1", "created_by_name": "Administrator", "description": "", "img": "", "deleted": "0", "assigned_user_id": "seed_jim_id", "assigned_user_name": "Jim Brennan", "team_name": [ { "id": "East", "name": "East", "name_2": "", "primary": true } ], "parent_id": "50b90565-e748-ed77-d9d7-511178f5acae", "parent_type": "Opportunities", "account_name": "", "account_id": "", "likely_case": "75000", "best_case": "75000", "worst_case": "75000", "base_rate": "1", "currency_id": "-99", "currency_name": "", "currency_symbol": "", "date_closed": "2013-02-10", "date_closed_timestamp": "1360531443", "sales_stage": "Perception Analysis", "probability": "70", "commit_stage": "include", "draft": "1", "my_favorite": false, "_acl": { "fields": {} } } ] }
```

Last Modified: 05/31/2017 05:54pm

/ForecastManagerWorksheets/filter POST

Returns a collection of ForecastManagerWorksheet models

Summary:

This end point is used to return selective ManagerWorksheets with the optional filter parameter.

Query Parameters:

| Param | Description | Optional |
|--------|----------------------------|----------|
| filter | What you want to filter on | Optional |

Possible Errors

| Error | Description |
|-------------------------|--|
| 412 - Invalid Parameter | When the passed in timeperiod and/or user can not be found |
| 403 - Not Authorized | If you are not an administrator, but you are trying to view another managers forecast manager worksheet, you will receive a 403 Not Authorized Error |

Url Example:

```
/rest/v10/ForecastManagerWorksheets/filter { "filter":[ {"assigned_user_id" :  
"seed_jim_id"}, {"timeperiod_id":"e546185a-5889-ea75-84c8-511178d1a5ba"}  
], }
```

Output Example:

```
{ "next_offset": -1, "records": [ { "id":  
"ad3908c7-83a3-f360-c223-51117844c208", "name": "Grow-Fast Inc -  
1000 units", "date_entered": "2013-02-05T21:22:00+00:00",  
"date_modified": "2013-02-05T21:22:00+00:00", "modified_user_id": "1",  
"modified_by_name": "Administrator", "created_by": "1",  
"created_by_name": "Administrator", "description": "", "img":  
"", "deleted": "0", "assigned_user_id": "seed_jim_id",  
"assigned_user_name": "Jim Brennan", "team_name": [ {  
"id": "East", "name": "East", "name_2": "",  
"primary": true } ], "parent_id":
```

```

"50b90565-e748-ed77-d9d7-511178f5acae",          "parent_type":
"Opportunities",          "account_name": "",          "account_id": "",
"likely_case": "75000",          "best_case": "75000",          "worst_case":
"75000",          "base_rate": "1",          "currency_id": "-99",
"currency_name": "",          "currency_symbol": "",          "date_closed":
"2013-02-10",          "date_closed_timestamp": "1360531443",
"sales_stage": "Perception Analysis",          "probability": "70",
"commit_stage": "include",          "draft": "1",          "my_favorite": false,
"_acl": {          "fields": { }          }          }

```

Last Modified: 05/31/2017 05:54pm

/ForecastManagerWorksheets/:timeperiod_id GET

Returns a collection of ForecastManagerWorksheet models

Summary:

This end point is used to return the ManagerWorksheets for a user at a given timeperiod. If no timeperiod is provide it will use the default timeperiod, likewise if no user is provided, it will default to the logged in user.

Url Parameters:

| Param | Description | Optional |
|---------------|---|----------|
| timeperiod_id | Show for a specific time period, defaults to the current time period if one is not passed | Optional |
| user_id | Show for a specific user, defaults to current user if not defined | Optional |

Possible Errors

| Error | Description |
|-------------------------|--|
| 412 - Invalid Parameter | When the passed in timeperiod and/or user can not be found |
| 403 - Not Authorized | If you are not an administrator, but you are trying to view another managers forecast manager worksheet, you will receive a 403 Not Authorized Error |

Url Example:

/rest/v10/ForecastManagerWorksheets/:timeperiod_id/:user_id

Output Example:

```
{
  "next_offset":-1,
  "records":[
    {
      "id":"401594f5-5b46-fb66-1627-55771fe8723e",
      "name":"Sally Bronsen",
      "date_modified":"2015-06-09T13:13:26-04:00",
      "created_by":"1",
      "quota":"1932.444445",
      "best_case":"29848.000000",
      "best_case_adjusted":"37310.000000",
      "likely_case":"28694.444445",
      "likely_case_adjusted":"35868.055556",
      "worst_case":"27540.888889",
      "worst_case_adjusted":"34426.111111",
      "timeperiod_id":"adb78e81-3fbd-b4e0-287f-55771fd04a06",
      "draft":true,
      "is_manager":false,
      "user_id":"seed_sally_id",
      "opp_count":10,
      "pipeline_opp_count":3,
      "pipeline_amount":"2415.555556",
      "closed_amount":"26278.888889",
      "manager_saved":true,
      "show_history_log":0,
      "following":false,
      "assigned_user_id":"seed_sarah_id",
      "assigned_user_name":"Sarah Smith",
      "team_name":[
        {
          "id":"1",
          "name":"Global",
          "primary":true
        }
      ],
      "name_2": "",
      "currency_id":"-99",
      "base_rate":"1.000000",
      "_acl":{
        "fields":{
          "id":"28226f12-a3c9-bd84-041e-55771f064c4e",
          "name":"Max Jensen",
          "date_modified":"2015-06-09T13:13:26-04:00",
          "created_by":"1",
          "quota":"3141.333332",
          "best_case":"15848.222223",
          "best_case_adjusted":"13206.851853",
          "likely_case":"14928.222222",
          "likely_case_adjusted":"12440.185185",
          "worst_case":"14008.222223",
          "worst_case_adjusted":"11673.518519",
          "timeperiod_id":"adb78e81-3fbd-b4e0-287f-55771fd04a06",
          "draft":true,
          "is_manager":false,
          "user_id":"seed_max_id",
          "opp_count":12,
          "pipeline_opp_count":3,
          "pipeline_amount":"2617.777777",
          "closed_amount":"12310.444445",
          "manager_saved":true,

```

```

"show_history_log":0,          "following":false,
"assigned_user_id":"seed_sarah_id",      "assigned_user_name":"Sarah Smith",
"team_name":[                {          "id":"1",          "name":"Global",
"name_2":"","                "primary":true      }          ],
"currency_id":"-99",         "base_rate":"1.000000",      "_acl":{
"fields":{                   }          },      "_module":"ForecastManagerWorksheets"
},      {          "id":"1aca66af-f069-bba4-28a1-55771fe27506",
"name":"","                  "date_modified":"2015-06-09T13:13:26-04:00",
"created_by":"1",           "quota":"10142.333333",
"best_case":"37625.000000",    "best_case_adjusted":"37625.000000",
"likely_case":"34241.333333",  "likely_case_adjusted":"34241.333333",
"worst_case":"30857.666667",  "worst_case_adjusted":"30857.666667",
"timeperiod_id":"adb78e81-3fbd-b4e0-287f-55771fd04a06",      "draft":true,
"is_manager":true,          "user_id":"seed_sarah_id",    "opp_count":13,
"pipeline_opp_count":6,     "pipeline_amount":"10142.333333",
"closed_amount":"24099.000000",  "manager_saved":true,
"show_history_log":0,          "following":false,
"assigned_user_id":"seed_sarah_id",      "assigned_user_name":"Sarah Smith",
"team_name":[                {          "id":"1",          "name":"Global",
"name_2":"","                "primary":true      }          ],
"currency_id":"-99",         "base_rate":"1.000000",      "_acl":{
"fields":{                   }          },      "_module":"ForecastManagerWorksheets"
}      ]      }

```

Last Modified: 05/31/2017 05:54pm

/ForecastManagerWorksheets/:timeperiod_id/:user_id GET

Returns a collection of ForecastManagerWorksheet models

Summary:

This end point is used to return the ManagerWorksheets for a user at a given timeperiod. If no timeperiod is provide it will use the default timeperiod, likewise if no user is provided, it will default to the logged in user.

Url Parameters:

| Param | Description | Optional |
|---------------|---|----------|
| timeperiod_id | Show for a specific time period, defaults to the current time period if one is not passed | Optional |
| user_id | Show for a specific user, defaults to current user if not defined | Optional |

Possible Errors

| Error | Description |
|-------------------------|--|
| 412 - Invalid Parameter | When the passed in timeperiod and/or user can not be found |
| 403 - Not Authorized | If you are not an administrator, but you are trying to view another managers forecast manager worksheet, you will receive a 403 Not Authorized Error |

Url Example:

/rest/v10/ForecastManagerWorksheets/:timeperiod_id/:user_id

Output Example:

```
{
  "next_offset":-1,
  "records":[
    {
      "id":"401594f5-5b46-fb66-1627-55771fe8723e",
      "name":"Sally Bronsen",
      "date_modified":"2015-06-09T13:13:26-04:00",
      "created_by":"1",
      "quota":"1932.444445",
      "best_case":"29848.000000",
      "best_case_adjusted":"37310.000000",
      "likely_case":"28694.444445",
      "likely_case_adjusted":"35868.055556",
      "worst_case":"27540.888889",
      "worst_case_adjusted":"34426.111111",
      "timeperiod_id":"adb78e81-3fbd-b4e0-287f-55771fd04a06",
      "draft":true,
      "is_manager":false,
      "user_id":"seed_sally_id",
      "opp_count":10,
      "pipeline_opp_count":3,
      "pipeline_amount":"2415.555556",
      "closed_amount":"26278.888889",
      "manager_saved":true,
    }
  ]
}
```

```

"show_history_log":0,          "following":false,
"assigned_user_id":"seed_sarah_id",          "assigned_user_name":"Sarah Smith",
"team_name":[                {                "id":"1",                "name":"Global",
"name_2":"","                "primary":true                }                ],
"currency_id":"-99",          "base_rate":"1.000000",          "_acl":{
"fields":{                    }                    },          "_module":"ForecastManagerWorksheets"
},          {                "id":"28226f12-a3c9-bd84-041e-55771f064c4e",
"name":"Max Jensen",          "date_modified":"2015-06-09T13:13:26-04:00",
"created_by":"1",          "quota":"3141.333332",
"best_case":"15848.222223",          "best_case_adjusted":"13206.851853",
"likely_case":"14928.222222",          "likely_case_adjusted":"12440.185185",
"worst_case":"14008.222223",          "worst_case_adjusted":"11673.518519",
"timeperiod_id":"adb78e81-3fbd-b4e0-287f-55771fd04a06",          "draft":true,
"is_manager":false,          "user_id":"seed_max_id",          "opp_count":12,
"pipeline_opp_count":3,          "pipeline_amount":"2617.777777",
"closed_amount":"12310.444445",          "manager_saved":true,
"show_history_log":0,          "following":false,
"assigned_user_id":"seed_sarah_id",          "assigned_user_name":"Sarah Smith",
"team_name":[                {                "id":"1",                "name":"Global",
"name_2":"","                "primary":true                }                ],
"currency_id":"-99",          "base_rate":"1.000000",          "_acl":{
"fields":{                    }                    },          "_module":"ForecastManagerWorksheets"
},          {                "id":"1aca66af-f069-bba4-28a1-55771fe27506",
"name":"","                "date_modified":"2015-06-09T13:13:26-04:00",
"created_by":"1",          "quota":"10142.333333",
"best_case":"37625.000000",          "best_case_adjusted":"37625.000000",
"likely_case":"34241.333333",          "likely_case_adjusted":"34241.333333",
"worst_case":"30857.666667",          "worst_case_adjusted":"30857.666667",
"timeperiod_id":"adb78e81-3fbd-b4e0-287f-55771fd04a06",          "draft":true,
"is_manager":true,          "user_id":"seed_sarah_id",          "opp_count":13,
"pipeline_opp_count":6,          "pipeline_amount":"10142.333333",
"closed_amount":"24099.000000",          "manager_saved":true,
"show_history_log":0,          "following":false,
"assigned_user_id":"seed_sarah_id",          "assigned_user_name":"Sarah Smith",
"team_name":[                {                "id":"1",                "name":"Global",
"name_2":"","                "primary":true                }                ],
"currency_id":"-99",          "base_rate":"1.000000",          "_acl":{
"fields":{                    }                    },          "_module":"ForecastManagerWorksheets"
}          ]          }

```

Last Modified: 05/31/2017 05:54pm

/ForecastWorksheets GET

Returns a collection of ForecastWorksheet models

Summary:

This end point is used to return the Worksheets for a user at a given timeperiod. If no timeperiod is provide it will use the default timeperiod, likewise if no user is provided, it will default to the logged in user.

Also this end points use the visibility requirements set forth by the application, that if the records being requested belong to you, you will get the draft version, If the worksheet is not yours, you will only see the committed version.

Url Parameters:

| Param | Description | Optional |
|---------------|---|----------|
| timeperiod_id | Show for a specific time period, defaults to the current time period if one is not passed | Optional |
| user_id | Show for a specific user, defaults to current user if not defined | Optional |

Query Parameters:

| Param | Description | Optional |
|-------|--|----------|
| type | Which type to return, currently the ones that are supported are, opportunity and product | Optional |

Possible Errors

| Error | Description |
|-------------------------|--|
| 412 - Invalid Parameter | When the passed in timeperiod and/or user can not be found |
| 403 - Not Authorized | If you are not a manager, but you are trying to view another sales rep forecast worksheet, you will receive a 403 Not Authorized Error |

Url Example:

/rest/v10/ForecastWorksheets/:timeperiod_id/:user_id

Output Example:

```
{ "next_offset": -1,      "records": [      {      "id":
"ad3908c7-83a3-f360-c223-51117844c208",      "name": "Grow-Fast Inc -
1000 units",      "date_entered": "2013-02-05T21:22:00+00:00",
"date_modified": "2013-02-05T21:22:00+00:00",      "modified_user_id": "1",
"modified_by_name": "Administrator",      "created_by": "1",
"created_by_name": "Administrator",      "description": "",      "img":
"",      "deleted": "0",      "assigned_user_id": "seed_jim_id",
"assigned_user_name": "Jim Brennan",      "team_name": [      {
"id": "East",      "name": "East",      "name_2": "",
"primary": true      }      ],      "parent_id":
"50b90565-e748-ed77-d9d7-511178f5acae",      "parent_type":
"Opportunities",      "account_name": "",      "account_id": "",
"likely_case": "75000",      "best_case": "75000",      "worst_case":
"75000",      "base_rate": "1",      "currency_id": "-99",
"currency_name": "",      "currency_symbol": "",      "date_closed":
"2013-02-10",      "date_closed_timestamp": "1360531443",
"sales_stage": "Perception Analysis",      "probability": "70",
"commit_stage": "include",      "draft": "1",      "my_favorite": false,
"_acl": {      "fields": {}      }      }      ]      }
```

Last Modified: 05/31/2017 05:54pm

/ForecastWorksheets/export GET

Overview

Returns a record set in CSV format along with HTTP headers to indicate content type.

Request Arguments

| Name | Type | Description | Required |
|--------|-------|---|----------|
| uid | Array | A list of bean ids. | False |
| filter | Array | The filter expression. More information on filter expressions can be found in <code>/<module>/filter</code> . | False |
| sample | Array | Indicates whether to download sample data. | False |

Request

Exporting Records by Specific IDs

```
{
  "uid": "d43243c6-9b8e-2973-ae2-512d09bc34b4"
}
```

Exporting Records by a List of IDs

```
{
  "uid": [
    "d43243c6-9b8e-2973-ae2-512d09bc34b4",
    "b3e87a3f-cd8f-7b86-467a-512d09e8d240"
  ]
}
```

Exporting Records Using a Filter

```
{
  "filter":[
    {
      "name":"airline"
    }
  ]
}
```

Exporting a Sample Result Set

```
{
  "sample":true
}
```

Response Arguments

| Name | Type | Description |
|--------------|--------|--|
| <csv export> | String | Returns the selected records in CSV format with the content headers. |

Response

```
result: HTTP/1.1 200 OK
Date: Fri, 01 Mar 2013 04:05:55 GMT
Server: Apache/2.2.22 (Unix) DAV/2 PHP/5.3.17
X-Powered-By: PHP/5.3.17
Expires: Mon, 26 Jul 1997 05:00:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0,
pre-check=0
Pragma: cache
Content-Disposition: attachment; filename=Accounts.csv
Content-transfer-encoding: binary
```

Last-Modified: Fri, 01 Mar 2013 04:05:55 GMT
Cache-Control: post-check=0, pre-check=0
Content-Length: 1080
Connection: close
Content-Type: application/octet-stream; charset=UTF-8

"Name", "ID", "Website", "Email Address", "Office Phone", "Alternate
Phone", "Fax", "Billing Street", "Billing City", "Billing State", "Billing
Postal Code", "Billing Country", "Shipping Street", "Shipping
City", "Shipping State", "Shipping Postal Code", "Shipping
Country", "Description", "Type", "Industry", "Annual
Revenue", "Employees", "SIC Code", "Ticker Symbol", "Parent Account
ID", "Ownership", "Campaign ID", "Rating", "Assigned User Name", "Assigned
To", "Team ID", "Teams", "Team Set ID", "Date Created", "Date
Modified", "Modified By", "Created
By", "Deleted", "Image", "last_activity_date", "Linkedin Company
ID", "Facebook Account", "Twitter Account", "Google Plus ID"
"Arts & Crafts
Inc", "d43243c6-9b8e-2973-ae2-512d09bc34b4", "", "", "(052)
034-1853", "", "", "777 West Filmore Ln", "Santa
Monica", "CA", "35354", "USA", "777 West Filmore Ln", "Santa
Monica", "CA", "35354", "USA", "", "Customer", "Transportation", "", "", "", "",
"", "", "", "", "sally", "seed_sally_id", "West", "West", "West", "02/26/2013
07:12 pm", "02/26/2013 07:12 pm", "1", "1", "0", "", "02/26/2013 07:12
pm", "", "", "", ""

Change Log

| Version | Change |
|---------|--------------------------------------|
| v10 | Added /<module>/export GET endpoint. |

Last Modified: 05/31/2017 05:29pm

/ForecastWorksheets/filter GET

Returns a collection of ForecastWorksheet models

Summary:

This end point is used to return the Worksheets for a user at a given timeperiod. If no timeperiod is provide it will use the default timeperiod, likewise if no user is provided, it will default to the logged in user.

Also this end points use the visibility requirements set forth by the application, that if the records being requested belong to you, you will get the draft version, If the worksheet is not yours, you will only see the committed version.

Query Parameters:

| Param | Description | Optional |
|--------|--|----------|
| filter | What you want to filter on | Optional |
| type | Which type to return, currently the ones that are supported are, opportunity and product | Optional |

Possible Errors

| Error | Description |
|-------------------------|--|
| 412 - Invalid Parameter | When the passed in timeperiod and/or user can not be found |
| 403 - Not Authorized | If you are not a manager, but you are trying to view another sales rep forecast worksheet, you will receive a 403 Not Authorized Error |

Url Example:

```
/rest/v10/ForecastWorksheets/:timeperiod_id/:user_id { "filter":[  
{"assigned_user_id" : "seed_jim_id"},  
{"timeperiod_id":"e546185a-5889-ea75-84c8-511178d1a5ba"} ], }
```

Output Example:

```

{ "next_offset": -1,      "records": [      {      "id":
"ad3908c7-83a3-f360-c223-51117844c208",      "name": "Grow-Fast Inc -
1000 units",      "date_entered": "2013-02-05T21:22:00+00:00",
"date_modified": "2013-02-05T21:22:00+00:00",      "modified_user_id": "1",
"modified_by_name": "Administrator",      "created_by": "1",
"created_by_name": "Administrator",      "description": "",      "img":
"",      "deleted": "0",      "assigned_user_id": "seed_jim_id",
"assigned_user_name": "Jim Brennan",      "team_name": [      {
"id": "East",      "name": "East",      "name_2": "",
"primary": true      }      ],      "parent_id":
"50b90565-e748-ed77-d9d7-511178f5acae",      "parent_type":
"Opportunities",      "account_name": "",      "account_id": "",
"likely_case": "75000",      "best_case": "75000",      "worst_case":
"75000",      "base_rate": "1",      "currency_id": "-99",
"currency_name": "",      "currency_symbol": "",      "date_closed":
"2013-02-10",      "date_closed_timestamp": "1360531443",
"sales_stage": "Perception Analysis",      "probability": "70",
"commit_stage": "include",      "draft": "1",      "my_favorite": false,
"_acl": {      "fields": {      }      }      }      }

```

Last Modified: 05/31/2017 05:54pm

/ForecastWorksheets/filter POST

Returns a collection of ForecastWorksheet models

Summary:

This end point is used to return the Worksheets for a user at a given timeperiod. If no timeperiod is provide it will use the default timeperiod, likewise if no user is provided, it will default to the logged in user.

Also this end points use the visibility requirements set forth by the application, that if the records being requested belong to you, you will get the draft version, If the worksheet is not yours, you will only see the committed version.

Query Parameters:

| Param | Description | Optional |
|--------|--|----------|
| filter | What you want to filter on | Optional |
| type | Which type to return, currently the ones that are supported are, opportunity and product | Optional |

Possible Errors

| Error | Description |
|-------------------------|--|
| 412 - Invalid Parameter | When the passed in timeperiod and/or user can not be found |
| 403 - Not Authorized | If you are not a manager, but you are trying to view another sales rep forecast worksheet, you will receive a 403 Not Authorized Error |

Url Example:

```
/rest/v10/ForecastWorksheets/:timeperiod_id/:user_id { "filter":[
{"assigned_user_id" : "seed_jim_id"},
{"timeperiod_id":"e546185a-5889-ea75-84c8-511178d1a5ba"} ], }
```

Output Example:

```
{ "next_offset": -1, "records": [ { "id":
"ad3908c7-83a3-f360-c223-51117844c208", "name": "Grow-Fast Inc -
1000 units", "date_entered": "2013-02-05T21:22:00+00:00",
"date_modified": "2013-02-05T21:22:00+00:00", "modified_user_id": "1",
"modified_by_name": "Administrator", "created_by": "1",
"created_by_name": "Administrator", "description": "", "img":
"", "deleted": "0", "assigned_user_id": "seed_jim_id",
"assigned_user_name": "Jim Brennan", "team_name": [ {
"id": "East", "name": "East", "name_2": "",
"primary": true } ], "parent_id":
"50b90565-e748-ed77-d9d7-511178f5acae", "parent_type":
"Opportunities", "account_name": "", "account_id": "",
```

```

"likely_case": "75000",          "best_case": "75000",          "worst_case":
"75000",          "base_rate": "1",          "currency_id": "-99",
"currency_name": "",          "currency_symbol": "",          "date_closed":
"2013-02-10",          "date_closed_timestamp": "1360531443",
"sales_stage": "Perception Analysis",          "probability": "70",
"commit_stage": "include",          "draft": "1",          "my_favorite": false,
"_acl": {          "fields": {}          }          }

```

Last Modified: 05/31/2017 05:54pm

/ForecastWorksheets/:record PUT

Saves a collection of ForecastWorksheet models

Summary:

This endpoint is used to save the json Data for an array of worksheet data array entries

Query Parameters:

| Param | Description | Optional |
|---|-------------|----------|
| This endpoint does not accept any parameters. | | |

Input Example:

```

{ "amount" : "10000.000000",          "assigned_user_id" : "seed_max_id",
"base_rate" : "1",          "best_case" : "25000",          "commit_stage" : "include",
"currency_id" : "-99",          "current_user" : "seed_max_id",          "date_closed" :
"2013-01-14",          "date_modified" : "2013-01-14T10:58:27-05:00",          "draft" :
1,          "id" : "347beb60-d57c-b3aa-5922-50f42b6c27d4",          "likely_case" :
"15000",          "name" : "RR. Talker Co - 1000 units",          "probability" : "90",
"product_id" : "34b0d547-adaf-03ea-146c-50f42b3e6f04",          "sales_stage" :
"Value Proposition",          "timeperiod_id" :
"36f7085a-5889-ea75-84c8-50f42bd1a5ba",          "version" : 1,
"w_date_modified" : "2013-01-14T10:58:27-05:00",          "worksheet_id" :
"e0adb532-7d1c-eb49-b102-50f42b455164",          "worst_case" : "10000.000000"
}

```

Output Example:

```
{ "amount" : "15000.000000",      "assigned_user_id" : "seed_max_id",
"base_rate" : "1",      "best_case" : "25000.000000",      "commit_stage" :
"include",      "currency_id" : "-99",      "date_closed" : "2013-01-14",
"date_modified" : "2013-01-15T10:52:31-05:00",      "id" :
"347beb60-d57c-b3aa-5922-50f42b6c27d4",      "likely_case" : "15000.000000",
"name" : "RR. Talker Co - 1000 units",      "probability" : "90",
"product_id" : "34b0d547-adaf-03ea-146c-50f42b3e6f04",      "sales_stage" :
"Value Proposition",      "version" : 1,      "w_date_modified" :
"2013-01-14T10:58:27-05:00",      "worksheet_id" :
"e0adb532-7d1c-eb49-b102-50f42b455164",      "worst_case" : "10000.000000"
}
```

Last Modified: 05/31/2017 05:54pm

/ForecastWorksheets/:timeperiod_id GET

Returns a collection of ForecastWorksheet models

Summary:

This end point is used to return the Worksheets for a user at a given timeperiod. If no timeperiod is provide it will use the default timeperiod, likewise if no user is provided, it will default to the logged in user.

Also this end points use the visibility requirements set forth by the application, that if the records being requested belong to you, you will get the draft version, If the worksheet is not yours, you will only see the committed version.

Url Parameters:

| Param | Description | Optional |
|---------------|---|----------|
| timeperiod_id | Show for a specific time period, defaults to the current time period if one | Optional |

| | | |
|---------|---|----------|
| | is not passed | |
| user_id | Show for a specific user, defaults to current user if not defined | Optional |

Query Parameters:

| Param | Description | Optional |
|-------|--|----------|
| type | Which type to return, currently the ones that are supported are, opportunity and product | Optional |

Possible Errors

| Error | Description |
|-------------------------|--|
| 412 - Invalid Parameter | When the passed in timeperiod and/or user can not be found |
| 403 - Not Authorized | If you are not a manager, but you are trying to view another sales rep forecast worksheet, you will receive a 403 Not Authorized Error |

Url Example:

/rest/v10/ForecastWorksheets/:timeperiod_id/:user_id

Output Example:

```
{ "next_offset": -1, "records": [ { "id": "ad3908c7-83a3-f360-c223-51117844c208", "name": "Grow-Fast Inc - 1000 units", "date_entered": "2013-02-05T21:22:00+00:00", "date_modified": "2013-02-05T21:22:00+00:00", "modified_user_id": "1", "modified_by_name": "Administrator", "created_by": "1", "created_by_name": "Administrator", "description": "", "img":
```

```

    "",
    "deleted": "0",
    "assigned_user_id": "seed_jim_id",
    "assigned_user_name": "Jim Brennan",
    "team_name": [
    "id": "East",
    "name": "East",
    "name_2": "",
    "primary": true
    },
    "parent_id":
    "50b90565-e748-ed77-d9d7-511178f5acae",
    "parent_type":
    "Opportunities",
    "account_name": "",
    "account_id": "",
    "likely_case": "75000",
    "best_case": "75000",
    "worst_case":
    "75000",
    "base_rate": "1",
    "currency_id": "-99",
    "currency_name": "",
    "currency_symbol": "",
    "date_closed":
    "2013-02-10",
    "date_closed_timestamp": "1360531443",
    "sales_stage": "Perception Analysis",
    "probability": "70",
    "commit_stage": "include",
    "draft": "1",
    "my_favorite": false,
    "_acl": {
    "fields": {}
    }
  }
}
}
}

```

Last Modified: 05/31/2017 05:54pm

/ForecastWorksheets/:timeperiod_id/:user_id GET

Returns a collection of ForecastWorksheet models

Summary:

This end point is used to return the Worksheets for a user at a given timeperiod. If no timeperiod is provide it will use the default timeperiod, likewise if no user is provided, it will default to the logged in user.

Also this end points use the visibility requirements set forth by the application, that if the records being requested belong to you, you will get the draft version, If the worksheet is not yours, you will only see the committed version.

Url Parameters:

| Param | Description | Optional |
|---------------|---|----------|
| timeperiod_id | Show for a specific time period, defaults to the current time period if one is not passed | Optional |
| user_id | Show for a specific user, | Optional |

| | | |
|--|---|--|
| | defaults to current user if not defined | |
|--|---|--|

Query Parameters:

| Param | Description | Optional |
|-------|--|----------|
| type | Which type to return, currently the ones that are supported are, opportunity and product | Optional |

Possible Errors

| Error | Description |
|-------------------------|--|
| 412 - Invalid Parameter | When the passed in timeperiod and/or user can not be found |
| 403 - Not Authorized | If you are not a manager, but you are trying to view another sales rep forecast worksheet, you will receive a 403 Not Authorized Error |

Url Example:

/rest/v10/ForecastWorksheets/:timeperiod_id/:user_id

Output Example:

```
{ "next_offset": -1, "records": [ { "id": "ad3908c7-83a3-f360-c223-51117844c208", "name": "Grow-Fast Inc - 1000 units", "date_entered": "2013-02-05T21:22:00+00:00", "date_modified": "2013-02-05T21:22:00+00:00", "modified_user_id": "1", "modified_by_name": "Administrator", "created_by": "1", "created_by_name": "Administrator", "description": "", "img": "", "deleted": "0", "assigned_user_id": "seed_jim_id", "assigned_user_name": "Jim Brennan", "team_name": [ {
```

```

"id": "East",
"primary": true
"50b90565-e748-ed77-d9d7-511178f5acae",
"Opportunities",
"likely_case": "75000",
"75000",
"currency_name": "",
"2013-02-10",
"sales_stage": "Perception Analysis",
"commit_stage": "include",
"_acl": {
      "name": "East",
    },
    "parent_id":
"parent_type":
"account_name": "",
"account_id": "",
"best_case": "75000",
"worst_case":
"base_rate": "1",
"currency_id": "-99",
"currency_symbol": "",
"date_closed":
"date_closed_timestamp": "1360531443",
"probability": "70",
"draft": "1",
"my_favorite": false,
"fields": {}
}
}

```

Last Modified: 05/31/2017 05:54pm

/Forecasts GET

Overview

Updates a record of a specified type as a favorite for the current user.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

| Name | Type | Description |
|----------------|---------------------|---|
| <record field> | <record field type> | Returns the fields for the selected record. |

Response

```

{
  "id": "bdd59d85-687b-1739-b00a-512d09f6db9e",
  "name": "Insight Marketing Inc",

```

```
"date_entered":"2013-02-26T19:12:00+00:00",
"date_modified":"2013-02-26T19:12:00+00:00",
"modified_user_id":"1",
"modified_by_name":"Administrator",
"created_by":"1",
"created_by_name":"Administrator",
"description":"",
"img":"",
"last_activity_date":"2013-02-26T19:12:00+00:00",
"deleted":false,
"assigned_user_id":"seed_max_id",
"assigned_user_name":"Max Jensen",
"team_name":[
  {
    "id":"East",
    "name":"East",
    "name_2":"",
    "primary":false
  },
  {
    "id":1,
    "name":"Global",
    "name_2":"",
    "primary":false
  },
  {
    "id":"West",
    "name":"West",
    "name_2":"",
    "primary":true
  }
],
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"account_type":"Customer",
"industry":"Electronics",
"annual_revenue":"",
"phone_fax":"",
"billing_address_street":"345 Sugar Blvd.",
"billing_address_street_2":"",
"billing_address_street_3":"",
```

```
"billing_address_street_4": "",
"billing_address_city": "San Mateo",
"billing_address_state": "CA",
"billing_address_postalcode": "56019",
"billing_address_country": "USA",
"rating": "",
"phone_office": "(927) 136-9572",
"phone_alternate": "",
"website": "www.sectionvegan.de",
"ownership": "",
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "345 Sugar Blvd.",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "San Mateo",
"shipping_address_state": "CA",
"shipping_address_postalcode": "56019",
"shipping_address_country": "USA",
"email1": "kid.support.vegan@example.info",
"parent_id": "",
"sic_code": "",
"parent_name": "",
"email_opt_out": false,
"invalid_email": false,
"email": [
  {
    "email_address": "kid.support.vegan@example.info",
    "opt_out": "0",
    "invalid_email": "0",
    "primary_address": "1"
  },
  {
    "email_address": "phone.kid@example.cn",
    "opt_out": "0",
    "invalid_email": "0",
    "primary_address": "0"
  }
],
"campaign_id": "",
"campaign_name": "",
"my_favorite": true,
```

```
  "_acl":{
    "fields":{
      }
    }
  }
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added /<module>/:record/favorite PUT endpoint. |

Last Modified: 05/31/2017 05:31pm

/Forecasts/config POST

Creates and/or updates the config settings for the Forecasts module

Summary:

This endpoint is used to create and/or update the config settings for the Forecasts module as json data. It extends the core config endpoint by adding the functionality to create the Timeperiods and updates existing Opportunities for use in the Forecasts module.

Query Parameters:

| Param | Description | Optional |
|-------|-------------|----------|
|-------|-------------|----------|

Input Example:

```
{  "is_setup": 1,      "is_upgrade": 0,      "has_commits": 1,
  "timeperiod_type": "chronological",  "timeperiod_interval": "Annual",
  "timeperiod_leaf_interval": "Quarter",  "timeperiod_start_date":
```

```

"2013-01-01",          "timeperiod_shown_forward": 2,
"timeperiod_shown_backward": 2,          "forecast_ranges": "show_binary",
"buckets_dom": "commit_stage_binary_dom",          "show_binary_ranges": {
"include": {          "min": 70,          "max": 100          },
"exclude": {          "min": 0,          "max": 69          }          },
"show_buckets_ranges": {          "include": {          "min": 85,
"max": 100          },          "upside": {          "min": 70,
"max": 84          },          "exclude": {          "min": 0,
"max": 69          }          },          "sales_stage_won": [          "Closed
Won"          ],          "sales_stage_lost": [          "Closed Lost"          ],
"show_worksheet_likely": 1,          "show_worksheet_best": 1,
"show_worksheet_worst": 0,          "show_projected_likely": 1,
"show_projected_best": 1,          "show_projected_worst": 0,
"show_forecasts_commit_warnings": 1          }

```

Output Example:

```

{          "is_setup": 1,          "is_upgrade": 0,          "has_commits": 1,
"timeperiod_type": "chronological",          "timeperiod_interval": "Annual",
"timeperiod_leaf_interval": "Quarter",          "timeperiod_start_date":
"2013-01-01",          "timeperiod_shown_forward": 2,
"timeperiod_shown_backward": 2,          "forecast_ranges": "show_binary",
"buckets_dom": "commit_stage_binary_dom",          "show_binary_ranges": {
"include": {          "min": 70,          "max": 100          },
"exclude": {          "min": 0,          "max": 69          }          },
"show_buckets_ranges": {          "include": {          "min": 85,
"max": 100          },          "upside": {          "min": 70,
"max": 84          },          "exclude": {          "min": 0,
"max": 69          }          },          "sales_stage_won": [          "Closed
Won"          ],          "sales_stage_lost": [          "Closed Lost"          ],
"show_worksheet_likely": 1,          "show_worksheet_best": 1,
"show_worksheet_worst": 0,          "show_projected_likely": 1,
"show_projected_best": 1,          "show_projected_worst": 0,
"show_forecasts_commit_warnings": 1          }

```

Last Modified: 05/31/2017 05:54pm

/Forecasts/config PUT

Creates and/or updates the config settings for the Forecasts module

Summary:

This endpoint is used to create and/or update the config settings for the Forecasts module as json data. It extends the core config endpoint by adding the functionality to create the Timeperiods and updates existing Opportunities for use in the Forecasts module.

Query Parameters:

| Param | Description | Optional |
|-------|-------------|----------|
|-------|-------------|----------|

Input Example:

```
{  "is_setup": 1,      "is_upgrade": 0,      "has_commits": 1,
"timeperiod_type": "chronological",      "timeperiod_interval": "Annual",
"timeperiod_leaf_interval": "Quarter",      "timeperiod_start_date":
"2013-01-01",      "timeperiod_shown_forward": 2,
"timeperiod_shown_backward": 2,      "forecast_ranges": "show_binary",
"buckets_dom": "commit_stage_binary_dom",      "show_binary_ranges": {
"include": {      "min": 70,      "max": 100      },
"exclude": {      "min": 0,      "max": 69      }      },
"show_buckets_ranges": {      "include": {      "min": 85,
"max": 100      },      "upside": {      "min": 70,
"max": 84      },      "exclude": {      "min": 0,
"max": 69      }      },      "sales_stage_won": [      "Closed
Won"      ],      "sales_stage_lost": [      "Closed Lost"      ],
"show_worksheet_likely": 1,      "show_worksheet_best": 1,
"show_worksheet_worst": 0,      "show_projected_likely": 1,
"show_projected_best": 1,      "show_projected_worst": 0,
"show_forecasts_commit_warnings": 1      }
```

Output Example:

```
{  "is_setup": 1,      "is_upgrade": 0,      "has_commits": 1,
"timeperiod_type": "chronological",      "timeperiod_interval": "Annual",
"timeperiod_leaf_interval": "Quarter",      "timeperiod_start_date":
"2013-01-01",      "timeperiod_shown_forward": 2,
"timeperiod_shown_backward": 2,      "forecast_ranges": "show_binary",
"buckets_dom": "commit_stage_binary_dom",      "show_binary_ranges": {
"include": {      "min": 70,      "max": 100      },
```

```

"exclude": { "min": 0, "max": 69 },
"show_buckets_ranges": { "include": { "min": 85,
"max": 100 }, "upside": { "min": 70,
"max": 84 }, "exclude": { "min": 0,
"max": 69 }, "sales_stage_won": [ "Closed
Won" ], "sales_stage_lost": [ "Closed Lost" ],
"show_worksheet_likely": 1, "show_worksheet_best": 1,
"show_worksheet_worst": 0, "show_projected_likely": 1,
"show_projected_best": 1, "show_projected_worst": 0,
"show_forecasts_commit_warnings": 1 }

```

Last Modified: 05/31/2017 05:54pm

/Forecasts/enum/selectedTimePeriod GET

ForecastsApi Timeperiod filter info

Summary:

This returns the timeperiods with respect to the forward/backward options selected. For example, if we elect to use quarterly timeperiods and 2 forward and 2 backward timeperiod intervals are set, then 5 years worth of timeperiods will be returned (2 backward + current + 2 forward).

Query Parameters:

| Param | Description | Optional |
|---|-------------|----------|
| This endpoint does not accept any parameters. | | |

Input Example:

Output Example:

```

{ "d181230a-a7e0-3176-d10f-50f8530a51ce" : "Q1 (01/01/2012 - 03/31/2012)",
"d2bc58ef-21c4-27d5-e416-50f853db29f9" : "Q2 (04/01/2012 - 06/30/2012)",
"d3db853e-94d6-b5ac-98af-50f8530689be" : "Q3 (07/01/2012 - 09/30/2012)",
"d519f521-5303-1cfc-24f5-50f8537f5b54" : "Q4 (10/01/2012 - 12/31/2012)",

```

```
"dcc2885a-5889-ea75-84c8-50f853d1a5ba" : "Q1 (01/01/2013 - 03/31/2013)",
"ddc24224-c6c9-04aa-7869-50f853db2ea2" : "Q2 (04/01/2013 - 06/30/2013)",
"deadcf3-41ed-5f6b-ce5b-50f853de6ef6" : "Q3 (07/01/2013 - 09/30/2013)",
"dfaa3724-6295-8ddb-6d14-50f853047fcc" : "Q4 (10/01/2013 - 12/31/2013)",
"e1ceee19-c8dd-afdd-e797-50f8538d900a" : "Q1 (01/01/2014 - 03/31/2014)",
"e307594a-c12a-6be6-74b7-50f85351c574" : "Q2 (04/01/2014 - 06/30/2014)",
"e469239c-3f71-a48e-cda1-50f853214b6a" : "Q3 (07/01/2014 - 09/30/2014)",
"e57f4889-cefd-4356-6d82-50f85350decd" : "Q4 (10/01/2014 - 12/31/2014)" }
```

Last Modified: 05/31/2017 05:54pm

/Forecasts/init GET

ForecastsApi - init

Summary:

This endpoint is used to return initialization data for the Forecasts module.

Query Parameters:

| Param | Description | Optional |
|---|-------------|----------|
| This endpoint does not accept any parameters. | | |

Input Example:

Output Example:

```
{  "initData": {    "timezone": "GMT",    "currency_id": "-99",    "currency_symbol": "$",    "currency_rate": 1,    "decimal_separator": ".",    "language": "en_us",    "display_name": "Global",    "selectedUser": {      "datepref": "m/d/Y",      "currency_name": "US Dollars",      "currency_iso": "USD",      "decimal_precision": 2,      "number_grouping_separator": "",      "default_teams": [        {          "id": 1,          "name": "Global",          "name_2": ""        }      ]    },    "preferences": {      "timepref": "h:ia"    }  }
```

```

"primary":true      }]      },      "module_list":      [
"Home",      "Accounts",      "Contacts",      "Opportunities",
"Leads","Calendar",      "Reports",      "Quotes",
"Documents",      "Emails",      "Campaigns",      "Calls",
"Meetings",      "Tasks",      "Notes",      "Forecasts",
"Cases",      "Prospects",      "ProspectLists",      "Bugs",
"Employees"      ],      "type":"user",      "id":"seed_jim_id",
"full_name":"Jim Brennan",      "user_name":"jim",
"picture":"46d1fbfb-c258-ce81-fa3a-50f809925709",      "acl":      {
"Forecasts":{"fields":[],"admin":"no","_hash":"095777549714234c8192a7a7963c38
b8"},
"ForecastWorksheets":{"fields":[],"admin":"no","_hash":"095777549714234c8192a
7a7963c38b8"},
"ForecastManagerWorksheets":{"fields":[],"admin":"no","_hash":"09577754971423
4c8192a7a7963c38b8"},      },      "my_teams":      [
{"id":"8833dfef-c1f4-e8b3-ca3b-50f8093616be","name":"Chris Olliver"},
{"id":"East","name":"East"},{"id":1,"name":"Global"},
{"id":"4ae6baba-f356-7374-7eb4-50f809745551","name":"Jim Brennan"},
{"id":"707a642a-710b-37f1-88f6-50f8098205c0","name":"Max Jensen"},
{"id":"63308ab9-f663-bae8-90a5-50f80951c0fe","name":"Sally Bronsen"},
{"id":"5776c705-8e6e-4bfc-44e5-50f809d12c0e","name":"Sarah Smith"},
{"id":"West","name":"West"},
{"id":"7cf7ee0b-f88e-b566-d4d3-50f809dcb717","name":"Will Westin"}      ],
"showOpps":false,      "first_name":"Jim",      "last_name":"Brennan",
"admin":"yes"      },      "forecasts_setup":1      },      "defaultSelections":      {
"timeperiod_id":      {      "id":"a2ab9ad3-2101-36f8-7ba3-50f809b3b62c",
"label":"Q1 (01\01\2013 - 03\31\2013)"      },      "ranges":["include"],
"group_by":"forecast",      "dataset":"likely"      }      }

```

Last Modified: 11/30/2016 09:17pm

/Forecasts/reportees/:user_id GET

ForecastsApi Reportees

Summary:

This endpoint is used to return the json Data for an array of users and their state for the forecasts users tree filter

Query Parameters:

| Param | Description | Optional |
|---------|---|----------|
| user_id | Show for a specific user, defaults to current user if not defined | Optional |
| level | Level of child notes, defaults to 1, -1 for all levels | Optional |

Input Example:

```
{ 'user_id':'seed_max_id' }
```

Output Example:

```
{  "attr":{      "id":"jstree_node_jim",      "rel":"root"    },  "children":[    {      "attr":{        "id":"jstree_node_myopps_jim",        "rel":"my_opportunities"      },      "children":[        ],      "data":"Opportunities (Jim Brennan)",      "metadata":{        "first_name":"Jim",        "full_name":"Jim Brennan",        "id":"seed_jim_id",        "last_name":"Brennan",        "level":"1",        "reports_to_id":"",        "user_name":"jim"      },      "state":""    },    {      "attr":{        "id":"jstree_node_will",        "rel":"manager"      },      "children":[        ],      "data":"Will Westin",      "metadata":{        "first_name":"Will",        "full_name":"Will Westin",        "id":"seed_will_id",        "last_name":"Westin",        "level":"2",        "reports_to_id":"seed_jim_id",        "user_name":"will"      },      "state":"closed"    },    {      "attr":{        "id":"jstree_node_sarah",        "rel":"manager"      },      "children":[        ],      "data":"Sarah Smith",      "metadata":{        "first_name":"Sarah",        "full_name":"Sarah Smith",        "id":"seed_sarah_id",        "last_name":"Smith",        "level":"2",        "reports_to_id":"seed_jim_id",        "user_name":"sarah"      },      "state":"closed"    }  ],  "data":"Jim Brennan",  "metadata":{    "first_name":"Jim",    "full_name":"Jim Brennan",    "id":"seed_jim_id",    "last_name":"Brennan",    "level":"1",    "reports_to_id":"",    "user_name":"jim"  },  "state":"open" }
```

Last Modified: 05/31/2017 05:54pm

/Forecasts/:timeperiod_id/progressManager/:user_id GET

Projected Manager Data

Summary:

This endpoint is used to return the json Data not already in client view for the Forecasts manager projected panel.

Query Parameters:

| Param | Description | Optional |
|---------------|---|----------|
| user_id | Show for a specific user, defaults to current user if not defined | Optional |
| timeperiod_id | Show for a specific time period, defaults to the current time period if one is not passed | Optional |

Input Example:

```
{  user_id:seed_sally_id  
timeperiod_id:36f7085a-5889-ea75-84c8-50f42bd1a5ba }
```

Output Example:

```
{  closed_amount: 0  opportunities: 13  pipeline_revenue: 470000  
quota_amount: 382000.000000 }
```

Last Modified: 05/31/2017 05:54pm

/Forecasts/:timeperiod_id/progressRep/:user_id GET

Projected Rep Data

Summary:

This endpoint is used to return the json Data not already in client view for the Forecasts rep projected panel.

Query Parameters:

| Param | Description | Optional |
|---------------|---|----------|
| user_id | Show for a specific user, defaults to current user if not defined | Optional |
| timeperiod_id | Show for a specific time period, defaults to the current time period if one is not passed | Optional |

Input Example:

```
{  user_id:seed_sally_id
timeperiod_id:36f7085a-5889-ea75-84c8-50f42bd1a5ba }
```

Output Example:

```
{  quota_amount: "80000.000000" }
```

Last Modified: 05/31/2017 05:54pm

/Forecasts/:timeperiod_id/quotas/direct/:user_id GET

ForecastsQuotasApi - Get

Summary:

This endpoint is used to return quota information for a specific user ID, in a specific timeperiod and by a specific type..

Query Parameters:

| Param | Description | Optional |
|---------------|---------------------------------|----------|
| timeperiod_id | Show for a specific time period | Required |
| quota_type | "rollup" or "direct" quota | Required |
| user_id | Show for a specific user | Required |

Input Example:

Output Example:

```
{  "currency_id": "-99",  "amount": "75.000000",  "date_modified": "2013-09-17 21:23:32",  "formatted_amount": "$75.00",  "is_top_level_manager": true }
```

Last Modified: 11/30/2016 09:22pm

/Forecasts/:timeperiod_id/quotas/rollup/:user_id GET

ForecastsQuotasApi - Get

Summary:

This endpoint is used to return quota information for a specific user ID, in a specific timeperiod and by a specific type..

Query Parameters:

| Param | Description | Optional |
|-------|-------------|----------|
|-------|-------------|----------|

| | | |
|---------------|---------------------------------|----------|
| timeperiod_id | Show for a specific time period | Required |
| quota_type | "rollup" or "direct" quota | Required |
| user_id | Show for a specific user | Required |

Input Example:

Output Example:

```
{ "currency_id":"-99", "amount":"75.000000", "date_modified":"2013-09-17 21:23:32", "formatted_amount":"$75.00", "is_top_level_manager":true }
```

Last Modified: 11/30/2016 09:22pm

/Forecasts/:timeperiod_id/:user_id/chart/:display_manager GET

Retrieve a Forecasting Information In SugarChart Format

Summary:

This endpoint is used to return the json Data for SugarCharts to use to display the needed chart.

Query Parameters:

| Param | Description | Optional |
|-----------------|--|----------|
| timeperiod_id | Show for a specific time period | |
| user_id | Show for a specific user | |
| display_manager | Pipeline or Committed are valid values. | |
| dataset | Which Forecast dataset to show, valid values are | Optional |


```

"$61,500.00"          ],          "gvalue" : 25500,          "gvaluelabel" :
"$25,500.00",        "label" : "February 2013",        "links" : [ "" ],
"valuelabels" : [ "$25,500.00" ],          "values" : [ 25500 ]          },          {
"goalmarkervalue" : [ "111000.00",          "61500.00"          ],
"goalmarkervalue" : [ "$111,000.00",          "$61,500.00"          ],
"gvalue" : 0,          "gvaluelabel" : "$0.00",          "label" : "March 2013",
"links" : [ "" ],          "valuelabels" : [ "$0.00" ],          "values" : [ 0 ]
}          ]          }

```

Last Modified: 05/31/2017 05:54pm

/Forecasts/user/:user_id GET

ForecastsApi - user

Summary:

This endpoint is used to return a user's id, user_name, full_name, first_name, last_name, and is_manager param given a user's id.

Query Parameters:

| Param | Description | Optional |
|---------|---|----------|
| user_id | Returns user data for this specific user id | |

Input Example:

```
{ user_id:seed_sally_id }
```

Output Example:

```
{
  first_name: "Sally"
  full_name: "Sally Bronsen"
  id:
  "seed_sally_id"
  is_manager: false
  last_name: "Bronsen"
  user_name: "sally"
}
```

Last Modified: 05/31/2017 05:54pm

/KBContents GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

| Name | Type | Description | Required |
|--------|--------|--|----------|
| filter | String | The filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query parameters in one of two ways for GET requests: <ol style="list-style-type: none">1. By specifying individual filter arguments as distinct parameters. | False |

| Name | Type | Description | Required |
|-----------|---------|--|----------|
| | | <p>Example:
filter[0][id]=1.
By specifying the whole filter as a single JSON-encoded string. Note that this syntax is currently not supported on certain modules.</p> <p>Example:
filter=[{"id":"1"}].</p> | |
| filter_id | String | Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND. | False |
| max_num | Integer | A maximum number of records to return. Default is 20. | False |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |
| fields | String | Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters | False |

| | | | |
|-----------------------|--------|---|-------|
| | | <p>(applicable to link and collection fields). The fields <code>id</code> and <code>date_modified</code> will always be returned.</p> <p>Example:
 <code>name,account_type,description,{ "name": "opportunities", "fields": ["id", "name", "sales_status"], "order_by": "date_closed: desc" }</code></p> <p>For more details on additional field parameters, see Relate API and Collection API.</p> | |
| <code>view</code> | String | <p>Instead of defining the <code>fields</code> argument, the <code>view</code> argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the <code>fields</code> argument. Common views are "record" and "list".</p> <p>Example: <code>record</code></p> | False |
| <code>order_by</code> | String | <p>How to sort the returned records, in a comma delimited list with the direction appended</p> | False |

| | | | |
|---------|---------|--|-------|
| | | to the column name after a colon.
Example:
name:DESC,account_type:DESC,date_modified:ASC | |
| q | String | A search expression, will search on this module. Cannot be used at the same time as a filter expression or id. | False |
| deleted | Boolean | Boolean to show deleted records in the result set. | False |

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
  "filter":[
    {
      "name":"Nelson Inc"
    }
  ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering

on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
  "filter": [
    {
      "name": {
        "$starts": "Nelson"
      }
    }
  ]
}
```

Below is a list of operation types:

| Operation | Description |
|--------------|---|
| \$equals | Performs an exact match on that field. |
| \$not_equals | Performs an exact match on that field. |
| \$not_equals | Matches on non-matching values. |
| \$starts | Matches on anything that starts with the value. |
| \$ends | Matches anything that ends with the value. |
| \$contains | Matches anything that contains the value |
| \$in | Finds anything where field matches one of the values as specified as an array. |
| \$not_in | Finds anything where field does not matches any of the values as specified as an array. |
| \$is_null | Checks if the field is null. This operation does not need a value specified. |
| \$not_null | Checks if the field is not null. This operation does not need a value specified. |
| \$lt | Matches when the field is less than the value. |

| Operation | Description |
|-----------|---|
| \$lte | Matches when the field is less than or equal to the value. |
| \$gt | Matches when the field is greater than the value. |
| \$gte | Matches when the field is greater than or equal to the value. |

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "\$and" and "\$or".

Example

```
{
  "filter": [
    {
      "$or": [
        {
          "name": "Nelson Inc"
        },
        {
          "name": "Nelson LLC"
        }
      ]
    }
  ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently

accepted module expressions are "\$favorite" and "\$owner".

Example

```
{
  "filter":[
    {
      "$favorite":"_this"
    }
  ]
}
```

Response Arguments

| Name | Type | Description |
|-------------|---------|---|
| next_offset | Integer | Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records. |
| records | Array | An array of results containing matched records. |

Response

```
{
  "next_offset":-1,
  "records":[
    {
      "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
      "name":"Dale Spivey",
      "date_modified":"2013-02-28T05:03:00+00:00",
      "description":"",
      "opportunities": [
        {
```



```

        _module: "Opportunities",
        "id": "b0701501-1fab-8ae7-3942-540da93f5017",
        "name": "360 Vacations - 228 Units",
        "date_modified": "2014-09-08T16:05:00+03:00",
        "sales_status": "New"
    },
],
"_acl": {
    "fields": {
    }
}
},
{
    "id": "95e17367-9b3d-0e26-22dc-512d0961fedf",
    "name": "Florence Haddock",
    "date_modified": "2013-02-26T19:12:00+00:00",
    "description": "",
    "opportunities": [
        {
            _module: "Opportunities"
            date_modified: "2014-09-08T16:05:00+03:00"
            id: "9ce7c088-8ee4-7cd3-18f1-540da944d4c0"
            name: "360 Vacations - 312 Units"
            sales_status: "New"
        },
    ],
    "_acl": {
        "fields": {
        }
    }
}
]
}

```

Change Log

| Version | Change |
|---------|--------------------------------------|
| v10 | Added /<module>/filter GET endpoint. |

Last Modified: 05/31/2017 05:31pm

/KBContents/config POST

Creates and/or updates the config settings for the KBContents module

Summary:

This endpoint is used to create and/or update the config settings for the KBContents module as json data. It extends the core config endpoint by adding the functionality to process deleted languages and documents related to them.

Query Parameters:

| Param | Description | Optional |
|-------|-------------|----------|
|-------|-------------|----------|

Input Example:

```
{  "languages":[{    "en":"English",    "primary":true  }},{  "de":"German",    "primary":false  }],  "category_root":"31696245-0438-ff7a-9144-544f8c659daf",  "deleted_languages":["es","ru"] }
```

Output Example:

```
{  "languages":[{    "en":"English",    "primary":true  }},{  "de":"German",    "primary":false  }],  "category_root":"31696245-0438-ff7a-9144-544f8c659daf",  "deleted_languages":["es","ru"] }
```

Last Modified: 05/31/2017 05:54pm

/KBContents/config PUT

Creates and/or updates the config settings for the KBContents module

Summary:

This endpoint is used to create and/or update the config settings for the KBContents module as json data. It extends the core config endpoint by adding the functionality to process deleted languages and documents related to them.

Query Parameters:

| Param | Description | Optional |
|-------|-------------|----------|
|-------|-------------|----------|

Input Example:

```
{  "languages":[{    "en":"English",    "primary":true  },{    "de":"German",    "primary":false  }],  "category_root":"31696245-0438-ff7a-9144-544f8c659daf",  "deleted_languages":["es","ru"] }
```

Output Example:

```
{  "languages":[{    "en":"English",    "primary":true  },{    "de":"German",    "primary":false  }],  "category_root":"31696245-0438-ff7a-9144-544f8c659daf",  "deleted_languages":["es","ru"] }
```

Last Modified: 05/31/2017 05:54pm

/KBContents/duplicateCheck POST

Overview

Runs a duplicate check against specified data.

Request Arguments

| Name | Type | Description | Required |
|----------------|---------------------|----------------------------------|----------|
| <record field> | <record field type> | The name value list of fields to | True |

| Name | Type | Description | Required |
|------|------|-------------|----------|
| | | populate. | |

Request

```
{
  "name": "Airline"
}
```

Response Arguments

| Name | Type | Description |
|----------------|---------------------|--|
| <record field> | <record field type> | Returns the potential duplicate records. |

Response

```
{
  "next_offset": -1,
  "records": [
    {
      "id": "c4c26496-5dbc-67dd-bf5c-512d0923889e",
      "name": "Airline Maintenance Co",
      "date_entered": "2013-02-26T19:12:00+00:00",
      "date_modified": "2013-02-26T19:12:00+00:00",
      "modified_user_id": "1",
      "modified_by_name": "Administrator",
      "created_by": "1",
      "created_by_name": "Administrator",
      "description": "",
      "img": "",
      "last_activity_date": "2013-02-26T19:12:00+00:00",
      "deleted": false,
      "assigned_user_id": "seed_sally_id",
      "assigned_user_name": "Sally Bronsen",
    }
  ]
}
```

```
"team_name":[
  {
    "id":"East",
    "name":"East",
    "name_2":"",
    "primary":false
  },
  {
    "id":1,
    "name":"Global",
    "name_2":"",
    "primary":false
  },
  {
    "id":"West",
    "name":"West",
    "name_2":"",
    "primary":true
  }
],
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"account_type":"Customer",
"industry":"Other",
"annual_revenue":"",
"phone_fax":"",
"billing_address_street":"123 Anywhere Street",
"billing_address_street_2":"",
"billing_address_street_3":"",
"billing_address_street_4":"",
"billing_address_city":"Sunnyvale",
"billing_address_state":"CA",
"billing_address_postalcode":"48566",
"billing_address_country":"USA",
"rating":"",
"phone_office":"(648) 452-3486",
"phone_alternate":"",
"website":"www.kidqa.it",
"ownership":"",
"employees":"",
"ticker_symbol":"",
```

```
"shipping_address_street": "123 Anywhere Street",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "Sunnyvale",
"shipping_address_state": "CA",
"shipping_address_postalcode": "48566",
"shipping_address_country": "USA",
"email1": "vegan.im@example.tw",
"parent_id": "",
"sic_code": "",
"parent_name": "",
"email_opt_out": false,
"invalid_email": false,
"email": [
  {
    "email_address": "vegan.im@example.tw",
    "opt_out": "0",
    "invalid_email": "0",
    "primary_address": "1"
  },
  {
    "email_address": "phone85@example.tv",
    "opt_out": "0",
    "invalid_email": "0",
    "primary_address": "0"
  }
],
"campaign_id": "",
"campaign_name": "",
"my_favorite": false,
"_acl": {
  "fields": {

  }
},
"duplicate_check_rank": 0
}
]
```

Change Log

| Version | Change |
|---------|---|
| v10 | Added /<module>/duplicateCheck POST endpoint. |

Last Modified: 05/31/2017 05:29pm

/KBContents/filter GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

| Name | Type | Description | Required |
|--------|--------|---|----------|
| filter | String | The filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query | False |

| Name | Type | Description | Required |
|-----------|---------|--|----------|
| | | parameters in one of two ways for GET requests: <ol style="list-style-type: none"> 1. By specifying individual filter arguments as distinct parameters. Example: filter[0][id]=1. 2. By specifying the whole filter as a single JSON-encoded string. Note that this syntax is currently not supported on certain modules. Example: filter=[{"id": "1"}]. | |
| filter_id | String | Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND. | False |
| max_num | Integer | A maximum number of records to return. Default is 20. | False |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |

| | | | |
|--------|--------|--|-------|
| fields | String | <p>Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters (applicable to link and collection fields). The fields id and date_modified will always be returned.</p> <p>Example:
 name,account_type,description,{"name":"opportunities","fields":["id","name","sales_status"],"order_by":"date_closed:desc"}</p> <p>For more details on additional field parameters, see Relate API and Collection API.</p> | False |
| view | String | <p>Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common</p> | False |

| | | | |
|----------|---------|--|-------|
| | | views are "record" and "list".
Example: record | |
| order_by | String | How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon.
Example:
name:DESC,account_type:DESC,date_modified:ASC | False |
| q | String | A search expression, will search on this module. Cannot be used at the same time as a filter expression or id. | False |
| deleted | Boolean | Boolean to show deleted records in the result set. | False |

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
  "filter":[
    {
      "name":"Nelson Inc"
    }
  ]
}
```

```
    }
  ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
  "filter": [
    {
      "name": {
        "$starts": "Nelson"
      }
    }
  ]
}
```

Below is a list of operation types:

| Operation | Description |
|---------------------------|---|
| <code>\$equals</code> | Performs an exact match on that field. |
| <code>\$not_equals</code> | Performs an exact match on that field. |
| <code>\$not_equals</code> | Matches on non-matching values. |
| <code>\$starts</code> | Matches on anything that starts with the value. |
| <code>\$ends</code> | Matches anything that ends with the value. |
| <code>\$contains</code> | Matches anything that contains the value |
| <code>\$in</code> | Finds anything where field matches one of the values as specified as an array. |
| <code>\$not_in</code> | Finds anything where field does not matches any of the values as specified as an array. |

| Operation | Description |
|------------|--|
| \$is_null | Checks if the field is null. This operation does not need a value specified. |
| \$not_null | Checks if the field is not null. This operation does not need a value specified. |
| \$lt | Matches when the field is less than the value. |
| \$lte | Matches when the field is less than or equal to the value. |
| \$gt | Matches when the field is greater than the value. |
| \$gte | Matches when the field is greater than or equal to the value. |

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "\$and" and "\$or".

Example

```
{
  "filter": [
    {
      "$or": [
        {
          "name": "Nelson Inc"
        },
        {
          "name": "Nelson LLC"
        }
      ]
    }
  ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "\$favorite" and "\$owner".

Example

```
{
  "filter":[
    {
      "$favorite":"_this"
    }
  ]
}
```

Response Arguments

| Name | Type | Description |
|-------------|---------|---|
| next_offset | Integer | Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records. |
| records | Array | An array of results containing matched records. |

Response

```
{
  "next_offset":-1,
```

```
"records":[
  {
    "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
    "name":"Dale Spivey",
    "date_modified":"2013-02-28T05:03:00+00:00",
    "description":"",
    "opportunities": [
      {
        _module: "Opportunities",
        "id": "b0701501-1fab-8ae7-3942-540da93f5017",
        "name": "360 Vacations - 228 Units",
        "date_modified": "2014-09-08T16:05:00+03:00",
        "sales_status": "New"
      },
    ],
    "_acl": {
      "fields": {
      }
    }
  },
  {
    "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
    "name":"Florence Haddock",
    "date_modified":"2013-02-26T19:12:00+00:00",
    "description":"",
    "opportunities": [
      {
        _module: "Opportunities"
        date_modified: "2014-09-08T16:05:00+03:00"
        id: "9ce7c088-8ee4-7cd3-18f1-540da944d4c0"
        name: "360 Vacations - 312 Units"
        sales_status: "New"
      },
    ],
    "_acl": {
      "fields": {
      }
    }
  }
]
```

Change Log

| Version | Change |
|---------|--------------------------------------|
| v10 | Added /<module>/filter GET endpoint. |

Last Modified: 05/31/2017 05:29pm

/KBContents/:record/link POST

Overview

Creates relationships to a pre-existing record.

Request Arguments

| Name | Type | Description | Required |
|---------------------|----------|--|----------|
| <relationship link> | <string> | Link between targeted and related records. | True |
| <record ID> | <string> | The name value list of related records. Each item of the list may be either string equal to related item ID, or map containing record ID ("id" key is required in this case) and addition relationship properties. | True |

Request

```
{
```

```

link_name: "accounts"
ids: [
  "da6a3741-2a81-ba7f-f249-512d0932e94e",
  {
    "id": "e689173e-c953-1e14-c215-512d0927e7a2",
    "role": "owner"
  }
]
}

```

Response Arguments

| Name | Type | Description |
|-----------------|-------|---------------------------------------|
| record | Array | The record linked to related records. |
| related_records | Array | Records that were associated. |

Response

```

"record": {
  "id": "da6a3741-2a81-ba7f-f249-512d0932e94e",
  "name": "Slender Broadband Inc - 1000 units",
  "date_entered": "2013-02-26T19:12:00+00:00",
  "date_modified": "2013-02-26T19:12:00+00:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
  "description": "",
  "img": "",
  "last_activity_date": "2013-02-28T18:21:00+00:00",
  "deleted": false,
  "assigned_user_id": "seed_max_id",
  "assigned_user_name": "Max Jensen",
  "team_name": [
    {

```

```
        "id": "East",
        "name": "East",
        "name_2": "",
        "primary": false
    },
    {
        "id": "West",
        "name": "West",
        "name_2": "",
        "primary": true
    }
],
"opportunity_type": "",
"account_name": "Slender Broadband Inc",
"account_id": "181461c6-dc81-1115-1fe0-512d092e8f15",
"campaign_id": "",
"campaign_name": "",
"lead_source": "Campaign",
"amount": "25000",
"base_rate": "1",
"amount_usdollar": "25000",
"currency_id": "-99",
"currency_name": "",
"currency_symbol": "",
"date_closed": "2013-02-27",
"date_closed_timestamp": "1361992480",
"next_step": "",
"sales_stage": "Needs Analysis",
"sales_status": "New",
"probability": "90",
"best_case": "25000",
"worst_case": "25000",
"commit_stage": "include",
"my_favorite": false,
"_acl": {
    "fields": {
    }
}
},
"related_records": [
    {
        "id": "e689173e-c953-1e14-c215-512d0927e7a2",
```

```
"name": "Gus Dales",
"date_entered": "2013-02-26T19:12:00+00:00",
"date_modified": "2013-02-26T19:12:00+00:00",
"modified_user_id": "1",
"modified_by_name": "Administrator",
"created_by": "1",
"created_by_name": "Administrator",
"description": "",
"img": "",
"deleted": false,
"assigned_user_id": "seed_sally_id",
"assigned_user_name": "Sally Bronsen",
"team_name": [
  {
    "id": "West",
    "name": "West",
    "name_2": "",
    "primary": true
  }
],
"salutation": "",
"first_name": "Gus",
"last_name": "Dales",
"full_name": "Gus Dales",
"title": "Director Operations",
"linkedin": "",
"facebook": "",
"twitter": "",
"googleplus": "",
"department": "",
"do_not_call": false,
"phone_home": "(661) 120-2292",
"email": [
  {
    "email_address":
"section.sugar.section@example.it",
    "opt_out": "1",
    "invalid_email": "0",
    "primary_address": "0"
  },
  {
    "email_address": "support.qa.kid@example.co.uk",
    "opt_out": "0",
```

```
        "invalid_email": "0",
        "primary_address": "1"
    }
],
"phone_mobile": "(294) 447-9707",
"phone_work": "(036) 840-3216",
"phone_other": "",
"phone_fax": "",
"email1": "support.qa.kid@example.co.uk",
"email2": "section.sugar.section@example.it",
"invalid_email": false,
"email_opt_out": false,
"primary_address_street": "48920 San Carlos Ave",
"primary_address_street_2": "",
"primary_address_street_3": "",
"primary_address_city": "Persistence",
"primary_address_state": "CA",
"primary_address_postalcode": "54556",
"primary_address_country": "USA",
"alt_address_street": "",
"alt_address_street_2": "",
"alt_address_street_3": "",
"alt_address_city": "",
"alt_address_state": "",
"alt_address_postalcode": "",
"alt_address_country": "",
"assistant": "",
"assistant_phone": "",
"picture": "",
"email_and_name1": "",
"lead_source": "Support Portal User Registration",
"account_name": "Arts & Crafts Inc",
"account_id": "d43243c6-9b8e-2973-ae2-512d09bc34b4",
"opportunity_role_fields": "",
"opportunity_role_id": "",
"opportunity_role": "Technical Advisor",
"reports_to_id": "",
"report_to_name": "",
"portal_name": "GusDales145",
"portal_active": true,
"portal_password": "$1$JxYr6tmM$b.O6.KF42jP46RadSwz0N0",
"portal_password1": "",
"portal_app": "",
```

```
"preferred_language": "en_us",
"campaign_id": "",
"campaign_name": "",
"c_accept_status_fields": "",
"m_accept_status_fields": "",
"accept_status_id": "",
"accept_status_name": "",
"sync_contact": "",
"my_favorite": false,
"_acl": {
  "fields": {
  }
}
},
{
  "id": "da6a3741-2a81-ba7f-f249-512d0932e94e",
  "name": "Slender Broadband Inc - 1000 units",
  "date_entered": "2013-02-26T19:12:00+00:00",
  "date_modified": "2013-02-26T19:12:00+00:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
  "description": "",
  "img": "",
  "last_activity_date": "2013-02-28T18:36:00+00:00",
  "deleted": false,
  "assigned_user_id": "seed_max_id",
  "assigned_user_name": "Max Jensen",
  "team_name": [
    {
      "id": "East",
      "name": "East",
      "name_2": "",
      "primary": false
    },
    {
      "id": "West",
      "name": "West",
      "name_2": "",
      "primary": true
    }
  ],
}
```

```

"opportunity_type": "",
"account_name": "Slender Broadband Inc",
"account_id": "181461c6-dc81-1115-1fe0-512d092e8f15",
"campaign_id": "",
"campaign_name": "",
"lead_source": "Campaign",
"amount": "25000",
"base_rate": "1",
"amount_usdollar": "25000",
"currency_id": "-99",
"currency_name": "",
"currency_symbol": "",
"date_closed": "2013-02-27",
"date_closed_timestamp": "1361992480",
"next_step": "",
"sales_stage": "Needs Analysis",
"sales_status": "New",
"probability": "90",
"best_case": "25000",
"worst_case": "25000",
"commit_stage": "include",
"my_favorite": false,
"_acl": {
  "fields": {
    }
  }
}
]
}

```

Change Log

| Version | Change |
|-------------|---|
| v10 (7.6.0) | Added support for additional relationship values. |
| v10 (7.1.5) | Added /<module>/:record/link POST endpoint. |

Last Modified: 05/31/2017 05:26pm

/KBContents/:record/notuseful PUT

Overview

Vote for Knowledge Base article.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

| Name | Type | Description |
|----------------|---------------------|--|
| <record field> | <record field type> | Returns the fields for the updated record. |

Response

```
{
  "active_date": "2014-01-01",
  "active_rev": 1,
  "approved": false,
  "assigned_user_id": "seed_will_id",
  "assigned_user_link": {"full_name": "Will Westin", "id":
"seed_will_id"},
  "assigned_user_name": "Will Westin",
  "attachment_list": [],
  "cases": {"name": "", "id": ""},
  "category_id": "3e844f1c-5ce1-6d6d-496d-5714901e6666",
  "category_name": "Database",
  "created_by": "1",
  "created_by_link": {"full_name": "Administrator", "id": "1"},
  "created_by_name": "Administrator",
  "date_entered": "2016-04-18T07:43:43+00:00",
  "date_modified": "2016-04-18T07:43:43+00:00",
```

```

"deleted": false,
"description": "",
"exp_date": "2014-12-31",
"file_mime_type": "",
"following": false,
"id": "87758ab5-75eb-451f-2bd7-571490d29d7a",
"is_external": false,
"kbarticle_id": "8862943c-022e-0385-908e-5714908ac792",
"kbarticle_name": "Resetting the device",
"kbarticles_kbcontents": {"name": "Resetting the device", "id":
"8862943c-022e-0385-908e-5714908ac792"},
"kbdocument_body": "

```

When things are not working as expected...

```

", "kbdocument_id": "87a20b6b-3172-ad19-0ca5-571490bec118",
"kbdocument_name": "Resetting the device", "kbdocuments_kbcontents": {"name":
"Resetting the device", "id": "87a20b6b-3172-ad19-0ca5-571490bec118"},
"kbsapprover_id": "", "kbsapprover_name": "", "kbsapprovers_kbcontents":
{"full_name": "", "id": ""}, "kbscase_id": "", "kbscase_name": "", "language": "en",
"modified_by_name": "Administrator", "modified_user_id": "1",
"modified_user_link": {"full_name": "Administrator", "id": "1"}, "my_favorite": false,
"name": "Resetting the device", "notuseful": 1 "related_languages": ["en"],
"revision": 1 "status": "in-review", "tag": [], "team_count": "", "team_count_link":
{"team_count": "", "id": "1"}, "team_name": [{"id": 1, "name": "Global", "name_2":
"", "primary": true}], "useful": 0, "usefulness_user_vote": "-1", "viewcount": 4 }

```

Change Log

| Version | Change |
|---------|---|
| v10 | Added /KBContents/:record/useful PUT endpoint. |
| v10 | Added /KBContents/:record/notuseful PUT endpoint. |

Last Modified: 05/31/2017 05:26pm

/KBContents/:record/useful PUT

Overview

Vote for Knowledge Base article.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

| Name | Type | Description |
|----------------|---------------------|--|
| <record field> | <record field type> | Returns the fields for the updated record. |

Response

```
{
  "active_date": "2014-01-01",
  "active_rev": 1,
  "approved": false,
  "assigned_user_id": "seed_will_id",
  "assigned_user_link": {"full_name": "Will Westin", "id":
"seed_will_id"},
  "assigned_user_name": "Will Westin",
  "attachment_list": [],
  "cases": {"name": "", "id": ""},
  "category_id": "3e844f1c-5ce1-6d6d-496d-5714901e6666",
  "category_name": "Database",
  "created_by": "1",
  "created_by_link": {"full_name": "Administrator", "id": "1"},
  "created_by_name": "Administrator",
  "date_entered": "2016-04-18T07:43:43+00:00",
  "date_modified": "2016-04-18T07:43:43+00:00",
  "deleted": false,
  "description": "",
  "exp_date": "2014-12-31",
  "file_mime_type": "",
```

```

"following": false,
"id": "87758ab5-75eb-451f-2bd7-571490d29d7a",
"is_external": false,
"kbarticle_id": "8862943c-022e-0385-908e-5714908ac792",
"kbarticle_name": "Resetting the device",
"kbarticles_kbcontents": {"name": "Resetting the device", "id":
"8862943c-022e-0385-908e-5714908ac792"},
"kbdocument_body": "

```

When things are not working as expected...

```

", "kbdocument_id": "87a20b6b-3172-ad19-0ca5-571490bec118",
"kbdocument_name": "Resetting the device", "kbdocuments_kbcontents": {"name":
"Resetting the device", "id": "87a20b6b-3172-ad19-0ca5-571490bec118"},
"kbsapprover_id": "", "kbsapprover_name": "", "kbsapprovers_kbcontents":
{"full_name": "", "id": ""}, "kbscase_id": "", "kbscase_name": "", "language": "en",
"modified_by_name": "Administrator", "modified_user_id": "1",
"modified_user_link": {"full_name": "Administrator", "id": "1"}, "my_favorite": false,
"name": "Resetting the device", "notuseful": 1 "related_languages": ["en"],
"revision": 1 "status": "in-review", "tag": [], "team_count": "", "team_count_link":
{"team_count": "", "id": "1"}, "team_name": [{"id": 1, "name": "Global", "name_2":
"", "primary": true}], "useful": 0, "usefulness_user_vote": "-1", "viewcount": 4 }

```

Change Log

| Version | Change |
|---------|---|
| v10 | Added /KBContents/:record/useful PUT endpoint. |
| v10 | Added /KBContents/:record/notuseful PUT endpoint. |

Last Modified: 05/31/2017 05:26pm

/Leads POST

Create Lead with optional post-save actions

Summary:

This endpoint is used to create a Lead with the optional post-save actions for Convert Target/Prospect and Create Lead from Email operations.

Query Parameters:

| Param | Description | Optional |
|------------------|--|----------|
| prospect_id | Pass a prospect id if Lead is being created as part of a Convert Target/Prospect process | Optional |
| inbound_email_id | Pass an inbound email id if Lead is being created from an Email | Optional |

Last Modified: 11/30/2016 09:16pm

/Leads/:leadId/convert POST

Convert Lead to a Contact and optionally link it to a new or existing instance of the modules specified

Summary:

This endpoint is used to convert the specified Lead to a Contact and optionally link that Contact to a new or existing instance of the modules specified. The types of modules that can be specified are limited to those that have been configured by the system administrator.

Post Parameters:

| Parameter | Description |
|--------------------|--|
| modules (required) | An object containing the Contacts module to be created as part of the conversion, along with (optionally) any modules that this new Contact record is to be related to. If relate-to modules are |

also specified, they must consist of a valid module type and either the id of an existing module of that type, or if new, the full record to be created for that type. Note that the allowed module types are defined by the System Administrator.

Example

```
{
  "modules": {
    "Contacts": {
      "deleted": "0",
      "do_not_call": false,
      "portal_active": "0",
      "preferred_language": "en_us",
      "assigned_user_id": "1",
      "assigned_user_name": "Administrator",
      "salutation": "",
      "first_name": "Darius",
      "last_name": "Paisley",
      "title": "Director Operations",
      "department": "",
      "description": "",
      "team_name": [
        {
          "id": "East",
          "name": "East",
          "name_2": "",
          "primary": true
        },
        {
          "id": 1,
          "name": "Global",
          "name_2": "",
          "primary": false
        },
        {
          "id": "West",
          "name": "West",
          "name_2": ""
        }
      ]
    }
  }
}
```

```
        "primary": false
      }
    ],
    "phone_home": "(890) 241-5509",
    "phone_mobile": "(738) 338-2546",
    "phone_work": "(579) 448-8879",
    "phone_fax": "",
    "primary_address_street": "123 Anywhere Street",
    "primary_address_city": "Salt Lake City",
    "primary_address_state": "CA",
    "primary_address_postalcode": "81738",
    "primary_address_country": "USA",
    "campaign_id": "",
    "campaign_name": "",
    "email": [
      {
        "email_address": "kid75@example.it",
        "invalid_email": false,
        "opt_out": false,
        "primary_address": true,
        "reply_to_address": false
      }
    ]
  },
  "Accounts": {
    "id": "1c212ff9-c0d5-866d-d3ae-526e6cd47a31",
    "name": "Lexington Shores Corp",
    "date_entered": "2013-10-28T09:52:46-04:00",
    "date_modified": "2013-10-28T09:52:46-04:00",
    "modified_user_id": "1",
    "modified_by_name": "Administrator",
    "created_by": "1",
    "created_by_name": "Administrator",
    "description": "",
    "deleted": false,
    "assigned_user_id": "seed_sally_id",
    "assigned_user_name": "Sally Bronsen",
    "team_count": "",
    "team_name": [
      {
        "id": "West",
        "name": "West",
        "name_2": "",

```

```
        "primary": true
    }
],
"linkedin": "",
"facebook": "",
"twitter": "",
"googleplus": "",
"account_type": "Customer",
"industry": "Electronics",
"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": "Santa Fe",
"billing_address_state": "CA",
"billing_address_postalcode": "63785",
"billing_address_country": "USA",
"rating": "",
"phone_office": "(641) 347-6902",
"phone_alternate": "",
"website": "www.imphone.de",
"ownership": "",
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "111 Silicon Valley Road",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "Santa Fe",
"shipping_address_state": "CA",
"shipping_address_postalcode": "63785",
"shipping_address_country": "USA",
"email": [
    {
        "email_address": "section.qa@example.tw",
        "invalid_email": false,
        "opt_out": false,
        "primary_address": true,
        "reply_to_address": false
    },
    {
```

```
        "email_address": "vegan29@example.de",
        "invalid_email": false,
        "opt_out": false,
        "primary_address": false,
        "reply_to_address": false
    }
],
"email1": "section.qa@example.tw",
"parent_id": "",
"sic_code": "",
"parent_name": "",
"email_opt_out": false,
"invalid_email": false,
"campaign_id": "",
"campaign_name": "",
"my_favorite": false,
"_acl": {
    "fields": {}
},
"following": false,
"_module": "Accounts",
"duplicate_check_rank": 8
},
"Opportunities": {
    "id": "5529e246-c42f-04e0-1989-526e6d3029c6",
    "name": "Lexington Shores Corp - 311 Units",
    "date_entered": "2013-10-28T09:52:46-04:00",
    "date_modified": "2013-10-28T09:52:46-04:00",
    "modified_user_id": "1",
    "modified_by_name": "Administrator",
    "created_by": "1",
    "created_by_name": "Administrator",
    "description": "",
    "deleted": false,
    "assigned_user_id": "seed_sally_id",
    "assigned_user_name": "Sally Bronsen",
    "team_count": "",
    "team_name": [
        {
            "id": "West",
            "name": "West",
            "name_2": "",
            "primary": true
        }
    ]
}
```

```
    }
  ],
  "opportunity_type": "",
  "account_name": "Lexington Shores Corp",
  "account_id": "1c212ff9-c0d5-866d-d3ae-526e6cd47a31",
  "campaign_id": "",
  "campaign_name": "",
  "lead_source": "Partner",
  "amount": 10413,
  "base_rate": 1,
  "amount_usdollar": 10413,
  "currency_id": "-99",
  "currency_name": "",
  "currency_symbol": "",
  "date_closed": "2014-03-30",
  "date_closed_timestamp": 1396187804,
  "next_step": "",
  "sales_stage": "Prospecting",
  "sales_status": "New",
  "probability": 10,
  "best_case": 11795,
  "worst_case": 9031,
  "commit_stage": "exclude",
  "total_revenue_line_items": 5,
  "closed_revenue_line_items": 1,
  "contact_role": "",
  "my_favorite": false,
  "_acl": {
    "fields": {}
  },
  "following": false,
  "_module": "Opportunities",
  "duplicate_check_rank": 0
}
}
```

Last Modified: 11/30/2016 09:20pm

/Leads/:record/freebusy GET

Get a lead's FreeBusy schedule

Summary:

This endpoint returns a list of time slots for which the specified person is busy.

Request

GET /Leads/:id/freebusy

Response

```
{
  "id": "foo"
  "module": "Users",
  "freebusy": [
    {
      "start": "2014-08-24T08:45:00-04:00",
      "end": "2014-08-24T09:15:00-04:00"
    },
    {
      "start": "2014-08-30T05:45:00-04:00",
      "end": "2014-08-30T06:15:00-04:00"
    },
    {
      "start": "2014-09-12T15:45:00-04:00",
      "end": "2014-09-12T16:15:00-04:00"
    }
  ]
}
```

Last Modified: 05/31/2017 05:54pm

/Leads/register POST

Overview

Creates new Leads.

Request Arguments

| Name | Type | Description | Required |
|----------------|---------------------|--|----------|
| <record field> | <record field type> | The name value list of fields to populate. | True |

Request

```
{  
  "first_name": "John",  
  "last_name": "Smith"  
}
```

Response Arguments

| Name | Type | Description |
|----------------|---------------------|--|
| <record field> | <record field type> | Returns the fields for the newly created record. |

Response

```
{  
  "id": "ecba9f86-4a4a-def6-359c-505a5b33f014",  
  "name": "John Smith",  
  "date_entered": "2012-09-19T23:54:54+0000",  
  "date_modified": "2012-09-19T23:54:54+0000",  
  "modified_user_id": "1",  
  "created_by": "1",  
  "deleted": 0,  
  "team_id": "1",  
}
```

```
"team_set_id": "1",
"first_name": "John",
"last_name": "Smith",
"full_name": "John Smith",
"do_not_call": false,
"converted": false,
"lead_source": "Support Portal User Registration",
"status": "New",
"preferred_language": "en_us",
"email": [

],
"my_favorite": false
}
```

Change Log

| Version | Change |
|---------|--------------------------------------|
| v10 | Added /Leads/register POST endpoint. |

Last Modified: 05/31/2017 05:29pm

/Mail POST

Overview

Create an email and send or save as draft.

Query String Parameters

This endpoint does not accept any query string parameters.

Input Parameters

| Name | Type | Description | Required |
|------|------|-------------|----------|
|------|------|-------------|----------|

| Name | Type | Description | Required |
|---------------|--------|---|----------|
| email_config | String | ID of the outbound email configuration to use when sending this email | True |
| to_addresses | Array | Array of name/email address pairs for the TO field, when present, only email subfield is required (not name). | True |
| cc_addresses | Array | Array of name/email address pairs for the CC field, when present, email subfield is required. | False |
| bcc_addresses | Array | Array of name/email address pairs for the BCC field, when present, email subfield is required. | False |
| subject | String | Subject of the email | False |
| html_body | String | HTML body of the email | False |
| text_body | String | Text body of the email | False |
| status | String | Indicates the status of the email - 'draft' or 'ready' (ready to be sent) | True |
| related | Object | Contains 'parent_type' and 'parent_id' to relate this email to (for example 'Contact' and a contact's id) | False |

| | | | |
|-------------|--------|---|-------|
| teams | Object | Team(s) to assign this email to. 'primary' attribute is an id string for the primary team and 'other' attribute is an array of id strings for the other teams - only primary is required | True |
| attachments | Array | Array of file attachments - each attachment consists of a 'type' (where the attachment came from: upload, document, or template), 'id' (of the file upload, note, document revision, etc), and 'name' (the file name) | False |

Input Example

```
{
  "email_config": "abc181a2-5c05-b879-8e68-502279a8c401",
  "to_addresses": [
    {
      "name": "John Doe",
      "email": "john_doe@foo.com"
    },
    {
      "name": "David Madison",
      "email": "david_madison@bar.com"
    }
  ],
  "cc_addresses": [
    {
```

```
    "name": "Tom Swift",
    "email": "tswift@baz.com"
  }
],
"bcc_addresses": null,
"subject": "Minneapolis Convention",
"html_body": "<html><body>Hello World</body></html>",
"text_body": "Hello World"
"status": "ready",
"related": {
  "type": "Contacts",
  "id": "61cf0f8d-938c-c9b2-53ad-51ed7bbcf83b"
},
"teams": {
  "primary": "dabec868-696c-f458-e204-50227995ab50",
  "others": [
    "c3094c88-c95f-2e17-4553-50227996ad20",
    "abcde868-696c-f458-e704-58369095ab62"
  ]
},
"attachments": [
  {
    "type": "upload",
    "id": "cfbe4551-548d-f602-b228-45387645fc12",
    "name": "company_logo.jpg"
  },
  {
    "type": "document",
    "id": "876112a4-89c1-4ba7-a05a-7729a7a76818"
  },
  {
    "type": "template",
    "id": "002cfe6c-98e9-4342-bdb1-1660d0788872"
  }
],
}
```

Result

| Name | Type | Description |
|----------------------|---------------------------|--|
| <email record field> | <email record field type> | Returns the fields for the newly created email record. |

Output Example

```
{
  "team_id": "9c61c46a-a7c5-df71-481c-51d48232f820",
  "team_set_id": "9c61c46a-a7c5-df71-481c-51d48232f820",
  "id": "d9c165d0-8863-ba61-dc85-51ed8016c476",
  "date_entered": "2013-07-22 18:57:57",
  "date_modified": "2013-07-22 18:57:57",
  "assigned_user_id": "1",
  "assigned_user_name": "",
  "modified_user_id": "1",
  "modified_by_name": "admin",
  "created_by": "1",
  "created_by_name": "",
  "deleted": 0,
  "from_addr_name": "SugarCRM",
  "reply_to_addr": "",
  "to_addrs_names": "john_doe@foo.com, david_madison@bar.com",
  "cc_addrs_names": "tswift@baz.com",
  "description_html": "<html><body>Hello World</body></html>",
  "description": "Hello World",
  "date_sent": "2013-07-22 18:57:00",
  "name": "Minneapolis Convention",
  "type": "out",
  "status": "sent",
  "parent_type": "Contacts",
  "parent_id": "61cf0f8d-938c-c9b2-53ad-51ed7bbcf83b",
}
```

Change Log

| Version | Change |
|---------|--------|
|---------|--------|

| | |
|-----|----------------------------|
| v10 | Added /Mail POST endpoint. |
|-----|----------------------------|

Last Modified: 05/31/2017 05:31pm

/Mail/archive POST

Overview

Archives an email.

Query String Parameters

This endpoint does not accept any query string parameters.

Input Parameters

| Name | Type | Description | Required |
|---------------|--------|---|----------|
| date_sent | String | Date of email sent | True |
| from_address | String | From email address | True |
| to_addresses | Array | Array of name/email address pairs for the TO field, when present, only email subfield is required (not name). | True |
| cc_addresses | Array | Array of name/email address pairs for the CC field, when present, email subfield is required. | False |
| bcc_addresses | Array | Array of name/email address pairs for the BCC field, when present, | False |

| Name | Type | Description | Required |
|------------------|--------|--|----------|
| | | email subfield is required. | |
| subject | String | Subject of the email | True |
| html_body | String | HTML body of the email | False |
| text_body | String | Text body of the email | False |
| status | String | Indicates the status of the email - 'draft' or 'ready' (ready to be sent) | True |
| related | Object | Contains 'parent_type' and 'parent_id' to relate this email to (for example 'Contact' and a contact's id) | False |
| teams | Object | Team(s) to assign this email to. 'primary' attribute is an id string for the primary team and 'other' attribute is an array of id strings for the other teams - only primary is required | True |
| assigned_user_id | String | ID of a user this record is assigned to. | False |
| attachments | Array | Array of file attachments - each attachment consists of a 'type' (where the attachment came from: upload, document, or template), 'id' (of the file upload, | False |

| | | | |
|--|--|---|--|
| | | note, document revision, etc), and 'name' (the file name) | |
|--|--|---|--|

Input Example

```
{
  "date_sent": "2014-02-07T00:00:00",
  "from_address": "test@foo.com"
  "to_addresses": [
    {
      "name": "John Doe",
      "email": "john_doe@foo.com",
      "module": "Leads"
    },
    {
      "name": "David Madison",
      "email": "david_madison@bar.com",
      "module": "Leads"
    }
  ],
  "cc_addresses": [
    {
      "name": "Tom Swift",
      "email": "tswift@baz.com"
    }
  ],
  "bcc_addresses": null,
  "subject": "Minneapolis Convention",
  "html_body": "<html><body>Hello World</body></html>",
  "text_body": "Hello World"
  "status": "archive",
  "related": {
    "type": "Contacts",
    "id": "61cf0f8d-938c-c9b2-53ad-51ed7bbcf83b"
  },
  "teams": {
    "primary": "dabec868-696c-f458-e204-50227995ab50",
    "others": [
      "c3094c88-c95f-2e17-4553-50227996ad20",
```

```

        "abcde868-696c-f458-e704-58369095ab62"
    ]
},
"assigned_user_id": "seed_jim_id",
"attachments": [
    {
        "type": "upload",
        "id": "cfbe4551-548d-f602-b228-45387645fc12",
        "name": "company_logo.jpg"
    },
    {
        "type": "document",
        "id": "876112a4-89c1-4ba7-a05a-7729a7a76818"
    },
    {
        "type": "template",
        "id": "002cfe6c-98e9-4342-bdb1-1660d0788872"
    }
],
}

```

Result

| Name | Type | Description |
|----------------------|---------------------------|--|
| <email record field> | <email record field type> | Returns the fields for the newly created email record. |

Output Example

```

{
    "team_id": "9c61c46a-a7c5-df71-481c-51d48232f820",
    "team_set_id": "9c61c46a-a7c5-df71-481c-51d48232f820",
    "id": "d9c165d0-8863-ba61-dc85-51ed8016c476",
    "date_entered": "2013-07-22 18:57:57",
    "date_modified": "2013-07-22 18:57:57",
    "assigned_user_id": "1",
}

```

```
"assigned_user_name": "",
"modified_user_id": "1",
"modified_by_name": "admin",
"created_by": "1",
"created_by_name": "",
"deleted": 0,
"from_addr_name": "SugarCRM",
"reply_to_addr": "",
"to_addrs_names": "john_doe@foo.com, david_madison@bar.com",
"cc_addrs_names": "tswift@baz.com",
"description_html": "<html><body>Hello World</body></html>",
"description": "Hello World",
"date_sent": "2013-07-22 18:57:00",
"name": "Minneapolis Convention",
"type": "out",
"status": "sent",
"parent_type": "Contacts",
"parent_id": "61cf0f8d-938c-c9b2-53ad-51ed7bbcf83b",
"from_address": "test@foo.com"
}
```

Change Log

| Version | Change |
|---------|------------------------------------|
| v10 | Added /Mail/archive POST endpoint. |

Last Modified: 05/31/2017 05:29pm

/Mail/attachment POST

Overview

Upload an email attachment

Query String Parameters

This endpoint does not accept any query string parameters.

Input Parameters

| Name | Type | Description | Required |
|------------------|--------|---------------------|----------|
| email_attachment | String | The file to upload. | True |

Input Example

```
{"email_attachment": "@\\path\\to\\ExampleDocument.txt"}
```

Result

| Name | Type | Description |
|----------------------|---------------------------|--|
| <email record field> | <email record field type> | Returns the fields for the newly created email record. |

Output Example

```
{  
  "guid": "61b19f41-0ac9-0f2f-d9c5-51eecaf89396",  
  "name": "ExampleDocument.txt",  
  "nameForDisplay": "ExampleDocument.txt"  
}
```

Change Log

| Version | Change |
|---------|---------------------------------------|
| v10 | Added /Mail/attachment POST endpoint. |

Last Modified: 05/31/2017 05:29pm

/Mail/attachment/cache DELETE

Overview

Clears the user's attachment cache directory

Query Parameters:

This endpoint does not accept any parameters.

Input Example:

This endpoint does not accept any input.

Output Example:

If successful, the endpoint responds with a bool of true.

Change Log

| Version | Change |
|---------|--|
| v10 | Added /Mail/attachment/:file_guid DELETE endpoint. |

Last Modified: 05/31/2017 05:26pm

/Mail/attachment/:file_guid DELETE

Overview

Delete an updated email attachment (where :file_guid is the guid value returned from the /Mail/attachment API)

Query Parameters:

This endpoint does not accept any parameters.

Input Example:

This endpoint does not accept any input.

Output Example:

If successful, the endpoint responds with a bool of true.

Change Log

| Version | Change |
|---------|--|
| v10 | Added /Mail/attachment/:file_guid DELETE endpoint. |

Last Modified: 05/31/2017 05:26pm

/Meetings POST

Overview

Create a single event or a series of event records.

Request Arguments

| Name | Type | Description | Required |
|----------------|---------------------|--|----------|
| <record field> | <record field type> | The name value list of fields to populate. | True |

Request

```
{
  "name": "Department Meeting",
  "description": "Weekly Department Meeting",
  "date_start": "yyyy-mm-ddThh:mm:ss-00:00",
  "date_end": "yyyy-mm-ddThh:mm:ss-00:00",
  "repeat_type": "Weekly",
  "repeat_interval": "1", /* Every Week */
  "repeat_dow": "25", /* Tue and Fri */
  "repeat_count": "4", /* 4 Recurrences */
  "repeat_until": "",
}
```

Response Arguments

| Name | Description | Start Date | End Date |
|----------------|--|------------|----------|
| <record field> | Returns the fields for the newly created record. | | |

Response

```
{
}
```

Change Log

| Version | Change |
|---------|--------------------------------|
| v10 | Added /<module> POST endpoint. |

Last Modified: 05/31/2017 05:31pm

/Meetings/:record DELETE

Overview

Deletes either a single event record or a series of event records

Request Arguments

| Name | Type | Description | Required |
|----------------|---------|--|----------|
| all_reurrences | Boolean | Flag to delete all events in a series. | False |

Request

```
http://{site_url}/rest/v10/Meetings/2c9e5c09-6824-0d14-f5cb-5130321ac3cf?all_reurrences=true
```

Response Arguments

| Name | Type | Description |
|------|--------|---------------------------------------|
| id | String | Returns the ID of the deleted record. |

Response

```
{
  "id": "11cf0d0a-40af-8cb1-9da0-5057a5f511f9"
}
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added /<module>/:record DELETE endpoint. |

Last Modified: 05/31/2017 05:28pm

/Meetings/:record PUT

Overview

Update a calendar event record of the specified type.

Request Arguments

| Name | Type | Description | Required |
|----------------|---------|--|----------|
| all_reurrences | Boolean | Flag to update all events in a series. | False |

Request

```
http://{site_url}/rest/v10/Meetings/2c9e5c09-6824-0d14-f5cb-5130321ac3cf?all_reurrences=true
```

```
{
  "name": "Department Meeting",
  "description": "Weekly Department Meeting",
  "date_start": "yyyy-mm-ddThh:mm:ss-00:00",
  "date_end": "yyyy-mm-ddThh:mm:ss-00:00",
  "repeat_type": "Weekly",
  "repeat_interval": "1", /* Every Week */
  "repeat_dow": "25", /* Tue and Fri */
  "repeat_count": "4", /* 4 Recurrences */
  "repeat_until": "",
}
```

Response Arguments

| Name | Description | Start Date | End Date |
|----------------|--|------------|----------|
| <record field> | Returns the fields for the updated record. | | |

Response

```
{  
}
```

Change Log

| Version | Change |
|---------|---------------------------------------|
| v10 | Added /<module>/:record PUT endpoint. |

Last Modified: 05/31/2017 05:28pm

/Meetings/:record/external GET

Retrieves info about launching an external meeting

Summary:

This endpoint is used to retrieve info about launching an external meeting. This includes whether the current user has permission to host and/or join the meeting along with the host/join URLs.

Query Parameters:

| Param | Description | Optional |
|---|-------------|----------|
| This endpoint does not accept any parameters. | | |

Input Example:

Output Example:

```
{  "is_host_option_allowed": false,  "host_url": "",  "is_join_option_allowed": true,  "join_url": "//link.to/external/meeting" }
```

Last Modified: 05/31/2017 05:54pm

/Notifications GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

| Name | Type | Description | Required |
|-----------|---------|---|----------|
| filter | String | <p>The filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query parameters in one of two ways for GET requests:</p> <ol style="list-style-type: none"> 1. By specifying individual filter arguments as distinct parameters.
Example:
filter[0][id]=1. 2. By specifying the whole filter as a single JSON-encoded string. Note that this syntax is currently not supported on certain modules.
Example:
filter=[{"id": "1"}]. | False |
| filter_id | String | Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND. | False |
| max_num | Integer | A maximum number of records to | False |

| Name | Type | Description | Required |
|--------|---------|---|----------|
| | | return. Default is 20. | |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |
| fields | String | Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters (applicable to link and collection fields). The fields id and date_modified will always be returned.
Example:
name,account_type,description,{"name":"opportunities","fields":["id","name","sales_status"],"order_by":"date_closed:desc"}
For more details on additional field parameters, see Relate API and Collection API. | False |
| view | String | Instead of defining the fields argument, the view argument can be used instead. The field list is | False |

| | | | |
|----------|---------|---|-------|
| | | constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list".
Example: record | |
| order_by | String | How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon.
Example:
name:DESC,account_type:DESC,date_modified:ASC | False |
| q | String | A search expression, will search on this module. Cannot be used at the same time as a filter expression or id. | False |
| deleted | Boolean | Boolean to show deleted records in the result set. | False |

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This

will only find exact matches.

Example

```
{
  "filter":[
    {
      "name":"Nelson Inc"
    }
  ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
  "filter":[
    {
      "name":{
        "$starts":"Nelson"
      }
    }
  ]
}
```

Below is a list of operation types:

| Operation | Description |
|---------------------------|---|
| <code>\$equals</code> | Performs an exact match on that field. |
| <code>\$not_equals</code> | Performs an exact match on that field. |
| <code>\$not_equals</code> | Matches on non-matching values. |
| <code>\$starts</code> | Matches on anything that starts with the value. |
| <code>\$ends</code> | Matches anything that ends with the |

| Operation | Description |
|------------|---|
| | value. |
| \$contains | Matches anything that contains the value |
| \$in | Finds anything where field matches one of the values as specified as an array. |
| \$not_in | Finds anything where field does not matches any of the values as specified as an array. |
| \$is_null | Checks if the field is null. This operation does not need a value specified. |
| \$not_null | Checks if the field is not null. This operation does not need a value specified. |
| \$lt | Matches when the field is less than the value. |
| \$lte | Matches when the field is less than or equal to the value. |
| \$gt | Matches when the field is greater than the value. |
| \$gte | Matches when the field is greater than or equal to the value. |

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "\$and" and "\$or".

Example

```
{
  "filter": [
    {
      "$or": [
        {
```

```

        "name": "Nelson Inc"
      },
      {
        "name": "Nelson LLC"
      }
    ]
  }
]
}

```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "\$favorite" and "\$owner".

Example

```

{
  "filter": [
    {
      "$favorite": "_this"
    }
  ]
}

```

Response Arguments

| Name | Type | Description |
|-------------|---------|---|
| next_offset | Integer | Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records. |
| records | Array | An array of results |

| Name | Type | Description |
|------|------|-----------------------------|
| | | containing matched records. |

Response

```
{
  "next_offset": -1,
  "records": [
    {
      "id": "fa300a0e-0ad1-b322-9601-512d0983c19a",
      "name": "Dale Spivey",
      "date_modified": "2013-02-28T05:03:00+00:00",
      "description": "",
      "opportunities": [
        {
          _module: "Opportunities",
          "id": "b0701501-1fab-8ae7-3942-540da93f5017",
          "name": "360 Vacations - 228 Units",
          "date_modified": "2014-09-08T16:05:00+03:00",
          "sales_status": "New"
        }
      ],
      "_acl": {
        "fields": {
        }
      }
    }
  ],
  {
    "id": "95e17367-9b3d-0e26-22dc-512d0961fedf",
    "name": "Florence Haddock",
    "date_modified": "2013-02-26T19:12:00+00:00",
    "description": "",
    "opportunities": [
      {
        _module: "Opportunities"
        date_modified: "2014-09-08T16:05:00+03:00"
        id: "9ce7c088-8ee4-7cd3-18f1-540da944d4c0"
        name: "360 Vacations - 312 Units"
        sales_status: "New"
      }
    ],
  }
}
```

```
    ],
    "_acl": {
      "fields": {
      }
    }
  }
]
}
```

Change Log

| Version | Change |
|---------|--------------------------------------|
| v10 | Added /<module>/filter GET endpoint. |

Last Modified: 05/31/2017 05:30pm

/Opportunities/config POST

Overview

Opportunity Config Save

Summary

This saves the config changes to the Opportunity Module, when the opps_view_by attribute changes, it will perform the nessicary migrations for the Metadata and the actual Data.

Request Arguments

| Name | Type | Description | Required |
|----------------|---------------------|--------------------------------------|----------|
| <config field> | <config field type> | The list of config fields to update. | False |

Request

```
{  
  "MyConfigOption": "MyConfigValue"  
}
```

Response Arguments

| Name | Type | Description |
|----------------|---------------------|---|
| <config field> | <config field type> | The list of config fields will be returned. |

Response

```
{  
  "MyConfigOption": "MyConfigValue"  
}
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added /Opportunities/config POST endpoint. |

Last Modified: 05/31/2017 05:29pm

/Opportunities/enum/:field GET

Overview

Retrieves the enum values for a specific field.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

| Name | Type | Description |
|---------------|-------|--------------------------------------|
| <list values> | Array | Returns the enum values for a field. |

Response

```
{
  "": "",
  "Analyst": "Analyst",
  "Competitor": "Competitor",
  "Customer": "Customer",
  "Integrator": "Integrator",
  "Investor": "Investor",
  "Partner": "Partner",
  "Press": "Press",
  "Prospect": "Prospect",
  "Reseller": "Reseller",
  "Other": "Other"
}
```

Change Log

| Version | Change |
|---------|---|
| v10 | Added /<module>/enum/:field GET endpoint. |

Last Modified: 05/31/2017 05:27pm

/PdfManager GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

| Name | Type | Description | Required |
|--------|--------|--|----------|
| filter | String | The filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query parameters in one of two ways for GET requests: <ol style="list-style-type: none">1. By specifying individual filter arguments as distinct parameters. | False |

| Name | Type | Description | Required |
|-----------|---------|---|---|
| | | | <p>Example:
filter[0][id]=1.
By specifying the whole filter as a single JSON-encoded string. Note that this syntax is currently not supported on certain modules.
Example:
filter=[{"id":"1"}].</p> |
| filter_id | String | Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND. | False |
| max_num | Integer | A maximum number of records to return. Default is 20. | False |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |
| fields | String | Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters | False |

| | | | |
|-----------------------|--------|---|-------|
| | | <p>(applicable to link and collection fields). The fields <code>id</code> and <code>date_modified</code> will always be returned.</p> <p>Example:
 <code>name,account_type,description,{ "name": "opportunities", "fields": ["id", "name", "sales_status"], "order_by": "date_closed: desc" }</code></p> <p>For more details on additional field parameters, see Relate API and Collection API.</p> | |
| <code>view</code> | String | <p>Instead of defining the <code>fields</code> argument, the <code>view</code> argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the <code>fields</code> argument. Common views are "record" and "list".</p> <p>Example: <code>record</code></p> | False |
| <code>order_by</code> | String | <p>How to sort the returned records, in a comma delimited list with the direction appended</p> | False |

| | | | |
|---------|---------|--|-------|
| | | to the column name after a colon.
Example:
name:DESC,account_type:DESC,date_modified:ASC | |
| q | String | A search expression, will search on this module. Cannot be used at the same time as a filter expression or id. | False |
| deleted | Boolean | Boolean to show deleted records in the result set. | False |

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
  "filter":[
    {
      "name":"Nelson Inc"
    }
  ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering

on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
  "filter": [
    {
      "name": {
        "$starts": "Nelson"
      }
    }
  ]
}
```

Below is a list of operation types:

| Operation | Description |
|--------------|---|
| \$equals | Performs an exact match on that field. |
| \$not_equals | Performs an exact match on that field. |
| \$not_equals | Matches on non-matching values. |
| \$starts | Matches on anything that starts with the value. |
| \$ends | Matches anything that ends with the value. |
| \$contains | Matches anything that contains the value |
| \$in | Finds anything where field matches one of the values as specified as an array. |
| \$not_in | Finds anything where field does not matches any of the values as specified as an array. |
| \$is_null | Checks if the field is null. This operation does not need a value specified. |
| \$not_null | Checks if the field is not null. This operation does not need a value specified. |
| \$lt | Matches when the field is less than the value. |

| Operation | Description |
|-----------|---|
| \$lte | Matches when the field is less than or equal to the value. |
| \$gt | Matches when the field is greater than the value. |
| \$gte | Matches when the field is greater than or equal to the value. |

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "\$and" and "\$or".

Example

```
{
  "filter": [
    {
      "$or": [
        {
          "name": "Nelson Inc"
        },
        {
          "name": "Nelson LLC"
        }
      ]
    }
  ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently

accepted module expressions are "\$favorite" and "\$owner".

Example

```
{
  "filter":[
    {
      "$favorite":"_this"
    }
  ]
}
```

Response Arguments

| Name | Type | Description |
|-------------|---------|---|
| next_offset | Integer | Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records. |
| records | Array | An array of results containing matched records. |

Response

```
{
  "next_offset":-1,
  "records":[
    {
      "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
      "name":"Dale Spivey",
      "date_modified":"2013-02-28T05:03:00+00:00",
      "description":"",
      "opportunities": [
        {
```

```

        _module: "Opportunities",
        "id": "b0701501-1fab-8ae7-3942-540da93f5017",
        "name": "360 Vacations - 228 Units",
        "date_modified": "2014-09-08T16:05:00+03:00",
        "sales_status": "New"
    },
],
"_acl": {
    "fields": {
    }
}
},
{
    "id": "95e17367-9b3d-0e26-22dc-512d0961fedf",
    "name": "Florence Haddock",
    "date_modified": "2013-02-26T19:12:00+00:00",
    "description": "",
    "opportunities": [
        {
            _module: "Opportunities"
            date_modified: "2014-09-08T16:05:00+03:00"
            id: "9ce7c088-8ee4-7cd3-18f1-540da944d4c0"
            name: "360 Vacations - 312 Units"
            sales_status: "New"
        },
    ],
    "_acl": {
        "fields": {
        }
    }
}
]
}

```

Change Log

| Version | Change |
|---------|--------------------------------------|
| v10 | Added /<module>/filter GET endpoint. |

Last Modified: 05/31/2017 05:30pm

/RevenueLineItems/:record/quote POST

Convert a Revenue Line Item to a quote.

Summary:

This endpoint is used to convert a single RevenueLineItem to a quote

Query Parameters:

| Param | Description | Optional |
|--------|------------------------------------|----------|
| record | Which Revenue Line Item to convert | false |

Valid Urls:

- /rest/v10/RevenueLineItems/:record/quote/

Output Example:

```
{      id:"123-456-789-0",      name:"My Test Quote"  }
```

Last Modified: 05/31/2017 05:54pm

/Tags/:record PUT

Overview

Update a record of the specified type.

Request Arguments

| Name | Type | Description | Required |
|----------------|---------------------|--|----------|
| <record field> | <record field type> | The name value list of fields to populate. | True |

Request

```
{
  "name": "New Account Name",
  "phone_office": "(555) 888-5555",
  "website": "www.newsite.com",
  "meetings": {
    {
      "add": [ "21e3385e-404f-b470-407e-54044e3d8024" ],
      "delete": [ "21e3385e-404f-b470-407e-54044e3d8023" ],
      "create": [
        {
          "name": "Test Meeting"
        }
      ]
    }
  ]
}
```

Link fields

It is possible to add or delete record relations to other records and create new related records.

| Action | Type | Description |
|--------|------|---|
| add | List | List of related records ID. See <code>RelateRecordApi</code> for details. |
| delete | List | List of related records ID. |
| create | List | List of name to value arrays of related records. |

Response Arguments

| Name | Type | Description |
|----------------|---------------------|--|
| <record field> | <record field type> | Returns the fields for the updated record. |

Response

```
{
  "id": "f222265a-b755-da89-0bc7-512d09b800b6",
  "name": "New Account Name",
  "date_entered": "2013-02-26T19:12:00+00:00",
  "date_modified": "2013-02-27T22:49:00+00:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
  "description": "",
  "img": "",
  "last_activity_date": "2013-02-26T19:12:00+00:00",
  "deleted": false,
  "assigned_user_id": "seed_sarah_id",
  "assigned_user_name": "Sarah Smith",
  "team_name": [
    {
      "id": 1,
      "name": "Global",
      "name_2": "",
      "primary": false
    },
    {
      "id": "West",
      "name": "West",
      "name_2": "",
      "primary": true
    }
  ],
  "linkedin": "",
  "facebook": "",
  "twitter": ""
}
```

```
"googleplus": "",
"account_type": "Customer",
"industry": "Hospitality",
"annual_revenue": "",
"phone_fax": "",
"billing_address_street": "9 IBM Path",
"billing_address_street_2": "",
"billing_address_street_3": "",
"billing_address_street_4": "",
"billing_address_city": "San Francisco",
"billing_address_state": "CA",
"billing_address_postalcode": "43635",
"billing_address_country": "USA",
"rating": "",
"phone_office": "(555) 888-5555",
"phone_alternate": "",
"website": "www.newsite.com",
"ownership": "",
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "9 IBM Path",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "San Francisco",
"shipping_address_state": "CA",
"shipping_address_postalcode": "43635",
"shipping_address_country": "USA",
"email1": "kid86@example.net",
"parent_id": "",
"sic_code": "",
"parent_name": "",
"email_opt_out": false,
"invalid_email": false,
"email": [
  {
    "email_address": "kid86@example.net",
    "opt_out": "0",
    "invalid_email": "0",
    "primary_address": "1"
  },
  {
    "email_address": "the.dev@example.name",
```

```

        "opt_out": "0",
        "invalid_email": "0",
        "primary_address": "0"
    }
],
"campaign_id": "",
"campaign_name": "",
"my_favorite": false,
"_acl": {
    "fields": {
    }
}
}

```

Change Log

| Version | Change |
|---------|---------------------------------------|
| v10 | Added /<module>/:record PUT endpoint. |

Last Modified: 05/31/2017 05:28pm

/Teams/:record/link POST

Overview

Creates relationships to a pre-existing record.

Request Arguments

| Name | Type | Description | Required |
|---------------------|----------|--|----------|
| <relationship link> | <string> | Link between targeted and related records. | True |

| Name | Type | Description | Required |
|-------------|----------|--|----------|
| <record ID> | <string> | The name value list of related records. Each item of the list may be either string equal to related item ID, or map containing record ID ("id" key is required in this case) and addition relationship properties. | True |

Request

```
{
  link_name: "accounts"
  ids: [
    "da6a3741-2a81-ba7f-f249-512d0932e94e",
    {
      "id": "e689173e-c953-1e14-c215-512d0927e7a2",
      "role": "owner"
    }
  ]
}
```

Response Arguments

| Name | Type | Description |
|-----------------|-------|---------------------------------------|
| record | Array | The record linked to related records. |
| related_records | Array | Records that were associated. |

Response

```
"record": {
  "id": "da6a3741-2a81-ba7f-f249-512d0932e94e",
  "name": "Slender Broadband Inc - 1000 units",
  "date_entered": "2013-02-26T19:12:00+00:00",
  "date_modified": "2013-02-26T19:12:00+00:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
  "description": "",
  "img": "",
  "last_activity_date": "2013-02-28T18:21:00+00:00",
  "deleted": false,
  "assigned_user_id": "seed_max_id",
  "assigned_user_name": "Max Jensen",
  "team_name": [
    {
      "id": "East",
      "name": "East",
      "name_2": "",
      "primary": false
    },
    {
      "id": "West",
      "name": "West",
      "name_2": "",
      "primary": true
    }
  ],
  "opportunity_type": "",
  "account_name": "Slender Broadband Inc",
  "account_id": "181461c6-dc81-1115-1fe0-512d092e8f15",
  "campaign_id": "",
  "campaign_name": "",
  "lead_source": "Campaign",
  "amount": "25000",
  "base_rate": "1",
  "amount_usdollar": "25000",
  "currency_id": "-99",
  "currency_name": "",
  "currency_symbol": "",
```

```
"date_closed": "2013-02-27",
"date_closed_timestamp": "1361992480",
"next_step": "",
"sales_stage": "Needs Analysis",
"sales_status": "New",
"probability": "90",
"best_case": "25000",
"worst_case": "25000",
"commit_stage": "include",
"my_favorite": false,
"_acl": {
  "fields": {
  }
}
},
"related_records": [
  {
    "id": "e689173e-c953-1e14-c215-512d0927e7a2",
    "name": "Gus Dales",
    "date_entered": "2013-02-26T19:12:00+00:00",
    "date_modified": "2013-02-26T19:12:00+00:00",
    "modified_user_id": "1",
    "modified_by_name": "Administrator",
    "created_by": "1",
    "created_by_name": "Administrator",
    "description": "",
    "img": "",
    "deleted": false,
    "assigned_user_id": "seed_sally_id",
    "assigned_user_name": "Sally Bronsen",
    "team_name": [
      {
        "id": "West",
        "name": "West",
        "name_2": "",
        "primary": true
      }
    ],
    "salutation": "",
    "first_name": "Gus",
    "last_name": "Dales",
    "full_name": "Gus Dales",
```

```
"title": "Director Operations",
"linkedin": "",
"facebook": "",
"twitter": "",
"googleplus": "",
"department": "",
"do_not_call": false,
"phone_home": "(661) 120-2292",
"email": [
  {
    "email_address":
"section.sugar.section@example.it",
    "opt_out": "1",
    "invalid_email": "0",
    "primary_address": "0"
  },
  {
    "email_address": "support.qa.kid@example.co.uk",
    "opt_out": "0",
    "invalid_email": "0",
    "primary_address": "1"
  }
],
"phone_mobile": "(294) 447-9707",
"phone_work": "(036) 840-3216",
"phone_other": "",
"phone_fax": "",
"email1": "support.qa.kid@example.co.uk",
"email2": "section.sugar.section@example.it",
"invalid_email": false,
"email_opt_out": false,
"primary_address_street": "48920 San Carlos Ave",
"primary_address_street_2": "",
"primary_address_street_3": "",
"primary_address_city": "Persistence",
"primary_address_state": "CA",
"primary_address_postalcode": "54556",
"primary_address_country": "USA",
"alt_address_street": "",
"alt_address_street_2": "",
"alt_address_street_3": "",
"alt_address_city": "",
"alt_address_state": "",
```

```
"alt_address_postalcode": "",
"alt_address_country": "",
"assistant": "",
"assistant_phone": "",
"picture": "",
"email_and_name1": "",
"lead_source": "Support Portal User Registration",
"account_name": "Arts & Crafts Inc",
"account_id": "d43243c6-9b8e-2973-ae2-512d09bc34b4",
"opportunity_role_fields": "",
"opportunity_role_id": "",
"opportunity_role": "Technical Advisor",
"reports_to_id": "",
"report_to_name": "",
"portal_name": "GusDales145",
"portal_active": true,
"portal_password": "$1$JxYr6tmM$b.O6.KF42jP46RadSwz0N0",
"portal_password1": "",
"portal_app": "",
"preferred_language": "en_us",
"campaign_id": "",
"campaign_name": "",
"c_accept_status_fields": "",
"m_accept_status_fields": "",
"accept_status_id": "",
"accept_status_name": "",
"sync_contact": "",
"my_favorite": false,
"_acl": {
  "fields": {
  }
}
},
{
  "id": "da6a3741-2a81-ba7f-f249-512d0932e94e",
  "name": "Slender Broadband Inc - 1000 units",
  "date_entered": "2013-02-26T19:12:00+00:00",
  "date_modified": "2013-02-26T19:12:00+00:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
  "description": "",
```

```
"img": "",
"last_activity_date": "2013-02-28T18:36:00+00:00",
"deleted": false,
"assigned_user_id": "seed_max_id",
"assigned_user_name": "Max Jensen",
"team_name": [
  {
    "id": "East",
    "name": "East",
    "name_2": "",
    "primary": false
  },
  {
    "id": "West",
    "name": "West",
    "name_2": "",
    "primary": true
  }
],
"opportunity_type": "",
"account_name": "Slender Broadband Inc",
"account_id": "181461c6-dc81-1115-1fe0-512d092e8f15",
"campaign_id": "",
"campaign_name": "",
"lead_source": "Campaign",
"amount": "25000",
"base_rate": "1",
"amount_usdollar": "25000",
"currency_id": "-99",
"currency_name": "",
"currency_symbol": "",
"date_closed": "2013-02-27",
"date_closed_timestamp": "1361992480",
"next_step": "",
"sales_stage": "Needs Analysis",
"sales_status": "New",
"probability": "90",
"best_case": "25000",
"worst_case": "25000",
"commit_stage": "include",
"my_favorite": false,
"_acl": {
  "fields": {
```

```
}  
  }  
} ]
```

Change Log

| Version | Change |
|-------------|---|
| v10 (7.6.0) | Added support for additional relationship values. |
| v10 (7.1.5) | Added /<module>/:record/link POST endpoint. |

Last Modified: 05/31/2017 05:26pm

/Teams/:record/link/:link_name/:remote_id DELETE

Overview

Deletes an existing relationship between two records.

Request Arguments

This endpoint does not accept any arguments.

Response Arguments

| Name | Type | Description |
|----------------|-------|---|
| record | Array | The record to disassociate from the related record. |
| related_record | Array | The record that was disassociated. |

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

Response

```
{
  "record": {
    "id": "da6a3741-2a81-ba7f-f249-512d0932e94e",
    "name": "Slender Broadband Inc - 1000 units",
    "date_entered": "2013-02-26T19:12:00+00:00",
    "date_modified": "2013-02-26T19:12:00+00:00",
    "modified_user_id": "1",
    "modified_by_name": "Administrator",
    "created_by": "1",
    "created_by_name": "Administrator",
    "description": "",
    "img": "",
    "last_activity_date": "2013-02-28T18:36:00+00:00",
    "deleted": false,
    "assigned_user_id": "seed_max_id",
    "assigned_user_name": "Max Jensen",
    "team_name": [
      {
        "id": "East",
        "name": "East",
        "name_2": "",
        "primary": false
      },
      {
        "id": "West",
        "name": "West",
        "name_2": "",
        "primary": true
      }
    ],
    "opportunity_type": "",
    "account_name": "Slender Broadband Inc",
    "account_id": "181461c6-dc81-1115-1fe0-512d092e8f15",
    "campaign_id": "",
    "campaign_name": "",
    "lead_source": "Campaign",
    "amount": "25000",
  }
}
```

```
"base_rate": "1",
"amount_usdollar": "25000",
"currency_id": "-99",
"currency_name": "",
"currency_symbol": "",
"date_closed": "2013-02-27",
"date_closed_timestamp": "1361992480",
"next_step": "",
"sales_stage": "Needs Analysis",
"sales_status": "New",
"probability": "90",
"best_case": "25000",
"worst_case": "25000",
"commit_stage": "include",
"my_favorite": false,
"_acl": {
  "fields": {
    }
  }
},
"related_record": {
  "id": "e689173e-c953-1e14-c215-512d0927e7a2",
  "name": "Gus Dales",
  "date_entered": "2013-02-26T19:12:00+00:00",
  "date_modified": "2013-02-26T19:12:00+00:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
  "description": "",
  "img": "",
  "deleted": false,
  "assigned_user_id": "seed_sally_id",
  "assigned_user_name": "Sally Bronsen",
  "team_name": [
    {
      "id": "West",
      "name": "West",
      "name_2": "",
      "primary": true
    }
  ]
}
```

```
],
"salutation": "",
"first_name": "Gus",
"last_name": "Dales",
"full_name": "Gus Dales",
"title": "Director Operations",
"linkedin": "",
"facebook": "",
"twitter": "",
"googleplus": "",
"department": "",
"do_not_call": false,
"phone_home": "(661) 120-2292",
"email": [
  {
    "email_address": "section.sugar.section@example.it",
    "opt_out": "1",
    "invalid_email": "0",
    "primary_address": "0"
  },
  {
    "email_address": "support.qa.kid@example.co.uk",
    "opt_out": "0",
    "invalid_email": "0",
    "primary_address": "1"
  }
],
"phone_mobile": "(294) 447-9707",
"phone_work": "(036) 840-3216",
"phone_other": "",
"phone_fax": "",
"email1": "support.qa.kid@example.co.uk",
"email2": "section.sugar.section@example.it",
"invalid_email": false,
"email_opt_out": false,
"primary_address_street": "48920 San Carlos Ave",
"primary_address_street_2": "",
"primary_address_street_3": "",
"primary_address_city": "Persistence",
"primary_address_state": "CA",
"primary_address_postalcode": "54556",
"primary_address_country": "USA",
```

```
"alt_address_street": "",
"alt_address_street_2": "",
"alt_address_street_3": "",
"alt_address_city": "",
"alt_address_state": "",
"alt_address_postalcode": "",
"alt_address_country": "",
"assistant": "",
"assistant_phone": "",
"picture": "",
"email_and_name1": "",
"lead_source": "Support Portal User Registration",
"account_name": "Arts & Crafts Inc",
"account_id": "d43243c6-9b8e-2973-ae2-512d09bc34b4",
"opportunity_role_fields": "",
"opportunity_role_id": "",
"opportunity_role": "",
"reports_to_id": "",
"report_to_name": "",
"portal_name": "GusDales145",
"portal_active": true,
"portal_password": "$1$JxYr6tmM$b.O6.KF42jp46RadSwz0N0",
"portal_password1": "",
"portal_app": "",
"preferred_language": "en_us",
"campaign_id": "",
"campaign_name": "",
"c_accept_status_fields": "",
"m_accept_status_fields": "",
"accept_status_id": "",
"accept_status_name": "",
"sync_contact": "",
"my_favorite": false,
"_acl": {
  "fields": {
    }
  }
}
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added
/<module>/:record/link/:link_name/:remote_id DELETE endpoint. |

Last Modified: 05/31/2017 05:25pm

/Teams/:record/link/:link_name/:remote_id POST

Overview

Creates a relationship to a pre-existing record.

Query String Parameters

This endpoint does not accept any arguments.

Response Arguments

| Name | Type | Description |
|----------------|-------|--|
| record | Array | The record linked to the related record. |
| related_record | Array | The record that was associated. |

Response

```
{
  "record": {
    "id": "da6a3741-2a81-ba7f-f249-512d0932e94e",
    "name": "Slender Broadband Inc - 1000 units",
```

```
"date_entered":"2013-02-26T19:12:00+00:00",
"date_modified":"2013-02-26T19:12:00+00:00",
"modified_user_id":"1",
"modified_by_name":"Administrator",
"created_by":"1",
"created_by_name":"Administrator",
"description":"",
"img":"",
"last_activity_date":"2013-02-28T18:21:00+00:00",
"deleted":false,
"assigned_user_id":"seed_max_id",
"assigned_user_name":"Max Jensen",
"team_name":[
  {
    "id":"East",
    "name":"East",
    "name_2":"",
    "primary":false
  },
  {
    "id":"West",
    "name":"West",
    "name_2":"",
    "primary":true
  }
],
"opportunity_type":"",
"account_name":"Slender Broadband Inc",
"account_id":"181461c6-dc81-1115-1fe0-512d092e8f15",
"campaign_id":"",
"campaign_name":"",
"lead_source":"Campaign",
"amount":"25000",
"base_rate":"1",
"amount_usdollar":"25000",
"currency_id":"-99",
"currency_name":"",
"currency_symbol":"",
"date_closed":"2013-02-27",
"date_closed_timestamp":"1361992480",
"next_step":"",
"sales_stage":"Needs Analysis",
```

```
"sales_status": "New",
"probability": "90",
"best_case": "25000",
"worst_case": "25000",
"commit_stage": "include",
"my_favorite": false,
"_acl": {
  "fields": {
    }
  }
},
"related_record": {
  "id": "e689173e-c953-1e14-c215-512d0927e7a2",
  "name": "Gus Dales",
  "date_entered": "2013-02-26T19:12:00+00:00",
  "date_modified": "2013-02-26T19:12:00+00:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
  "description": "",
  "img": "",
  "deleted": false,
  "assigned_user_id": "seed_sally_id",
  "assigned_user_name": "Sally Bronsen",
  "team_name": [
    {
      "id": "West",
      "name": "West",
      "name_2": "",
      "primary": true
    }
  ],
  "salutation": "",
  "first_name": "Gus",
  "last_name": "Dales",
  "full_name": "Gus Dales",
  "title": "Director Operations",
  "linkedin": "",
  "facebook": "",
  "twitter": ""
```

```
"googleplus":"","  
"department":"","  
"do_not_call":false,  
"phone_home":"(661) 120-2292",  
"email":[  
  {  
    "email_address":"section.sugar.section@example.it",  
    "opt_out":"1",  
    "invalid_email":"0",  
    "primary_address":"0"  
  },  
  {  
    "email_address":"support.qa.kid@example.co.uk",  
    "opt_out":"0",  
    "invalid_email":"0",  
    "primary_address":"1"  
  }  
],  
"phone_mobile":"(294) 447-9707",  
"phone_work":"(036) 840-3216",  
"phone_other":"","  
"phone_fax":"","  
"email1":"support.qa.kid@example.co.uk",  
"email2":"section.sugar.section@example.it",  
"invalid_email":false,  
"email_opt_out":false,  
"primary_address_street":"48920 San Carlos Ave",  
"primary_address_street_2":"","  
"primary_address_street_3":"","  
"primary_address_city":"Persistence",  
"primary_address_state":"CA",  
"primary_address_postalcode":"54556",  
"primary_address_country":"USA",  
"alt_address_street":"","  
"alt_address_street_2":"","  
"alt_address_street_3":"","  
"alt_address_city":"","  
"alt_address_state":"","  
"alt_address_postalcode":"","  
"alt_address_country":"","  
"assistant":"","  
"assistant_phone":"","
```

```

"picture": "",
"email_and_name1": "",
"lead_source": "Support Portal User Registration",
"account_name": "Arts & Crafts Inc",
"account_id": "d43243c6-9b8e-2973-ae2-512d09bc34b4",
"opportunity_role_fields": "",
"opportunity_role_id": "",
"opportunity_role": "Technical Advisor",
"reports_to_id": "",
"report_to_name": "",
"portal_name": "GusDales145",
"portal_active": true,
"portal_password": "$1$JxYr6tmM$b.O6.KF42jP46RadSwz0N0",
"portal_password1": "",
"portal_app": "",
"preferred_language": "en_us",
"campaign_id": "",
"campaign_name": "",
"c_accept_status_fields": "",
"m_accept_status_fields": "",
"accept_status_id": "",
"accept_status_name": "",
"sync_contact": "",
"my_favorite": false,
"_acl": {
  "fields": {
    }
  }
}
}
}

```

Change Log

| Version | Change |
|---------|--|
| v10 | Added
/<module>/:record/link/:link_name/:remote_id POST endpoint. |

/TimePeriods GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

| Name | Type | Description | Required |
|--------|--------|--|----------|
| filter | String | The filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query parameters in one of two ways for GET requests:
<ol style="list-style-type: none">1. By specifying individual | False |

| Name | Type | Description | Required |
|-----------|---------|---|----------|
| | | filter arguments as distinct parameters. Example: filter[0][id]=1. By specifying the whole filter as a single JSON-encoded string. Note that this syntax is currently not supported on certain modules. Example: filter=[{"id":"1"}]. | |
| filter_id | String | Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND. | False |
| max_num | Integer | A maximum number of records to return. Default is 20. | False |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |
| fields | String | Comma delimited list of fields to return. Each field may be represented | False |

| | | | |
|------|--------|---|-------|
| | | <p>either by string, or by map containing field name and additional field parameters (applicable to link and collection fields). The fields <code>id</code> and <code>date_modified</code> will always be returned.</p> <p>Example:
 <code>name,account_type,description,{ "name": "opportunities", "fields": ["id", "name", "sales_status"], "order_by": "date_closed: desc" }</code></p> <p>For more details on additional field parameters, see Relate API and Collection API.</p> | |
| view | String | <p>Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list".</p> | False |

| | | Example: record | |
|----------|---------|--|-------|
| order_by | String | How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon.
Example:
name:DESC,account_type:DESC,date_modified:ASC | False |
| q | String | A search expression, will search on this module. Cannot be used at the same time as a filter expression or id. | False |
| deleted | Boolean | Boolean to show deleted records in the result set. | False |

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
  "filter": [
    {
      "name": "Nelson Inc"
    }
  ]
}
```

```

    }
  ]
}

```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```

{
  "filter": [
    {
      "name": {
        "$starts": "Nelson"
      }
    }
  ]
}

```

Below is a list of operation types:

| Operation | Description |
|--------------|--|
| \$equals | Performs an exact match on that field. |
| \$not_equals | Performs an exact match on that field. |
| \$not_equals | Matches on non-matching values. |
| \$starts | Matches on anything that starts with the value. |
| \$ends | Matches anything that ends with the value. |
| \$contains | Matches anything that contains the value |
| \$in | Finds anything where field matches one of the values as specified as an array. |
| \$not_in | Finds anything where field does not |

| Operation | Description |
|------------|--|
| | matches any of the values as specified as an array. |
| \$is_null | Checks if the field is null. This operation does not need a value specified. |
| \$not_null | Checks if the field is not null. This operation does not need a value specified. |
| \$lt | Matches when the field is less than the value. |
| \$lte | Matches when the field is less than or equal to the value. |
| \$gt | Matches when the field is greater than the value. |
| \$gte | Matches when the field is greater than or equal to the value. |

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "\$and" and "\$or".

Example

```
{
  "filter": [
    {
      "$or": [
        {
          "name": "Nelson Inc"
        },
        {
          "name": "Nelson LLC"
        }
      ]
    }
  ]
}
```

```
    }
  ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "\$favorite" and "\$owner".

Example

```
{
  "filter":[
    {
      "$favorite":"_this"
    }
  ]
}
```

Response Arguments

| Name | Type | Description |
|-------------|---------|---|
| next_offset | Integer | Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records. |
| records | Array | An array of results containing matched records. |

Response

```
{
  "next_offset":-1,
  "records":[
    {
      "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
      "name":"Dale Spivey",
      "date_modified":"2013-02-28T05:03:00+00:00",
      "description":"",
      "opportunities": [
        {
          _module: "Opportunities",
          "id": "b0701501-1fab-8ae7-3942-540da93f5017",
          "name": "360 Vacations - 228 Units",
          "date_modified": "2014-09-08T16:05:00+03:00",
          "sales_status": "New"
        },
      ],
      "_acl": {
        "fields": {
        }
      }
    },
    {
      "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
      "name":"Florence Haddock",
      "date_modified":"2013-02-26T19:12:00+00:00",
      "description":"",
      "opportunities": [
        {
          _module: "Opportunities"
          date_modified: "2014-09-08T16:05:00+03:00"
          id: "9ce7c088-8ee4-7cd3-18f1-540da944d4c0"
          name: "360 Vacations - 312 Units"
          sales_status: "New"
        },
      ],
      "_acl": {
        "fields": {
        }
      }
    }
  ]
}
```

```
}
  }
]
}
```

Change Log

| Version | Change |
|---------|--------------------------------------|
| v10 | Added /<module>/filter GET endpoint. |

Last Modified: 05/31/2017 05:30pm

/TimePeriods/current GET

Get the get the current timeperiod

Summary:

This endpoint is used to get the current TimePeriod. If one is not found a 404 is returned

Output Example:

```
{      "id":"e394485a-5889-ea75-84c8-51543bd1a5ba",      "name":"Q1
(01\01\2013 - 03\31\2013)",
"parent_id":"e28efe2d-c51c-be45-3d84-51543b4d2910",
"start_date":"2013-01-01",      "start_date_timestamp":"1356998400",
"end_date":"2013-03-31",      "end_date_timestamp":"1364774399",
"created_by":"1",      "date_entered":"2013-03-28 12:46:36",
"date_modified":"2013-03-28 12:46:36",      "deleted":"0",      "is_fiscal":"0",
"is_fiscal_year":"0",      "leaf_cycle":"1",      "type":"Quarter",
"related_timeperiods":""      }
```

Last Modified: 05/31/2017 05:54pm

/TimePeriods/:date GET

Return a Timeperiod by a given date

Summary:

This endpoint is used to get a TimePeriod for a given date. If one is not found a 404 is returned.

Output Example:

```
{      "id":"e394485a-5889-ea75-84c8-51543bd1a5ba",      "name":"Q1
(01\01\2013 - 03\31\2013)",
"parent_id":"e28efe2d-c51c-be45-3d84-51543b4d2910",
"start_date":"2013-01-01",      "start_date_timestamp":"1356998400",
"end_date":"2013-03-31",      "end_date_timestamp":"1364774399",
"created_by":"1",      "date_entered":"2013-03-28 12:46:36",
"date_modified":"2013-03-28 12:46:36",      "deleted":"0",      "is_fiscal":"0",
"is_fiscal_year":"0",      "leaf_cycle":"1",      "type":"Quarter",
"related_timeperiods":""      }
```

Last Modified: 05/31/2017 05:54pm

/TimePeriods/filter GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long.

Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

| Name | Type | Description | Required |
|--------|--------|--|----------|
| filter | String | <p>The filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query parameters in one of two ways for GET requests:</p> <ol style="list-style-type: none">1. By specifying individual filter arguments as distinct parameters.
Example:
filter[0][id]=1.2. By specifying the whole filter as a single JSON-encoded string. Note that this syntax is currently not supported on certain modules.
Example: | False |

| Name | Type | Description | Required |
|-----------|---------|--|----------------------|
| | | | filter=[{"id":"1"}]. |
| filter_id | String | Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND. | False |
| max_num | Integer | A maximum number of records to return. Default is 20. | False |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |
| fields | String | Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters (applicable to link and collection fields). The fields id and date_modified will always be returned.
Example:
name,account_type,description,{"name":"opportunities","fields":["id","name","sales_status"],"order_by":"date_closed:desc"} | False |

| | | | |
|----------|--------|---|-------|
| | | For more details on additional field parameters, see Relate API and Collection API. | |
| view | String | Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list".
Example: record | False |
| order_by | String | How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon.
Example:
name:DESC,account_type:DESC,date_modified:ASC | False |
| q | String | A search expression, will search on this module. Cannot be used at the same time as a filter expression or id. | False |

| | | | |
|---------|---------|--|-------|
| deleted | Boolean | Boolean to show deleted records in the result set. | False |
|---------|---------|--|-------|

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
  "filter": [
    {
      "name": "Nelson Inc"
    }
  ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
  "filter": [
    {
      "name": {
        "$starts": "Nelson"
      }
    }
  ]
}
```



```

    }
  ]
}

```

Below is a list of operation types:

| Operation | Description |
|--------------|---|
| \$equals | Performs an exact match on that field. |
| \$not_equals | Performs an exact match on that field. |
| \$not_equals | Matches on non-matching values. |
| \$starts | Matches on anything that starts with the value. |
| \$ends | Matches anything that ends with the value. |
| \$contains | Matches anything that contains the value |
| \$in | Finds anything where field matches one of the values as specified as an array. |
| \$not_in | Finds anything where field does not matches any of the values as specified as an array. |
| \$is_null | Checks if the field is null. This operation does not need a value specified. |
| \$not_null | Checks if the field is not null. This operation does not need a value specified. |
| \$lt | Matches when the field is less than the value. |
| \$lte | Matches when the field is less than or equal to the value. |
| \$gt | Matches when the field is greater than the value. |
| \$gte | Matches when the field is greater than or equal to the value. |

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "\$and" and "\$or".

Example

```
{
  "filter":[
    {
      "$or":[
        {
          "name":"Nelson Inc"
        },
        {
          "name":"Nelson LLC"
        }
      ]
    }
  ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "\$favorite" and "\$owner".

Example

```
{
  "filter":[
    {
      "$favorite":"_this"
    }
  ]
}
```

```
  ]  
}
```

Response Arguments

| Name | Type | Description |
|-------------|---------|---|
| next_offset | Integer | Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records. |
| records | Array | An array of results containing matched records. |

Response

```
{  
  "next_offset":-1,  
  "records": [  
    {  
      "id": "fa300a0e-0ad1-b322-9601-512d0983c19a",  
      "name": "Dale Spivey",  
      "date_modified": "2013-02-28T05:03:00+00:00",  
      "description": "",  
      "opportunities": [  
        {  
          _module: "Opportunities",  
          "id": "b0701501-1fab-8ae7-3942-540da93f5017",  
          "name": "360 Vacations - 228 Units",  
          "date_modified": "2014-09-08T16:05:00+03:00",  
          "sales_status": "New"  
        },  
      ],  
      "_acl": {  
        "fields": {
```

```

    }
  },
  {
    "id": "95e17367-9b3d-0e26-22dc-512d0961fedf",
    "name": "Florence Haddock",
    "date_modified": "2013-02-26T19:12:00+00:00",
    "description": "",
    "opportunities": [
      {
        _module: "Opportunities"
        date_modified: "2014-09-08T16:05:00+03:00"
        id: "9ce7c088-8ee4-7cd3-18f1-540da944d4c0"
        name: "360 Vacations - 312 Units"
        sales_status: "New"
      },
    ],
    "_acl": {
      "fields": {
      }
    }
  }
]
}

```

Change Log

| Version | Change |
|---------|--------------------------------------|
| v10 | Added /<module>/filter GET endpoint. |

Last Modified: 05/31/2017 05:29pm

/Users GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

| Name | Type | Description | Required |
|--------|--------|---|----------|
| filter | String | The filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query parameters in one of two ways for GET requests: <ol style="list-style-type: none">1. By specifying individual filter arguments as distinct parameters. Example: filter[0][id]=1.2. By specifying the whole | False |

| Name | Type | Description | Required |
|-----------|---------|---|--|
| | | | filter as a single JSON-encoded string. Note that this syntax is currently not supported on certain modules. Example: filter=[{"id":"1"}]. |
| filter_id | String | Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND. | False |
| max_num | Integer | A maximum number of records to return. Default is 20. | False |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |
| fields | String | Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters (applicable to link and collection fields). The fields id | False |

| | | | |
|-----------------------|--------|--|-------|
| | | <p>and <code>date_modified</code> will always be returned.</p> <p>Example:
 <code>name,account_type,description,{"name":"opportunities","fields":["id","name","sales_status"],"order_by":"date_closed:desc"}</code></p> <p>For more details on additional field parameters, see Relate API and Collection API.</p> | |
| <code>view</code> | String | <p>Instead of defining the <code>fields</code> argument, the <code>view</code> argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the <code>fields</code> argument. Common views are "record" and "list".</p> <p>Example: <code>record</code></p> | False |
| <code>order_by</code> | String | <p>How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon.</p> | False |

| | | | |
|---------|---------|--|-------|
| | | Example:
name:DESC,account_type:DESC,date_modified:ASC | |
| q | String | A search expression, will search on this module. Cannot be used at the same time as a filter expression or id. | False |
| deleted | Boolean | Boolean to show deleted records in the result set. | False |

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
  "filter":[
    {
      "name":"Nelson Inc"
    }
  ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering

on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
  "filter": [
    {
      "name": {
        "$starts": "Nelson"
      }
    }
  ]
}
```

Below is a list of operation types:

| Operation | Description |
|--------------|---|
| \$equals | Performs an exact match on that field. |
| \$not_equals | Performs an exact match on that field. |
| \$not_equals | Matches on non-matching values. |
| \$starts | Matches on anything that starts with the value. |
| \$ends | Matches anything that ends with the value. |
| \$contains | Matches anything that contains the value |
| \$in | Finds anything where field matches one of the values as specified as an array. |
| \$not_in | Finds anything where field does not matches any of the values as specified as an array. |
| \$is_null | Checks if the field is null. This operation does not need a value specified. |
| \$not_null | Checks if the field is not null. This operation does not need a value specified. |
| | |

| Operation | Description |
|-----------|---|
| \$lt | Matches when the field is less than the value. |
| \$lte | Matches when the field is less than or equal to the value. |
| \$gt | Matches when the field is greater than the value. |
| \$gte | Matches when the field is greater than or equal to the value. |

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "\$and" and "\$or".

Example

```
{
  "filter": [
    {
      "$or": [
        {
          "name": "Nelson Inc"
        },
        {
          "name": "Nelson LLC"
        }
      ]
    }
  ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The

current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "\$favorite" and "\$owner".

Example

```
{
  "filter":[
    {
      "$favorite":"_this"
    }
  ]
}
```

Response Arguments

| Name | Type | Description |
|-------------|---------|---|
| next_offset | Integer | Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records. |
| records | Array | An array of results containing matched records. |

Response

```
{
  "next_offset":-1,
  "records":[
    {
      "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
      "name":"Dale Spivey",
```

```

    "date_modified": "2013-02-28T05:03:00+00:00",
    "description": "",
    "opportunities": [
      {
        _module: "Opportunities",
        "id": "b0701501-1fab-8ae7-3942-540da93f5017",
        "name": "360 Vacations - 228 Units",
        "date_modified": "2014-09-08T16:05:00+03:00",
        "sales_status": "New"
      },
    ],
    "_acl": {
      "fields": {
      }
    }
  },
  {
    "id": "95e17367-9b3d-0e26-22dc-512d0961fedf",
    "name": "Florence Haddock",
    "date_modified": "2013-02-26T19:12:00+00:00",
    "description": "",
    "opportunities": [
      {
        _module: "Opportunities"
        date_modified: "2014-09-08T16:05:00+03:00"
        id: "9ce7c088-8ee4-7cd3-18f1-540da944d4c0"
        name: "360 Vacations - 312 Units"
        sales_status: "New"
      },
    ],
    "_acl": {
      "fields": {
      }
    }
  }
]
}

```

Change Log

| Version | Change |
|---------|--------------------------------------|
| v10 | Added /<module>/filter GET endpoint. |

Last Modified: 05/31/2017 05:31pm

/Users/:record DELETE

Delete a User and return its ID

Summary:

This endpoint is used to delete a User. Returns the ID of the deleted User.

Last Modified: 11/30/2016 09:18pm

/Users/:record/freebusy GET

Get a user's FreeBusy schedule

Summary:

This endpoint returns a list of time slots for which the specified person is busy.

Request

GET /Users/:id/freebusy

Response

```
{
  "id": "foo"
  "module": "Users",
  "freebusy": [
    {
```

```

        "start": "2014-08-24T08:45:00-04:00",
        "end": "2014-08-24T09:15:00-04:00"
    },
    {
        "start": "2014-08-30T05:45:00-04:00",
        "end": "2014-08-30T06:15:00-04:00"
    },
    {
        "start": "2014-09-12T15:45:00-04:00",
        "end": "2014-09-12T16:15:00-04:00"
    }
]
}

```

Last Modified: 05/31/2017 05:54pm

/Users/:record/link POST

Overview

Creates relationships to a pre-existing record.

Request Arguments

| Name | Type | Description | Required |
|---------------------|----------|--|----------|
| <relationship link> | <string> | Link between targeted and related records. | True |
| <record ID> | <string> | The name value list of related records. Each item of the list may be either string equal to related item ID, or map containing record ID ("id" key | True |

| Name | Type | Description | Required |
|------|------|-------------|---|
| | | | is required in this case) and addition relationship properties. |

Request

```
{
  link_name: "accounts"
  ids: [
    "da6a3741-2a81-ba7f-f249-512d0932e94e",
    {
      "id": "e689173e-c953-1e14-c215-512d0927e7a2",
      "role": "owner"
    }
  ]
}
```

Response Arguments

| Name | Type | Description |
|-----------------|-------|---------------------------------------|
| record | Array | The record linked to related records. |
| related_records | Array | Records that were associated. |

Response

```
"record": {
  "id": "da6a3741-2a81-ba7f-f249-512d0932e94e",
  "name": "Slender Broadband Inc - 1000 units",
  "date_entered": "2013-02-26T19:12:00+00:00",
```

```
"date_modified": "2013-02-26T19:12:00+00:00",
"modified_user_id": "1",
"modified_by_name": "Administrator",
"created_by": "1",
"created_by_name": "Administrator",
"description": "",
"img": "",
"last_activity_date": "2013-02-28T18:21:00+00:00",
"deleted": false,
"assigned_user_id": "seed_max_id",
"assigned_user_name": "Max Jensen",
"team_name": [
  {
    "id": "East",
    "name": "East",
    "name_2": "",
    "primary": false
  },
  {
    "id": "West",
    "name": "West",
    "name_2": "",
    "primary": true
  }
],
"opportunity_type": "",
"account_name": "Slender Broadband Inc",
"account_id": "181461c6-dc81-1115-1fe0-512d092e8f15",
"campaign_id": "",
"campaign_name": "",
"lead_source": "Campaign",
"amount": "25000",
"base_rate": "1",
"amount_usdollar": "25000",
"currency_id": "-99",
"currency_name": "",
"currency_symbol": "",
"date_closed": "2013-02-27",
"date_closed_timestamp": "1361992480",
"next_step": "",
"sales_stage": "Needs Analysis",
"sales_status": "New",
```

```
"probability": "90",
"best_case": "25000",
"worst_case": "25000",
"commit_stage": "include",
"my_favorite": false,
"_acl": {
  "fields": {
  }
}
},
"related_records": [
  {
    "id": "e689173e-c953-1e14-c215-512d0927e7a2",
    "name": "Gus Dales",
    "date_entered": "2013-02-26T19:12:00+00:00",
    "date_modified": "2013-02-26T19:12:00+00:00",
    "modified_user_id": "1",
    "modified_by_name": "Administrator",
    "created_by": "1",
    "created_by_name": "Administrator",
    "description": "",
    "img": "",
    "deleted": false,
    "assigned_user_id": "seed_sally_id",
    "assigned_user_name": "Sally Bronsen",
    "team_name": [
      {
        "id": "West",
        "name": "West",
        "name_2": "",
        "primary": true
      }
    ],
    "salutation": "",
    "first_name": "Gus",
    "last_name": "Dales",
    "full_name": "Gus Dales",
    "title": "Director Operations",
    "linkedin": "",
    "facebook": "",
    "twitter": "",
```

```
"googleplus": "",
"department": "",
"do_not_call": false,
"phone_home": "(661) 120-2292",
"email": [
  {
    "email_address":
"section.sugar.section@example.it",
    "opt_out": "1",
    "invalid_email": "0",
    "primary_address": "0"
  },
  {
    "email_address": "support.qa.kid@example.co.uk",
    "opt_out": "0",
    "invalid_email": "0",
    "primary_address": "1"
  }
],
"phone_mobile": "(294) 447-9707",
"phone_work": "(036) 840-3216",
"phone_other": "",
"phone_fax": "",
"email1": "support.qa.kid@example.co.uk",
"email2": "section.sugar.section@example.it",
"invalid_email": false,
"email_opt_out": false,
"primary_address_street": "48920 San Carlos Ave",
"primary_address_street_2": "",
"primary_address_street_3": "",
"primary_address_city": "Persistence",
"primary_address_state": "CA",
"primary_address_postalcode": "54556",
"primary_address_country": "USA",
"alt_address_street": "",
"alt_address_street_2": "",
"alt_address_street_3": "",
"alt_address_city": "",
"alt_address_state": "",
"alt_address_postalcode": "",
"alt_address_country": "",
"assistant": "",
```

```
"assistant_phone": "",
"picture": "",
"email_and_name1": "",
"lead_source": "Support Portal User Registration",
"account_name": "Arts & Crafts Inc",
"account_id": "d43243c6-9b8e-2973-ae2-512d09bc34b4",
"opportunity_role_fields": "",
"opportunity_role_id": "",
"opportunity_role": "Technical Advisor",
"reports_to_id": "",
"report_to_name": "",
"portal_name": "GusDales145",
"portal_active": true,
"portal_password": "$1$JxYr6tmM$b.O6.KF42jP46RadSwz0N0",
"portal_password1": "",
"portal_app": "",
"preferred_language": "en_us",
"campaign_id": "",
"campaign_name": "",
"c_accept_status_fields": "",
"m_accept_status_fields": "",
"accept_status_id": "",
"accept_status_name": "",
"sync_contact": "",
"my_favorite": false,
"_acl": {
  "fields": {
  }
}
},
{
  "id": "da6a3741-2a81-ba7f-f249-512d0932e94e",
  "name": "Slender Broadband Inc - 1000 units",
  "date_entered": "2013-02-26T19:12:00+00:00",
  "date_modified": "2013-02-26T19:12:00+00:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
  "description": "",
  "img": "",
  "last_activity_date": "2013-02-28T18:36:00+00:00",
```

```
"deleted": false,
"assigned_user_id": "seed_max_id",
"assigned_user_name": "Max Jensen",
"team_name": [
  {
    "id": "East",
    "name": "East",
    "name_2": "",
    "primary": false
  },
  {
    "id": "West",
    "name": "West",
    "name_2": "",
    "primary": true
  }
],
"opportunity_type": "",
"account_name": "Slender Broadband Inc",
"account_id": "181461c6-dc81-1115-1fe0-512d092e8f15",
"campaign_id": "",
"campaign_name": "",
"lead_source": "Campaign",
"amount": "25000",
"base_rate": "1",
"amount_usdollar": "25000",
"currency_id": "-99",
"currency_name": "",
"currency_symbol": "",
"date_closed": "2013-02-27",
"date_closed_timestamp": "1361992480",
"next_step": "",
"sales_stage": "Needs Analysis",
"sales_status": "New",
"probability": "90",
"best_case": "25000",
"worst_case": "25000",
"commit_stage": "include",
"my_favorite": false,
"_acl": {
  "fields": {
```

```
}
  ]
}
}
```

Change Log

| Version | Change |
|-------------|---|
| v10 (7.6.0) | Added support for additional relationship values. |
| v10 (7.1.5) | Added /<module>/:record/link POST endpoint. |

Last Modified: 05/31/2017 05:26pm

/Users/:record/link/:link_name/:remote_id DELETE

Overview

Deletes an existing relationship between two records.

Request Arguments

This endpoint does not accept any arguments.

Response Arguments

| Name | Type | Description |
|----------------|-------|---|
| record | Array | The record to disassociate from the related record. |
| related_record | Array | The record that was |

| Name | Type | Description |
|------|------|----------------|
| | | disassociated. |

Response

```
{
  "record": {
    "id": "da6a3741-2a81-ba7f-f249-512d0932e94e",
    "name": "Slender Broadband Inc - 1000 units",
    "date_entered": "2013-02-26T19:12:00+00:00",
    "date_modified": "2013-02-26T19:12:00+00:00",
    "modified_user_id": "1",
    "modified_by_name": "Administrator",
    "created_by": "1",
    "created_by_name": "Administrator",
    "description": "",
    "img": "",
    "last_activity_date": "2013-02-28T18:36:00+00:00",
    "deleted": false,
    "assigned_user_id": "seed_max_id",
    "assigned_user_name": "Max Jensen",
    "team_name": [
      {
        "id": "East",
        "name": "East",
        "name_2": "",
        "primary": false
      },
      {
        "id": "West",
        "name": "West",
        "name_2": "",
        "primary": true
      }
    ],
    "opportunity_type": "",
    "account_name": "Slender Broadband Inc",
    "account_id": "181461c6-dc81-1115-1fe0-512d092e8f15",
    "campaign_id": ""
  }
}
```

```
"campaign_name": "",
"lead_source": "Campaign",
"amount": "25000",
"base_rate": "1",
"amount_usdollar": "25000",
"currency_id": "-99",
"currency_name": "",
"currency_symbol": "",
"date_closed": "2013-02-27",
"date_closed_timestamp": "1361992480",
"next_step": "",
"sales_stage": "Needs Analysis",
"sales_status": "New",
"probability": "90",
"best_case": "25000",
"worst_case": "25000",
"commit_stage": "include",
"my_favorite": false,
"_acl": {
  "fields": {
    }
  }
},
"related_record": {
  "id": "e689173e-c953-1e14-c215-512d0927e7a2",
  "name": "Gus Dales",
  "date_entered": "2013-02-26T19:12:00+00:00",
  "date_modified": "2013-02-26T19:12:00+00:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
  "description": "",
  "img": "",
  "deleted": false,
  "assigned_user_id": "seed_sally_id",
  "assigned_user_name": "Sally Bronsen",
  "team_name": [
    {
      "id": "West",
      "name": "West",
    }
  ]
}
```

```
        "name_2": "",
        "primary": true
    }
],
"salutation": "",
"first_name": "Gus",
"last_name": "Dales",
"full_name": "Gus Dales",
"title": "Director Operations",
"linkedin": "",
"facebook": "",
"twitter": "",
"googleplus": "",
"department": "",
"do_not_call": false,
"phone_home": "(661) 120-2292",
"email": [
    {
        "email_address": "section.sugar.section@example.it",
        "opt_out": "1",
        "invalid_email": "0",
        "primary_address": "0"
    },
    {
        "email_address": "support.qa.kid@example.co.uk",
        "opt_out": "0",
        "invalid_email": "0",
        "primary_address": "1"
    }
],
"phone_mobile": "(294) 447-9707",
"phone_work": "(036) 840-3216",
"phone_other": "",
"phone_fax": "",
"email1": "support.qa.kid@example.co.uk",
"email2": "section.sugar.section@example.it",
"invalid_email": false,
"email_opt_out": false,
"primary_address_street": "48920 San Carlos Ave",
"primary_address_street_2": "",
"primary_address_street_3": "",
"primary_address_city": "Persistence",
```

```
"primary_address_state":"CA",
"primary_address_postalcode":"54556",
"primary_address_country":"USA",
"alt_address_street":"","
"alt_address_street_2":"","
"alt_address_street_3":"","
"alt_address_city":"","
"alt_address_state":"","
"alt_address_postalcode":"","
"alt_address_country":"","
"assistant":"","
"assistant_phone":"","
"picture":"","
"email_and_name1":"","
"lead_source":"Support Portal User Registration",
"account_name":"Arts & Crafts Inc",
"account_id":"d43243c6-9b8e-2973-ae2-512d09bc34b4",
"opportunity_role_fields":"","
"opportunity_role_id":"","
"opportunity_role":"","
"reports_to_id":"","
"report_to_name":"","
"portal_name":"GusDales145",
"portal_active":true,
"portal_password":"$1$JxYr6tmM$b.O6.KF42jP46RadSwz0N0",
"portal_password1":"","
"portal_app":"","
"preferred_language":"en_us",
"campaign_id":"","
"campaign_name":"","
"c_accept_status_fields":"","
"m_accept_status_fields":"","
"accept_status_id":"","
"accept_status_name":"","
"sync_contact":"","
"my_favorite":false,
"_acl":{
  "fields":{
    }
  }
}
```

}

Change Log

| Version | Change |
|---------|--|
| v10 | Added
/<module>/:record/link/:link_name/:remote_id DELETE endpoint. |

Last Modified: 05/31/2017 05:25pm

/Users/:record/link/:link_name/:remote_id POST

Overview

Creates a relationship to a pre-existing record.

Query String Parameters

This endpoint does not accept any arguments.

Response Arguments

| Name | Type | Description |
|----------------|-------|--|
| record | Array | The record linked to the related record. |
| related_record | Array | The record that was associated. |

Response

```
{
  "record": {
    "id": "da6a3741-2a81-ba7f-f249-512d0932e94e",
    "name": "Slender Broadband Inc - 1000 units",
    "date_entered": "2013-02-26T19:12:00+00:00",
    "date_modified": "2013-02-26T19:12:00+00:00",
    "modified_user_id": "1",
    "modified_by_name": "Administrator",
    "created_by": "1",
    "created_by_name": "Administrator",
    "description": "",
    "img": "",
    "last_activity_date": "2013-02-28T18:21:00+00:00",
    "deleted": false,
    "assigned_user_id": "seed_max_id",
    "assigned_user_name": "Max Jensen",
    "team_name": [
      {
        "id": "East",
        "name": "East",
        "name_2": "",
        "primary": false
      },
      {
        "id": "West",
        "name": "West",
        "name_2": "",
        "primary": true
      }
    ],
    "opportunity_type": "",
    "account_name": "Slender Broadband Inc",
    "account_id": "181461c6-dc81-1115-1fe0-512d092e8f15",
    "campaign_id": "",
    "campaign_name": "",
    "lead_source": "Campaign",
    "amount": "25000",
    "base_rate": "1",
    "amount_usdollar": "25000",
    "currency_id": "-99",
    "currency_name": "",
    "currency_symbol": ""
  }
}
```

```
"date_closed":"2013-02-27",
"date_closed_timestamp":"1361992480",
"next_step":"",
"sales_stage":"Needs Analysis",
"sales_status":"New",
"probability":"90",
"best_case":"25000",
"worst_case":"25000",
"commit_stage":"include",
"my_favorite":false,
"_acl":{
  "fields":{
    }
  }
},
"related_record":{
  "id":"e689173e-c953-1e14-c215-512d0927e7a2",
  "name":"Gus Dales",
  "date_entered":"2013-02-26T19:12:00+00:00",
  "date_modified":"2013-02-26T19:12:00+00:00",
  "modified_user_id":"1",
  "modified_by_name":"Administrator",
  "created_by":"1",
  "created_by_name":"Administrator",
  "description":"",
  "img":"",
  "deleted":false,
  "assigned_user_id":"seed_sally_id",
  "assigned_user_name":"Sally Bronsen",
  "team_name":[
    {
      "id":"West",
      "name":"West",
      "name_2":"",
      "primary":true
    }
  ],
  "salutation":"",
  "first_name":"Gus",
  "last_name":"Dales",
  "full_name":"Gus Dales",
```

```
"title":"Director Operations",
"linkedin":"","
"facebook":"","
"twitter":"","
"googleplus":"","
"department":"","
"do_not_call":false,
"phone_home":"(661) 120-2292",
"email":[
  {
    "email_address":"section.sugar.section@example.it",
    "opt_out":"1",
    "invalid_email":"0",
    "primary_address":"0"
  },
  {
    "email_address":"support.qa.kid@example.co.uk",
    "opt_out":"0",
    "invalid_email":"0",
    "primary_address":"1"
  }
],
"phone_mobile":"(294) 447-9707",
"phone_work":"(036) 840-3216",
"phone_other":"","
"phone_fax":"","
"email1":"support.qa.kid@example.co.uk",
"email2":"section.sugar.section@example.it",
"invalid_email":false,
"email_opt_out":false,
"primary_address_street":"48920 San Carlos Ave",
"primary_address_street_2":"","
"primary_address_street_3":"","
"primary_address_city":"Persistence",
"primary_address_state":"CA",
"primary_address_postalcode":"54556",
"primary_address_country":"USA",
"alt_address_street":"","
"alt_address_street_2":"","
"alt_address_street_3":"","
"alt_address_city":"","
"alt_address_state":"","
```

```
"alt_address_postalcode": "",
"alt_address_country": "",
"assistant": "",
"assistant_phone": "",
"picture": "",
"email_and_name1": "",
"lead_source": "Support Portal User Registration",
"account_name": "Arts & Crafts Inc",
"account_id": "d43243c6-9b8e-2973-ae2-512d09bc34b4",
"opportunity_role_fields": "",
"opportunity_role_id": "",
"opportunity_role": "Technical Advisor",
"reports_to_id": "",
"report_to_name": "",
"portal_name": "GusDales145",
"portal_active": true,
"portal_password": "$1$JxYr6tmM$b.O6.KF42jP46RadSwz0N0",
"portal_password1": "",
"portal_app": "",
"preferred_language": "en_us",
"campaign_id": "",
"campaign_name": "",
"c_accept_status_fields": "",
"m_accept_status_fields": "",
"accept_status_id": "",
"accept_status_name": "",
"sync_contact": "",
"my_favorite": false,
"_acl": {
  "fields": {
    }
  }
}
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added
/<module>/:record/link/:link_name/:remote_id POST endpoint. |

Last Modified: 05/31/2017 05:25pm

/VCardDownload GET

Overview

Downloads a vCard.

Request Arguments

| Name | Type | Description | Required |
|--------|--------|---|----------|
| id | String | The id of the vcard. | True |
| module | String | The name of the module to get the vcard from. | True |

Request

`http://{site_url}/rest/v10/VCardDownload?id=2c9e5c09-6824-0d14-f5cb-5130321ac3cf&module=Contacts`

Note: GET endpoint parameters are passed in the form of a query string.

Response Arguments

| Name | Type | Description |
|---------------------|---------|-------------------------|
| <vcard information> | String; | Record in vcard format. |

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

Response

```
BEGIN:VCARD
N;CHARSET=utf-8:Leone;Rosemarie;;
FN;CHARSET=utf-8: Rosemarie Leone
BDAY:
TEL;WORK;FAX:
TEL;HOME;CHARSET=utf-8:(692) 586-8287
TEL;CELL;CHARSET=utf-8:(117) 577-0969
TEL;WORK;CHARSET=utf-8:(844) 325-7679
EMAIL;INTERNET;CHARSET=utf-8:support.im@example.it
ADR;WORK;CHARSET=utf-8;;;777 West Filmore Ln;San Mateo;CA;74984;USA
ORG;CHARSET=utf-8:Q.R.&E. Corp;
TITLE;CHARSET=utf-8:Director Sales
END:VCARD
```

Change Log

| Version | Change |
|---------|------------------------------------|
| v10 | Added /VCardDownload GET endpoint. |

Last Modified: 05/31/2017 05:31pm

/bulk POST

Overview

Run API calls in bulk.

Summary

This request will run a sequence of API requests within one query. The requests are executed sequentially and their results are returned as one response. Some requests may return failure code, that does not interrupt the execution of the batch, and the overall request will still be considered successful.

Request Arguments

| Name | Type | Description | Required |
|----------|-------|----------------------|----------|
| requests | Array | The list of requests | True |

Each of the requests can have the following fields:

| Name | Type | Description | Required |
|---------|-------------|--|----------|
| url | String | The request URL, starting with version. | True |
| data | JSON String | The data for the POST/PUT body. Must be a JSON-encoded string. | False |
| headers | Array | The request headers | False |
| method | String | The HTTP method (default is GET) | False |

Request Example

```
{ "requests":  
  [  
    {  
      "url": "/v10/Accounts", "method": "POST", "data": "{ \"name\":
```

```
\ "test123\" }"
  },
  {
    "url": "/v10/Accounts", "method": "GET"
  }
]
}
```

Response

The response will contain an array of response objects, each of them will correspond to the individual request. The following fields are in the response objects:

| Name | Type | Description |
|----------|-----------------|---|
| contents | Array or String | The response contents, can be JSON object or string depending on what the individual request is supposed to return. |
| headers | Array | The response headers |
| status | Integer | HTTP status code of the response. Will be 2XX for successful requests and 4XX or 5XX for errors. |

Response Example

```
[
  {
    "contents": {
      "my_favorite": false,
      "following": true,
      "id": "7d2e21a6-8a76-a74f-bb53-535620211304",
    }
  }
]
```

```

        "name": "test123",
        "date_entered": "2014-04-22T03:56:24-04:00",
        "_module": "Accounts"
    },
    "headers": [],
    "status": 200
},
{
    "contents": {
        "next_offset": -1,
        "records": [
            {
                "my_favorite": false,
                "following": true,
                "id": "7d2e21a6-8a76-a74f-bb53-535620211304",
                "name": "test123",
                "date_entered": "2014-04-22T03:56:24-04:00",
                "date_modified": "2014-04-22T03:56:24-04:00",
            }
        ]
    },
    "headers": [],
    "status": null
}
]

```

Change Log

| Version | Change |
|---------|----------------------------|
| v10 | Added /bulk POST endpoint. |

Last Modified: 05/31/2017 05:32pm

/collection/:collection_name GET

Overview

Lists records from multiple modules at a time.

Summary

This endpoint will return a set of records fetched from multiple modules defined by `:collection_name`. The records will be filtered, sorted and truncated to `max_num` as a single collection. Collections may use a field mapping. For instance, the `due_date` of `Tasks` may be referred to as `date_end`. In this case the alias (`date_end`) should be used in `filter` and `order_by` expressions instead of real field name. Note that response data still contains the original fields for the module.

Request Arguments

| Name | Type | Description | Required |
|-----------------------|---------|---|----------|
| <code>fields</code> | String | See Filter API. If the collection definition uses aliases, then aliases should be used instead of real field names. | False |
| <code>max_num</code> | Integer | See Filter API. | False |
| <code>filter</code> | String | See Filter API. If collection definition uses aliases, then aliases should be used instead of real field names. | False |
| <code>order_by</code> | String | See Filter API. If collection definition uses aliases, then aliases should be used instead of real field names. | False |
| <code>offset</code> | Map | Individual offset for each module which | False |

| Name | Type | Description | Required |
|------|------|--|----------|
| | | the collection consists of. -1 (or any negative value) denotes that the module should be skipped. If module offset is not specified, it defaults to 0. | |

Response Arguments

| Name | Type | Description |
|-------------|-------|---|
| next_offset | Map | The next offset to retrieve records. -1 will be returned for the given module when there are no more records. |
| records | Array | The record result set. |

Response

```
{
  "next_offset": {
    "Calls": 1,
    "Meetings": -1,
    "Tasks": -2,
  },
  "records": [
    {
      "_module": "Calls",
      "id": "8703fbf3-0ffa-c288-8d2c-512f943ecdc3",
      "name": "Discuss review process",
    }
  ]
}
```

```

    "date_end": "2014-02-26T19:12:00+00:00"
  },
  {
    "_module": "Tasks",
    "id": "e1c495cb-af17-1b37-dd66-512f934fe155",
    "name": "Introduce all players",
    "due_date": "2014-02-26T19:12:00+00:00"
  },
  {
    "_module": "Tasks",
    "id": "456b7848-9959-5a64-cd34-512d0938add",
    "name": "Follow-up on proposal",
    "due_date": "2014-02-26T19:12:00+00:00"
  }
]
}

```

Change Log

| Version | Change |
|---------|--|
| v10 | Added /collection/:collection_name GET endpoint. |

Last Modified: 05/31/2017 05:30pm

/connector/dnb/Accounts/bal POST

D&B API -- Post Requests Overview

Interfaces with D&B Direct 2.0 API.

Key to most of the API calls is a D-U-N-S number

A D-U-N-S Number is a unique 9-digit number identifying a business on which a D&B Credit Report has been generated; and once assigned, is neither reused nor assigned to another business.

Query Parameters

| Param | Description |
|-------|--|
| qtype | Query Type: Name of D&B Operation To Invoke Below listed is a table of qtypes. Each of these qtypes represent the different D&B Operations / API's that can be invoked |
| qdata | JSON Object to be POSTed to the D&B operation |

Query Parameters

| QType | Description | QData |
|-----------|--|---|
| cmRequest | Performs Cleanse Match Request Invokes the GetCleanseMatch API | The details of the request elements are provided in the GetCleanseMatch Documentation |
| contacts | Gets Details Of A Contact Invokes the People - Enhanced | <pre>{ 'duns_num' : '', 'contact_id' : '' }</pre> |
| indMap | Converts Industry Code to Hoovers Industry Code Invokes the FindIndustry API | <pre>{ 'industryType' : , 'industryCode' : }</pre> |
| | | |

| | | |
|----------|---|--|
| industry | Mash up of indMap and industry get service.
1. First converts the SIC to HIC
2. Based on HIC invokes the Research - Industry Code API | {

'industryType' : ,

'industryCode' :

} |
|----------|---|--|

Last Modified: 05/31/2017 05:26pm

/connector/dnb/Contacts/bal POST

D&B API -- Post Requests Overview

Interfaces with D&B Direct 2.0 API.

Key to most of the API calls is a D-U-N-S number

A D-U-N-S Number is a unique 9-digit number identifying a business on which a D&B Credit Report has been generated; and once assigned, is neither reused nor assigned to another business.

Query Parameters

| Param | Description |
|-------|--|
| qtype | Query Type: Name of D&B Operation To Invoke Below listed is a table of qtypes. Each of these qtypes represent the different D&B Operations / API's that can be invoked |
| qdata | JSON Object to be POSTed to the D&B operation |

Query Parameters

| QType | Description | QData |
|-----------|---|---|
| cmRequest | Performs Cleanse Match Request Invokes the GetCleanseMatch API | The details of the request elements are provided in the GetCleanseMatch Documentation |
| contacts | Gets Details Of A Contact Invokes the People - Enhanced | <pre>{ 'duns_num' : '', 'contact_id' : '' }</pre> |
| indMap | Converts Industry Code to Hoovers Industry Code Invokes the FindIndustry API | <pre>{ 'industryType' : , 'industryCode' : }</pre> |
| industry | Mash up of indMap and industry get service.
1. First converts the SIC to HIC
2. Based on HIC invokes the Research - Industry Code API | <pre>{ 'industryType' : , 'industryCode' : }</pre> |

Last Modified: 05/31/2017 05:26pm

/connector/dnb/meter GET

D&B API -- Get Requests Overview

Interfaces with D&B Direct 2.0 API.

Key to most of the API calls is a D-U-N-S number

A D-U-N-S Number is a unique 9-digit number identifying a business on which a D&B Credit Report has been generated; and once assigned, is neither reused nor assigned to another business.

Query Parameters

| Param | Description |
|--------|--|
| qtype | Query Type: Name of D&B Operation To Invoke Below listed is a table of qtypes. Each of these qtypes represent the different D&B Operations / API's that can be invoked |
| qparam | Query Parameter: Parameter to be passed to the D&B Operation |

Query Parameters

| QType | Description | QParam | Description |
|--------------|--|---------|---|
| search | Performs a Keyword based search for a company name. Invokes the FindCompany API | keyword | String. Can contain space, api takes care of encoding the special characters. |
| findIndustry | Performs a Keyword based search for a industry name Invokes the FindIndustry API | keyword | String. Can contain space, api takes care of encoding the special characters. |
| profile | Gets Company Information For A | duns | string up to 15 chars. Pattern: |

| | D-U-N-S number | | {([0-9a-zA-Z]*)} |
|------------------|---|------|--|
| competitors | Gets a List Of Competitors For A D-U-N-S number. Invokes the Find Competitors API | duns | string up to 15 chars. Pattern: {([0-9a-zA-Z]*)} |
| financial | Gets Financials For A D-U-N-S number. Invokes the Financial Highlights API | duns | string up to 15 chars. Pattern: {([0-9a-zA-Z]*)} |
| familytree | Gets Corporate Linkage / Family Tree For A D-U-N-S number. Invokes the Corporate Linkage - Enhanced API | duns | string up to 15 chars. Pattern: {([0-9a-zA-Z]*)} |
| firmographic | Gets Firmographic Information For A D-U-N-S number. Invokes the Detailed Company Profile - Standard API | duns | string up to 15 chars. Pattern: {([0-9a-zA-Z]*)} |
| premfirmographic | Gets Firmographic Information For A D-U-N-S number. Invokes the Detailed Company Profile - Premium API | duns | string up to 15 chars. Pattern: {([0-9a-zA-Z]*)} |
| findContacts | Gets A List Of Contacts For A D-U-N-S number. Invokes the FindContact API | duns | string up to 15 chars. Pattern: {([0-9a-zA-Z]*)} |
| news | Gets News and | duns | string up to 15 |

| | | | |
|------------------|--|---|--|
| | Social Media Information for A D-U-N-S number. Invokes the News and Media Service API | | chars. Pattern: {[0-9a-zA-Z]*} |
| litefirmographic | Gets Basic Company Information A D-U-N-S number | duns | string up to 15 chars. Pattern: {[0-9a-zA-Z]*} |
| refreshcheck | Checks when a D-U-N-S number was last updated by D&B. | duns | string up to 15 chars. Pattern: {[0-9a-zA-Z]*} |
| industry | Get Research Information on a Hoovers Industry Code (HIC) Invokes the Research - Industry Code API | {IndustryTypeCode}-{IndustryCodeTypeCode}
Where IndustryCodeTypeCode must be 25838 (representing the Hoovers Industry Type Code) | |

Last Modified: 05/31/2017 05:28pm

/connector/dnb/:module/bulkimport POST

D&B Bulk Import API Overview

Bulk Imports D&B Objects -- Currently supporting Accounts module alone

Request Arguments

| Name | Type | Description | Required |
|------|------|-------------|----------|
|------|------|-------------|----------|

| Name | Type | Description | Required |
|----------|------------|--------------------------------|----------|
| bulkdata | JSON Array | List of objects to be imported | True |

Request Example

```
{
  "bulkdata":
  [
    {
      "name": "Account_1", "duns_num": "1"
    },
    {
      "name": "Account_2", "duns_num": "2"
    }
  ]
}
```

Response

The response will contain a json object reporting count of objects successfully imported and count of objects not imported due to duplicates present in the system

Response Example

```
{
  "importSuccess": 2,
  "duplicates": 0
}
```

Last Modified: 05/31/2017 05:26pm

/connector/dnb/:qtype POST

D&B API -- Post Requests Overview

Interfaces with D&B Direct 2.0 API.

Key to most of the API calls is a D-U-N-S number

A D-U-N-S Number is a unique 9-digit number identifying a business on which a D&B Credit Report has been generated; and once assigned, is neither reused nor assigned to another business.

Query Parameters

| Param | Description |
|-------|--|
| qtype | Query Type: Name of D&B Operation To Invoke Below listed is a table of qtypes. Each of these qtypes represent the different D&B Operations / API's that can be invoked |
| qdata | JSON Object to be POSTed to the D&B operation |

Query Parameters

| QType | Description | QData |
|-----------|--|---|
| cmRequest | Performs Cleanse Match Request Invokes the GetCleanseMatch API | The details of the request elements are provided in the GetCleanseMatch Documentation |
| contacts | Gets Details Of A Contact Invokes the People - Enhanced | {

'duns_num' : '' , |

| | | |
|----------|---|--|
| | | <pre>'contact_id' : '' }</pre> |
| indMap | Converts Industry Code to Hoovers Industry Code
Invokes the Invokes the FindIndustry API | <pre>{ 'industryType' : , 'industryCode' : }</pre> |
| industry | Mash up of indMap and industry get service.
1. First converts the SIC to HIC
2. Based on HIC invokes the Research - Industry Code API | <pre>{ 'industryType' : , 'industryCode' : }</pre> |

Last Modified: 05/31/2017 05:27pm

/connector/dnb/:qtype/:qparam GET

D&B API -- Get Requests Overview

Interfaces with D&B Direct 2.0 API.

Key to most of the API calls is a D-U-N-S number

A D-U-N-S Number is a unique 9-digit number identifying a business on which a D&B Credit Report has been generated; and once assigned, is neither reused nor assigned to another business.

Query Parameters

| Param | Description |
|-------|-------------|
|-------|-------------|

| | |
|--------|--|
| qtype | Query Type: Name of D&B Operation To Invoke Below listed is a table of qtypes. Each of these qtypes represent the different D&B Operations / API's that can be invoked |
| qparam | Query Parameter: Parameter to be passed to the D&B Operation |

Query Parameters

| QType | Description | QParam | Description |
|--------------|---|---------|---|
| search | Performs a Keyword based search for a company name. Invokes the FindCompany API | keyword | String. Can contain space, api takes care of encoding the special characters. |
| findIndustry | Performs a Keyword based search for a industry name Invokes the FindIndustry API | keyword | String. Can contain space, api takes care of encoding the special characters. |
| profile | Gets Company Information For A D-U-N-S number | duns | string up to 15 chars. Pattern: {[0-9a-zA-Z]*} |
| competitors | Gets a List Of Competitors For A D-U-N-S number. Invokes the Find Competitors API | duns | string up to 15 chars. Pattern: {[0-9a-zA-Z]*} |
| financial | Gets Financials For A D-U-N-S number. Invokes the Financial | duns | string up to 15 chars. Pattern: {[0-9a-zA-Z]*} |

| | Highlights API | | |
|------------------|---|------|--|
| familytree | Gets Corporate Linkage / Family Tree For A D-U-N-S number. Invokes the Corporate Linkage - Enhanced API | duns | string up to 15 chars. Pattern: {[0-9a-zA-Z]*} |
| firmographic | Gets Firmographic Information For A D-U-N-S number. Invokes the Detailed Company Profile - Standard API | duns | string up to 15 chars. Pattern: {[0-9a-zA-Z]*} |
| premfirmographic | Gets Firmographic Information For A D-U-N-S number. Invokes the Detailed Company Profile - Premium API | duns | string up to 15 chars. Pattern: {[0-9a-zA-Z]*} |
| findContacts | Gets A List Of Contacts For A D-U-N-S number. Invokes the FindContact API | duns | string up to 15 chars. Pattern: {[0-9a-zA-Z]*} |
| news | Gets News and Social Media Information for A D-U-N-S number. Invokes the News and Media Service API | duns | string up to 15 chars. Pattern: {[0-9a-zA-Z]*} |
| litefirmographic | Gets Basic Company Information A D-U-N-S number | duns | string up to 15 chars. Pattern: {[0-9a-zA-Z]*} |

| | | | |
|--------------|--|---|--|
| refreshcheck | Checks when a D-U-N-S number was last updated by D&B. | duns | string up to 15 chars. Pattern: {[0-9a-zA-Z]*} |
| industry | Get Research Information on a Hoovers Industry Code (HIC) Invokes the Research - Industry Code API | {IndustryTypeCode}-{IndustryCodeTypeCode}
Where IndustryCodeTypeCode must be 25838 (representing the Hoovers Industry Type Code) | |

Last Modified: 05/31/2017 05:26pm

/connectors GET

Overview

Gets general info about connectors.

Request Arguments

This endpoint does not accept any request arguments.

Result Arguments

Response

```
{
  "connectors": {
    "ext_rest_twitter": {
      "testing_enabled": true,
      "test_passed": false,
      "eapm_bean": false,

```

```
"field_mapping": {
  "beans": {
    "Accounts": {
      "name": "name",
      "id": "id"
    },
    "Contacts": {
      "name": "full_name",
      "id": "id"
    },
    "Leads": {
      "name": "account_name",
      "id": "id"
    },
    "Prospects": {
      "name": "account_name",
      "id": "id"
    }
  }
},
{id": "ext_rest_twitter",
name": "Twitter"
},
"ext_eapm_google": {
  "testing_enabled": false,
  "test_passed": false,
  "eapm_bean": false,
  "field_mapping": [],
  "id": "ext_eapm_google",
  "name": "Google"
},
"ext_eapm_gotomeeting": {
  "testing_enabled": false,
  "test_passed": false,
  "eapm_bean": false,
  "field_mapping": [],
  "id": "ext_eapm_gotomeeting",
  "name": "GoToMeeting"
},
"ext_eapm_ibmsmartcloud": {
```

```
    "testing_enabled": false,
    "test_passed": false,
    "eapm_bean": false,
    "field_mapping": [],
    "id": "ext_eapm_ibmsmartcloud",
    "name": "IBM SmartCloud"
  },
  "ext_eapm_webex": {
    "testing_enabled": false,
    "test_passed": false,
    "eapm_bean": false,
    "field_mapping": [],
    "id": "ext_eapm_webex",
    "name": "WebEx"
  },
  "ext_eapm_lotuslive": {
    "testing_enabled": false,
    "test_passed": false,
    "eapm_bean": false,
    "field_mapping": [],
    "id": "ext_eapm_lotuslive",
    "name": "LotusLive"
  },
  "ext_rest_dnb": {
    "testing_enabled": false,
    "test_passed": false,
    "eapm_bean": false,
    "field_mapping": {
      "beans": {
        "Accounts": {
          "name": "name",
          "id": "id"
        }
      }
    },
    "id": "ext_rest_dnb",
    "name": "DNB"
  },
  "_hash": "\u0000d4751e1632fd5d1d2c9069a1f176e6f"
},
```

```
    "_hash": "2cd9132603ead4f79715c69ee98e64f1"
  }
```

Change Log

| Version | Change |
|---------|-----------------------------------|
| v10 | Added /<connectors> GET endpoint. |

Last Modified: 05/31/2017 05:31pm

/connector/twitter/currentUser GET

Overview

Responds with twitter timeline if connector is set up in administration

Request Arguments

| Name | Type | Description | Required |
|-------|---------|-----------------------------------|----------|
| count | Integer | The number of tweets to retrieve. | False |

Response Arguments

| Name | Type | Description |
|------------|--------|-------------|
| <response> | String | |

Response

```
[
  {
    "created_at": "Tue Jun 25 23:45:35 +0000 2013",
    "id": 349674766946414592,
    "id_str": "349674766946414592",
    "text": "this tweet is awesome!",
    "source": "web",
    "truncated": false,
    "in_reply_to_status_id": null,
    "in_reply_to_status_id_str": null,
    "in_reply_to_user_id": 143456975,
    "in_reply_to_user_id_str": "14934565",
    "in_reply_to_screen_name": "testdesk",
    "user": {
      "id": 2630841,
      "id_str": "2630841",
      "name": "SugarCRM",
      "screen_name": "sugarcrm",
      "location": "Cupertino, CA",
      "description": "Everyone Sells. Everyone Supports. Everyone
Connects.",
      "url": "http://t.co/s9ejrBSlki",
      "entities": {
        "url": {
          "urls": [
            {
              "url": "http://t.co/s9ejrBSlki",
              "expanded_url": "http://www.sugarcrm.com",
              "display_url": "sugarcrm.com",
              "indices": [
                0,
                22
              ]
            }
          ]
        }
      },
      "description": {
        "urls": [
```

```
    ]
  }
},
"protected":false,
"followers_count":8761,
"friends_count":6966,
"listed_count":453,
"created_at":"Wed Mar 28 07:16:58 +0000 2007",
"favourites_count":27,
"utc_offset":-28800,
"time_zone":"Pacific Time (US \u0026 Canada)",
"geo_enabled":false,
"verified":false,
"statuses_count":7488,
"lang":"en",
"contributors_enabled":false,
"is_translator":false,
"profile_background_color":"98C7EA",

"profile_background_image_url":"http://a0.twimg.com/profile_backgro
und_images/551274192/sugar-twitter-background.png",

"profile_background_image_url_https":"https://si0.twimg.com/profile
_background_images/551274192/sugar-twitter-background.png",
  "profile_background_tile":false,

"profile_image_url":"http://a0.twimg.com/profile_images/2027721183
/Sugar_cube_RGB_180x180_normal.png",

"profile_image_url_https":"https://si0.twimg.com/profile_images/20
27721183/Sugar_cube_RGB_180x180_normal.png",
  "profile_link_color":"0045AD",
  "profile_sidebar_border_color":"FFFFFF",
  "profile_sidebar_fill_color":"CCEAFF",
  "profile_text_color":"000000",
  "profile_use_background_image":true,
  "default_profile":false,
  "default_profile_image":false,
  "following":null,
  "follow_request_sent":false,
```

```
    "notifications":null
  },
  "geo":null,
  "coordinates":null,
  "place":null,
  "contributors":null,
  "retweet_count":1,
  "favorite_count":0,
  "entities":{
    "hashtags":[

    ],
    "symbols":[

    ],
    "urls":[

    ],
    "user_mentions":[
      {
        "screen_name":"testdesk",
        "name":"testdesk",
        "id":143455,
        "id_str":"ertrt75",
        "indices":[
          0,
          8
        ]
      }
    ],
    "media":[

    ]
  },
  "favorited":false,
  "retweeted":false,
  "possibly_sensitive":false,
  "lang":"en"
}
]
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added /twitter/{twitterId} GET endpoint. |

Last Modified: 05/31/2017 05:28pm

/connector/twitter/:twitterId GET

Overview

Responds with twitter timeline if connector is set up in administration

Request Arguments

| Name | Type | Description | Required |
|-------|---------|-----------------------------------|----------|
| count | Integer | The number of tweets to retrieve. | False |

Response Arguments

| Name | Type | Description |
|------------|--------|-------------|
| <response> | String | |

Response

```
[
  {
    "created_at": "Tue Jun 25 23:45:35 +0000 2013",
    "id": 349674766946414592,
    "id_str": "349674766946414592",
    "text": "this tweet is awesome!",
    "source": "web",
    "truncated": false,
    "in_reply_to_status_id": null,
    "in_reply_to_status_id_str": null,
    "in_reply_to_user_id": 143456975,
    "in_reply_to_user_id_str": "14934565",
    "in_reply_to_screen_name": "testdesk",
    "user": {
      "id": 2630841,
      "id_str": "2630841",
      "name": "SugarCRM",
      "screen_name": "sugarcrm",
      "location": "Cupertino, CA",
      "description": "Everyone Sells. Everyone Supports. Everyone
Connects.",
      "url": "http://t.co/s9ejrBSlki",
      "entities": {
        "url": {
          "urls": [
            {
              "url": "http://t.co/s9ejrBSlki",
              "expanded_url": "http://www.sugarcrm.com",
              "display_url": "sugarcrm.com",
              "indices": [
                0,
                22
              ]
            }
          ]
        }
      },
      "description": {
        "urls": [

```

```
    }
  },
  "protected":false,
  "followers_count":8761,
  "friends_count":6966,
  "listed_count":453,
  "created_at":"Wed Mar 28 07:16:58 +0000 2007",
  "favourites_count":27,
  "utc_offset":-28800,
  "time_zone":"Pacific Time (US \u0026 Canada)",
  "geo_enabled":false,
  "verified":false,
  "statuses_count":7488,
  "lang":"en",
  "contributors_enabled":false,
  "is_translator":false,
  "profile_background_color":"98C7EA",

"profile_background_image_url":"http://a0.twimg.com/profile_backgro
und_images/551274192/sugar-twitter-background.png",

"profile_background_image_url_https":"https://si0.twimg.com/profile
_background_images/551274192/sugar-twitter-background.png",
  "profile_background_tile":false,

"profile_image_url":"http://a0.twimg.com/profile_images/2027721183
/Sugar_cube_RGB_180x180_normal.png",

"profile_image_url_https":"https://si0.twimg.com/profile_images/20
27721183/Sugar_cube_RGB_180x180_normal.png",
  "profile_link_color":"0045AD",
  "profile_sidebar_border_color":"FFFFFF",
  "profile_sidebar_fill_color":"CCEAFF",
  "profile_text_color":"000000",
  "profile_use_background_image":true,
  "default_profile":false,
  "default_profile_image":false,
  "following":null,
  "follow_request_sent":false,
  "notifications":null
```

```
    },
    "geo":null,
    "coordinates":null,
    "place":null,
    "contributors":null,
    "retweet_count":1,
    "favorite_count":0,
    "entities":{
      "hashtags":[

    ],
      "symbols":[

    ],
      "urls":[

    ],
      "user_mentions":[
        {
          "screen_name":"testdesk",
          "name":"testdesk",
          "id":143455,
          "id_str":"ertrt75",
          "indices":[
            0,
            8
          ]
        }
      ],
      "media":[

    ]
    },
    "favorited":false,
    "retweeted":false,
    "possibly_sensitive":false,
    "lang":"en"
  }
]
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added /twitter/{twitterId} GET endpoint. |

Last Modified: 05/31/2017 05:28pm

/css GET

Overview

Runs LessPHP for a platform and a theme and outputs an array of css files. If not found the css files will be compiled.

Request Arguments

| Name | Type | Description | Required |
|-----------|--------|---|---------------------------|
| platform | String | The theme platform - /themes/clients/ ***PLATFORM*** /themeName/. Accepted values are 'base' and 'portal'. | No. (defaults to base) |
| themeName | String | The theme name - /themes/clients/platform/ ***THEMENAME** */. | No. (defaults to default) |

Request

```
{
  platform: 'portal',
  themeName: 'default'
}
```

Response

```
{
  url: [
    'cache/themes/clients/base/default/bootstrap_97c9cf53841dbe471c20d169e3533763.css',
    'cache/themes/clients/base/default/sugar_14c42516aad866b7f80cc896ce5c5406.css',
  ]
}
```

Change Log

| Version | Change |
|---------|----------------------------|
| v10 | Added /<css> GET endpoint. |

Last Modified: 05/31/2017 05:31pm

/css/preview GET

Overview

Runs LessPHP for a platform and a theme and outputs the compiled CSS. It only allows to preview the theme because the file is not saved on the server.

Request Arguments

| Name | Type | Description | Required |
|-----------|---------|---|---------------------------|
| platform | String | The theme platform - /themes/clients/ ***PLATFORM*** /themeName/. Accepted values are 'base' and 'portal'. | No. (defaults to base) |
| themeName | String | The theme name - /themes/clients/platform/ ***THEMENAME** */. | No. (defaults to default) |
| min | Boolean | Whether or not to minify the css | No. (defaults to false) |

Request

```
{
  platform: 'portal',
  themeName: 'default',
  min: false
}
```

Response Arguments

Response

Content-type: text/css

```
.someClass {
  any-property: any-value;
}
```

Change Log

| Version | Change |
|---------|----------------------------------|
| v10 | Added /css/preview GET endpoint. |

Last Modified: 05/31/2017 05:30pm

/globalsearch GET

Overview

Global search

Summary

This endpoint exposes the global search capability using solely the Elasticsearch backend as an alternative to the /search endpoint. This endpoint can be used with a long list of request arguments. Instead of using GET request arguments, all described arguments can be used inside a JSON encoded request body using a the GET request. If your client has no support for GET requests with a body, the POST method can be used as an alternative.

Request Arguments

| Name | Type | Description | Required |
|------|--------|---|----------|
| q | String | The search expression. Multiple terms can be specified at once. All enabled fields will be searched in. The | False |

| Name | Type | Description | Required |
|-------------|---------|--|----------|
| | | <p>results are ordered by relevance which is based on a multitude of settings based on token counts, hit ratio, (weighted) boost values and type of field. Currently no operators are supported in the search expression itself. By refining the search expression more relevant results will be returned as top results. If no search expression is given results are returned based on last modified date.</p> | |
| module_list | String | <p>Comma delimited list of modules to search. If omitted, all search enabled modules will be queried. Note that when consuming the endpoint <code>/:module/globalsearch</code> that this parameter is ignored.
Example:
Accounts,Contacts</p> | False |
| max_num | Integer | A maximum number | False |

| | | | |
|------------|---------|--|-------|
| | | of records to return. Default is 20. | |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |
| highlights | Boolean | Whether or not to return highlighted results. Default is true. | False |
| sort | Array | Define the sort order of the results. By default the results are returned by relevance which is the preferred approach. Using this argument any search enabled field can be used to sort on. Keep in mind the not sorting by relevance may have a negative performance impact.
Example:
{ "date_modified": "desc", "name": "asc" } | False |

Request

```
{
}
```

Response Arguments

| Name | Type | Description |
|-------------|---------|---|
| next_offset | Integer | Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records. |
| records | Array | An array of results containing matched records. |

Response

```
{  
}
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added /globalsearch GET/POST endpoint. |
| v10 | Added /:module/globalsearch GET/POST endpoint. |

Last Modified: 05/31/2017 05:31pm

/globalsearch POST

Overview

Global search

Summary

This endpoint exposes the global search capability using solely the Elasticsearch backend as an alternative to the /search endpoint. This endpoint can be used with a long list of request arguments. Instead of using GET request arguments, all described arguments can be used inside a JSON encoded request body using a the GET request. If your client has no support for GET requests with a body, the POST method can be used as an alternative.

Request Arguments

| Name | Type | Description | Required |
|------|--------|---|----------|
| q | String | The search expression. Multiple terms can be specified at once. All enabled fields will be searched in. The results are ordered by relevance which is based on a multitude of settings based on token counts, hit ratio, (weighted) boost values and type of field. Currently no operators are supported in the search expression | False |

| Name | Type | Description | Required |
|-------------|---------|--|----------|
| | | itself. By refining the search expression more relevant results will be returned as top results. If no search expression is given results are returned based on last modified date. | |
| module_list | String | Comma delimited list of modules to search. If omitted, all search enabled modules will be queried. Note that when consuming the endpoint <code>/:module/globalsearch</code> that this parameter is ignored.
Example:
Accounts,Contacts | False |
| max_num | Integer | A maximum number of records to return. Default is 20. | False |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |
| highlights | Boolean | Whether or not to return highlighted results. Default is true. | False |

| | | | |
|------|-------|--|-------|
| sort | Array | Define the sort order of the results. By default the results are returned by relevance which is the preferred approach. Using this argument any search enabled field can be used to sort on. Keep in mind the not sorting by relevance may have a negative performance impact.
Example:
<code>{"date_modified":"desc","name":"asc"}</code> | False |
|------|-------|--|-------|

Request

```
{
}
```

Response Arguments

| Name | Type | Description |
|-------------|---------|---|
| next_offset | Integer | Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records. |

| Name | Type | Description |
|---------|-------|---|
| records | Array | An array of results containing matched records. |

Response

```
{  
}
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added /globalsearch GET/POST endpoint. |
| v10 | Added /:module/globalsearch GET/POST endpoint. |

Last Modified: 05/31/2017 05:32pm

/help GET

Overview

Fetches the help documentation

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

| Name | Type | Description |
|--------|--------|----------------------------------|
| <html> | String | Returns the html of the help doc |

Response

The HTML for this help document.

Change Log

| Version | Change |
|---------|---------------------------|
| v10 | Added /help GET endpoint. |

Last Modified: 05/31/2017 05:31pm

/help/exceptions GET

Overview

Fetches the documentation on which exceptions are thrown by the API, their HTTP codes, their default messages and a brief description of what the exception means.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

| Name | Type | Description |
|--------|--------|----------------------------------|
| Name | Type | Description |
| <html> | String | Returns the html of the help doc |

Response

The HTML document of the exception help page.

Change Log

| Version | Change |
|---------|--------------------------------------|
| v10 | Added /help/exceptions GET endpoint. |

Last Modified: 05/31/2017 05:30pm

/logger POST

Overview

Logs a message to the Sugar Log

Request Arguments

| Name | Type | Description | Required |
|---------|--------|--|----------|
| level | String | The log level | True |
| message | String | The message to log | True |
| channel | String | Prefix for the log message, defaults to Main | false |

Response Arguments

| Name | Type | Description |
|--------|------|---------------------------|
| status | bool | if the message was logged |

Response

```
{  
  "status": true,  
}
```

Change Log

| Version | Change |
|---------|------------------------------|
| v10 | Added /logger POST endpoint. |

Last Modified: 05/31/2017 05:32pm

/me GET

Overview

Returns the current user object.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

Response User

```
{
  "current_user": {
    "type": "user",
    "id": "1",
    "full_name": "Administrator",
    "timepref": "h:ia",
    "timezone": "America/Los_Angeles",
    "user_name": "admin"
  }
}
```

Response Portal User

```
{
  "current_user": {
    "type": "support_portal",
    "id": "1234-567",
    "user_id": "abcd-123",
    "full_name": "Bill Williamson",
    "timepref": "h:ia",
    "timezone": "America/Denver",
    "user_name": "SupportPortalApi"
  }
}
```

Change Log

| Version | Change |
|---------|-------------------------|
| v10 | Added /me GET endpoint. |

Last Modified: 05/31/2017 05:31pm

/me PUT

Overview

Returns the current user object.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

Response User Example

```
{
  "current_user": {
    "type": "user",
    "id": "1",
    "full_name": "Administrator",
    "timepref": "h:ia",
    "timezone": "America/Los_Angeles",
    "user_name": "admin"
  }
}
```

Response Portal User Example

```
{
  "current_user": {
    "type": "support_portal",
    "id": "1234-567",
    "user_id": "abcd-123",
    "full_name": "Bill Williamson",
  }
}
```

```
    "timepref": "h:ia",
    "timezone": "America/Denver",
    "user_name": "SupportPortalApi"
  }
}
```

Change Log

| Version | Change |
|---------|-------------------------|
| v10 | Added /me PUT endpoint. |

Last Modified: 05/31/2017 05:30pm

/me/following GET

Overview

Returns all of the current users followed records

Request Arguments

limit and offset are available query parameters.

Response Arguments

| Name | Type | Description |
|------|------|-------------|
| | | |

Response

```
{  
}
```

Change Log

| Version | Change |
|---------|-----------------------------------|
| v10 | Added /me/following GET endpoint. |

Last Modified: 05/31/2017 05:30pm

/me/password POST

Overview

Create a new record of a specified type.

Summary

This endpoint allows for verification of the user's password. If client type is support_portal, it will update the corrending Contact. Otherwise, it will update User

Request Arguments

| Name | Type | Description | Required |
|--------------------|--------|------------------------|----------|
| password_to_verify | String | The password to verify | True |

Request

```
{
  "password_to_verify": "mycurrentpassword"
}
```

Response Arguments

| Name | Type | Description |
|-------|---------|---|
| valid | Boolean | Returns the success of the password verification. |

Response

```
{
  "valid": true
}
```

Change Log

| Version | Change |
|---------|-----------------------------------|
| v10 | Added /me/password POST endpoint. |

Last Modified: 05/31/2017 05:29pm

/me/password PUT

Overview

Create a new record of a specified type.

Summary

This endpoint allows for changes to the user's password. If client type is support_portal, it will update the corrending Contact. Otherwise, it will update User

Request Arguments

| Name | Type | Description | Required |
|--------------|--------|-----------------------|----------|
| old_password | String | The current password. | True |
| new_password | String | The new password. | True |

Request

```
{  
  "old_password": "myoldpass",  
  "new_password": "mynewpass"  
}
```

Response Arguments

| Name | Type | Description |
|------------|---------|---|
| valid | Boolean | Returns the success of the password verification. |
| expiration | Date | When the password will expire. |

Response

```
{
  "valid":true,
  "expiration":null
}
```

Change Log

| Version | Change |
|---------|----------------------------------|
| v10 | Added /me/password PUT endpoint. |

Last Modified: 05/31/2017 05:28pm

/me/preference/:preference_name DELETE

Overview

Deletes a specific preference for the current user.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

| Name | Type | Description |
|------|------|-------------|
| | | |

Response

```
{
```

}

Change Log

| Version | Change |
|---------|--|
| v10 | Added
/me/preference/:preference_name
DELETE endpoint. |

Last Modified: 05/31/2017 05:26pm

/me/preference/:preference_name GET

Overview

Returns a specific preference for the current user.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

| Name | Type | Description |
|------|------|-------------|
| | | |

Response

{

}

Change Log

| Version | Change |
|---------|---|
| v10 | Added
/me/preference/:preference_name GET
endpoint. |

Last Modified: 05/31/2017 05:28pm

/me/preference/:preference_name POST

Overview

Creates a preference for the current user.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

| Name | Type | Description |
|------|------|-------------|
| | | |

Response

{

}

Change Log

| Version | Change |
|---------|--|
| v10 | Added
/me/preference/:preference_name POST
endpoint. |

Last Modified: 05/31/2017 05:27pm

/me/preference/:preference_name PUT

Overview

Updates a specific preference for the current user.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

| Name | Type | Description |
|------|------|-------------|
| | | |

Response

{

}

Change Log

| Version | Change |
|---------|---|
| v10 | Added
/me/preference/:preference_name PUT
endpoint. |

Last Modified: 05/31/2017 05:26pm

/me/preferences GET

Overview

Returns all the current user's stored preferences.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

| Name | Type | Description |
|------|------|-------------|
| | | |

Response

{

}

Change Log

| Version | Change |
|---------|-------------------------------------|
| v10 | Added /me/preferences GET endpoint. |

Last Modified: 05/31/2017 05:30pm

/me/preferences PUT

Overview

Mass updates preferences for the user.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

| Name | Type | Description |
|------|------|-------------|
| | | |

Response

{

}

Change Log

| Version | Change |
|---------|-------------------------------------|
| v10 | Added /me/preferences PUT endpoint. |

Last Modified: 05/31/2017 05:28pm

/<module>/Activities GET

Activities on a module's list view

Summary:

This endpoint lists activities across a particular module in the system. It uses a subscription model, and can be queried like a normal bean collection, but without search, ordering or filtering.

Query Parameters:

| Param | Description | Optional |
|---------|---|----------|
| max_num | A maximum number of records to return | Optional |
| offset | How many records to skip over before records are returned | Optional |

Input Example:

This endpoint does not accept any input

Output Example:

```
{
  "next_offset": 20, This will be set to -1 when there are no more
  records after this "page".
  "records": [{
    "id": "22fb8b16-de1d-f1dc-b15b-51240efde177",
    "date_entered": "2013-02-19T23:47:11+00:00",
    "date_modified": "2013-02-19T23:47:11+00:00",
    "created_by": "1",
    "deleted": "0",
    "parent_id": "f5bb0331-2c0f-5c7c-b4db-5123caac0056",
    "parent_type": "Contacts",
    "activity_type": "post", This will be the type of activity
    performed.
    "data": {
      "value": "This is a test post on a contact I'm subscribed
    to."
    },
    "comment_count": 0, This will be set to the total number of
    comments on the post.
    "last_comment": { This will be the last comment on the post,
    which can be used to create a comment model on the frontend.
      "deleted": 0,
      "data": []
    },
    "fields": [],
    "first_name": null,
    "last_name": "Administrator",
    "created_by_name": " Administrator" This will be the
    locale-formatted full name of the user.
  }, ... ]
}
```

Last Modified: 11/30/2016 09:17pm

/<module>/Activities/filter GET

Activities on a module's list view

Summary:

This endpoint lists activities across a particular module in the system. It uses a subscription model, and can be queried like a normal bean collection, but without search, ordering or filtering.

Query Parameters:

| Param | Description | Optional |
|---------|---|----------|
| max_num | A maximum number of records to return | Optional |
| offset | How many records to skip over before records are returned | Optional |

Input Example:

This endpoint does not accept any input

Output Example:

```
{
  "next_offset": 20, This will be set to -1 when there are no more
records after this "page".
  "records": [{
    "id": "22fb8b16-de1d-f1dc-b15b-51240efde177",
    "date_entered": "2013-02-19T23:47:11+00:00",
    "date_modified": "2013-02-19T23:47:11+00:00",
    "created_by": "1",
    "deleted": "0",
    "parent_id": "f5bb0331-2c0f-5c7c-b4db-5123caac0056",
    "parent_type": "Contacts",
```

```

    "activity_type": "post", This will be the type of activity
performed.
    "data": {
        "value": "This is a test post on a contact I'm subscribed
to."
    },
    "comment_count": 0, This will be set to the total number of
comments on the post.
    "last_comment": { This will be the last comment on the post,
which can be used to create a comment model on the frontend.
        "deleted": 0,
        "data": []
    },
    "fields": [],
    "first_name": null,
    "last_name": "Administrator",
    "created_by_name": " Administrator" This will be the
locale-formatted full name of the user.
}, ... ]
}

```

Last Modified: 11/30/2016 09:19pm

/<module>/MassUpdate DELETE

Overview

An API to mass delete records.

Query String Parameters

| Name | Type | Description | Required |
|-----------------------|-------|----------------------------|----------|
| massupdate_param
s | Array | Mass update
parameters. | True |

| Name | Type | Description | Required |
|-----------------------|-------|----------------|----------|
| massupdate_params.uid | Array | A list of ids. | True |

Request

Mass Deleting Records by ID in a Module

```
{
  "massupdate_params": {
    "uid": [
      "ebf22b86-94ea-1601-4f4f-512d09173438",
      "e3b71c55-d96b-80bb-1696-512d09672398"
    ]
  }
}
```

Response Arguments

| Name | Type | Description |
|--------|--------|--|
| Status | String | The status of the mass update. Possible value is 'done'. |

Output Done Example

```
{
  "status": "done"
}
```

Change Log

| Version | Change |
|---------|---|
| v10 | Added /<module>/MassUpdate DELETE endpoint. |

Last Modified: 05/31/2017 05:28pm

/<module>/MassUpdate PUT

Overview

An API to mass update records.

Query String Parameters

| Name | Type | Description | Required |
|----------------------------------|--------------|---|----------|
| massupdate_params | Array | Mass update parameters. | True |
| massupdate_params.uid | Array | A list of ids. | True |
| massupdate_params.[field name] | [field type] | The field to be modified. | False |
| massupdate_params.team_name | Array | Team array. | False |
| massupdate_params.team_name_type | String | Whether to replace or add teams. Possible values are 'add' and 'replace'. | False |

Request

Mass Updating Records by ID in a Module

```
{
  "massupdate_params": {
    "uid": [
      "f22d1955-97d8-387d-9866-512d09cc1520",
      "ef1b40aa-5815-4f8d-e909-512d09617ac8"
    ],
    "department": "Marketing"
  }
}
```

Mass Updating Records with Teams

```
{
  "massupdate_params": {
    "uid": [
      "f22d1955-97d8-387d-9866-512d09cc1520",
      "ef1b40aa-5815-4f8d-e909-512d09617ac8"
    ],
    "team_name": [
      {
        "id": "35eab226-c20d-48f4-4670-512d095c8c6f",
        "primary": true
      },
      {
        "id": "8f640aba-f356-7374-7eb4-512d09745551",
        "primary": false
      }
    ],
    "team_name_type": "replace"
  }
}
```

Mass Add Leads, Contacts or Prospects to a Target List

```
{
  "massupdate_params": {
```

```

"uid": [      /* Leads, Contacts, or Prospects to Add */
  "f3d90a49-14b4-a81c-6cac-526e6c71d33e",
  "f22cde0d-9a20-89d3-ca14-526e6c3c4d08",
  "f15f10bd-1445-5e20-9b5c-526e6ceb71d0"
],
"prospect_lists": [
  /* Prospect List(s) to Add them To */
  "bc5bc249-9c9c-52ad-52b9-526e71f0e18d"
]
}

```

Response Arguments

| Name | Type | Description |
|--------|--------|--|
| Status | String | The status of the mass update. Possible value is 'done'. |

Output Done Example

```

{
  "status": "done"
}

```

Change Log

| Version | Change |
|---------|--|
| v10 | Added /<module>/MassUpdate PUT endpoint. |

Last Modified: 05/31/2017 05:29pm

/<module> GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

| Name | Type | Description | Required |
|--------|--------|---|----------|
| filter | String | The filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query parameters in one of two ways for GET requests:
<ol style="list-style-type: none">1. By specifying individual filter | False |

| Name | Type | Description | Required |
|-----------|---------|--|----------|
| | | arguments as distinct parameters. Example: filter[0][id]=1. By specifying the whole filter as a single JSON-encoded string. Note that this syntax is currently not supported on certain modules. Example: filter=[{"id":"1"}]. | |
| filter_id | String | Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND. | False |
| max_num | Integer | A maximum number of records to return. Default is 20. | False |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |
| fields | String | Comma delimited list of fields to return. Each field | False |

| | | | |
|------|--------|--|-------|
| | | <p>may be represented either by string, or by map containing field name and additional field parameters (applicable to link and collection fields). The fields <code>id</code> and <code>date_modified</code> will always be returned.</p> <p>Example:
 <code>name,account_type,description,{"name":"opportunities","fields":["id","name","sales_status"],"order_by":"date_closed:desc"}</code></p> <p>For more details on additional field parameters, see Relate API and Collection API.</p> | |
| view | String | <p>Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common</p> | False |

| | | | |
|----------|---------|--|-------|
| | | views are "record" and "list".
Example: record | |
| order_by | String | How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon.
Example:
name:DESC,account_type:DESC,date_modified:ASC | False |
| q | String | A search expression, will search on this module. Cannot be used at the same time as a filter expression or id. | False |
| deleted | Boolean | Boolean to show deleted records in the result set. | False |

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
```

```
"filter":[
  {
    "name":"Nelson Inc"
  }
]
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
  "filter":[
    {
      "name":{
        "$starts":"Nelson"
      }
    }
  ]
}
```

Below is a list of operation types:

| Operation | Description |
|---------------------------|---|
| <code>\$equals</code> | Performs an exact match on that field. |
| <code>\$not_equals</code> | Performs an exact match on that field. |
| <code>\$not_equals</code> | Matches on non-matching values. |
| <code>\$starts</code> | Matches on anything that starts with the value. |
| <code>\$ends</code> | Matches anything that ends with the value. |
| <code>\$contains</code> | Matches anything that contains the value |

| Operation | Description |
|------------|---|
| \$in | Finds anything where field matches one of the values as specified as an array. |
| \$not_in | Finds anything where field does not matches any of the values as specified as an array. |
| \$is_null | Checks if the field is null. This operation does not need a value specified. |
| \$not_null | Checks if the field is not null. This operation does not need a value specified. |
| \$lt | Matches when the field is less than the value. |
| \$lte | Matches when the field is less than or equal to the value. |
| \$gt | Matches when the field is greater than the value. |
| \$gte | Matches when the field is greater than or equal to the value. |

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "\$and" and "\$or".

Example

```
{
  "filter": [
    {
      "$or": [
        {
          "name": "Nelson Inc"
        }
      ]
    }
  ]
}
```

```

        },
        {
            "name": "Nelson LLC"
        }
    ]
}

```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "\$favorite" and "\$owner".

Example

```

{
    "filter": [
        {
            "$favorite": "_this"
        }
    ]
}

```

Response Arguments

| Name | Type | Description |
|-------------|---------|---|
| next_offset | Integer | Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records. |

| Name | Type | Description |
|---------|-------|---|
| records | Array | An array of results containing matched records. |

Response

```
{
  "next_offset":-1,
  "records":[
    {
      "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
      "name":"Dale Spivey",
      "date_modified":"2013-02-28T05:03:00+00:00",
      "description":"",
      "opportunities": [
        {
          _module: "Opportunities",
          "id": "b0701501-1fab-8ae7-3942-540da93f5017",
          "name": "360 Vacations - 228 Units",
          "date_modified": "2014-09-08T16:05:00+03:00",
          "sales_status": "New"
        },
      ],
      "_acl": {
        "fields": {
        }
      }
    },
    {
      "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",
      "name":"Florence Haddock",
      "date_modified":"2013-02-26T19:12:00+00:00",
      "description":"",
      "opportunities": [
        {
          _module: "Opportunities"
          date_modified: "2014-09-08T16:05:00+03:00"
        }
      ]
    }
  ]
}
```

```

        id: "9ce7c088-8ee4-7cd3-18f1-540da944d4c0"
        name: "360 Vacations - 312 Units"
        sales_status: "New"
    },
],
"_acl": {
    "fields": {
    }
}
}
]
}

```

Change Log

| Version | Change |
|---------|--------------------------------------|
| v10 | Added /<module>/filter GET endpoint. |

Last Modified: 05/31/2017 05:31pm

/<module> POST

Overview

Create a new record of a specified type.

Request Arguments

| Name | Type | Description | Required |
|----------------|---------------------|--|----------|
| <record field> | <record field type> | The name value list of fields to populate. | True |

| Name | Type | Description | Required |
|------|------|-------------|----------|
|------|------|-------------|----------|

Request

```
{
  "name": "Example Account",
  "account_type": "Customer",
  "description": "My Example Account",
  "meetings": {
    {
      "add": ["21e3385e-404f-b470-407e-54044e3d8023"],
      "create": [
        {
          "name": "Test Meeting"
        }
      ]
    }
  }
}
```

Link fields

It is possible to add record relations to other records and create new related records.

| Action | Type | Description |
|--------|------|---|
| add | List | List of related records ID. See <code>RelateRecordApi</code> for details. |
| create | List | List of name to value arrays of related records. |

Response Arguments

| Name | Type | Description |
|----------------|---------------------|--|
| <record field> | <record field type> | Returns the fields for the newly created record. |

Response

```
{
  "id": "e91b1fa7-1bd8-3c71-be96-512e643f9ca4",
  "name": "Example Account",
  "date_entered": "2013-02-27T19:56:00+00:00",
  "date_modified": "2013-02-27T19:56:00+00:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
  "description": "My Example Account",
  "img": "",
  "deleted": false,
  "assigned_user_id": "",
  "assigned_user_name": "",
  "team_name": [
    {
      "id": 1,
      "name": "Global",
      "name_2": "",
      "primary": true
    }
  ],
  "linkedin": "",
  "facebook": "",
  "twitter": "",
  "googleplus": "",
  "account_type": "Customer",
  "industry": "",
  "annual_revenue": ""
}
```

```
"phone_fax": "",
"billing_address_street": "",
"billing_address_street_2": "",
"billing_address_street_3": "",
"billing_address_street_4": "",
"billing_address_city": "",
"billing_address_state": "",
"billing_address_postalcode": "",
"billing_address_country": "",
"rating": "",
"phone_office": "",
"phone_alternate": "",
"website": "",
"ownership": "",
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "",
"shipping_address_state": "",
"shipping_address_postalcode": "",
"shipping_address_country": "",
"email1": "",
"parent_id": "",
"sic_code": "",
"parent_name": "",
"email_opt_out": "",
"invalid_email": "",
"email": [

],
"campaign_id": "",
"campaign_name": "",
"my_favorite": false,
"_acl": {
  "fields": {
```

```
}  
  }  
}
```

Change Log

| Version | Change |
|-------------|--------------------------------|
| v10 (7.6.0) | Added support for link fields. |
| v10 | Added /<module> POST endpoint. |

Last Modified: 05/31/2017 05:31pm

/<module>/append/:target POST

Overview

Append new node to target node as last child.

Request Arguments

| Name | Type | Description | Required |
|--------|--------|---|----------|
| module | String | The name of sugar module that contains a nested set data and implements the NestedSetInterface. | True |
| target | String | The ID of record that will be used as target to append new node | True |

| Name | Type | Description | Required |
|------|------|-------------|----------|
|------|------|-------------|----------|

Request

```
{  
  "name" : "Children Node 1"  
}
```

Response Arguments

This endpoint does not return any response arguments.

Response

This endpoint returns a newly created bean

```
{ "my_favorite":false, "following":"","id":"59fa8dd7-0f2c-4bfd-364f-54495f77fa3f",  
"name":"Default title", "date_entered":"2014-10-23T23:03:22+03:00",  
"date_modified":"2014-10-23T23:03:22+03:00", "modified_user_id":"1",  
"modified_by_name":"Administrator", "created_by":"1",  
"created_by_name":"Administrator", "doc_owner":"","description":"","  
"deleted":false, "source_id":"","source_type":"","source_meta":"","  
"root":"be9b0c4a-8b78-1ffa-4f14-54481c2f6269", "lft":118, "rgt":119, "level":1,  
"_acl":{"fields":{}}}, "_module":"Categories" }
```

Change Log

| Version | Change |
|---------|---|
| v10 | Added /<module>/append/:target POST endpoint. |

Last Modified: 05/31/2017 05:27pm

/<module>/config GET

Overview

Retrieves the config settings for a given module.

Summary

This endpoint is normally used for the forecasting module.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

| Name | Type | Description |
|----------------|---------------------|---|
| <config field> | <config field type> | The list of config fields will be returned. |

Response

```
{  
  "is_setup":1,  
  "is_upgrade":0,  
  "has_commits":1,  
  "timeperiod_type":"chronological",  
  "timeperiod_interval":"Annual",  
  "timeperiod_leaf_interval":"Quarter",
```

```
"timeperiod_start_date":"2013-01-01",
"timeperiod_shown_forward":2,
"timeperiod_shown_backward":2,
"forecast_ranges":"show_binary",
"buckets_dom":"commit_stage_binary_dom",
"show_binary_ranges":{
  "include":{
    "min":70,
    "max":100
  },
  "exclude":{
    "min":0,
    "max":69
  }
},
"show_buckets_ranges":{
  "include":{
    "min":85,
    "max":100
  },
  "upside":{
    "min":70,
    "max":84
  },
  "exclude":{
    "min":0,
    "max":69
  }
},
"show_custom_buckets_ranges":{
  "include":{
    "min":85,
    "max":100
  },
  "upside":{
    "min":70,
    "max":84
  },
}
```

```
    "exclude": {
      "min": 0,
      "max": 69
    }
  },
  "sales_stage_won": [
    "Closed Won"
  ],
  "sales_stage_lost": [
    "Closed Lost"
  ],
  "show_worksheet_likely": 1,
  "show_worksheet_best": 1,
  "show_worksheet_worst": 0,
  "show_projected_likely": 1,
  "show_projected_best": 1,
  "show_projected_worst": 0,
  "show_forecasts_commit_warnings": 1
}
```

Change Log

| Version | Change |
|---------|--------------------------------------|
| v10 | Added /<module>/config GET endpoint. |

Last Modified: 05/31/2017 05:30pm

/<module>/config POST

Overview

Retrieves the config settings for a given module.

Summary

This endpoint is normally used to manage the forecasting module.

Request Arguments

| Name | Type | Description | Required |
|----------------|---------------------|--------------------------------------|----------|
| <config field> | <config field type> | The list of config fields to update. | False |

Request

```
{  
  "MyConfigOption": "MyConfigValue"  
}
```

Response Arguments

| Name | Type | Description |
|----------------|---------------------|---|
| <config field> | <config field type> | The list of config fields will be returned. |

Response

```
{  
  "MyConfigOption": "MyConfigValue"  
}
```

Change Log

| Version | Change |
|---------|---------------------------------------|
| v10 | Added /<module>/config POST endpoint. |

Last Modified: 05/31/2017 05:29pm

/<module>/config PUT

Overview

Retrieves the config settings for a given module.

Summary

This endpoint is normally used to manage the forecasting module.

Request Arguments

| Name | Type | Description | Required |
|----------------|---------------------|--------------------------------------|----------|
| <config field> | <config field type> | The list of config fields to update. | False |

Request

```
{
```

```
    "MyConfigOption": "MyConfigValue"  
  }
```

Response Arguments

| Name | Type | Description |
|----------------|---------------------|---|
| <config field> | <config field type> | The list of config fields will be returned. |

Response

```
{  
  "MyConfigOption": "MyConfigValue"  
}
```

Change Log

| Version | Change |
|---------|--------------------------------------|
| v10 | Added /<module>/config PUT endpoint. |

Last Modified: 05/31/2017 05:29pm

/<module>/count GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

| Name | Type | Description | Required |
|--------|--------|---|----------|
| filter | String | The filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query parameters in one of two ways for GET requests: <ol style="list-style-type: none">1. By specifying individual filter arguments as distinct parameters. Example: filter[0][id]=1.2. By specifying | False |

| Name | Type | Description | Required |
|-----------|---------|---|--|
| | | | the whole filter as a single JSON-encoded string. Note that this syntax is currently not supported on certain modules.
Example:
filter=[{"id":"1"}]. |
| filter_id | String | Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND. | False |
| max_num | Integer | A maximum number of records to return. Default is 20. | False |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |
| fields | String | Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and | False |

| | | | |
|------|--------|---|-------|
| | | <p>additional field parameters (applicable to link and collection fields). The fields <code>id</code> and <code>date_modified</code> will always be returned.</p> <p>Example:
 <code>name,account_type,description,{"name":"opportunities","fields":["id","name","sales_status"],"order_by":"date_closed:desc"}</code></p> <p>For more details on additional field parameters, see Relate API and Collection API.</p> | |
| view | String | <p>Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record"</p> | False |

| | | | |
|----------|---------|--|-------|
| | | and "list".
Example: record | |
| order_by | String | How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon.
Example:
name:DESC,account_type:DESC,date_modified:ASC | False |
| q | String | A search expression, will search on this module. Cannot be used at the same time as a filter expression or id. | False |
| deleted | Boolean | Boolean to show deleted records in the result set. | False |

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
  "filter": [
    {
      "name": "Nelson Inc"
    }
  ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
  "filter": [
    {
      "name": {
        "$starts": "Nelson"
      }
    }
  ]
}
```

Below is a list of operation types:

| Operation | Description |
|---------------------------|---|
| <code>\$equals</code> | Performs an exact match on that field. |
| <code>\$not_equals</code> | Performs an exact match on that field. |
| <code>\$not_equals</code> | Matches on non-matching values. |
| <code>\$starts</code> | Matches on anything that starts with the value. |

| Operation | Description |
|------------|---|
| \$ends | Matches anything that ends with the value. |
| \$contains | Matches anything that contains the value |
| \$in | Finds anything where field matches one of the values as specified as an array. |
| \$not_in | Finds anything where field does not matches any of the values as specified as an array. |
| \$is_null | Checks if the field is null. This operation does not need a value specified. |
| \$not_null | Checks if the field is not null. This operation does not need a value specified. |
| \$lt | Matches when the field is less than the value. |
| \$lte | Matches when the field is less than or equal to the value. |
| \$gt | Matches when the field is greater than the value. |
| \$gte | Matches when the field is greater than or equal to the value. |

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "\$and" and "\$or".

Example

```
{
  "filter":[
    {
      "$or":[
        {
          "name":"Nelson Inc"
        },
        {
          "name":"Nelson LLC"
        }
      ]
    }
  ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "\$favorite" and "\$owner".

Example

```
{
  "filter":[
    {
      "$favorite":"_this"
    }
  ]
}
```

Response Arguments

| Name | Type | Description |
|-------------|---------|---|
| next_offset | Integer | Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records. |
| records | Array | An array of results containing matched records. |

Response

```
{
  "next_offset":-1,
  "records":[
    {
      "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
      "name":"Dale Spivey",
      "date_modified":"2013-02-28T05:03:00+00:00",
      "description":"",
      "opportunities": [
        {
          _module: "Opportunities",
          "id": "b0701501-1fab-8ae7-3942-540da93f5017",
          "name": "360 Vacations - 228 Units",
          "date_modified": "2014-09-08T16:05:00+03:00",
          "sales_status": "New"
        },
      ],
      "_acl": {
        "fields": {
        }
      }
    }
  ]
}
```

```

    }
  },
  {
    "id": "95e17367-9b3d-0e26-22dc-512d0961fedf",
    "name": "Florence Haddock",
    "date_modified": "2013-02-26T19:12:00+00:00",
    "description": "",
    "opportunities": [
      {
        _module: "Opportunities"
        date_modified: "2014-09-08T16:05:00+03:00"
        id: "9ce7c088-8ee4-7cd3-18f1-540da944d4c0"
        name: "360 Vacations - 312 Units"
        sales_status: "New"
      },
    ],
    "_acl": {
      "fields": {
      }
    }
  }
]
}

```

Change Log

| Version | Change |
|---------|--------------------------------------|
| v10 | Added /<module>/filter GET endpoint. |

Last Modified: 05/31/2017 05:30pm

/<module>/duplicateCheck POST

Overview

Runs a duplicate check against specified data.

Request Arguments

| Name | Type | Description | Required |
|----------------|---------------------|--|----------|
| <record field> | <record field type> | The name value list of fields to populate. | True |

Request

```
{  
  "name": "Airline"  
}
```

Response Arguments

| Name | Type | Description |
|----------------|---------------------|--|
| <record field> | <record field type> | Returns the potential duplicate records. |

Response

```
{
```

```
"next_offset":-1,
"records":[
  {
    "id":"c4c26496-5dbc-67dd-bf5c-512d0923889e",
    "name":"Airline Maintenance Co",
    "date_entered":"2013-02-26T19:12:00+00:00",
    "date_modified":"2013-02-26T19:12:00+00:00",
    "modified_user_id":"1",
    "modified_by_name":"Administrator",
    "created_by":"1",
    "created_by_name":"Administrator",
    "description":"",
    "img":"",
    "last_activity_date":"2013-02-26T19:12:00+00:00",
    "deleted":false,
    "assigned_user_id":"seed_sally_id",
    "assigned_user_name":"Sally Bronsen",
    "team_name":[
      {
        "id":"East",
        "name":"East",
        "name_2":"",
        "primary":false
      },
      {
        "id":1,
        "name":"Global",
        "name_2":"",
        "primary":false
      },
      {
        "id":"West",
        "name":"West",
        "name_2":"",
        "primary":true
      }
    ],
    "linkedin":""
  }
]
```

```
"facebook":"","  
"twitter":"","  
"googleplus":"","  
"account_type":"Customer",  
"industry":"Other",  
"annual_revenue":"","  
"phone_fax":"","  
"billing_address_street":"123 Anywhere Street",  
"billing_address_street_2":"","  
"billing_address_street_3":"","  
"billing_address_street_4":"","  
"billing_address_city":"Sunnyvale",  
"billing_address_state":"CA",  
"billing_address_postalcode":"48566",  
"billing_address_country":"USA",  
"rating":"","  
"phone_office":"(648) 452-3486",  
"phone_alternate":"","  
"website":"www.kidqa.it",  
"ownership":"","  
"employees":"","  
"ticker_symbol":"","  
"shipping_address_street":"123 Anywhere Street",  
"shipping_address_street_2":"","  
"shipping_address_street_3":"","  
"shipping_address_street_4":"","  
"shipping_address_city":"Sunnyvale",  
"shipping_address_state":"CA",  
"shipping_address_postalcode":"48566",  
"shipping_address_country":"USA",  
"email1":"vegan.im@example.tw",  
"parent_id":"","  
"sic_code":"","  
"parent_name":"","  
"email_opt_out":false,  
"invalid_email":false,  
"email":[  
  {
```

```

        "email_address": "vegan.im@example.tw",
        "opt_out": "0",
        "invalid_email": "0",
        "primary_address": "1"
    },
    {
        "email_address": "phone85@example.tv",
        "opt_out": "0",
        "invalid_email": "0",
        "primary_address": "0"
    }
],
"campaign_id": "",
"campaign_name": "",
"my_favorite": false,
"_acl": {
    "fields": {

    }
},
"duplicate_check_rank": 0
}
]
}

```

Change Log

| Version | Change |
|---------|---|
| v10 | Added /<module>/duplicateCheck POST endpoint. |

Last Modified: 05/31/2017 05:29pm

/<module>/enum/:field GET

Overview

Retrieves the enum values for a specific field.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

| Name | Type | Description |
|---------------|-------|--------------------------------------|
| <list values> | Array | Returns the enum values for a field. |

Response

```
{
  "": "",
  "Analyst": "Analyst",
  "Competitor": "Competitor",
  "Customer": "Customer",
  "Integrator": "Integrator",
  "Investor": "Investor",
  "Partner": "Partner",
  "Press": "Press",
  "Prospect": "Prospect",
  "Reseller": "Reseller",
  "Other": "Other"
}
```

Change Log

| Version | Change |
|---------|---|
| v10 | Added /<module>/enum/:field GET endpoint. |

Last Modified: 05/31/2017 05:28pm

/<module>/export/:record_list_id GET

Overview

Returns a record set in CSV format along with HTTP headers to indicate content type.

Request Arguments

| Name | Type | Description | Required |
|--------|-------|---|----------|
| uid | Array | A list of bean ids. | False |
| filter | Array | The filter expression. More information on filter expressions can be found in /<module>/filter. | False |
| sample | Array | Indicates whether to download sample data. | False |

| Name | Type | Description | Required |
|------|------|-------------|----------|
|------|------|-------------|----------|

Request

Exporting Records by Specific IDs

```
{
  "uid": "d43243c6-9b8e-2973-ae2-512d09bc34b4"
}
```

Exporting Records by a List of IDs

```
{
  "uid": [
    "d43243c6-9b8e-2973-ae2-512d09bc34b4",
    "b3e87a3f-cd8f-7b86-467a-512d09e8d240"
  ]
}
```

Exporting Records Using a Filter

```
{
  "filter": [
    {
      "name": "airline"
    }
  ]
}
```

Exporting a Sample Result Set

```
{
  "sample":true
}
```

Response Arguments

| Name | Type | Description |
|--------------|--------|--|
| <csv export> | String | Returns the selected records in CSV format with the content headers. |

Response

```
result: HTTP/1.1 200 OK
Date: Fri, 01 Mar 2013 04:05:55 GMT
Server: Apache/2.2.22 (Unix) DAV/2 PHP/5.3.17
X-Powered-By: PHP/5.3.17
Expires: Mon, 26 Jul 1997 05:00:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0,
pre-check=0
Pragma: cache
Content-Disposition: attachment; filename=Accounts.csv
Content-transfer-encoding: binary
Last-Modified: Fri, 01 Mar 2013 04:05:55 GMT
Cache-Control: post-check=0, pre-check=0
Content-Length: 1080
Connection: close
Content-Type: application/octet-stream; charset=UTF-8
```

```
"Name", "ID", "Website", "Email Address", "Office Phone", "Alternate
```

Phone", "Fax", "Billing Street", "Billing City", "Billing State", "Billing Postal Code", "Billing Country", "Shipping Street", "Shipping City", "Shipping State", "Shipping Postal Code", "Shipping Country", "Description", "Type", "Industry", "Annual Revenue", "Employees", "SIC Code", "Ticker Symbol", "Parent Account ID", "Ownership", "Campaign ID", "Rating", "Assigned User Name", "Assigned To", "Team ID", "Teams", "Team Set ID", "Date Created", "Date Modified", "Modified By", "Created By", "Deleted", "Image", "last_activity_date", "Linkedin Company ID", "Facebook Account", "Twitter Account", "Google Plus ID" "Arts & Crafts Inc", "d43243c6-9b8e-2973-ae2-512d09bc34b4", "", "", "(052) 034-1853", "", "", "777 West Filmore Ln", "Santa Monica", "CA", "35354", "USA", "777 West Filmore Ln", "Santa Monica", "CA", "35354", "USA", "", "Customer", "Transportation", "", "", "", "", "", "sally", "seed_sally_id", "West", "West", "West", "02/26/2013 07:12 pm", "02/26/2013 07:12 pm", "1", "1", "0", "", "02/26/2013 07:12 pm", "", "", "", ""

Change Log

| Version | Change |
|---------|--------------------------------------|
| v10 | Added /<module>/export GET endpoint. |

Last Modified: 05/31/2017 05:28pm

/<module>/file/vcard_import POST

Overview

Imports a person record from a vcard.

Request Arguments

| Name | Type | Description | Required |
|--------|--------|---|----------|
| format | String | The data format. Currently accepts 'sugar-html-json'. | True |
| module | String | The name of the module to import the vcard to. | True |
| file | String | The vcard contents. | True |

Request

```
{  
  "format": "sugar-html-json",  
  "module": "Contacts"  
  "file": "@\path\to\ExampleContact.vcf"  
}
```

Response Arguments

| Name | Type | Description |
|------|--------|-------------------------------|
| file | String | The id of the imported vcard. |

Response

```
{
  "file": "2c9e5c09-6824-0d14-f5cb-5130321ac3cf"
}
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added /<module>/file/vcard_import POST endpoint. |

Last Modified: 05/31/2017 05:27pm

/<module>/filter GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

| Name | Type | Description | Required |
|--------|--------|--|----------|
| filter | String | <p>The filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query parameters in one of two ways for GET requests:</p> <ol style="list-style-type: none">1. By specifying individual filter arguments as distinct parameters.
Example:
filter[0][id]=1.2. By specifying the whole filter as a single JSON-encoded string. Note that this syntax is currently not supported on certain modules.
Example: | False |

| Name | Type | Description | Required |
|-----------|---------|--|----------------------|
| | | | filter=[{"id":"1"}]. |
| filter_id | String | Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND. | False |
| max_num | Integer | A maximum number of records to return. Default is 20. | False |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |
| fields | String | Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters (applicable to link and collection fields). The fields id and date_modified will always be returned.
Example:
name,account_type, | False |

| | | | |
|----------|--------|---|-------|
| | | <pre>description, {"name": "opportunities", "fields": ["id", "name", "sales_status"], "order_by": "date_closed: desc"}</pre> <p>For more details on additional field parameters, see Relate API and Collection API.</p> | |
| view | String | <p>Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list".
Example: record</p> | False |
| order_by | String | <p>How to sort the returned records, in a comma delimited list with the direction appended to the column name</p> | False |

| | | | |
|---------|---------|--|-------|
| | | after a colon.
Example:
name:DESC,account_type:DESC,date_modified:ASC | |
| q | String | A search expression, will search on this module. Cannot be used at the same time as a filter expression or id. | False |
| deleted | Boolean | Boolean to show deleted records in the result set. | False |

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
  "filter":[
    {
      "name":"Nelson Inc"
    }
  ]
}
```

```
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
  "filter":[
    {
      "name":{
        "$starts":"Nelson"
      }
    }
  ]
}
```

Below is a list of operation types:

| Operation | Description |
|--------------|---|
| \$equals | Performs an exact match on that field. |
| \$not_equals | Performs an exact match on that field. |
| \$not_equals | Matches on non-matching values. |
| \$starts | Matches on anything that starts with the value. |
| \$ends | Matches anything that ends with the value. |
| \$contains | Matches anything that contains the value |

| Operation | Description |
|------------|---|
| \$in | Finds anything where field matches one of the values as specified as an array. |
| \$not_in | Finds anything where field does not matches any of the values as specified as an array. |
| \$is_null | Checks if the field is null. This operation does not need a value specified. |
| \$not_null | Checks if the field is not null. This operation does not need a value specified. |
| \$lt | Matches when the field is less than the value. |
| \$lte | Matches when the field is less than or equal to the value. |
| \$gt | Matches when the field is greater than the value. |
| \$gte | Matches when the field is greater than or equal to the value. |

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "\$and" and "\$or".

Example

```
{
  "filter": [
```

```
{
  "$or": [
    {
      "name": "Nelson Inc"
    },
    {
      "name": "Nelson LLC"
    }
  ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "\$favorite" and "\$owner".

Example

```
{
  "filter": [
    {
      "$favorite": "_this"
    }
  ]
}
```

Response Arguments

| Name | Type | Description |
|-------------|---------|---|
| next_offset | Integer | Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records. |
| records | Array | An array of results containing matched records. |

Response

```
{
  "next_offset": -1,
  "records": [
    {
      "id": "fa300a0e-0ad1-b322-9601-512d0983c19a",
      "name": "Dale Spivey",
      "date_modified": "2013-02-28T05:03:00+00:00",
      "description": "",
      "opportunities": [
        {
          _module: "Opportunities",
          "id": "b0701501-1fab-8ae7-3942-540da93f5017",
          "name": "360 Vacations - 228 Units",
          "date_modified": "2014-09-08T16:05:00+03:00",
          "sales_status": "New"
        }
      ],
      "_acl": {
        "fields": {
        }
      }
    }
  ]
}
```

```

    }
  },
  {
    "id": "95e17367-9b3d-0e26-22dc-512d0961fedf",
    "name": "Florence Haddock",
    "date_modified": "2013-02-26T19:12:00+00:00",
    "description": "",
    "opportunities": [
      {
        _module: "Opportunities"
        date_modified: "2014-09-08T16:05:00+03:00"
        id: "9ce7c088-8ee4-7cd3-18f1-540da944d4c0"
        name: "360 Vacations - 312 Units"
        sales_status: "New"
      },
    ],
    "_acl": {
      "fields": {
      }
    }
  }
]
}

```

Change Log

| Version | Change |
|---------|--------------------------------------|
| v10 | Added /<module>/filter GET endpoint. |

Last Modified: 05/31/2017 05:30pm

/<module>/filter POST

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

| Name | Type | Description | Required |
|-----------|---------|---|----------|
| filter | String | The filter expression. Filter expressions are explained below. | False |
| filter_id | String | Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND. | False |
| max_num | Integer | A maximum number of records to return. Default is | False |

| Name | Type | Description | Required |
|----------|---------|--|----------|
| | | 20. | |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |
| fields | String | Comma delimited list of fields to return. The field date_modified will always be returned. Example: name,account_type,description | False |
| view | String | Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list". Example: record | False |
| order_by | String | How to sort the | False |

| | | | |
|--|--|--|--|
| | | returned records, in a comma delimited list with the direction appended to the column name after a colon.
Example:
name:DESC,account_type:DESC,date_modified:ASC | |
|--|--|--|--|

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
  "filter": [
    {
      "name": "Nelson Inc"
    }
  ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering

on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
  "filter": [
    {
      "name": {
        "$starts": "Nelson"
      }
    }
  ]
}
```

Below is a list of operation types:

| Operation | Description |
|---------------------------|---|
| <code>\$equals</code> | Performs an exact match on that field. |
| <code>\$not_equals</code> | Performs an exact match on that field. |
| <code>\$not_equals</code> | Matches on non-matching values. |
| <code>\$starts</code> | Matches on anything that starts with the value. |
| <code>\$ends</code> | Matches anything that ends with the value. |
| <code>\$contains</code> | Matches anything that contains the value |
| <code>\$in</code> | Finds anything where field matches one of the values as specified as an array. |
| <code>\$not_in</code> | Finds anything where field does not matches any of the values as specified as an array. |
| <code>\$is_null</code> | Checks if the field is null. This operation |

| Operation | Description |
|------------|--|
| | does not need a value specified. |
| \$not_null | Checks if the field is not null. This operation does not need a value specified. |
| \$lt | Matches when the field is less than the value. |
| \$lte | Matches when the field is less than or equal to the value. |
| \$gt | Matches when the field is greater than the value. |
| \$gte | Matches when the field is greater than or equal to the value. |

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "\$and" and "\$or".

Example

```
{
  "filter": [
    {
      "$or": [
        {
          "name": "Nelson Inc"
        },
        {
          "name": "Nelson LLC"
        }
      ]
    }
  ]
}
```

```
    }
  ]
}
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "\$favorite" and "\$owner".

Example

```
{
  "filter":[
    {
      "$favorite":"_this"
    }
  ]
}
```

Response Arguments

| Name | Type | Description |
|-------------|---------|--|
| next_offset | Integer | Displays the next offset for retrieval of additional results. -1 will be returned when there are no more |

| Name | Type | Description |
|---------|-------|---|
| | | records. |
| records | Array | An array of results containing matched records. |

Response

```
{
  "next_offset":-1,
  "records":[
    {
      "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
      "name":"Dale Spivey",
      "date_entered":"2013-02-26T19:12:00+00:00",
      "date_modified":"2013-02-28T05:03:00+00:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "description":"",
      "img":"",
      "deleted":false,
      "assigned_user_id":"seed_sally_id",
      "assigned_user_name":"Sally Bronsen",
      "team_name":[
        {
          "id":"East",
          "name":"East",
          "name_2":"",
          "primary":false
        }
      ],
    }
  ]
}
```

```
        "id":1,
        "name":"Global",
        "name_2":"",
        "primary":false
    },
    {
        "id":"West",
        "name":"West",
        "name_2":"",
        "primary":true
    }
],
"salutation":"",
"first_name":"Dale",
"last_name":"Spivey",
"full_name":"Dale Spivey",
"title":"VP Operations",
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(523) 825-4311",
"email":[
    {
        "email_address":"sugar.dev.sugar@example.co.jp",
        "opt_out":"0",
        "invalid_email":"0",
        "primary_address":"1"
    },
    {
        "email_address":"the.support@example.biz",
        "opt_out":"0",
        "invalid_email":"0",
        "primary_address":"0"
    }
]
```

```
    }
  ],
  "phone_mobile": "(373) 861-0757",
  "phone_work": "(212) 542-9596",
  "phone_other": "",
  "phone_fax": "",
  "email1": "sugar.dev.sugar@example.co.jp",
  "email2": "the.support@example.biz",
  "invalid_email": false,
  "email_opt_out": false,
  "primary_address_street": "345 Sugar Blvd.",
  "primary_address_street_2": "",
  "primary_address_street_3": "",
  "primary_address_city": "Denver",
  "primary_address_state": "CA",
  "primary_address_postalcode": "87261",
  "primary_address_country": "USA",
  "alt_address_street": "",
  "alt_address_street_2": "",
  "alt_address_street_3": "",
  "alt_address_city": "",
  "alt_address_state": "",
  "alt_address_postalcode": "",
  "alt_address_country": "",
  "assistant": "",
  "assistant_phone": "",
  "picture": "",
  "email_and_name1": "",
  "lead_source": "Campaign",
  "account_name": "Smallville Resources Inc",
  "account_id": "d5db6292-5c24-eb61-e202-512d09f0134e",
  "opportunity_role_fields": "",
  "opportunity_role_id": "",
  "opportunity_role": "",
  "reports_to_id": "",
  "report_to_name": "",
```

```
"portal_name": "DaleSpivey97",
"portal_active": true,
"portal_password": "$1$yKMAONHM$Y5S.8CY.WZCZCwfGD1a1Q\/",
"portal_password1": "",
"portal_app": "",
"preferred_language": "en_us",
"campaign_id": "",
"campaign_name": "",
"c_accept_status_fields": "",
"m_accept_status_fields": "",
"accept_status_id": "",
"accept_status_name": "",
"sync_contact": "",
"my_favorite": false,
"_acl": {
  "fields": {
    }
  }
},
{
  "id": "95e17367-9b3d-0e26-22dc-512d0961fedf",
  "name": "Florence Haddock",
  "date_entered": "2013-02-26T19:12:00+00:00",
  "date_modified": "2013-02-26T19:12:00+00:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
  "description": "",
  "img": "",
  "deleted": false,
  "assigned_user_id": "seed_sally_id",
  "assigned_user_name": "Sally Bronsen",
  "team_name": [
    {
```

```
        "id": "East",
        "name": "East",
        "name_2": "",
        "primary": false
    },
    {
        "id": 1,
        "name": "Global",
        "name_2": "",
        "primary": false
    },
    {
        "id": "West",
        "name": "West",
        "name_2": "",
        "primary": true
    }
],
"salutation": "",
"first_name": "Florence",
"last_name": "Haddock",
"full_name": "Florence Haddock",
"title": "Director Sales",
"linkedin": "",
"facebook": "",
"twitter": "",
"googleplus": "",
"department": "",
"do_not_call": false,
"phone_home": "(729) 845-3137",
"email": [
    {
        "email_address": "dev.vegan@example.de",
        "opt_out": "1",
        "invalid_email": "0",
        "primary_address": "0"
    }
]
```

```
    },
    {
      "email_address": "section71@example.it",
      "opt_out": "0",
      "invalid_email": "0",
      "primary_address": "1"
    }
  ],
  "phone_mobile": "(246) 233-1382",
  "phone_work": "(565) 696-6981",
  "phone_other": "",
  "phone_fax": "",
  "email1": "section71@example.it",
  "email2": "dev.vegan@example.de",
  "invalid_email": false,
  "email_opt_out": false,
  "primary_address_street": "111 Silicon Valley Road",
  "primary_address_street_2": "",
  "primary_address_street_3": "",
  "primary_address_city": "Denver",
  "primary_address_state": "CA",
  "primary_address_postalcode": "79900",
  "primary_address_country": "USA",
  "alt_address_street": "",
  "alt_address_street_2": "",
  "alt_address_street_3": "",
  "alt_address_city": "",
  "alt_address_state": "",
  "alt_address_postalcode": "",
  "alt_address_country": "",
  "assistant": "",
  "assistant_phone": "",
  "picture": "",
  "email_and_name1": "",
  "lead_source": "Support Portal User Registration",
  "account_name": "Smallville Resources Inc",
```

```

"account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",
"opportunity_role_fields":"","
"opportunity_role_id":"","
"opportunity_role":"","
"reports_to_id":"","
"report_to_name":"","
"portal_name":"FlorenceHaddock169",
"portal_active":true,
"portal_password":"$1$WFhTbK6$JF9BCGSqL\NCRbhueX5ia0",
"portal_password1":"","
"portal_app":"","
"preferred_language":"en_us",
"campaign_id":"","
"campaign_name":"","
"c_accept_status_fields":"","
"m_accept_status_fields":"","
"accept_status_id":"","
"accept_status_name":"","
"sync_contact":"","
"my_favorite":false,
"_acl":{
  "fields":{
    }
  }
}
]
}

```

Change Log

| Version | Change |
|---------|---------------------------------------|
| v10 | Added /<module>/filter POST endpoint. |

Last Modified: 05/31/2017 05:29pm

/<module>/filter/count GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

| Name | Type | Description | Required |
|-----------|--------|---|----------|
| filter | String | The filter expression. Filter expressions are explained below. | False |
| filter_id | String | Identifier for a preexisting filter. If filter is also set, the | False |

| Name | Type | Description | Required |
|---------|---------|--|----------|
| | | two filters are joined with an AND. | |
| max_num | Integer | A maximum number of records to return. Default is 20. | False |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |
| fields | String | Comma delimited list of fields to return. The field date_modified will always be returned. Example: name,account_type,description | False |
| view | String | Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields | False |

| | | | |
|----------|--------|--|-------|
| | | argument. Common views are "record" and "list".
Example: record | |
| order_by | String | How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon.
Example:
name:DESC,account_type:DESC,date_modified:ASC | False |

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
  "filter": [
    {
      "name": "Nelson Inc"
    }
  ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
  "filter": [
    {
      "name": {
        "$starts": "Nelson"
      }
    }
  ]
}
```

Below is a list of operation types:

| Operation | Description |
|--------------|---|
| \$equals | Performs an exact match on that field. |
| \$not_equals | Performs an exact match on that field. |
| \$not_equals | Matches on non-matching values. |
| \$starts | Matches on anything that starts with the value. |
| \$ends | Matches anything that ends with the value. |
| \$contains | Matches anything that contains the value |
| \$in | Finds anything where field matches one |

| Operation | Description |
|------------|---|
| | of the values as specified as an array. |
| \$not_in | Finds anything where field does not matches any of the values as specified as an array. |
| \$is_null | Checks if the field is null. This operation does not need a value specified. |
| \$not_null | Checks if the field is not null. This operation does not need a value specified. |
| \$lt | Matches when the field is less than the value. |
| \$lte | Matches when the field is less than or equal to the value. |
| \$gt | Matches when the field is greater than the value. |
| \$gte | Matches when the field is greater than or equal to the value. |

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "\$and" and "\$or".

Example

```
{
  "filter": [
    {
```

```
    "$or": [
      {
        "name": "Nelson Inc"
      },
      {
        "name": "Nelson LLC"
      }
    ]
  }
]
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "\$favorite" and "\$owner".

Example

```
{
  "filter": [
    {
      "$favorite": "_this"
    }
  ]
}
```

Response Arguments

| Name | Type | Description |
|-------------|---------|---|
| next_offset | Integer | Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records. |
| records | Array | An array of results containing matched records. |

Response

```
{
  "next_offset":-1,
  "records":[
    {
      "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
      "name":"Dale Spivey",
      "date_entered":"2013-02-26T19:12:00+00:00",
      "date_modified":"2013-02-28T05:03:00+00:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "description":"",
      "img":"",
      "deleted":false,
      "assigned_user_id":"seed_sally_id",
      "assigned_user_name":"Sally Bronsen",
      "team_name":[
        {
          "id":"East",
```

```
        "name": "East",
        "name_2": "",
        "primary": false
    },
    {
        "id": 1,
        "name": "Global",
        "name_2": "",
        "primary": false
    },
    {
        "id": "West",
        "name": "West",
        "name_2": "",
        "primary": true
    }
],
"salutation": "",
"first_name": "Dale",
"last_name": "Spivey",
"full_name": "Dale Spivey",
"title": "VP Operations",
"linkedin": "",
"facebook": "",
"twitter": "",
"googleplus": "",
"department": "",
"do_not_call": false,
"phone_home": "(523) 825-4311",
"email": [
    {
        "email_address": "sugar.dev.sugar@example.co.jp",
        "opt_out": "0",
        "invalid_email": "0",
        "primary_address": "1"
    }
],
```

```
    {
      "email_address": "the.support@example.biz",
      "opt_out": "0",
      "invalid_email": "0",
      "primary_address": "0"
    }
  ],
  "phone_mobile": "(373) 861-0757",
  "phone_work": "(212) 542-9596",
  "phone_other": "",
  "phone_fax": "",
  "email1": "sugar.dev.sugar@example.co.jp",
  "email2": "the.support@example.biz",
  "invalid_email": false,
  "email_opt_out": false,
  "primary_address_street": "345 Sugar Blvd.",
  "primary_address_street_2": "",
  "primary_address_street_3": "",
  "primary_address_city": "Denver",
  "primary_address_state": "CA",
  "primary_address_postalcode": "87261",
  "primary_address_country": "USA",
  "alt_address_street": "",
  "alt_address_street_2": "",
  "alt_address_street_3": "",
  "alt_address_city": "",
  "alt_address_state": "",
  "alt_address_postalcode": "",
  "alt_address_country": "",
  "assistant": "",
  "assistant_phone": "",
  "picture": "",
  "email_and_name1": "",
  "lead_source": "Campaign",
  "account_name": "Smallville Resources Inc",
  "account_id": "d5db6292-5c24-eb61-e202-512d09f0134e",
```

```
"opportunity_role_fields":"","  
"opportunity_role_id":"","  
"opportunity_role":"","  
"reports_to_id":"","  
"report_to_name":"","  
"portal_name":"DaleSpivey97",  
"portal_active":true,  
"portal_password":"$1$yKMAONHM$Y5S.8CY.WZCZCwfGD1a1Q\/",  
"portal_password1":"","  
"portal_app":"","  
"preferred_language":"en_us",  
"campaign_id":"","  
"campaign_name":"","  
"c_accept_status_fields":"","  
"m_accept_status_fields":"","  
"accept_status_id":"","  
"accept_status_name":"","  
"sync_contact":"","  
"my_favorite":false,  
"_acl":{  
  "fields":{  
  
  }  
}  
},  
{  
  "id":"95e17367-9b3d-0e26-22dc-512d0961fedf",  
  "name":"Florence Haddock",  
  "date_entered":"2013-02-26T19:12:00+00:00",  
  "date_modified":"2013-02-26T19:12:00+00:00",  
  "modified_user_id":"1",  
  "modified_by_name":"Administrator",  
  "created_by":"1",  
  "created_by_name":"Administrator",  
  "description":"","  
  "img":"","
```

```
"deleted":false,
"assigned_user_id":"seed_sally_id",
"assigned_user_name":"Sally Bronsen",
"team_name":[
  {
    "id":"East",
    "name":"East",
    "name_2":"",
    "primary":false
  },
  {
    "id":1,
    "name":"Global",
    "name_2":"",
    "primary":false
  },
  {
    "id":"West",
    "name":"West",
    "name_2":"",
    "primary":true
  }
],
"salutation":"",
"first_name":"Florence",
"last_name":"Haddock",
"full_name":"Florence Haddock",
"title":"Director Sales",
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(729) 845-3137",
"email":[
```

```
{
  "email_address": "dev.vegan@example.de",
  "opt_out": "1",
  "invalid_email": "0",
  "primary_address": "0"
},
{
  "email_address": "section71@example.it",
  "opt_out": "0",
  "invalid_email": "0",
  "primary_address": "1"
}
],
"phone_mobile": "(246) 233-1382",
"phone_work": "(565) 696-6981",
"phone_other": "",
"phone_fax": "",
"email1": "section71@example.it",
"email2": "dev.vegan@example.de",
"invalid_email": false,
"email_opt_out": false,
"primary_address_street": "111 Silicon Valley Road",
"primary_address_street_2": "",
"primary_address_street_3": "",
"primary_address_city": "Denver",
"primary_address_state": "CA",
"primary_address_postalcode": "79900",
"primary_address_country": "USA",
"alt_address_street": "",
"alt_address_street_2": "",
"alt_address_street_3": "",
"alt_address_city": "",
"alt_address_state": "",
"alt_address_postalcode": "",
"alt_address_country": "",
"assistant": ""
```

```
"assistant_phone":"","
"picture":"","
"email_and_name1":"","
"lead_source":"Support Portal User Registration",
"account_name":"Smallville Resources Inc",
"account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",
"opportunity_role_fields":"","
"opportunity_role_id":"","
"opportunity_role":"","
"reports_to_id":"","
"report_to_name":"","
"portal_name":"FlorenceHaddock169",
"portal_active":true,
"portal_password":"$1$NWFhTbK6$JF9BCGSqL\NCRbhueX5ia0",
"portal_password1":"","
"portal_app":"","
"preferred_language":"en_us",
"campaign_id":"","
"campaign_name":"","
"c_accept_status_fields":"","
"m_accept_status_fields":"","
"accept_status_id":"","
"accept_status_name":"","
"sync_contact":"","
"my_favorite":false,
"_acl":{
  "fields":{
    }
  }
}
]
```

Change Log

| Version | Change |
|---------|---------------------------------------|
| v10 | Added /<module>/filter POST endpoint. |

Last Modified: 05/31/2017 05:28pm

/<module>/filter/count POST

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

| Name | Type | Description | Required |
|--------|--------|-------------------------------|----------|
| filter | String | The filter expression. Filter | False |

| Name | Type | Description | Required |
|-----------|---------|--|----------|
| | | expressions are explained below. | |
| filter_id | String | Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND. | False |
| max_num | Integer | A maximum number of records to return. Default is 20. | False |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |
| fields | String | Comma delimited list of fields to return. The field date_modified will always be returned. Example: name,account_type,description | False |
| view | String | Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view | False |

| | | | |
|----------|--------|--|-------|
| | | definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list".
Example: record | |
| order_by | String | How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon.
Example:
name:DESC,account_type:DESC,date_modified:ASC | False |

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
  "filter": [
```

```
{
  "name": "Nelson Inc"
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
  "filter": [
    {
      "name": {
        "$starts": "Nelson"
      }
    }
  ]
}
```

Below is a list of operation types:

| Operation | Description |
|---------------------------|---|
| <code>\$equals</code> | Performs an exact match on that field. |
| <code>\$not_equals</code> | Performs an exact match on that field. |
| <code>\$not_equals</code> | Matches on non-matching values. |
| <code>\$starts</code> | Matches on anything that starts with the value. |
| <code>\$ends</code> | Matches anything that ends with the |

| Operation | Description |
|------------|---|
| | value. |
| \$contains | Matches anything that contains the value |
| \$in | Finds anything where field matches one of the values as specified as an array. |
| \$not_in | Finds anything where field does not matches any of the values as specified as an array. |
| \$is_null | Checks if the field is null. This operation does not need a value specified. |
| \$not_null | Checks if the field is not null. This operation does not need a value specified. |
| \$lt | Matches when the field is less than the value. |
| \$lte | Matches when the field is less than or equal to the value. |
| \$gt | Matches when the field is greater than the value. |
| \$gte | Matches when the field is greater than or equal to the value. |

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "\$and" and "\$or".

Example

```
{
  "filter":[
    {
      "$or":[
        {
          "name":"Nelson Inc"
        },
        {
          "name":"Nelson LLC"
        }
      ]
    }
  ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "\$favorite" and "\$owner".

Example

```
{
  "filter":[
    {
      "$favorite":"_this"
    }
  ]
}
```

Response Arguments

| Name | Type | Description |
|-------------|---------|---|
| next_offset | Integer | Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records. |
| records | Array | An array of results containing matched records. |

Response

```
{
  "next_offset": -1,
  "records": [
    {
      "id": "fa300a0e-0ad1-b322-9601-512d0983c19a",
      "name": "Dale Spivey",
      "date_entered": "2013-02-26T19:12:00+00:00",
      "date_modified": "2013-02-28T05:03:00+00:00",
      "modified_user_id": "1",
      "modified_by_name": "Administrator",
      "created_by": "1",
      "created_by_name": "Administrator",
      "description": "",
      "img": "",
      "deleted": false,
    }
  ]
}
```

```
"assigned_user_id":"seed_sally_id",
"assigned_user_name":"Sally Bronsen",
"team_name":[
  {
    "id":"East",
    "name":"East",
    "name_2":"",
    "primary":false
  },
  {
    "id":1,
    "name":"Global",
    "name_2":"",
    "primary":false
  },
  {
    "id":"West",
    "name":"West",
    "name_2":"",
    "primary":true
  }
],
"salutation":"",
"first_name":"Dale",
"last_name":"Spivey",
"full_name":"Dale Spivey",
"title":"VP Operations",
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(523) 825-4311",
"email":[
  {
```

```
        "email_address": "sugar.dev.sugar@example.co.jp",
        "opt_out": "0",
        "invalid_email": "0",
        "primary_address": "1"
    },
    {
        "email_address": "the.support@example.biz",
        "opt_out": "0",
        "invalid_email": "0",
        "primary_address": "0"
    }
],
"phone_mobile": "(373) 861-0757",
"phone_work": "(212) 542-9596",
"phone_other": "",
"phone_fax": "",
"email1": "sugar.dev.sugar@example.co.jp",
"email2": "the.support@example.biz",
"invalid_email": false,
"email_opt_out": false,
"primary_address_street": "345 Sugar Blvd.",
"primary_address_street_2": "",
"primary_address_street_3": "",
"primary_address_city": "Denver",
"primary_address_state": "CA",
"primary_address_postalcode": "87261",
"primary_address_country": "USA",
"alt_address_street": "",
"alt_address_street_2": "",
"alt_address_street_3": "",
"alt_address_city": "",
"alt_address_state": "",
"alt_address_postalcode": "",
"alt_address_country": "",
"assistant": "",
"assistant_phone": "",
```

```
"picture": "",
"email_and_name1": "",
"lead_source": "Campaign",
"account_name": "Smallville Resources Inc",
"account_id": "d5db6292-5c24-eb61-e202-512d09f0134e",
"opportunity_role_fields": "",
"opportunity_role_id": "",
"opportunity_role": "",
"reports_to_id": "",
"report_to_name": "",
"portal_name": "DaleSpivey97",
"portal_active": true,
"portal_password": "$1$yKMAONHM$Y5S.8CY.WZCZCwfGD1a1Q\/",
"portal_password1": "",
"portal_app": "",
"preferred_language": "en_us",
"campaign_id": "",
"campaign_name": "",
"c_accept_status_fields": "",
"m_accept_status_fields": "",
"accept_status_id": "",
"accept_status_name": "",
"sync_contact": "",
"my_favorite": false,
"_acl": {
  "fields": {
  }
}
},
{
  "id": "95e17367-9b3d-0e26-22dc-512d0961fedf",
  "name": "Florence Haddock",
  "date_entered": "2013-02-26T19:12:00+00:00",
  "date_modified": "2013-02-26T19:12:00+00:00",
  "modified_user_id": "1",
```

```
"modified_by_name": "Administrator",
"created_by": "1",
"created_by_name": "Administrator",
"description": "",
"img": "",
"deleted": false,
"assigned_user_id": "seed_sally_id",
"assigned_user_name": "Sally Bronsen",
"team_name": [
  {
    "id": "East",
    "name": "East",
    "name_2": "",
    "primary": false
  },
  {
    "id": 1,
    "name": "Global",
    "name_2": "",
    "primary": false
  },
  {
    "id": "West",
    "name": "West",
    "name_2": "",
    "primary": true
  }
],
"salutation": "",
"first_name": "Florence",
"last_name": "Haddock",
"full_name": "Florence Haddock",
"title": "Director Sales",
"linkedin": "",
"facebook": "",
"twitter": "",
```

```
"googleplus":"","  
"department":"","  
"do_not_call":false,  
"phone_home":"(729) 845-3137",  
"email":[  
  {  
    "email_address":"dev.vegan@example.de",  
    "opt_out":"1",  
    "invalid_email":"0",  
    "primary_address":"0"  
  },  
  {  
    "email_address":"section71@example.it",  
    "opt_out":"0",  
    "invalid_email":"0",  
    "primary_address":"1"  
  }  
],  
"phone_mobile":"(246) 233-1382",  
"phone_work":"(565) 696-6981",  
"phone_other":"","  
"phone_fax":"","  
"email1":"section71@example.it",  
"email2":"dev.vegan@example.de",  
"invalid_email":false,  
"email_opt_out":false,  
"primary_address_street":"111 Silicon Valley Road",  
"primary_address_street_2":"","  
"primary_address_street_3":"","  
"primary_address_city":"Denver",  
"primary_address_state":"CA",  
"primary_address_postalcode":"79900",  
"primary_address_country":"USA",  
"alt_address_street":"","  
"alt_address_street_2":"","  
"alt_address_street_3":"","
```

```
"alt_address_city":"","  
"alt_address_state":"","  
"alt_address_postalcode":"","  
"alt_address_country":"","  
"assistant":"","  
"assistant_phone":"","  
"picture":"","  
"email_and_name1":"","  
"lead_source":"Support Portal User Registration",  
"account_name":"Smallville Resources Inc",  
"account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",  
"opportunity_role_fields":"","  
"opportunity_role_id":"","  
"opportunity_role":"","  
"reports_to_id":"","  
"report_to_name":"","  
"portal_name":"FlorenceHaddock169",  
"portal_active":true,  
"portal_password":"$1$WfHtBk6$JF9BCGSqL\NCRbhueX5ia0",  
"portal_password1":"","  
"portal_app":"","  
"preferred_language":"en_us",  
"campaign_id":"","  
"campaign_name":"","  
"c_accept_status_fields":"","  
"m_accept_status_fields":"","  
"accept_status_id":"","  
"accept_status_name":"","  
"sync_contact":"","  
"my_favorite":false,  
"_acl":{  
  "fields":{  
  
  }  
}  
}
```

```
]
}
```

Change Log

| Version | Change |
|---------|---------------------------------------|
| v10 | Added /<module>/filter POST endpoint. |

Last Modified: 05/31/2017 05:27pm

/<module>/filter/:record GET

Overview

Lists filtered records given a predefined filter set id.

Summary

Note: This endpoint is deprecated. Please use GET /<module>/filter and supply a filter_id instead. This endpoint will return a set of records filtered by a given filter id. Additional filters can also be applied in addition to any existing filters defined by the filter record. Care will need to be taken to make sure this filter has appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

| Name | Type | Description | Required |
|---------|---------|--|----------|
| filter | String | The filter expression. Filter expressions are explained below. | False |
| max_num | Integer | A maximum number of records to return. Default is 20. | False |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |
| fields | String | Comma delimited list of fields to return. The field date_modified will always be returned. Example: name,account_type,description | False |
| view | String | Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This | False |

| Name | Type | Description | Required |
|----------|--------|--|----------|
| | | argument can be used in combination with the fields argument. Common views are "record" and "list".
Example: record | |
| order_by | String | How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon.
Example:
name:DESC,account_type:DESC,date_modified:ASC | False |

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
  "filter": [
```

```
{
  {
    "name": "Nelson Inc"
  }
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
  "filter": [
    {
      "name": {
        "$starts": "Nelson"
      }
    }
  ]
}
```

Below is a list of operation types:

| Operation | Description |
|---------------------------|---|
| <code>\$equals</code> | Performs an exact match on that field. |
| <code>\$not_equals</code> | Performs an exact match on that field. |
| <code>\$not_equals</code> | Matches on non-matching values. |
| <code>\$starts</code> | Matches on anything that starts with the value. |
| <code>\$ends</code> | Matches anything that ends with the |

| Operation | Description |
|------------|---|
| | value. |
| \$contains | Matches anything that contains the value |
| \$in | Finds anything where field matches one of the values as specified as an array. |
| \$not_in | Finds anything where field does not matches any of the values as specified as an array. |
| \$is_null | Checks if the field is null. This operation does not need a value specified. |
| \$not_null | Checks if the field is not null. This operation does not need a value specified. |
| \$lt | Matches when the field is less than the value. |
| \$lte | Matches when the field is less than or equal to the value. |
| \$gt | Matches when the field is greater than the value. |
| \$gte | Matches when the field is greater than or equal to the value. |

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "\$and" and "\$or".

Example

```
{
  "filter":[
    {
      "$or":[
        {
          "name":"Nelson Inc"
        },
        {
          "name":"Nelson LLC"
        }
      ]
    }
  ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "\$favorite" and "\$owner".

Example

```
{
  "filter":[
    {
      "$favorite":"_this"
    }
  ]
}
```

Response Arguments

| Name | Type | Description |
|-------------|---------|---|
| next_offset | Integer | Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records. |
| records | Array | An array of results containing matched records. |

Response

```
{
  "next_offset": -1,
  "records": [
    {
      "id": "bbd4690e-26bb-e473-91e7-512d09c95157",
      "name": "Rubble Group Inc",
      "date_entered": "2013-02-26T19:12:00+00:00",
      "date_modified": "2013-02-26T19:12:00+00:00",
      "modified_user_id": "1",
      "modified_by_name": "Administrator",
      "created_by": "1",
      "created_by_name": "Administrator",
      "description": "",
      "img": "",
      "last_activity_date": "2013-02-26T19:12:00+00:00",
```

```
"deleted":false,
"assigned_user_id":"seed_chris_id",
"assigned_user_name":"Chris Olliver",
"team_name":[
  {
    "id":"East",
    "name":"East",
    "name_2":"",
    "primary":true
  },
  {
    "id":1,
    "name":"Global",
    "name_2":"",
    "primary":false
  }
],
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"account_type":"Customer",
"industry":"Manufacturing",
"annual_revenue":"",
"phone_fax":"",
"billing_address_street":"9 IBM Path",
"billing_address_street_2":"",
"billing_address_street_3":"",
"billing_address_street_4":"",
"billing_address_city":"Salt Lake City",
"billing_address_state":"NY",
"billing_address_postalcode":"96885",
"billing_address_country":"USA",
"rating":"",
"phone_office":"(877) 305-0310",
"phone_alternate":"",
```

```
"website": "www.beanssupport.edu",
"ownership": "",
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "9 IBM Path",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "Salt Lake City",
"shipping_address_state": "NY",
"shipping_address_postalcode": "96885",
"shipping_address_country": "USA",
"email1": "sugar81@example.de",
"parent_id": "",
"sic_code": "",
"parent_name": "",
"email_opt_out": false,
"invalid_email": false,
"email": [
  {
    "email_address": "sugar81@example.de",
    "opt_out": "0",
    "invalid_email": "0",
    "primary_address": "1"
  },
  {
    "email_address": "qa.sales@example.info",
    "opt_out": "0",
    "invalid_email": "0",
    "primary_address": "0"
  }
],
"campaign_id": "",
"campaign_name": "",
"my_favorite": false,
"_acl": {
```

```
    "fields":{
    }
  },
  {
    "id":"edc82e24-acc9-9218-80f8-512d0902a509",
    "name":"Consulting Group LTD",
    "date_entered":"2013-02-26T19:12:00+00:00",
    "date_modified":"2013-02-26T19:12:00+00:00",
    "modified_user_id":"1",
    "modified_by_name":"Administrator",
    "created_by":"1",
    "created_by_name":"Administrator",
    "description":"",
    "img":"",
    "last_activity_date":"2013-02-26T19:12:00+00:00",
    "deleted":false,
    "assigned_user_id":"seed_chris_id",
    "assigned_user_name":"Chris Olliver",
    "team_name":[
      {
        "id":"East",
        "name":"East",
        "name_2":"",
        "primary":true
      },
      {
        "id":"West",
        "name":"West",
        "name_2":"",
        "primary":false
      }
    ],
    "linkedin":"",
    "facebook":"",
```

```
"twitter":"","  
"googleplus":"","  
"account_type":"Customer",  
"industry":"Media",  
"annual_revenue":"","  
"phone_fax":"","  
"billing_address_street":"999 Baker Way",  
"billing_address_street_2":"","  
"billing_address_street_3":"","  
"billing_address_street_4":"","  
"billing_address_city":"Cupertino",  
"billing_address_state":"NY",  
"billing_address_postalcode":"96252",  
"billing_address_country":"USA",  
"rating":"","  
"phone_office":"(181) 483-0547",  
"phone_alternate":"","  
"website":"www.devthe.info",  
"ownership":"","  
"employees":"","  
"ticker_symbol":"","  
"shipping_address_street":"999 Baker Way",  
"shipping_address_street_2":"","  
"shipping_address_street_3":"","  
"shipping_address_street_4":"","  
"shipping_address_city":"Cupertino",  
"shipping_address_state":"NY",  
"shipping_address_postalcode":"96252",  
"shipping_address_country":"USA",  
"email1":"section.phone@example.co.uk",  
"parent_id":"","  
"sic_code":"","  
"parent_name":"","  
"email_opt_out":false,  
"invalid_email":false,  
"email":[
```

```
    {
      "email_address": "section.phone@example.co.uk",
      "opt_out": "0",
      "invalid_email": "0",
      "primary_address": "1"
    },
    {
      "email_address": "info32@example.org",
      "opt_out": "0",
      "invalid_email": "0",
      "primary_address": "0"
    }
  ],
  "campaign_id": "",
  "campaign_name": "",
  "my_favorite": false,
  "_acl": {
    "fields": {

    }
  }
},
{
  "id": "a42675b0-f727-5fb2-dafd-512d09977e4e",
  "name": "Orange Insurance",
  "date_entered": "2013-02-26T19:12:00+00:00",
  "date_modified": "2013-02-26T19:12:00+00:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
  "description": "",
  "img": "",
  "last_activity_date": "2013-02-26T19:12:00+00:00",
  "deleted": false,
  "assigned_user_id": "seed_max_id",
```

```
"assigned_user_name": "Max Jensen",
"team_name": [
  {
    "id": "West",
    "name": "West",
    "name_2": "",
    "primary": true
  }
],
"linkedin": "",
"facebook": "",
"twitter": "",
"googleplus": "",
"account_type": "Customer",
"industry": "Machinery",
"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": "Santa Monica",
"billing_address_state": "CA",
"billing_address_postalcode": "93826",
"billing_address_country": "USA",
"rating": "",
"phone_office": "(570) 239-0233",
"phone_alternate": "",
"website": "www.vegansupport.biz",
"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": "Santa Monica",
"shipping_address_state": "CA",
"shipping_address_postalcode": "93826",
"shipping_address_country": "USA",
"email1": "sales.vegan.beans@example.us",
"parent_id": "",
"sic_code": "",
"parent_name": "",
"email_opt_out": false,
"invalid_email": false,
"email": [
  {
    "email_address": "sales.vegan.beans@example.us",
    "opt_out": "0",
    "invalid_email": "0",
    "primary_address": "1"
  },
  {
    "email_address": "hr.hr@example.name",
    "opt_out": "0",
    "invalid_email": "0",
    "primary_address": "0"
  }
],
"campaign_id": "",
"campaign_name": "",
"my_favorite": false,
"_acl": {
  "fields": {
  }
}
]
}
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added /<module>/filter/:record GET endpoint. |

Last Modified: 05/31/2017 05:28pm

/<module>/globalsearch GET

Overview

Global search

Summary

This endpoint exposes the global search capability using solely the Elasticsearch backend as an alternative to the /search endpoint. This endpoint can be used with a long list of request arguments. Instead of using GET request arguments, all described arguments can be used inside a JSON encoded request body using a the GET request. If your client has no support for GET requests with a body, the POST method can be used as an alternative.

Request Arguments

| Name | Type | Description | Required |
|------|--------|------------------------|----------|
| q | String | The search expression. | False |

| Name | Type | Description | Required |
|-------------|--------|---|----------|
| | | <p>Multiple terms can be specified at once. All enabled fields will be searched in. The results are ordered by relevance which is based on a multitude of settings based on token counts, hit ratio, (weighted) boost values and type of field. Currently no operators are supported in the search expression itself. By refining the search expression more relevant results will be returned as top results. If no search expression is given results are returned based on last modified date.</p> | |
| module_list | String | Comma delimited list of modules to search. If omitted, all search enabled modules will be queried. Note that | False |

| | | | |
|------------|---------|---|-------|
| | | when consuming the endpoint <code>/:module/globalsearch</code> that this parameter is ignored.
Example:
Accounts,Contacts | |
| max_num | Integer | A maximum number of records to return. Default is 20. | False |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |
| highlights | Boolean | Whether or not to return highlighted results. Default is true. | False |
| sort | Array | Define the sort order of the results. By default the results are returned by relevance which is the preferred approach. Using this argument any search enabled field can be used to sort on. Keep in mind the not sorting by relevance may have | False |

| | | | |
|--|--|---|--|
| | | a negative performance impact.
Example:
{"date_modified":"desc","name":"asc"} | |
|--|--|---|--|

Request

```
{
}
```

Response Arguments

| Name | Type | Description |
|-------------|---------|---|
| next_offset | Integer | Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records. |
| records | Array | An array of results containing matched records. |

Response

```
{
}
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added /globalsearch GET/POST endpoint. |
| v10 | Added /:module/globalsearch GET/POST endpoint. |

Last Modified: 05/31/2017 05:30pm

/<module>/globalsearch POST

Overview

Global search

Summary

This endpoint exposes the global search capability using solely the Elasticsearch backend as an alternative to the /search endpoint. This endpoint can be used with a long list of request arguments. Instead of using GET request arguments, all described arguments can be used inside a JSON encoded request body using a the GET request. If your client has no support for GET requests with a body, the POST method can be used as an alternative.

Request Arguments

| Name | Type | Description | Required |
|-------------|--------|--|----------|
| q | String | <p>The search expression. Multiple terms can be specified at once. All enabled fields will be searched in. The results are ordered by relevance which is based on a multitude of settings based on token counts, hit ratio, (weighted) boost values and type of field. Currently no operators are supported in the search expression itself. By refining the search expression more relevant results will be returned as top results. If no search expression is given results are returned based on last modified date.</p> | False |
| module_list | String | Comma delimited list of modules to search. If omitted, all search enabled | False |

| Name | Type | Description | Required |
|------------|---------|---|----------|
| | | modules will be queried. Note that when consuming the endpoint <code>/:module/globalsearch</code> that this parameter is ignored.
Example:
Accounts,Contacts | |
| max_num | Integer | A maximum number of records to return. Default is 20. | False |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |
| highlights | Boolean | Whether or not to return highlighted results. Default is true. | False |
| sort | Array | Define the sort order of the results. By default the results are returned by relevance which is the preferred approach. Using this argument any search enabled field can be used to | False |

| | | | |
|--|--|---|--|
| | | <p>sort on. Keep in mind the not sorting by relevance may have a negative performance impact.</p> <p>Example:
 <code>{"date_modified":"desc","name":"asc"}</code></p> | |
|--|--|---|--|

Request

```
{
}
```

Response Arguments

| Name | Type | Description |
|-------------|---------|---|
| next_offset | Integer | Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records. |
| records | Array | An array of results containing matched records. |

Response

```
{  
}
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added /globalsearch GET/POST endpoint. |
| v10 | Added /:module/globalsearch GET/POST endpoint. |

Last Modified: 05/31/2017 05:29pm

/<module>/insertafter/:target POST

Overview

Insert new node after target node.

Request Arguments

| Name | Type | Description | Required |
|--------|--------|---|----------|
| module | String | The name of sugar module that contains a nested set data and implements the | True |

| Name | Type | Description | Required |
|--------|--------|---|----------|
| | | NestedSetInterface. | |
| target | String | The ID of record that will be used as target to insert new node after | True |

Request

```
{
  "name" : "Children Node 1"
}
```

Response Arguments

This endpoint does not return any response arguments.

Response

This endpoint returns a newly created bean

```
{ "my_favorite":false, "following":"","id":"59fa8dd7-0f2c-4bfd-364f-54495f77fa3f",
"name":"Default title", "date_entered":"2014-10-23T23:03:22+03:00",
"date_modified":"2014-10-23T23:03:22+03:00", "modified_user_id":"1",
"modified_by_name":"Administrator", "created_by":"1",
"created_by_name":"Administrator", "doc_owner":"","description":"","
"deleted":false, "source_id":"","source_type":"","source_meta":"","
"root":"be9b0c4a-8b78-1ffa-4f14-54481c2f6269", "lft":118, "rgt":119, "level":1,
"_acl":{"fields":{}}}, "_module":"Categories" }
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added /<module>/insertafter/:target POST endpoint. |

Last Modified: 05/31/2017 05:27pm

/<module>/insertbefore/:target POST

Overview

Insert new node before target node.

Request Arguments

| Name | Type | Description | Required |
|--------|--------|---|----------|
| module | String | The name of sugar module that contains a nested set data and implements the NestedSetInterface. | True |
| target | String | The ID of record that will be used as target to insert new node before | True |

Request

```
{
  "name" : "Children Node 1"
}
```

Response Arguments

This endpoint does not return any response arguments.

Response

This endpoint returns a newly created bean

```
{ "my_favorite":false, "following":"","id":"59fa8dd7-0f2c-4bfd-364f-54495f77fa3f",
"name":"Default title", "date_entered":"2014-10-23T23:03:22+03:00",
"date_modified":"2014-10-23T23:03:22+03:00", "modified_user_id":"1",
"modified_by_name":"Administrator", "created_by":"1",
"created_by_name":"Administrator", "doc_owner":"","description":"","
"deleted":false, "source_id":"","source_type":"","source_meta":"","
"root":"be9b0c4a-8b78-1ffa-4f14-54481c2f6269", "lft":118, "rgt":119, "level":1,
"_acl":{"fields":{}}}, "_module":"Categories" }
```

Change Log

| Version | Change |
|---------|---|
| v10 | Added /<module>/insertbefore/:target POST endpoint. |

Last Modified: 05/31/2017 05:27pm

/<module>/prepend/:target POST

Overview

Append new node to target node as first child.

Request Arguments

| Name | Type | Description | Required |
|--------|--------|---|----------|
| module | String | The name of sugar module that contains a nested set data and implements the NestedSetInterface. | True |
| target | String | The ID of record that will be used as target to prepend new node | True |

Request

```
{  
  "name" : "Children Node 1"  
}
```

Response Arguments

This endpoint does not return any response arguments.

Response

This endpoint returns a newly created record GUID

```
{ "my_favorite":false, "following":"","id":"59fa8dd7-0f2c-4bfd-364f-54495f77fa3f",
"name":"Default title", "date_entered":"2014-10-23T23:03:22+03:00",
"date_modified":"2014-10-23T23:03:22+03:00", "modified_user_id":"1",
"modified_by_name":"Administrator", "created_by":"1",
"created_by_name":"Administrator", "doc_owner":"","description":"","
"deleted":false, "source_id":"","source_type":"","source_meta":"","
"root":"be9b0c4a-8b78-1ffa-4f14-54481c2f6269", "lft":118, "rgt":119, "level":1,
"_acl":{"fields":{}}}, "_module":"Categories" }
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added /<module>/prepend/:target POST endpoint. |

Last Modified: 05/31/2017 05:27pm

/<module>/:record DELETE

Overview

Delete a record of a specified type.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

| Name | Type | Description |
|------|--------|---------------------------------------|
| id | String | Returns the ID of the deleted record. |

Response

```
{  
  "id": "11cf0d0a-40af-8cb1-9da0-5057a5f511f9"  
}
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added /<module>/:record DELETE endpoint. |

Last Modified: 05/31/2017 05:28pm

/<module>/:record GET

Overview

Retrieves a record.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

| Name | Type | Description |
|----------------|---------------------|---|
| <record field> | <record field type> | Returns the fields for the selected record. |

Response

```
{
  "id": "bdd59d85-687b-1739-b00a-512d09f6db9e",
  "name": "Insight Marketing Inc",
  "date_entered": "2013-02-26T19:12:00+00:00",
  "date_modified": "2013-02-26T19:12:00+00:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
  "description": "",
  "img": "",
  "last_activity_date": "2013-02-26T19:12:00+00:00",
  "deleted": false,
  "assigned_user_id": "seed_max_id",
  "assigned_user_name": "Max Jensen",
  "team_name": [
    {
      "id": "East",
      "name": "East",
      "name_2": ""
    }
  ]
}
```

```
    "primary":false
  },
  {
    "id":1,
    "name":"Global",
    "name_2":"",
    "primary":false
  },
  {
    "id":"West",
    "name":"West",
    "name_2":"",
    "primary":true
  }
],
"linkedin":"","
"facebook":"","
"twitter":"","
"googleplus":"","
"account_type":"Customer",
"industry":"Electronics",
"annual_revenue":"","
"phone_fax":"","
"billing_address_street":"345 Sugar Blvd.",
"billing_address_street_2":"","
"billing_address_street_3":"","
"billing_address_street_4":"","
"billing_address_city":"San Mateo",
"billing_address_state":"CA",
"billing_address_postalcode":"56019",
"billing_address_country":"USA",
"rating":"","
"phone_office":"(927) 136-9572",
"phone_alternate":"","
"website":"www.sectionvegan.de",
"ownership":"","
```

```
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "345 Sugar Blvd.",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "San Mateo",
"shipping_address_state": "CA",
"shipping_address_postalcode": "56019",
"shipping_address_country": "USA",
"email1": "kid.support.vegan@example.info",
"parent_id": "",
"sic_code": "",
"parent_name": "",
"email_opt_out": false,
"invalid_email": false,
"email": [
  {
    "email_address": "kid.support.vegan@example.info",
    "opt_out": "0",
    "invalid_email": "0",
    "primary_address": "1"
  },
  {
    "email_address": "phone.kid@example.cn",
    "opt_out": "0",
    "invalid_email": "0",
    "primary_address": "0"
  }
],
"campaign_id": "",
"campaign_name": "",
"my_favorite": false,
"_acl": {
  "fields": {
```

```
}  
  }  
}
```

Change Log

| Version | Change |
|---------|---------------------------------------|
| v10 | Added /<module>/:record GET endpoint. |

Last Modified: 05/31/2017 05:30pm

/<module>/:record PUT

Overview

Update a record of the specified type.

Request Arguments

| Name | Type | Description | Required |
|----------------|---------------------|--|----------|
| <record field> | <record field type> | The name value list of fields to populate. | True |

Request

```

{
  "name": "New Account Name",
  "phone_office": "(555) 888-5555",
  "website": "www.newsite.com",
  "meetings": {
    {
      "add": ["21e3385e-404f-b470-407e-54044e3d8024"],
      "delete": ["21e3385e-404f-b470-407e-54044e3d8023"],
      "create": [
        {
          "name": "Test Meeting"
        }
      ]
    }
  ]
}

```

Link fields

It is possible to add or delete record relations to other records and create new related records.

| Action | Type | Description |
|--------|------|---|
| add | List | List of related records ID. See <code>RelateRecordApi</code> for details. |
| delete | List | List of related records ID. |
| create | List | List of name to value arrays of related records. |

Response Arguments

| Name | Type | Description |
|----------------|---------------------|--|
| <record field> | <record field type> | Returns the fields for the updated record. |

Response

```
{
  "id": "f222265a-b755-da89-0bc7-512d09b800b6",
  "name": "New Account Name",
  "date_entered": "2013-02-26T19:12:00+00:00",
  "date_modified": "2013-02-27T22:49:00+00:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
  "description": "",
  "img": "",
  "last_activity_date": "2013-02-26T19:12:00+00:00",
  "deleted": false,
  "assigned_user_id": "seed_sarah_id",
  "assigned_user_name": "Sarah Smith",
  "team_name": [
    {
      "id": 1,
      "name": "Global",
      "name_2": "",
      "primary": false
    },
    {
      "id": "West",
      "name": "West",
      "name_2": "",
      "primary": true
    }
  ]
}
```

```
    }  
  ],  
  "linkedin": "",  
  "facebook": "",  
  "twitter": "",  
  "googleplus": "",  
  "account_type": "Customer",  
  "industry": "Hospitality",  
  "annual_revenue": "",  
  "phone_fax": "",  
  "billing_address_street": "9 IBM Path",  
  "billing_address_street_2": "",  
  "billing_address_street_3": "",  
  "billing_address_street_4": "",  
  "billing_address_city": "San Francisco",  
  "billing_address_state": "CA",  
  "billing_address_postalcode": "43635",  
  "billing_address_country": "USA",  
  "rating": "",  
  "phone_office": "(555) 888-5555",  
  "phone_alternate": "",  
  "website": "www.newsite.com",  
  "ownership": "",  
  "employees": "",  
  "ticker_symbol": "",  
  "shipping_address_street": "9 IBM Path",  
  "shipping_address_street_2": "",  
  "shipping_address_street_3": "",  
  "shipping_address_street_4": "",  
  "shipping_address_city": "San Francisco",  
  "shipping_address_state": "CA",  
  "shipping_address_postalcode": "43635",  
  "shipping_address_country": "USA",  
  "email1": "kid86@example.net",  
  "parent_id": "",  
  "sic_code": "",
```

```
"parent_name": "",
"email_opt_out": false,
"invalid_email": false,
"email": [
  {
    "email_address": "kid86@example.net",
    "opt_out": "0",
    "invalid_email": "0",
    "primary_address": "1"
  },
  {
    "email_address": "the.dev@example.name",
    "opt_out": "0",
    "invalid_email": "0",
    "primary_address": "0"
  }
],
"campaign_id": "",
"campaign_name": "",
"my_favorite": false,
"_acl": {
  "fields": {
  }
}
}
```

Change Log

| Version | Change |
|---------|---------------------------------------|
| v10 | Added /<module>/:record PUT endpoint. |

Last Modified: 05/31/2017 05:29pm

/<module>/record_list POST

Overview

An API to create and save lists of records.

Summary

Create record list. Only POST request allowed.

Request Arguments

| Name | Type | Description | Required |
|---------|--------|------------------|----------|
| module | string | Module name | True |
| records | array | Array of records | True |

Request Example

```
POST /rest/v10/:module/record_list
```

```
{records: [ "e5c8bb3b-5eea-3d8a-c278-56c6affa6212",  
"e49395b4-d3b0-1e3f-600a-56c6afe7e34f" ] }
```

Response Arguments

| Name | Type | Description |
|------------------|--------|----------------|
| id | guid | record list id |
| assigned_user_id | guid | user id |
| module_name | string | Module name |
| records | array | record ids |
| date modified | date | date modified |

Output Example

```
{
  "id": "39d2c781-c0b7-446f-1d83-56c6e3cda510",
  "assigned_user_id": "1",
  "module_name": "Accounts",
  "records": [
    "e5c8bb3b-5eea-3d8a-c278-56c6affa6212",
    "e49395b4-d3b0-1e3f-600a-56c6afe7e34f",
    "c03ed8ee-60d3-c6a6-55c3-56c6af95e696",
    "d65573e3-c25a-f9b4-33f2-56c6afb8d856"
  ],
  "date_modified": "2016-02-19 09:42:44"
}
```

Last Modified: 05/31/2017 05:29pm

/<module>/record_list/:record_list_id DELETE

Overview

An API to delete lists of records

Summary

Delete record list. Only DELETE request allowed.

Request Arguments

| Name | Type | Description | Required |
|----------------|--------|----------------|----------|
| module | string | Module name | True |
| record_list_id | guid | record list id | True |

Request Example

```
DELETE /rest/v10/:module/record_list/:id
```

Response Arguments

Return boolean true.

Last Modified: 05/31/2017 05:26pm

/<module>/record_list/:record_list_id GET

Overview

An API to return record list data

Summary

Get record list data. Only GET request allowed.

Request Arguments

| Name | Type | Description | Required |
|----------------|--------|----------------|----------|
| module | string | Module name | True |
| record_list_id | guid | record list id | True |

Request Example

```
GET /rest/v10/:module/record_list/:id
```

Response Arguments

| Name | Type | Description |
|------------------|--------|----------------|
| id | guid | record list id |
| assigned_user_id | guid | user id |
| module_name | string | Module name |
| records | array | record ids |
| date modified | date | date modified |

Output Example

```
{
  "id": "39d2c781-c0b7-446f-1d83-56c6e3cda510",
  "assigned_user_id": "1",
  "module_name": "Accounts",
  "records": [
    "e5c8bb3b-5eea-3d8a-c278-56c6affa6212",
    "e49395b4-d3b0-1e3f-600a-56c6afe7e34f",
    "c03ed8ee-60d3-c6a6-55c3-56c6af95e696",
    "d65573e3-c25a-f9b4-33f2-56c6afb8d856"
  ],
  "date_modified": "2016-02-19 09:42:44"
}
```

Last Modified: 05/31/2017 05:27pm

/<module>/:record/audit GET

Overview

Returns data changes for a specific record.

Request Arguments

| Name | Type | Description | Required |
|--------|--------|-----------------|----------|
| record | String | The record id. | True |
| module | String | The name of the | True |

| Name | Type | Description | Required |
|------|------|-------------|----------|
| | | | module. |

Response Arguments

| Name | Type | Description |
|--------------|---------|---|
| <image file> | String; | Returns the image content with the content headers. |

Response

```
{
  "next_offset":-1,
  "records":[
    {
      "id":"b569e480-237a-5921-4382-512ff555fee7",
      "parent_id":"ef1b40aa-5815-4f8d-e909-512d09617ac8",
      "date_created":"03\/01\/2013 12:26am",
      "created_by":"admin",
      "field_name":"Team ID:",
      "data_type":"team_list",
      "before_value_string":"East",
      "after_value_string":"Will"
    },
    {
      "id":"bc0bee72-8398-a0a6-d731-512ff5bfefbc7",
      "parent_id":"ef1b40aa-5815-4f8d-e909-512d09617ac8",
      "date_created":"03\/01\/2013 12:26am",
      "created_by":"admin",
      "field_name":"Teams",
      "data_type":"id",

```

```
        "before_value_string": "East, Global, West",
        "after_value_string": "Jim, Will"
    }
]
}
```

Change Log

| Version | Change |
|---------|----------------------------|
| v10 | Added /Audit GET endpoint. |

Last Modified: 05/31/2017 05:28pm

/<module>/:record/children GET

Overview

Retrieves all children of selected record.

Request Arguments

| Name | Type | Description | Required |
|--------|--------|---|----------|
| module | String | The name of sugar module that contains a nested set data and implements the NestedSetInterface. | True |

| Name | Type | Description | Required |
|---------|--------|------------------|----------|
| :record | String | The ID of record | True |

Response Arguments

This endpoint does not return any response arguments.

Response

```
[{
  "id": "045c1de6-327b-11e4-818b-5404a67f3363",
  "name": "Financial",
  "date_entered": null,
  "date_modified": null,
  "modified_user_id": null,
  "created_by": null,
  "description": null,
  "deleted": "0",
  "source_id": null,
  "source_type": null,
  "source_meta": null,
  "root": "935d3e07-327a-11e4-818b-5404a67f3363",
  "lft": "11",
  "rgt": "14",
  "level": "2"
}, {
  "id": "0f65a6c7-327b-11e4-818b-5404a67f3363",
  "name": "Agreements",
  "date_entered": null,
  "date_modified": null,
  "modified_user_id": null,
  "created_by": null,
  "description": null,
```

```
"deleted": "0",
"source_id": null,
"source_type": null,
"source_meta": null,
"root": "935d3e07-327a-11e4-818b-5404a67f3363",
"lft": "15",
"rgt": "16",
"level": "2"
}, {
  "id": "14d30bf3-327b-11e4-818b-5404a67f3363",
  "name": "Clients",
  "date_entered": null,
  "date_modified": null,
  "modified_user_id": null,
  "created_by": null,
  "description": null,
  "deleted": "0",
  "source_id": null,
  "source_type": null,
  "source_meta": null,
  "root": "935d3e07-327a-11e4-818b-5404a67f3363",
  "lft": "17",
  "rgt": "18",
  "level": "2"
}]
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added /<module>/:record/children GET endpoint. |

Last Modified: 05/31/2017 05:28pm

/<module>/:record/collection/:collection_name GET

Overview

Lists related records from multiple links at a time.

Summary

This endpoint will return a set of related records fetched from multiple links defined by :collection_name. The records will be filtered, sorted and truncated to max_num as a single collection. Collections may use a field mapping. For instance, the due_date of Tasks may be referred to as date_end. In this case the alias (date_end) should be used in filter and order_by expressions instead of real field name. Note that response data still contains the original fields for the module.

Request Arguments

| Name | Type | Description | Required |
|---------|---------|---|----------|
| fields | String | See Filter API. If collection definition uses aliases, then aliases should be used instead of real field names. | False |
| view | String | See Filter API. | False |
| max_num | Integer | See Filter API. | False |
| filter | String | See Filter API. If collection definition uses aliases, then aliases should be | False |

| Name | Type | Description | Required |
|----------|--------|--|----------|
| | | used instead of real field names. | |
| order_by | String | See Filter API. If collection definition uses aliases, then aliases should be used instead of real field names. | False |
| offset | Map | Individual offset for each link which the collection consists of. -1 (or any negative value) denotes that the link should be skipped. If link offset is not specified, it defaults to 0. | False |

Response Arguments

| Name | Type | Description |
|-------------|-------|---|
| next_offset | Map | The next offset to retrieve records. -1 will be returned for the given link when there are no more records. |
| records | Array | The record result set. |
| errors | Map | Errors encountered while |

| Name | Type | Description |
|------|------|--|
| | | making requests for the individual links. These errors are returned for the client to consume, but do not affect the response status of the request. |

Response

```
{
  "next_offset": {
    "calls": 1,
    "meetings": -1,
    "tasks": -1,
  },
  "records": [
    {
      "_module": "Calls",
      "_link": "calls",
      "id": "8703fbf3-0ffa-c288-8d2c-512f943ecdc3",
      "name": "Discuss review process",
      "date_end": "2014-02-26T19:12:00+00:00"
    },
    {
      "_module": "Tasks",
      "_link": "tasks",
      "id": "e1c495cb-af17-1b37-dd66-512f934fe155",
      "name": "Introduce all players",
      "due_date": "2014-02-26T19:12:00+00:00"
    },
    {
      "_module": "Tasks",
      "_link": "tasks",
    }
  ]
}
```

```

        "id": "456b7848-9959-5a64-cd34-512d0938add",
        "name": "Follow-up on proposal",
        "due_date": "2014-02-26T19:12:00+00:00"
    },
    ],
    "errors": {
        "meetings": {
            "code": 403,
            "error": "not_authorized",
            "error_message": "You are not authorized to perform this
action. Contact your administrator if you need access."
        }
    }
}

```

Change Log

| Version | Change |
|---------|--|
| v10 | Added
/<module>/:record/collection/:collection_name GET endpoint. |

Last Modified: 05/31/2017 05:26pm

/<module>/:record/collection/:collection_name/count GET

Overview

Counts related records from multiple links at a time.

Summary

This endpoint will return count of records related by multiple links defined by `:collection_name`. The records may be filtered.

Request Arguments

| Name | Type | Description | Required |
|--------|--------|---|----------|
| filter | String | See Filter API. If collection definition uses aliases, then aliases should be used instead of real field names. | False |

Response Arguments

| Name | Type | Description |
|--------------|------|---------------------------|
| record_count | Map | Count of related records. |

Response

```
{
  "record_count": {
    "calls": 1,
    "meetings": 2,
    "tasks": 3
  }
}
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added
/<module>/:record/collection/:collection_name/count GET endpoint. |

Last Modified: 05/31/2017 05:25pm

/<module>/:record/favorite DELETE

Overview

Removes a record of a specified type as a favorite for the current user.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

| Name | Type | Description |
|----------------|---------------------|---|
| <record field> | <record field type> | Returns the fields for the selected record. |

Response

```
{
  "id": "bdd59d85-687b-1739-b00a-512d09f6db9e",
  "name": "Insight Marketing Inc",
  "date_entered": "2013-02-26T19:12:00+00:00",
  "date_modified": "2013-02-26T19:12:00+00:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
  "description": "",
  "img": "",
  "last_activity_date": "2013-02-26T19:12:00+00:00",
  "deleted": false,
  "assigned_user_id": "seed_max_id",
  "assigned_user_name": "Max Jensen",
  "team_name": [
    {
      "id": "East",
      "name": "East",
      "name_2": "",
      "primary": false
    },
    {
      "id": 1,
      "name": "Global",
      "name_2": "",
      "primary": false
    },
    {
      "id": "West",
      "name": "West",
      "name_2": "",
      "primary": true
    }
  ]
}
```

```
],
"linkedin": "",
"facebook": "",
"twitter": "",
"googleplus": "",
"account_type": "Customer",
"industry": "Electronics",
"annual_revenue": "",
"phone_fax": "",
"billing_address_street": "345 Sugar Blvd.",
"billing_address_street_2": "",
"billing_address_street_3": "",
"billing_address_street_4": "",
"billing_address_city": "San Mateo",
"billing_address_state": "CA",
"billing_address_postalcode": "56019",
"billing_address_country": "USA",
"rating": "",
"phone_office": "(927) 136-9572",
"phone_alternate": "",
"website": "www.sectionvegan.de",
"ownership": "",
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "345 Sugar Blvd.",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "San Mateo",
"shipping_address_state": "CA",
"shipping_address_postalcode": "56019",
"shipping_address_country": "USA",
"email1": "kid.support.vegan@example.info",
"parent_id": "",
"sic_code": "",
"parent_name": "",
```

```

"email_opt_out":false,
"invalid_email":false,
"email":[
  {
    "email_address":"kid.support.vegan@example.info",
    "opt_out":"0",
    "invalid_email":"0",
    "primary_address":"1"
  },
  {
    "email_address":"phone.kid@example.cn",
    "opt_out":"0",
    "invalid_email":"0",
    "primary_address":"0"
  }
],
"campaign_id":"","
"campaign_name":"","
"my_favorite":false,
"_acl":{
  "fields":{
  }
}
}

```

Change Log

| Version | Change |
|---------|---|
| v10 | Added /<module>/:record/favorite DELETE endpoint. |

Last Modified: 05/31/2017 05:26pm

/<module>/:record/favorite PUT

Overview

Updates a record of a specified type as a favorite for the current user.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

| Name | Type | Description |
|----------------|---------------------|---|
| <record field> | <record field type> | Returns the fields for the selected record. |

Response

```
{
  "id": "bdd59d85-687b-1739-b00a-512d09f6db9e",
  "name": "Insight Marketing Inc",
  "date_entered": "2013-02-26T19:12:00+00:00",
  "date_modified": "2013-02-26T19:12:00+00:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
  "description": "",
  "img": ""
```

```
"last_activity_date":"2013-02-26T19:12:00+00:00",
"deleted":false,
"assigned_user_id":"seed_max_id",
"assigned_user_name":"Max Jensen",
"team_name":[
  {
    "id":"East",
    "name":"East",
    "name_2":"",
    "primary":false
  },
  {
    "id":1,
    "name":"Global",
    "name_2":"",
    "primary":false
  },
  {
    "id":"West",
    "name":"West",
    "name_2":"",
    "primary":true
  }
],
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"account_type":"Customer",
"industry":"Electronics",
"annual_revenue":"",
"phone_fax":"",
"billing_address_street":"345 Sugar Blvd.",
"billing_address_street_2":"",
"billing_address_street_3":"",
"billing_address_street_4":"",
```

```
"billing_address_city": "San Mateo",
"billing_address_state": "CA",
"billing_address_postalcode": "56019",
"billing_address_country": "USA",
"rating": "",
"phone_office": "(927) 136-9572",
"phone_alternate": "",
"website": "www.sectionvegan.de",
"ownership": "",
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "345 Sugar Blvd.",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "San Mateo",
"shipping_address_state": "CA",
"shipping_address_postalcode": "56019",
"shipping_address_country": "USA",
"email1": "kid.support.vegan@example.info",
"parent_id": "",
"sic_code": "",
"parent_name": "",
"email_opt_out": false,
"invalid_email": false,
"email": [
  {
    "email_address": "kid.support.vegan@example.info",
    "opt_out": "0",
    "invalid_email": "0",
    "primary_address": "1"
  },
  {
    "email_address": "phone.kid@example.cn",
    "opt_out": "0",
    "invalid_email": "0",
```

```
        "primary_address": "0"
    }
],
"campaign_id": "",
"campaign_name": "",
"my_favorite": true,
"_acl": {
    "fields": {
    }
}
}
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added /<module>/:record/favorite PUT endpoint. |

Last Modified: 05/31/2017 05:26pm

/<module>/:record/file GET

Overview

Lists all populated fields of type "file" or of type "image" for a record.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

| Name | Type | Description |
|----------------------|---------|--|
| field | Array | The fields containing file properties. |
| field.content-type | String | The files content type. |
| field.content-length | Integer | The files content length. |
| field.name | String | The files ID. |
| field.width | Integer | The width of the image. |
| field.height | Integer | The height of the image. |
| field.uri | String | The URI of the file. |

Response

```
{
  "picture": {
    "content-type": "image/png",
    "content-length": 72512,
    "name": "1a7b8f5c-b11c-0094-c8d8-512e9daaa983",
    "width": 933,
    "height": 519,

"uri": "http://sugarcrm/rest/v10/Contacts/fa300a0e-0ad1-b322-9601-512d0983c19a/file/picture"
  },
  "record": {
    "my_favorite": false,
    "following": true,
    "id": "fa300a0e-0ad1-b322-9601-512d0983c19a",
    "name": "Test",
```

```
"date_modified": "2014-05-16T03:49:12+02:00",
"modified_user_id": "1",
"modified_by_name": "admin",
"created_by": "1",
"created_by_name": "Administrator",
"doc_owner": "1",
"description": "",
"deleted": false,
"assigned_user_id": "1",
"assigned_user_name": "Administrator",
"team_count": "",
"team_name": [
  {
    "id": 1,
    "name": "Global",
    "name_2": "",
    "primary": true
  }
],
"email": [],
"email1": "",
"email2": "",
"invalid_email": "",
"email_opt_out": "",
"email_addresses_non_primary": "",
"salutation": "",
"first_name": "",
"last_name": "Test",
"full_name": "Test",
"title": "",
"facebook": "",
"twitter": "",
"googleplus": "",
"department": "",
"do_not_call": false,
"phone_home": "",
```

```
"phone_mobile": "",
"phone_work": "",
"phone_other": "",
"phone_fax": "",
"primary_address_street": "",
"primary_address_city": "",
"primary_address_state": "",
"primary_address_postalcode": "",
"primary_address_country": "",
"alt_address_street": "",
"alt_address_city": "",
"alt_address_state": "",
"alt_address_postalcode": "",
"alt_address_country": "",
"assistant": "",
"assistant_phone": "",
"picture": "",
"email_and_name1": "",
"lead_source": "",
"account_name": "",
"account_id": "",
"dnb_principal_id": "",
"opportunity_role_fields": "",
"opportunity_role_id": "",
"opportunity_role": "",
"reports_to_id": "",
"report_to_name": "",
"birthdate": "",
"portal_name": "",
"portal_active": false,
"portal_password": null,
"portal_password1": null,
"portal_app": "",
"preferred_language": "en_us",
"campaign_id": "",
"campaign_name": "",
```

```
"c_accept_status_fields": "",
"m_accept_status_fields": "",
"accept_status_id": "",
"accept_status_name": "",
"accept_status_calls": "",
"accept_status_meetings": "",
"sync_contact": false,
"mkto_sync": false,
"mkto_id": null,
"mkto_lead_score": null,
"_acl": {
  "fields": {}
},
"_module": "Contacts"
}
}
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added /<module>/:record/file GET endpoint. |

Last Modified: 05/31/2017 05:28pm

/<module>/:record/file/:field DELETE

Overview

Removes an attachment from a field for a record and subsequently removes the file from the file system.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

| Name | Type | Description |
|----------------------|---------|--|
| field | Array | The fields containing file properties. |
| field.content-type | String | The files content type. |
| field.content-length | Integer | The files content length. |
| field.name | String | The files ID. |
| field.width | Integer | The width of the image. |
| field.height | Integer | The height of the image. |
| field.uri | String | The URI of the file. |

Response

```
{  
  "picture": {}  
}
```

Change Log

| Version | Change |
|---------|-------------------------------------|
| v10 | Added /<module>/:record/file/:field |

DELETE endpoint.

Last Modified: 05/31/2017 05:25pm

/<module>/:record/file/:field GET

Overview

Retrieves an attached file for a specific field on a record.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

The response from this request is an HTTP response with a forced download. This can be used in rendering images through the API or in offering immediate file downloads.

Response

```
Cache-Control: no-cache, must-revalidate Connection: keep-alive
Content-Encoding: gzip Content-Type: application/octet-stream Date:
Mon, 30 Jan 2012 17:00:46 GMT Expires: Mon, 30 Jan 2012 17:00:45 GMT
Last-Modified: Thu, 10 Nov 2011 19:01:31 GMT Transfer-Encoding:
chunked Vary: Accept-Encoding...
```

Change Log

| Version | Change |
|---------|---|
| v10 | Added /<module>/:record/file/:field GET endpoint. |

Last Modified: 05/31/2017 05:26pm

/<module>/:record/file/:field POST

Overview

Attaches a file to a field on a record.

Request Arguments

| Name | Type | Description | Required |
|--------------------|---------|---|--|
| format | String | The data format. Currently accepts 'sugar-html-json'. | True |
| delete_if_fails | Boolean | Indicates whether the API is to mark related record deleted if the file upload fails. | False |
| oauth_token | String | The oauth_token value. | False - Required if only if delete_if_fails is true. |
| <attachment field> | String | The field and file to populate. | True |

| Name | Type | Description | Required |
|------|------|---|----------|
| | | Example:
{"":"@\\path\\to\\ExampleDocument.txt"} | |

Request

```
{
  "format": "sugar-html-json",
  "delete_if_fails": true,
  "oauth_token": "43b6b327-cc70-c301-3299-512ffb99ad97",
  "<attachment field>": "@\\path\\to\\ExampleDocument.txt"
}
```

Response Arguments

| Name | Type | Description |
|----------------|---------|---------------------------|
| content-type | String | The files content type. |
| content-length | Integer | The files content length. |
| name | String | The files name. |
| uri | String | The URI of the file. |

Response

```
{
  "content-type": "text/plain",
```

```
"content-length":16,
"name":"ExampleDocument.txt",

"uri":"http://sugarcrm/rest/v10/Notes/ca66c92f-5a8b-28b4-4fc8-51
2d099b790b/file/<attachment
field>?format=sugar-html-json&delete_if_fails=1&oauth_token=6ec91cf3-1
f97-25b9-e0b1-512f8971b2d4"
}
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added /<module>/:record/file/:field POST endpoint. |

Last Modified: 05/31/2017 05:26pm

/<module>/:record/file/:field PUT

Overview

This endpoint takes a file or image and saves it to a record that already contains an attachment in the specified field. The PUT method is very similar to the POST method but differs slightly in how the request is constructed. PUT requests should send the file data as the body of the request. Optionally, a filename query parameter can be sent with the request to assign a name. Additionally, the PUT method can accept base64 encoded file data which will be decoded by the endpoint if the `content_transfer_encoding` parameter is set to 'base64'.

NOTE: In lieu of the `content_transfer_encoding` parameter, a request header of `X-Content-Transfer-Encoding` can also be sent with a value of 'base64'. In the event

both the content transfer encoding header and request parameter are sent, the header will be used.

Request Arguments

| Name | Type | Description | Required |
|---------------------------|--------|---|----------|
| filename | String | Filename to save the document as | False |
| content_transfer_encoding | String | When set to 'base64', indicates the file contents are base64 encoded and will result in the file contents being base64 decoded before being saved | False |

Request

PUT /rest/v10/Notes/abcd-1234/file/field/ HTTP/1.1 Host: localhost Connection: keep-alive Content-Length: 23456 Content-Type: application/document-doc ...This is where the bin data would be

Response Arguments

| Name | Type | Description |
|-------|-------|--|
| field | Array | The fields containing file properties. |

| Name | Type | Description |
|----------------------|---------|---------------------------|
| field.content-type | String | The files content type. |
| field.content-length | Integer | The files content length. |
| field.name | String | The files name. |
| field.width | Integer | The width of the image. |
| field.height | Integer | The height of the image. |
| field.uri | String | The URI of the file. |

Response

```
{
  "picture": {
    "content-type": "image/png",
    "content-length": 72512,
    "name": "1a7b8f5c-b11c-0094-c8d8-512e9daaa983",
    "width": 933,
    "height": 519,

"uri": "http:\\\\sugarcrm\\rest\\v10\\Contacts\\fa300a0e-0ad1-b322-9601-512d0983c19a\\file\\picture"
  }
  "attachment": {
    "content-type": "application/document-doc",
    "content-length": "873921",
    "name": "myFile.doc",
    "uri":
"http:\\\\sugarcrm\\rest\\v10\\Contacts\\f2f9aa4d-99a8-e86e-f4d5-512d0986effa\\file\\attachment"
  }
}
```

Change Log

| Version | Change |
|---------|---|
| v10 | Added /<module>/:record/file/:field PUT endpoint. |

Last Modified: 05/31/2017 05:25pm

/<module>/:record/link POST

Overview

Creates relationships to a pre-existing record.

Request Arguments

| Name | Type | Description | Required |
|---------------------|----------|---|----------|
| <relationship link> | <string> | Link between targeted and related records. | True |
| <record ID> | <string> | The name value list of related records. Each item of the list may be either string equal to related item ID, or map containing record ID ("id" key is required in this case) and addition | True |

| Name | Type | Description | Required |
|------|------|--------------------------|----------|
| | | relationship properties. | |

Request

```
{
  link_name: "accounts"
  ids: [
    "da6a3741-2a81-ba7f-f249-512d0932e94e",
    {
      "id": "e689173e-c953-1e14-c215-512d0927e7a2",
      "role": "owner"
    }
  ]
}
```

Response Arguments

| Name | Type | Description |
|-----------------|-------|---------------------------------------|
| record | Array | The record linked to related records. |
| related_records | Array | Records that were associated. |

Response

```
"record": {
```

```
"id": "da6a3741-2a81-ba7f-f249-512d0932e94e",
"name": "Slender Broadband Inc - 1000 units",
"date_entered": "2013-02-26T19:12:00+00:00",
"date_modified": "2013-02-26T19:12:00+00:00",
"modified_user_id": "1",
"modified_by_name": "Administrator",
"created_by": "1",
"created_by_name": "Administrator",
"description": "",
"img": "",
"last_activity_date": "2013-02-28T18:21:00+00:00",
"deleted": false,
"assigned_user_id": "seed_max_id",
"assigned_user_name": "Max Jensen",
"team_name": [
  {
    "id": "East",
    "name": "East",
    "name_2": "",
    "primary": false
  },
  {
    "id": "West",
    "name": "West",
    "name_2": "",
    "primary": true
  }
],
"opportunity_type": "",
"account_name": "Slender Broadband Inc",
"account_id": "181461c6-dc81-1115-1fe0-512d092e8f15",
"campaign_id": "",
"campaign_name": "",
"lead_source": "Campaign",
"amount": "25000",
"base_rate": "1",
```

```
"amount_usdollar": "25000",
"currency_id": "-99",
"currency_name": "",
"currency_symbol": "",
"date_closed": "2013-02-27",
"date_closed_timestamp": "1361992480",
"next_step": "",
"sales_stage": "Needs Analysis",
"sales_status": "New",
"probability": "90",
"best_case": "25000",
"worst_case": "25000",
"commit_stage": "include",
"my_favorite": false,
"_acl": {
  "fields": {
  }
}
},
"related_records": [
  {
    "id": "e689173e-c953-1e14-c215-512d0927e7a2",
    "name": "Gus Dales",
    "date_entered": "2013-02-26T19:12:00+00:00",
    "date_modified": "2013-02-26T19:12:00+00:00",
    "modified_user_id": "1",
    "modified_by_name": "Administrator",
    "created_by": "1",
    "created_by_name": "Administrator",
    "description": "",
    "img": "",
    "deleted": false,
    "assigned_user_id": "seed_sally_id",
    "assigned_user_name": "Sally Bronsen",
    "team_name": [
```

```
        {
            "id": "West",
            "name": "West",
            "name_2": "",
            "primary": true
        }
    ],
    "salutation": "",
    "first_name": "Gus",
    "last_name": "Dales",
    "full_name": "Gus Dales",
    "title": "Director Operations",
    "linkedin": "",
    "facebook": "",
    "twitter": "",
    "googleplus": "",
    "department": "",
    "do_not_call": false,
    "phone_home": "(661) 120-2292",
    "email": [
        {
            "email_address":
"section.sugar.section@example.it",
            "opt_out": "1",
            "invalid_email": "0",
            "primary_address": "0"
        },
        {
            "email_address": "support.qa.kid@example.co.uk",
            "opt_out": "0",
            "invalid_email": "0",
            "primary_address": "1"
        }
    ],
    "phone_mobile": "(294) 447-9707",
    "phone_work": "(036) 840-3216",
```

```
"phone_other": "",
"phone_fax": "",
"email1": "support.ga.kid@example.co.uk",
"email2": "section.sugar.section@example.it",
"invalid_email": false,
"email_opt_out": false,
"primary_address_street": "48920 San Carlos Ave",
"primary_address_street_2": "",
"primary_address_street_3": "",
"primary_address_city": "Persistence",
"primary_address_state": "CA",
"primary_address_postalcode": "54556",
"primary_address_country": "USA",
"alt_address_street": "",
"alt_address_street_2": "",
"alt_address_street_3": "",
"alt_address_city": "",
"alt_address_state": "",
"alt_address_postalcode": "",
"alt_address_country": "",
"assistant": "",
"assistant_phone": "",
"picture": "",
"email_and_name1": "",
"lead_source": "Support Portal User Registration",
"account_name": "Arts & Crafts Inc",
"account_id": "d43243c6-9b8e-2973-ae2-512d09bc34b4",
"opportunity_role_fields": "",
"opportunity_role_id": "",
"opportunity_role": "Technical Advisor",
"reports_to_id": "",
"report_to_name": "",
"portal_name": "GusDales145",
"portal_active": true,
"portal_password": "$1$JxYr6tmM$b.O6.KF42jP46RadSwz0N0",
"portal_password1": "",
```

```
"portal_app": "",
"preferred_language": "en_us",
"campaign_id": "",
"campaign_name": "",
"c_accept_status_fields": "",
"m_accept_status_fields": "",
"accept_status_id": "",
"accept_status_name": "",
"sync_contact": "",
"my_favorite": false,
"_acl": {
  "fields": {
  }
}
},
{
  "id": "da6a3741-2a81-ba7f-f249-512d0932e94e",
  "name": "Slender Broadband Inc - 1000 units",
  "date_entered": "2013-02-26T19:12:00+00:00",
  "date_modified": "2013-02-26T19:12:00+00:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
  "description": "",
  "img": "",
  "last_activity_date": "2013-02-28T18:36:00+00:00",
  "deleted": false,
  "assigned_user_id": "seed_max_id",
  "assigned_user_name": "Max Jensen",
  "team_name": [
    {
      "id": "East",
      "name": "East",
      "name_2": "",
      "primary": false
    }
  ]
}
```

```
    },
    {
      "id": "West",
      "name": "West",
      "name_2": "",
      "primary": true
    }
  ],
  "opportunity_type": "",
  "account_name": "Slender Broadband Inc",
  "account_id": "181461c6-dc81-1115-1fe0-512d092e8f15",
  "campaign_id": "",
  "campaign_name": "",
  "lead_source": "Campaign",
  "amount": "25000",
  "base_rate": "1",
  "amount_usdollar": "25000",
  "currency_id": "-99",
  "currency_name": "",
  "currency_symbol": "",
  "date_closed": "2013-02-27",
  "date_closed_timestamp": "1361992480",
  "next_step": "",
  "sales_stage": "Needs Analysis",
  "sales_status": "New",
  "probability": "90",
  "best_case": "25000",
  "worst_case": "25000",
  "commit_stage": "include",
  "my_favorite": false,
  "_acl": {
    "fields": {
      }
    }
  }
}
```

```
    ]  
}
```

Change Log

| Version | Change |
|-------------|---|
| v10 (7.6.0) | Added support for additional relationship values. |
| v10 (7.1.5) | Added /<module>/:record/link POST endpoint. |

Last Modified: 05/31/2017 05:27pm

/<module>/:record/link/activities GET

Activities on a module's list view

Summary:

This endpoint lists activities across a particular module in the system. It does not use a subscription model unlike the other endpoints. It can be queried as a regular bean collection, but without search, ordering or filtering.

Query Parameters:

| Param | Description | Optional |
|---------|---------------------------------------|----------|
| max_num | A maximum number of records to return | Optional |

| | | |
|--------|---|----------|
| offset | How many records to skip over before records are returned | Optional |
|--------|---|----------|

Input Example:

This endpoint does not accept any input

Output Example:

```
{
  "next_offset": 20, This will be set to -1 when there are no more
  records after this "page".
  "records": [{
    "id": "22fb8b16-de1d-f1dc-b15b-51240efde177",
    "date_entered": "2013-02-19T23:47:11+00:00",
    "date_modified": "2013-02-19T23:47:11+00:00",
    "created_by": "1",
    "deleted": "0",
    "parent_id": "f5bb0331-2c0f-5c7c-b4db-5123caac0056",
    "parent_type": "Contacts",
    "activity_type": "post", This will be the type of activity
    performed.
    "data": {
      "value": "This is a test post on a contact I'm subscribed
      to."
    },
    "comment_count": 0, This will be set to the total number of
    comments on the post.
    "last_comment": { This will be the last comment on the post,
    which can be used to create a comment model on the frontend.
      "deleted": 0,
      "data": []
    },
  },
```

```
    "fields": [],
    "first_name": null,
    "last_name": "Administrator",
    "created_by_name": " Administrator" This will be the
locale-formatted full name of the user.
  }, ... ]
}
```

Last Modified: 11/30/2016 09:21pm

/<module>/:record/link/activities/filter GET

Activities on a module's list view

Summary:

This endpoint lists activities across a particular module in the system. It does not use a subscription model unlike the other endpoints. It can be queried as a regular bean collection, but without search, ordering or filtering.

Query Parameters:

| Param | Description | Optional |
|---------|---|----------|
| max_num | A maximum number of records to return | Optional |
| offset | How many records to skip over before records are returned | Optional |

Input Example:

This endpoint does not accept any input

Output Example:

```
{
  "next_offset": 20, This will be set to -1 when there are no more
records after this "page".
  "records": [{
    "id": "22fb8b16-de1d-f1dc-b15b-51240efde177",
    "date_entered": "2013-02-19T23:47:11+00:00",
    "date_modified": "2013-02-19T23:47:11+00:00",
    "created_by": "1",
    "deleted": "0",
    "parent_id": "f5bb0331-2c0f-5c7c-b4db-5123caac0056",
    "parent_type": "Contacts",
    "activity_type": "post", This will be the type of activity
performed.
    "data": {
      "value": "This is a test post on a contact I'm subscribed
to."
    },
    "comment_count": 0, This will be set to the total number of
comments on the post.
    "last_comment": { This will be the last comment on the post,
which can be used to create a comment model on the frontend.
      "deleted": 0,
      "data": []
    },
    "fields": [],
    "first_name": null,
    "last_name": "Administrator",
    "created_by_name": " Administrator" This will be the
```

```
locale-formatted full name of the user.  
    }, ... ]  
}
```

Last Modified: 11/30/2016 09:22pm

/<module>/:record/link/history GET

Overview

Lists history filtered records.

Summary

This endpoint will return a set of history modules (meetings, calls, notes, tasks, emails) records filtered by an expression.

Request Arguments

| Name | Type | Description | Required |
|---------|---------|--|----------|
| filter | String | The filter expression. Filter expressions are explained below. | False |
| max_num | Integer | A maximum number of records to return. Default is 20. | False |
| offset | Integer | The number of | False |

| Name | Type | Description | Required |
|-------------|---------|--|----------|
| | | records to skip over before records are returned. Default is 0. | |
| order_by | String | How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon.
Example:
name:DESC,account_type:DESC,date_modified:ASC | False |
| deleted | Boolean | Boolean to show deleted records in the result set. | False |
| module_list | String | A list of modules from the valid modules [Tasks, Calls, Emails, Notes, Meetings] that need to be included | False |

Last Modified: 05/31/2017 05:26pm

/<module>/:record/link/:link_name GET

Overview

Lists related filtered records.

Summary

This endpoint will return a set of related records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Care will need to be taken to make sure this filter has appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

| Name | Type | Description | Required |
|---------|---------|---|----------|
| filter | String | The filter expression. Filter expressions are explained below. | False |
| max_num | Integer | A maximum number of records to return. Default is 20. | False |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |
| fields | String | Comma delimited list of fields to return. The field date_modified will | False |

| Name | Type | Description | Required |
|----------|--------|---|----------|
| | | <p>always be returned. This argument can be combined with the view argument. Example: name,account_type,description</p> | |
| view | String | <p>Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list". Example: record</p> | False |
| order_by | String | <p>How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon. Example: name:DESC,accoun</p> | False |

| | | | |
|--|--|-----------------------------------|--|
| | | t_type:DESC,date_
modified:ASC | |
|--|--|-----------------------------------|--|

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
  "filter": [
    {
      "name": "Nelson Inc"
    }
  ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
  "filter": [
```

```

    {
      "name": {
        "$starts": "Nelson"
      }
    }
  ]
}

```

Below is a list of operation types:

| Operation | Description |
|--------------|---|
| \$equals | Performs an exact match on that field. |
| \$not_equals | Performs an exact match on that field. |
| \$not_equals | Matches on non-matching values. |
| \$starts | Matches on anything that starts with the value. |
| \$ends | Matches anything that ends with the value. |
| \$contains | Matches anything that contains the value |
| \$in | Finds anything where field matches one of the values as specified as an array. |
| \$not_in | Finds anything where field does not matches any of the values as specified as an array. |
| \$is_null | Checks if the field is null. This operation does not need a value specified. |
| \$not_null | Checks if the field is not null. This operation does not need a value specified. |
| \$lt | Matches when the field is less than the value. |
| \$lte | Matches when the field is less than or |

| Operation | Description |
|-----------|---|
| | equal to the value. |
| \$gt | Matches when the field is greater than the value. |
| \$gte | Matches when the field is greater than or equal to the value. |

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "\$and" and "\$or".

Example

```
{
  "filter": [
    {
      "$or": [
        {
          "name": "Nelson Inc"
        },
        {
          "name": "Nelson LLC"
        }
      ]
    }
  ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "\$favorite" and "\$owner".

Example

```
{
  "filter":[
    {
      "$favorite":"_this"
    }
  ]
}
```

Response Arguments

| Name | Type | Description |
|-------------|---------|---|
| next_offset | Integer | Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records. |
| records | Array | An array of results containing matched records. |

Reponse

```
{
  "next_offset":-1,
  "records":[
    {
      "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
      "name":"Dale Spivey",
      "date_entered":"2013-02-26T19:12:00+00:00",
      "date_modified":"2013-02-28T05:03:00+00:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "description":"",
      "img":"",
      "deleted":false,
      "assigned_user_id":"seed_sally_id",
      "assigned_user_name":"Sally Bronsen",
      "team_name":[
        {
          "id":"East",
          "name":"East",
          "name_2":"",
          "primary":false
        },
        {
          "id":1,
          "name":"Global",
          "name_2":"",
          "primary":false
        },
        {
          "id":"West",
          "name":"West",
          "name_2":"",
```

```
        "primary":true
    }
],
"salutation":"","
"first_name":"Dale",
"last_name":"Spivey",
"full_name":"Dale Spivey",
"title":"VP Operations",
"linkedin":"","
"facebook":"","
"twitter":"","
"googleplus":"","
"department":"","
"do_not_call":false,
"phone_home":"(523) 825-4311",
"email":[
    {
        "email_address":"sugar.dev.sugar@example.co.jp",
        "opt_out":"0",
        "invalid_email":"0",
        "primary_address":"1"
    },
    {
        "email_address":"the.support@example.biz",
        "opt_out":"0",
        "invalid_email":"0",
        "primary_address":"0"
    }
],
"phone_mobile":"(373) 861-0757",
"phone_work":"(212) 542-9596",
"phone_other":"","
"phone_fax":"","
"email1":"sugar.dev.sugar@example.co.jp",
"email2":"the.support@example.biz",
"invalid_email":false,
```

```
"email_opt_out":false,
"primary_address_street":"345 Sugar Blvd.",
"primary_address_street_2":"","
"primary_address_street_3":"","
"primary_address_city":"Denver",
"primary_address_state":"CA",
"primary_address_postalcode":"87261",
"primary_address_country":"USA",
"alt_address_street":"","
"alt_address_street_2":"","
"alt_address_street_3":"","
"alt_address_city":"","
"alt_address_state":"","
"alt_address_postalcode":"","
"alt_address_country":"","
"assistant":"","
"assistant_phone":"","
"picture":"","
"email_and_name1":"","
"lead_source":"Campaign",
"account_name":"Smallville Resources Inc",
"account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",
"opportunity_role_fields":"","
"opportunity_role_id":"","
"opportunity_role":"","
"reports_to_id":"","
"report_to_name":"","
"portal_name":"DaleSpivey97",
"portal_active":true,
"portal_password":"$1$yKMAONHM$Y5S.8CY.WZCZCwfGD1a1Q\/",
"portal_password1":"","
"portal_app":"","
"preferred_language":"en_us",
"campaign_id":"","
"campaign_name":"","
"c_accept_status_fields":"","
```

```
"m_accept_status_fields": "",
"accept_status_id": "",
"accept_status_name": "",
"sync_contact": "",
"my_favorite": false,
"_acl": {
  "fields": {
    }
  }
},
{
  "id": "95e17367-9b3d-0e26-22dc-512d0961fedf",
  "name": "Florence Haddock",
  "date_entered": "2013-02-26T19:12:00+00:00",
  "date_modified": "2013-02-26T19:12:00+00:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
  "description": "",
  "img": "",
  "deleted": false,
  "assigned_user_id": "seed_sally_id",
  "assigned_user_name": "Sally Bronsen",
  "team_name": [
    {
      "id": "East",
      "name": "East",
      "name_2": "",
      "primary": false
    },
    {
      "id": 1,
      "name": "Global",
      "name_2": ""
    }
  ]
}
```

```
        "primary":false
    },
    {
        "id":"West",
        "name":"West",
        "name_2":"",
        "primary":true
    }
],
"salutation":"",
"first_name":"Florence",
"last_name":"Haddock",
"full_name":"Florence Haddock",
"title":"Director Sales",
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(729) 845-3137",
"email":[
    {
        "email_address":"dev.vegan@example.de",
        "opt_out":"1",
        "invalid_email":"0",
        "primary_address":"0"
    },
    {
        "email_address":"section71@example.it",
        "opt_out":"0",
        "invalid_email":"0",
        "primary_address":"1"
    }
],
"phone_mobile":"(246) 233-1382",
```

```
"phone_work": "(565) 696-6981",
"phone_other": "",
"phone_fax": "",
"email1": "section71@example.it",
"email2": "dev.vegan@example.de",
"invalid_email": false,
"email_opt_out": false,
"primary_address_street": "111 Silicon Valley Road",
"primary_address_street_2": "",
"primary_address_street_3": "",
"primary_address_city": "Denver",
"primary_address_state": "CA",
"primary_address_postalcode": "79900",
"primary_address_country": "USA",
"alt_address_street": "",
"alt_address_street_2": "",
"alt_address_street_3": "",
"alt_address_city": "",
"alt_address_state": "",
"alt_address_postalcode": "",
"alt_address_country": "",
"assistant": "",
"assistant_phone": "",
"picture": "",
"email_and_name1": "",
"lead_source": "Support Portal User Registration",
"account_name": "Smallville Resources Inc",
"account_id": "d5db6292-5c24-eb61-e202-512d09f0134e",
"opportunity_role_fields": "",
"opportunity_role_id": "",
"opportunity_role": "",
"reports_to_id": "",
"report_to_name": "",
"portal_name": "FlorenceHaddock169",
"portal_active": true,
"portal_password": "$1$nWFhTbK6$JF9BCGSqL\NCRbhueX5ia0",
```

```

"portal_password1": "",
"portal_app": "",
"preferred_language": "en_us",
"campaign_id": "",
"campaign_name": "",
"c_accept_status_fields": "",
"m_accept_status_fields": "",
"accept_status_id": "",
"accept_status_name": "",
"sync_contact": "",
"my_favorite": false,
"_acl": {
  "fields": {
    }
  }
}
]
}

```

Change Log

| Version | Change |
|---------|--|
| v10 | Added
/<module>/:record/link/:link_name/filter
GET endpoint. |

Last Modified: 05/31/2017 05:26pm

/<module>/:record/link/:link_name POST

Overview

Creates a related record.

Request Arguments

| Name | Type | Description | Required |
|----------------|---------------------|--|----------|
| <record field> | <record field type> | The name value list of fields to populate. | True |

Request

```
{  
  "first_name": "Bill",  
  "last_name": "Edwards"  
}
```

Response Arguments

| Name | Type | Description |
|----------------|-------|--|
| record | Array | The record associated to the newly created record. |
| related_record | Array | The record that was created and associated. |

Response

```
{
  "record": {
    "id": "da6a3741-2a81-ba7f-f249-512d0932e94e",
    "name": "Slender Broadband Inc - 1000 units",
    "date_entered": "2013-02-26T19:12:00+00:00",
    "date_modified": "2013-02-26T19:12:00+00:00",
    "modified_user_id": "1",
    "modified_by_name": "Administrator",
    "created_by": "1",
    "created_by_name": "Administrator",
    "description": "",
    "img": "",
    "deleted": false,
    "assigned_user_id": "seed_max_id",
    "assigned_user_name": "Max Jensen",
    "team_name": [
      {
        "id": "East",
        "name": "East",
        "name_2": "",
        "primary": false
      },
      {
        "id": "West",
        "name": "West",
        "name_2": "",
        "primary": true
      }
    ],
    "opportunity_type": "",
    "account_name": "Slender Broadband Inc",
    "account_id": "181461c6-dc81-1115-1fe0-512d092e8f15",
    "campaign_id": "",
    "campaign_name": ""
  }
}
```

```
"lead_source": "Campaign",
"amount": "25000",
"base_rate": "1",
"amount_usdollar": "25000",
"currency_id": "-99",
"currency_name": "",
"currency_symbol": "",
"date_closed": "2013-02-27",
"date_closed_timestamp": "1361992480",
"next_step": "",
"sales_stage": "Needs Analysis",
"sales_status": "New",
"probability": "90",
"best_case": "25000",
"worst_case": "25000",
"commit_stage": "include",
"my_favorite": false,
"_acl": {
  "fields": {
    }
  }
},
"related_record": {
  "id": "e1c495cb-af17-1b37-dd66-512f934fe155",
  "name": "Bill Edwards",
  "date_entered": "2013-02-28T17:25:00+00:00",
  "date_modified": "2013-02-28T17:25:00+00:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
  "description": "",
  "img": "",
  "deleted": false,
  "assigned_user_id": ""
```

```
"assigned_user_name": "",
"team_name": [
  {
    "id": 1,
    "name": "Global",
    "name_2": "",
    "primary": true
  }
],
"salutation": "",
"first_name": "Bill",
"last_name": "Edwards",
"full_name": "Bill Edwards",
"title": "",
"linkedin": "",
"facebook": "",
"twitter": "",
"googleplus": "",
"department": "",
"do_not_call": false,
"phone_home": "",
"email": [

],
"phone_mobile": "",
"phone_work": "",
"phone_other": "",
"phone_fax": "",
"email1": "",
"email2": "",
"invalid_email": "",
"email_opt_out": "",
"primary_address_street": "",
"primary_address_street_2": "",
"primary_address_street_3": "",
"primary_address_city": "",
```

```
"primary_address_state":"","  
"primary_address_postalcode":"","  
"primary_address_country":"","  
"alt_address_street":"","  
"alt_address_street_2":"","  
"alt_address_street_3":"","  
"alt_address_city":"","  
"alt_address_state":"","  
"alt_address_postalcode":"","  
"alt_address_country":"","  
"assistant":"","  
"assistant_phone":"","  
"picture":"","  
"email_and_name1":"","  
"lead_source":"","  
"account_id":"","  
"opportunity_role_fields":"","  
"opportunity_role_id":"","  
"opportunity_role":"","  
"reports_to_id":"","  
"report_to_name":"","  
"portal_name":"","  
"portal_active":false,  
"portal_password":"","  
"portal_password1":"","  
"portal_app":"","  
"preferred_language":"en_us",  
"campaign_id":"","  
"campaign_name":"","  
"c_accept_status_fields":"","  
"m_accept_status_fields":"","  
"accept_status_id":"","  
"accept_status_name":"","  
"sync_contact":"","  
"my_favorite":false,  
"_acl":{
```

```
        "fields":{
          }
        }
      }
    }
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added
/<module>/:record/link/:link_name
POST endpoint. |

Last Modified: 05/31/2017 05:26pm

/<module>/:record/link/:link_name/add_record_list/:remote_id POST

Overview

Creates relationships to pre-existing record from a record list.

URL Arguments

| Name | Type | Description | Required |
|-------------|----------|---------------------|----------|
| <record ID> | <string> | Target record ID. | True. |
| <report ID> | <string> | Report ID for Saved | True. |

| Name | Type | Description | Required |
|---------------------|----------|--|----------|
| | | Report. | |
| <relationship link> | <string> | Link between targeted and related records. | True. |

Request

Response Arguments

| Name | Type | Description |
|-----------------|-------|---------------------------------------|
| record | Array | The record linked to related records. |
| related_records | Array | Record IDs that were associated. |

Response

```

"record": {
  "id": "da6a3741-2a81-ba7f-f249-512d0932e94e",
  "name": "Slender Broadband Inc - 1000 units",
  "date_entered": "2013-02-26T19:12:00+00:00",
  "date_modified": "2013-02-26T19:12:00+00:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
  "description": "",
  "img": "",

```

```
"last_activity_date": "2013-02-28T18:21:00+00:00",
"deleted": false,
"assigned_user_id": "seed_max_id",
"assigned_user_name": "Max Jensen",
"team_name": [
  {
    "id": "East",
    "name": "East",
    "name_2": "",
    "primary": false
  },
  {
    "id": "West",
    "name": "West",
    "name_2": "",
    "primary": true
  }
],
"opportunity_type": "",
"account_name": "Slender Broadband Inc",
"account_id": "181461c6-dc81-1115-1fe0-512d092e8f15",
"campaign_id": "",
"campaign_name": "",
"lead_source": "Campaign",
"amount": "25000",
"base_rate": "1",
"amount_usdollar": "25000",
"currency_id": "-99",
"currency_name": "",
"currency_symbol": "",
"date_closed": "2013-02-27",
"date_closed_timestamp": "1361992480",
"next_step": "",
"sales_stage": "Needs Analysis",
"sales_status": "New",
"probability": "90",
```

```

    "best_case": "25000",
    "worst_case": "25000",
    "commit_stage": "include",
    "my_favorite": false,
    "_acl": {
      "fields": {
        }
      }
    },
    "related_records": [
      success: [
        "e689173e-c953-1e14-c215-512d0927e7a2",
        "da6a3741-2a81-ba7f-f249-512d0932e94e",
        "181461c6-dc81-1115-1fe0-512d092e8f15"
      ],
      error: []
    ]
  }

```

Change Log

| Version | Change |
|-------------|--|
| v10 (7.2.0) | Added
/<module>/:record/link/:link_name/add_record_list/:remote_id POST endpoint. |

Last Modified: 05/31/2017 05:25pm

/<module>/:record/link/:link_name/count GET

Overview

Lists related filtered records.

Summary

This endpoint will return a set of related records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Care will need to be taken to make sure this filter has appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

| Name | Type | Description | Required |
|---------|---------|---|----------|
| filter | String | The filter expression. Filter expressions are explained below. | False |
| max_num | Integer | A maximum number of records to return. Default is 20. | False |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |
| fields | String | Comma delimited list of fields to | False |

| Name | Type | Description | Required |
|----------|--------|--|----------|
| | | return. The field date_modified will always be returned. This argument can be combined with the view argument. Example: name,account_type,description | |
| view | String | Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list". Example: record | False |
| order_by | String | How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon. | False |

| | | | |
|--|--|---|--|
| | | Example:
name:DESC,account_type:DESC,date_modified:ASC | |
|--|--|---|--|

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
  "filter": [
    {
      "name": "Nelson Inc"
    }
  ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```

{
  "filter":[
    {
      "name":{
        "$starts":"Nelson"
      }
    }
  ]
}

```

Below is a list of operation types:

| Operation | Description |
|--------------|---|
| \$equals | Performs an exact match on that field. |
| \$not_equals | Performs an exact match on that field. |
| \$not_equals | Matches on non-matching values. |
| \$starts | Matches on anything that starts with the value. |
| \$ends | Matches anything that ends with the value. |
| \$contains | Matches anything that contains the value |
| \$in | Finds anything where field matches one of the values as specified as an array. |
| \$not_in | Finds anything where field does not matches any of the values as specified as an array. |
| \$is_null | Checks if the field is null. This operation does not need a value specified. |
| \$not_null | Checks if the field is not null. This operation does not need a value specified. |
| \$lt | Matches when the field is less than the |

| Operation | Description |
|-----------|---|
| | value. |
| \$lte | Matches when the field is less than or equal to the value. |
| \$gt | Matches when the field is greater than the value. |
| \$gte | Matches when the field is greater than or equal to the value. |

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "\$and" and "\$or".

Example

```
{
  "filter": [
    {
      "$or": [
        {
          "name": "Nelson Inc"
        },
        {
          "name": "Nelson LLC"
        }
      ]
    }
  ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "\$favorite" and "\$owner".

Example

```
{
  "filter":[
    {
      "$favorite":"_this"
    }
  ]
}
```

Response Arguments

| Name | Type | Description |
|-------------|---------|---|
| next_offset | Integer | Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records. |
| records | Array | An array of results containing matched records. |

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

Reponse

```
{
  "next_offset":-1,
  "records":[
    {
      "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
      "name":"Dale Spivey",
      "date_entered":"2013-02-26T19:12:00+00:00",
      "date_modified":"2013-02-28T05:03:00+00:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "description":"",
      "img":"",
      "deleted":false,
      "assigned_user_id":"seed_sally_id",
      "assigned_user_name":"Sally Bronsen",
      "team_name":[
        {
          "id":"East",
          "name":"East",
          "name_2":"",
          "primary":false
        },
        {
          "id":1,
          "name":"Global",
          "name_2":"",
          "primary":false
        }
      ]
    }
  ]
}
```

```
    {
      "id": "West",
      "name": "West",
      "name_2": "",
      "primary": true
    }
  ],
  "salutation": "",
  "first_name": "Dale",
  "last_name": "Spivey",
  "full_name": "Dale Spivey",
  "title": "VP Operations",
  "linkedin": "",
  "facebook": "",
  "twitter": "",
  "googleplus": "",
  "department": "",
  "do_not_call": false,
  "phone_home": "(523) 825-4311",
  "email": [
    {
      "email_address": "sugar.dev.sugar@example.co.jp",
      "opt_out": "0",
      "invalid_email": "0",
      "primary_address": "1"
    },
    {
      "email_address": "the.support@example.biz",
      "opt_out": "0",
      "invalid_email": "0",
      "primary_address": "0"
    }
  ],
  "phone_mobile": "(373) 861-0757",
  "phone_work": "(212) 542-9596",
```

```
"phone_other":"","  
"phone_fax":"","  
"email1":"sugar.dev.sugar@example.co.jp",  
"email2":"the.support@example.biz",  
"invalid_email":false,  
"email_opt_out":false,  
"primary_address_street":"345 Sugar Blvd.",  
"primary_address_street_2":"","  
"primary_address_street_3":"","  
"primary_address_city":"Denver",  
"primary_address_state":"CA",  
"primary_address_postalcode":"87261",  
"primary_address_country":"USA",  
"alt_address_street":"","  
"alt_address_street_2":"","  
"alt_address_street_3":"","  
"alt_address_city":"","  
"alt_address_state":"","  
"alt_address_postalcode":"","  
"alt_address_country":"","  
"assistant":"","  
"assistant_phone":"","  
"picture":"","  
"email_and_name1":"","  
"lead_source":"Campaign",  
"account_name":"Smallville Resources Inc",  
"account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",  
"opportunity_role_fields":"","  
"opportunity_role_id":"","  
"opportunity_role":"","  
"reports_to_id":"","  
"report_to_name":"","  
"portal_name":"DaleSpivey97",  
"portal_active":true,  
"portal_password":"$1$yKMAONHM$Y5S.8CY.WZCZCwfGD1a1Q\/",
```

```
"portal_password1": "",
"portal_app": "",
"preferred_language": "en_us",
"campaign_id": "",
"campaign_name": "",
"c_accept_status_fields": "",
"m_accept_status_fields": "",
"accept_status_id": "",
"accept_status_name": "",
"sync_contact": "",
"my_favorite": false,
"_acl": {
  "fields": {
    }
  }
},
{
  "id": "95e17367-9b3d-0e26-22dc-512d0961fedf",
  "name": "Florence Haddock",
  "date_entered": "2013-02-26T19:12:00+00:00",
  "date_modified": "2013-02-26T19:12:00+00:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
  "description": "",
  "img": "",
  "deleted": false,
  "assigned_user_id": "seed_sally_id",
  "assigned_user_name": "Sally Bronsen",
  "team_name": [
    {
      "id": "East",
      "name": "East",
    }
  ]
}
```

```
        "name_2": "",
        "primary": false
    },
    {
        "id": 1,
        "name": "Global",
        "name_2": "",
        "primary": false
    },
    {
        "id": "West",
        "name": "West",
        "name_2": "",
        "primary": true
    }
],
"salutation": "",
"first_name": "Florence",
"last_name": "Haddock",
"full_name": "Florence Haddock",
"title": "Director Sales",
"linkedin": "",
"facebook": "",
"twitter": "",
"googleplus": "",
"department": "",
"do_not_call": false,
"phone_home": "(729) 845-3137",
"email": [
    {
        "email_address": "dev.vegan@example.de",
        "opt_out": "1",
        "invalid_email": "0",
        "primary_address": "0"
    },

```

```
{
  "email_address": "section71@example.it",
  "opt_out": "0",
  "invalid_email": "0",
  "primary_address": "1"
}
],
"phone_mobile": "(246) 233-1382",
"phone_work": "(565) 696-6981",
"phone_other": "",
"phone_fax": "",
"email1": "section71@example.it",
"email2": "dev.vegan@example.de",
"invalid_email": false,
"email_opt_out": false,
"primary_address_street": "111 Silicon Valley Road",
"primary_address_street_2": "",
"primary_address_street_3": "",
"primary_address_city": "Denver",
"primary_address_state": "CA",
"primary_address_postalcode": "79900",
"primary_address_country": "USA",
"alt_address_street": "",
"alt_address_street_2": "",
"alt_address_street_3": "",
"alt_address_city": "",
"alt_address_state": "",
"alt_address_postalcode": "",
"alt_address_country": "",
"assistant": "",
"assistant_phone": "",
"picture": "",
"email_and_name1": "",
"lead_source": "Support Portal User Registration",
"account_name": "Smallville Resources Inc",
```

```

"account_id":"d5db6292-5c24-eb61-e202-512d09f0134e",
"opportunity_role_fields":"","
"opportunity_role_id":"","
"opportunity_role":"","
"reports_to_id":"","
"report_to_name":"","
"portal_name":"FlorenceHaddock169",
"portal_active":true,
"portal_password":"$1$nWFhTbK6$JF9BCGSqL\NCRbhueX5ia0",
"portal_password1":"","
"portal_app":"","
"preferred_language":"en_us",
"campaign_id":"","
"campaign_name":"","
"c_accept_status_fields":"","
"m_accept_status_fields":"","
"accept_status_id":"","
"accept_status_name":"","
"sync_contact":"","
"my_favorite":false,
"_acl":{
  "fields":{
    }
  }
}
]
}

```

Change Log

| Version | Change |
|---------|--------|
|---------|--------|

| | |
|-----|--|
| v10 | Added
/<module>/:record/link/:link_name/filter
GET endpoint. |
|-----|--|

Last Modified: 05/31/2017 05:25pm

/<module>/:record/link/:link_name/filter GET

Overview

Lists related filtered records.

Summary

This endpoint will return a set of related records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Care will need to be taken to make sure this filter has appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

| Name | Type | Description | Required |
|--------|--------|--|----------|
| filter | String | The filter expression. Filter expressions are explained below. | False |

| Name | Type | Description | Required |
|---------|---------|--|----------|
| max_num | Integer | A maximum number of records to return. Default is 20. | False |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |
| fields | String | Comma delimited list of fields to return. The field date_modified will always be returned. This argument can be combined with the view argument. Example: name,account_type,description | False |
| view | String | Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be | False |

| | | | |
|----------|--------|--|-------|
| | | used in combination with the fields argument. Common views are "record" and "list".
Example: record | |
| order_by | String | How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon.
Example:
name:DESC,account_type:DESC,date_modified:ASC | False |

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
  "filter": [
    {
```

```
        "name": "Nelson Inc"
      }
    ]
  }
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
  "filter": [
    {
      "name": {
        "$starts": "Nelson"
      }
    }
  ]
}
```

Below is a list of operation types:

| Operation | Description |
|---------------------------|---|
| <code>\$equals</code> | Performs an exact match on that field. |
| <code>\$not_equals</code> | Performs an exact match on that field. |
| <code>\$not_equals</code> | Matches on non-matching values. |
| <code>\$starts</code> | Matches on anything that starts with the value. |
| <code>\$ends</code> | Matches anything that ends with the |

| Operation | Description |
|------------|---|
| | value. |
| \$contains | Matches anything that contains the value |
| \$in | Finds anything where field matches one of the values as specified as an array. |
| \$not_in | Finds anything where field does not matches any of the values as specified as an array. |
| \$is_null | Checks if the field is null. This operation does not need a value specified. |
| \$not_null | Checks if the field is not null. This operation does not need a value specified. |
| \$lt | Matches when the field is less than the value. |
| \$lte | Matches when the field is less than or equal to the value. |
| \$gt | Matches when the field is greater than the value. |
| \$gte | Matches when the field is greater than or equal to the value. |

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "\$and" and "\$or".

Example

```
{
  "filter":[
    {
      "$or":[
        {
          "name":"Nelson Inc"
        },
        {
          "name":"Nelson LLC"
        }
      ]
    }
  ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "\$favorite" and "\$owner".

Example

```
{
  "filter":[
    {
      "$favorite":"_this"
    }
  ]
}
```

}

Response Arguments

| Name | Type | Description |
|-------------|---------|---|
| next_offset | Integer | Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records. |
| records | Array | An array of results containing matched records. |

Response

```
{
  "next_offset": -1,
  "records": [
    {
      "id": "fa300a0e-0ad1-b322-9601-512d0983c19a",
      "name": "Dale Spivey",
      "date_entered": "2013-02-26T19:12:00+00:00",
      "date_modified": "2013-02-28T05:03:00+00:00",
      "modified_user_id": "1",
      "modified_by_name": "Administrator",
      "created_by": "1",
      "created_by_name": "Administrator",
      "description": ""
    }
  ]
}
```

```
"img":"","  
"deleted":false,  
"assigned_user_id":"seed_sally_id",  
"assigned_user_name":"Sally Bronsen",  
"team_name":[  
  {  
    "id":"East",  
    "name":"East",  
    "name_2":"","  
    "primary":false  
  },  
  {  
    "id":1,  
    "name":"Global",  
    "name_2":"","  
    "primary":false  
  },  
  {  
    "id":"West",  
    "name":"West",  
    "name_2":"","  
    "primary":true  
  }  
],  
"salutation":"","  
"first_name":"Dale",  
"last_name":"Spivey",  
"full_name":"Dale Spivey",  
"title":"VP Operations",  
"linkedin":"","  
"facebook":"","  
"twitter":"","  
"googleplus":"","  
"department":"","  
"do_not_call":false,
```

```
"phone_home": "(523) 825-4311",
"email": [
  {
    "email_address": "sugar.dev.sugar@example.co.jp",
    "opt_out": "0",
    "invalid_email": "0",
    "primary_address": "1"
  },
  {
    "email_address": "the.support@example.biz",
    "opt_out": "0",
    "invalid_email": "0",
    "primary_address": "0"
  }
],
"phone_mobile": "(373) 861-0757",
"phone_work": "(212) 542-9596",
"phone_other": "",
"phone_fax": "",
"email1": "sugar.dev.sugar@example.co.jp",
"email2": "the.support@example.biz",
"invalid_email": false,
"email_opt_out": false,
"primary_address_street": "345 Sugar Blvd.",
"primary_address_street_2": "",
"primary_address_street_3": "",
"primary_address_city": "Denver",
"primary_address_state": "CA",
"primary_address_postalcode": "87261",
"primary_address_country": "USA",
"alt_address_street": "",
"alt_address_street_2": "",
"alt_address_street_3": "",
"alt_address_city": "",
"alt_address_state": "",
```

```
"alt_address_postalcode": "",
"alt_address_country": "",
"assistant": "",
"assistant_phone": "",
"picture": "",
"email_and_name1": "",
"lead_source": "Campaign",
"account_name": "Smallville Resources Inc",
"account_id": "d5db6292-5c24-eb61-e202-512d09f0134e",
"opportunity_role_fields": "",
"opportunity_role_id": "",
"opportunity_role": "",
"reports_to_id": "",
"report_to_name": "",
"portal_name": "DaleSpivey97",
"portal_active": true,
"portal_password": "$1$yKMAONHM$Y5S.8CY.WZCZCwfGD1a1Q\/",
"portal_password1": "",
"portal_app": "",
"preferred_language": "en_us",
"campaign_id": "",
"campaign_name": "",
"c_accept_status_fields": "",
"m_accept_status_fields": "",
"accept_status_id": "",
"accept_status_name": "",
"sync_contact": "",
"my_favorite": false,
"_acl": {
  "fields": {
    }
  }
},
{
```

```
"id": "95e17367-9b3d-0e26-22dc-512d0961fedf",
"name": "Florence Haddock",
"date_entered": "2013-02-26T19:12:00+00:00",
"date_modified": "2013-02-26T19:12:00+00:00",
"modified_user_id": "1",
"modified_by_name": "Administrator",
"created_by": "1",
"created_by_name": "Administrator",
"description": "",
"img": "",
"deleted": false,
"assigned_user_id": "seed_sally_id",
"assigned_user_name": "Sally Bronsen",
"team_name": [
  {
    "id": "East",
    "name": "East",
    "name_2": "",
    "primary": false
  },
  {
    "id": 1,
    "name": "Global",
    "name_2": "",
    "primary": false
  },
  {
    "id": "West",
    "name": "West",
    "name_2": "",
    "primary": true
  }
],
"salutation": "",
"first_name": "Florence",
```

```
"last_name": "Haddock",
"full_name": "Florence Haddock",
"title": "Director Sales",
"linkedin": "",
"facebook": "",
"twitter": "",
"googleplus": "",
"department": "",
"do_not_call": false,
"phone_home": "(729) 845-3137",
"email": [
  {
    "email_address": "dev.vegan@example.de",
    "opt_out": "1",
    "invalid_email": "0",
    "primary_address": "0"
  },
  {
    "email_address": "section71@example.it",
    "opt_out": "0",
    "invalid_email": "0",
    "primary_address": "1"
  }
],
"phone_mobile": "(246) 233-1382",
"phone_work": "(565) 696-6981",
"phone_other": "",
"phone_fax": "",
"email1": "section71@example.it",
"email2": "dev.vegan@example.de",
"invalid_email": false,
"email_opt_out": false,
"primary_address_street": "111 Silicon Valley Road",
"primary_address_street_2": "",
"primary_address_street_3": "",
```

```
"primary_address_city": "Denver",
"primary_address_state": "CA",
"primary_address_postalcode": "79900",
"primary_address_country": "USA",
"alt_address_street": "",
"alt_address_street_2": "",
"alt_address_street_3": "",
"alt_address_city": "",
"alt_address_state": "",
"alt_address_postalcode": "",
"alt_address_country": "",
"assistant": "",
"assistant_phone": "",
"picture": "",
"email_and_name1": "",
"lead_source": "Support Portal User Registration",
"account_name": "Smallville Resources Inc",
"account_id": "d5db6292-5c24-eb61-e202-512d09f0134e",
"opportunity_role_fields": "",
"opportunity_role_id": "",
"opportunity_role": "",
"reports_to_id": "",
"report_to_name": "",
"portal_name": "FlorenceHaddock169",
"portal_active": true,
"portal_password": "$1$WfHtBk6$JF9BCGSqL\NcrbhueX5ia0",
"portal_password1": "",
"portal_app": "",
"preferred_language": "en_us",
"campaign_id": "",
"campaign_name": "",
"c_accept_status_fields": "",
"m_accept_status_fields": "",
"accept_status_id": "",
"accept_status_name": "",
```

```
    "sync_contact": "",
    "my_favorite": false,
    "_acl": {
      "fields": {
        }
      }
    }
  ]
}
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added
/<module>/:record/link/:link_name/filter
GET endpoint. |

Last Modified: 05/31/2017 05:25pm

/<module>/:record/link/:link_name/filter/count GET

Overview

Lists related filtered records.

Summary

This endpoint will return a set of related records filtered by an expression. The

filter can be applied to multiple fields and have multiple and/or conditions in it. Care will need to be taken to make sure this filter has appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

| Name | Type | Description | Required |
|---------|---------|--|----------|
| filter | String | The filter expression. Filter expressions are explained below. | False |
| max_num | Integer | A maximum number of records to return. Default is 20. | False |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |
| fields | String | Comma delimited list of fields to return. The field date_modified will always be returned. This argument can be combined with the view argument. | False |

| Name | Type | Description | Required |
|----------|--------|---|----------|
| | | Example:
name,account_type,
description | |
| view | String | Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list".
Example: record | False |
| order_by | String | How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon.
Example:
name:DESC,account_type:DESC,date_modified:ASC | False |

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
  "filter": [
    {
      "name": "Nelson Inc"
    }
  ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
  "filter": [
    {
      "name": {
```

```

    }
  }
]
}
    "$starts": "Nelson"
  }
}

```

Below is a list of operation types:

| Operation | Description |
|--------------|---|
| \$equals | Performs an exact match on that field. |
| \$not_equals | Performs an exact match on that field. |
| \$not_equals | Matches on non-matching values. |
| \$starts | Matches on anything that starts with the value. |
| \$ends | Matches anything that ends with the value. |
| \$contains | Matches anything that contains the value |
| \$in | Finds anything where field matches one of the values as specified as an array. |
| \$not_in | Finds anything where field does not matches any of the values as specified as an array. |
| \$is_null | Checks if the field is null. This operation does not need a value specified. |
| \$not_null | Checks if the field is not null. This operation does not need a value specified. |
| \$lt | Matches when the field is less than the value. |
| \$lte | Matches when the field is less than or |

| Operation | Description |
|-----------|---|
| | equal to the value. |
| \$gt | Matches when the field is greater than the value. |
| \$gte | Matches when the field is greater than or equal to the value. |

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "\$and" and "\$or".

Example

```
{
  "filter": [
    {
      "$or": [
        {
          "name": "Nelson Inc"
        },
        {
          "name": "Nelson LLC"
        }
      ]
    }
  ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "\$favorite" and "\$owner".

Example

```
{
  "filter":[
    {
      "$favorite":"_this"
    }
  ]
}
```

Response Arguments

| Name | Type | Description |
|-------------|---------|---|
| next_offset | Integer | Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records. |
| records | Array | An array of results containing matched records. |

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

Response

```
{
  "next_offset":-1,
  "records":[
    {
      "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
      "name":"Dale Spivey",
      "date_entered":"2013-02-26T19:12:00+00:00",
      "date_modified":"2013-02-28T05:03:00+00:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "description":"",
      "img":"",
      "deleted":false,
      "assigned_user_id":"seed_sally_id",
      "assigned_user_name":"Sally Bronsen",
      "team_name":[
        {
          "id":"East",
          "name":"East",
          "name_2":"",
          "primary":false
        },
        {
          "id":1,
          "name":"Global",
          "name_2":"",
          "primary":false
        }
      ]
    }
  ]
}
```

```
    },
    {
      "id": "West",
      "name": "West",
      "name_2": "",
      "primary": true
    }
  ],
  "salutation": "",
  "first_name": "Dale",
  "last_name": "Spivey",
  "full_name": "Dale Spivey",
  "title": "VP Operations",
  "linkedin": "",
  "facebook": "",
  "twitter": "",
  "googleplus": "",
  "department": "",
  "do_not_call": false,
  "phone_home": "(523) 825-4311",
  "email": [
    {
      "email_address": "sugar.dev.sugar@example.co.jp",
      "opt_out": "0",
      "invalid_email": "0",
      "primary_address": "1"
    },
    {
      "email_address": "the.support@example.biz",
      "opt_out": "0",
      "invalid_email": "0",
      "primary_address": "0"
    }
  ]
}
```

```
],
"phone_mobile": "(373) 861-0757",
"phone_work": "(212) 542-9596",
"phone_other": "",
"phone_fax": "",
"email1": "sugar.dev.sugar@example.co.jp",
"email2": "the.support@example.biz",
"invalid_email": false,
"email_opt_out": false,
"primary_address_street": "345 Sugar Blvd.",
"primary_address_street_2": "",
"primary_address_street_3": "",
"primary_address_city": "Denver",
"primary_address_state": "CA",
"primary_address_postalcode": "87261",
"primary_address_country": "USA",
"alt_address_street": "",
"alt_address_street_2": "",
"alt_address_street_3": "",
"alt_address_city": "",
"alt_address_state": "",
"alt_address_postalcode": "",
"alt_address_country": "",
"assistant": "",
"assistant_phone": "",
"picture": "",
"email_and_name1": "",
"lead_source": "Campaign",
"account_name": "Smallville Resources Inc",
"account_id": "d5db6292-5c24-eb61-e202-512d09f0134e",
"opportunity_role_fields": "",
"opportunity_role_id": "",
"opportunity_role": ""
```

```
"reports_to_id": "",
"report_to_name": "",
"portal_name": "DaleSpivey97",
"portal_active": true,
"portal_password": "$1$yKMAONHM$Y5S.8CY.WZCZCwfGD1a1Q\/",
"portal_password1": "",
"portal_app": "",
"preferred_language": "en_us",
"campaign_id": "",
"campaign_name": "",
"c_accept_status_fields": "",
"m_accept_status_fields": "",
"accept_status_id": "",
"accept_status_name": "",
"sync_contact": "",
"my_favorite": false,
"_acl": {
  "fields": {
    }
  }
},
{
  "id": "95e17367-9b3d-0e26-22dc-512d0961fedf",
  "name": "Florence Haddock",
  "date_entered": "2013-02-26T19:12:00+00:00",
  "date_modified": "2013-02-26T19:12:00+00:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
  "description": "",
  "img": ""
```

```
"deleted":false,
"assigned_user_id":"seed_sally_id",
"assigned_user_name":"Sally Bronsen",
"team_name":[
  {
    "id":"East",
    "name":"East",
    "name_2":"",
    "primary":false
  },
  {
    "id":1,
    "name":"Global",
    "name_2":"",
    "primary":false
  },
  {
    "id":"West",
    "name":"West",
    "name_2":"",
    "primary":true
  }
],
"salutation":"",
"first_name":"Florence",
"last_name":"Haddock",
"full_name":"Florence Haddock",
"title":"Director Sales",
"linkedin":"",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
```

```
"do_not_call":false,
"phone_home":"(729) 845-3137",
"email":[
  {
    "email_address":"dev.vegan@example.de",
    "opt_out":"1",
    "invalid_email":"0",
    "primary_address":"0"
  },
  {
    "email_address":"section71@example.it",
    "opt_out":"0",
    "invalid_email":"0",
    "primary_address":"1"
  }
],
"phone_mobile":"(246) 233-1382",
"phone_work":"(565) 696-6981",
"phone_other":"","
"phone_fax":"","
"email1":"section71@example.it",
"email2":"dev.vegan@example.de",
"invalid_email":false,
"email_opt_out":false,
"primary_address_street":"111 Silicon Valley Road",
"primary_address_street_2":"","
"primary_address_street_3":"","
"primary_address_city":"Denver",
"primary_address_state":"CA",
"primary_address_postalcode":"79900",
"primary_address_country":"USA",
"alt_address_street":"","
"alt_address_street_2":"","
```

```
"alt_address_street_3": "",
"alt_address_city": "",
"alt_address_state": "",
"alt_address_postalcode": "",
"alt_address_country": "",
"assistant": "",
"assistant_phone": "",
"picture": "",
"email_and_name1": "",
"lead_source": "Support Portal User Registration",
"account_name": "Smallville Resources Inc",
"account_id": "d5db6292-5c24-eb61-e202-512d09f0134e",
"opportunity_role_fields": "",
"opportunity_role_id": "",
"opportunity_role": "",
"reports_to_id": "",
"report_to_name": "",
"portal_name": "FlorenceHaddock169",
"portal_active": true,
"portal_password": "$1$NWFhTbK6$JF9BCGSqL\NCRbhueX5ia0",
"portal_password1": "",
"portal_app": "",
"preferred_language": "en_us",
"campaign_id": "",
"campaign_name": "",
"c_accept_status_fields": "",
"m_accept_status_fields": "",
"accept_status_id": "",
"accept_status_name": "",
"sync_contact": "",
"my_favorite": false,
"_acl": {
  "fields": {
```

```
}  
  ]  
  }  
  }  
}
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added
/<module>/:record/link/:link_name/filter
GET endpoint. |

Last Modified: 05/31/2017 05:25pm

/<module>/:record/link/:link_name/:remote_id DELETE

Overview

Deletes an existing relationship between two records.

Request Arguments

This endpoint does not accept any arguments.

Response Arguments

| Name | Type | Description |
|----------------|-------|---|
| record | Array | The record to disassociate from the related record. |
| related_record | Array | The record that was disassociated. |

Response

```
{
  "record": {
    "id": "da6a3741-2a81-ba7f-f249-512d0932e94e",
    "name": "Slender Broadband Inc - 1000 units",
    "date_entered": "2013-02-26T19:12:00+00:00",
    "date_modified": "2013-02-26T19:12:00+00:00",
    "modified_user_id": "1",
    "modified_by_name": "Administrator",
    "created_by": "1",
    "created_by_name": "Administrator",
    "description": "",
    "img": "",
    "last_activity_date": "2013-02-28T18:36:00+00:00",
    "deleted": false,
    "assigned_user_id": "seed_max_id",
    "assigned_user_name": "Max Jensen",
    "team_name": [
      {
        "id": "East",
        "name": "East",

```

```
        "name_2": "",
        "primary": false
    },
    {
        "id": "West",
        "name": "West",
        "name_2": "",
        "primary": true
    }
],
"opportunity_type": "",
"account_name": "Slender Broadband Inc",
"account_id": "181461c6-dc81-1115-1fe0-512d092e8f15",
"campaign_id": "",
"campaign_name": "",
"lead_source": "Campaign",
"amount": "25000",
"base_rate": "1",
"amount_usdollar": "25000",
"currency_id": "-99",
"currency_name": "",
"currency_symbol": "",
"date_closed": "2013-02-27",
"date_closed_timestamp": "1361992480",
"next_step": "",
"sales_stage": "Needs Analysis",
"sales_status": "New",
"probability": "90",
"best_case": "25000",
"worst_case": "25000",
"commit_stage": "include",
"my_favorite": false,
"_acl": {
```

```
    "fields":{
      }
    },
  "related_record":{
    "id":"e689173e-c953-1e14-c215-512d0927e7a2",
    "name":"Gus Dales",
    "date_entered":"2013-02-26T19:12:00+00:00",
    "date_modified":"2013-02-26T19:12:00+00:00",
    "modified_user_id":"1",
    "modified_by_name":"Administrator",
    "created_by":"1",
    "created_by_name":"Administrator",
    "description":"",
    "img":"",
    "deleted":false,
    "assigned_user_id":"seed_sally_id",
    "assigned_user_name":"Sally Bronsen",
    "team_name":[
      {
        "id":"West",
        "name":"West",
        "name_2":"",
        "primary":true
      }
    ],
    "salutation":"",
    "first_name":"Gus",
    "last_name":"Dales",
    "full_name":"Gus Dales",
    "title":"Director Operations",
    "linkedin":""
  }
```

```
"facebook": "",
"twitter": "",
"googleplus": "",
"department": "",
"do_not_call": false,
"phone_home": "(661) 120-2292",
"email": [
  {
    "email_address": "section.sugar.section@example.it",
    "opt_out": "1",
    "invalid_email": "0",
    "primary_address": "0"
  },
  {
    "email_address": "support.qa.kid@example.co.uk",
    "opt_out": "0",
    "invalid_email": "0",
    "primary_address": "1"
  }
],
"phone_mobile": "(294) 447-9707",
"phone_work": "(036) 840-3216",
"phone_other": "",
"phone_fax": "",
"email1": "support.qa.kid@example.co.uk",
"email2": "section.sugar.section@example.it",
"invalid_email": false,
"email_opt_out": false,
"primary_address_street": "48920 San Carlos Ave",
"primary_address_street_2": "",
"primary_address_street_3": "",
"primary_address_city": "Persistence",
"primary_address_state": "CA",
```

```
"primary_address_postalcode": "54556",
"primary_address_country": "USA",
"alt_address_street": "",
"alt_address_street_2": "",
"alt_address_street_3": "",
"alt_address_city": "",
"alt_address_state": "",
"alt_address_postalcode": "",
"alt_address_country": "",
"assistant": "",
"assistant_phone": "",
"picture": "",
"email_and_name1": "",
"lead_source": "Support Portal User Registration",
"account_name": "Arts & Crafts Inc",
"account_id": "d43243c6-9b8e-2973-ae2-512d09bc34b4",
"opportunity_role_fields": "",
"opportunity_role_id": "",
"opportunity_role": "",
"reports_to_id": "",
"report_to_name": "",
"portal_name": "GusDales145",
"portal_active": true,
"portal_password": "$1$JxYr6tmM$b.06.KF42jP46RadSwz0N0",
"portal_password1": "",
"portal_app": "",
"preferred_language": "en_us",
"campaign_id": "",
"campaign_name": "",
"c_accept_status_fields": "",
"m_accept_status_fields": "",
"accept_status_id": "",
"accept_status_name": ""
```

```
    "sync_contact": "",
    "my_favorite": false,
    "_acl": {
      "fields": {
        }
      }
    }
  }
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added
/<module>/:record/link/:link_name/:remote_id DELETE endpoint. |

Last Modified: 05/31/2017 05:25pm

/<module>/:record/link/:link_name/:remote_id GET

Overview

Retrieves a related record with relationship role information.

Request Arguments

This endpoint does not accept any arguments.

Response Arguments

| Name | Type | Description |
|----------------|---------------------|--|
| <record field> | <record field type> | Returns the fields for the newly created record. |

Response

```
{
  "id": "e689173e-c953-1e14-c215-512d0927e7a2",
  "name": "Gus Dales",
  "date_entered": "2013-02-26T19:12:00+00:00",
  "date_modified": "2013-02-26T19:12:00+00:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
  "description": "",
  "img": "",
  "deleted": false,
  "assigned_user_id": "seed_sally_id",
  "assigned_user_name": "Sally Bronsen",
  "team_name": [
    {
      "id": "West",
      "name": "West",
      "name_2": ""
    }
  ]
}
```

```
        "primary":true
    }
],
"salutation":"","
"first_name":"Gus",
"last_name":"Dales",
"full_name":"Gus Dales",
"title":"Director Operations",
"linkedin":"","
"facebook":"","
"twitter":"","
"googleplus":"","
"department":"","
"do_not_call":false,
"phone_home":"(661) 120-2292",
"email":[
    {
        "email_address":"section.sugar.section@example.it",
        "opt_out":"1",
        "invalid_email":"0",
        "primary_address":"0"
    },
    {
        "email_address":"support.qa.kid@example.co.uk",
        "opt_out":"0",
        "invalid_email":"0",
        "primary_address":"1"
    }
],
"phone_mobile":"(294) 447-9707",
"phone_work":"(036) 840-3216",
"phone_other":"","
"phone_fax":"","
```

```
"email1":"support.qa.kid@example.co.uk",
"email2":"section.sugar.section@example.it",
"invalid_email":false,
"email_opt_out":false,
"primary_address_street":"48920 San Carlos Ave",
"primary_address_street_2":"","
"primary_address_street_3":"","
"primary_address_city":"Persistence",
"primary_address_state":"CA",
"primary_address_postalcode":"54556",
"primary_address_country":"USA",
"alt_address_street":"","
"alt_address_street_2":"","
"alt_address_street_3":"","
"alt_address_city":"","
"alt_address_state":"","
"alt_address_postalcode":"","
"alt_address_country":"","
"assistant":"","
"assistant_phone":"","
"picture":"","
"email_and_name1":"","
"lead_source":"Support Portal User Registration",
"account_name":"Arts & Crafts Inc",
"account_id":"d43243c6-9b8e-2973-ae2-512d09bc34b4",
"opportunity_role_fields":"","
"opportunity_role_id":"","
"opportunity_role":"Technical Advisor",
"reports_to_id":"","
"report_to_name":"","
"portal_name":"GusDales145",
"portal_active":true,
"portal_password":"$1$JxYr6tmM$b.O6.KF42jP46RadSwz0N0",
```

```
"portal_password1": "",
"portal_app": "",
"preferred_language": "en_us",
"campaign_id": "",
"campaign_name": "",
"c_accept_status_fields": "",
"m_accept_status_fields": "",
"accept_status_id": "",
"accept_status_name": "",
"sync_contact": "",
"my_favorite": false,
"_acl": {
  "fields": {
    }
  }
}
```

Change Log

| Version | Change |
|---------|---|
| v10 | Added
/<module>/:record/link/:link_name/:remote_id GET endpoint. |

Last Modified: 05/31/2017 05:25pm

/<module>/:record/link/:link_name/:remote_id POST

Overview

Creates a relationship to a pre-existing record.

Query String Parameters

This endpoint does not accept any arguments.

Response Arguments

| Name | Type | Description |
|----------------|-------|--|
| record | Array | The record linked to the related record. |
| related_record | Array | The record that was associated. |

Response

```
{
  "record": {
    "id": "da6a3741-2a81-ba7f-f249-512d0932e94e",
    "name": "Slender Broadband Inc - 1000 units",
    "date_entered": "2013-02-26T19:12:00+00:00",
    "date_modified": "2013-02-26T19:12:00+00:00",
    "modified_user_id": "1",
    "modified_by_name": "Administrator",
    "created_by": "1",
```

```
"created_by_name": "Administrator",
"description": "",
"img": "",
"last_activity_date": "2013-02-28T18:21:00+00:00",
"deleted": false,
"assigned_user_id": "seed_max_id",
"assigned_user_name": "Max Jensen",
"team_name": [
  {
    "id": "East",
    "name": "East",
    "name_2": "",
    "primary": false
  },
  {
    "id": "West",
    "name": "West",
    "name_2": "",
    "primary": true
  }
],
"opportunity_type": "",
"account_name": "Slender Broadband Inc",
"account_id": "181461c6-dc81-1115-1fe0-512d092e8f15",
"campaign_id": "",
"campaign_name": "",
"lead_source": "Campaign",
"amount": "25000",
"base_rate": "1",
"amount_usdollar": "25000",
"currency_id": "-99",
"currency_name": "",
"currency_symbol": "",
```

```
"date_closed": "2013-02-27",
"date_closed_timestamp": "1361992480",
"next_step": "",
"sales_stage": "Needs Analysis",
"sales_status": "New",
"probability": "90",
"best_case": "25000",
"worst_case": "25000",
"commit_stage": "include",
"my_favorite": false,
"_acl": {
  "fields": {
    }
  }
},
"related_record": {
  "id": "e689173e-c953-1e14-c215-512d0927e7a2",
  "name": "Gus Dales",
  "date_entered": "2013-02-26T19:12:00+00:00",
  "date_modified": "2013-02-26T19:12:00+00:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
  "description": "",
  "img": "",
  "deleted": false,
  "assigned_user_id": "seed_sally_id",
  "assigned_user_name": "Sally Bronsen",
  "team_name": [
    {
      "id": "West",
```

```
        "name": "West",
        "name_2": "",
        "primary": true
    }
],
"salutation": "",
"first_name": "Gus",
"last_name": "Dales",
"full_name": "Gus Dales",
"title": "Director Operations",
"linkedin": "",
"facebook": "",
"twitter": "",
"googleplus": "",
"department": "",
"do_not_call": false,
"phone_home": "(661) 120-2292",
"email": [
    {
        "email_address": "section.sugar.section@example.it",
        "opt_out": "1",
        "invalid_email": "0",
        "primary_address": "0"
    },
    {
        "email_address": "support.qa.kid@example.co.uk",
        "opt_out": "0",
        "invalid_email": "0",
        "primary_address": "1"
    }
],
"phone_mobile": "(294) 447-9707",
"phone_work": "(036) 840-3216",
```

```
"phone_other": "",
"phone_fax": "",
"email1": "support.qa.kid@example.co.uk",
"email2": "section.sugar.section@example.it",
"invalid_email": false,
"email_opt_out": false,
"primary_address_street": "48920 San Carlos Ave",
"primary_address_street_2": "",
"primary_address_street_3": "",
"primary_address_city": "Persistence",
"primary_address_state": "CA",
"primary_address_postalcode": "54556",
"primary_address_country": "USA",
"alt_address_street": "",
"alt_address_street_2": "",
"alt_address_street_3": "",
"alt_address_city": "",
"alt_address_state": "",
"alt_address_postalcode": "",
"alt_address_country": "",
"assistant": "",
"assistant_phone": "",
"picture": "",
"email_and_name1": "",
"lead_source": "Support Portal User Registration",
"account_name": "Arts & Crafts Inc",
"account_id": "d43243c6-9b8e-2973-ae2-512d09bc34b4",
"opportunity_role_fields": "",
"opportunity_role_id": "",
"opportunity_role": "Technical Advisor",
"reports_to_id": "",
"report_to_name": "",
"portal_name": "GusDales145",
```

```

"portal_active":true,
"portal_password":"$1$JxYr6tmM$b.06.KF42jP46RadSwz0N0",
"portal_password1":"",
"portal_app":"",
"preferred_language":"en_us",
"campaign_id":"",
"campaign_name":"",
"c_accept_status_fields":"",
"m_accept_status_fields":"",
"accept_status_id":"",
"accept_status_name":"",
"sync_contact":"",
"my_favorite":false,
"_acl":{
  "fields":{
    }
  }
}

```

Change Log

| Version | Change |
|---------|--|
| v10 | Added
/<module>/:record/link/:link_name/:remote_id POST endpoint. |

Last Modified: 05/31/2017 05:25pm

`/<module>/:record/link/:link_name/:remote_id PUT`

Overview

Updates relationship specific information on an existing relationship.

Request Arguments

| Name | Type | Description | Required |
|---|--|--|----------|
| <code><relationship field></code> | <code><relationship field type></code> | The name value list of relationship fields to populate. An example is the <code>contact_role</code> field on Opportunities and Contacts. | True |

Request

```
{  
  "contact_role": "Primary Decision Maker"  
}
```

Response Arguments

| Name | Type | Description |
|----------------|-------|--|
| record | Array | The record linked to the related record. |
| related_record | Array | The record that was associated. |

Response

```
{
  "record": {
    "id": "da6a3741-2a81-ba7f-f249-512d0932e94e",
    "name": "Slender Broadband Inc - 1000 units",
    "date_entered": "2013-02-26T19:12:00+00:00",
    "date_modified": "2013-02-26T19:12:00+00:00",
    "modified_user_id": "1",
    "modified_by_name": "Administrator",
    "created_by": "1",
    "created_by_name": "Administrator",
    "description": "",
    "img": "",
    "deleted": false,
    "assigned_user_id": "seed_max_id",
    "assigned_user_name": "Max Jensen",
    "team_name": [
      {
        "id": "East",
        "name": "East",
        "name_2": "",
        "primary": false
      }
    ]
  }
}
```

```
    },
    {
      "id": "West",
      "name": "West",
      "name_2": "",
      "primary": true
    }
  ],
  "opportunity_type": "",
  "account_name": "Slender Broadband Inc",
  "account_id": "181461c6-dc81-1115-1fe0-512d092e8f15",
  "campaign_id": "",
  "campaign_name": "",
  "lead_source": "Campaign",
  "amount": "25000",
  "base_rate": "1",
  "amount_usdollar": "25000",
  "currency_id": "-99",
  "currency_name": "",
  "currency_symbol": "",
  "date_closed": "2013-02-27",
  "date_closed_timestamp": "1361992480",
  "next_step": "",
  "sales_stage": "Needs Analysis",
  "sales_status": "New",
  "probability": "90",
  "best_case": "25000",
  "worst_case": "25000",
  "commit_stage": "include",
  "my_favorite": false,
  "_acl": {
    "fields": {
```

```
    }
  }
},
"related_record":{
  "id":"e1c495cb-af17-1b37-dd66-512f934fe155",
  "name":"Bill Edwards",
  "date_entered":"2013-02-28T17:25:00+00:00",
  "date_modified":"2013-02-28T17:25:00+00:00",
  "modified_user_id":"1",
  "modified_by_name":"Administrator",
  "created_by":"1",
  "created_by_name":"Administrator",
  "description":"",
  "img":"",
  "deleted":false,
  "assigned_user_id":"",
  "assigned_user_name":"",
  "team_name":[
    {
      "id":1,
      "name":"Global",
      "name_2":"",
      "primary":true
    }
  ],
  "salutation":"",
  "first_name":"Bill",
  "last_name":"Edwards",
  "full_name":"Bill Edwards",
  "title":"",
  "linkedin":"",
  "facebook":"",
  "twitter":""
}
```

```
"googleplus": "",
"department": "",
"do_not_call": false,
"phone_home": "",
"email": [
],
"phone_mobile": "",
"phone_work": "",
"phone_other": "",
"phone_fax": "",
"email1": "",
"email2": "",
"invalid_email": "",
"email_opt_out": "",
"primary_address_street": "",
"primary_address_street_2": "",
"primary_address_street_3": "",
"primary_address_city": "",
"primary_address_state": "",
"primary_address_postalcode": "",
"primary_address_country": "",
"alt_address_street": "",
"alt_address_street_2": "",
"alt_address_street_3": "",
"alt_address_city": "",
"alt_address_state": "",
"alt_address_postalcode": "",
"alt_address_country": "",
"assistant": "",
"assistant_phone": "",
"picture": "",
"email_and_name1": "",
```

```
"lead_source":"","
"account_id":"","
"opportunity_role_fields":"","
"opportunity_role_id":"","
"opportunity_role":"Primary Decision Maker",
"reports_to_id":"","
"report_to_name":"","
"portal_name":"","
"portal_active":false,
"portal_password":"","
"portal_password1":"","
"portal_app":"","
"preferred_language":"en_us",
"campaign_id":"","
"campaign_name":"","
"c_accept_status_fields":"","
"m_accept_status_fields":"","
"accept_status_id":"","
"accept_status_name":"","
"sync_contact":"","
"my_favorite":false,
"_acl":{
  "fields":{
    }
  }
}
```

Change Log

| Version | Change |
|---------|---|
| v10 | Added
/<module>/:record/link/:link_name/:remote_id PUT endpoint. |

Last Modified: 05/31/2017 05:25pm

/<module>/:record/moveafter/:target PUT

Overview

Move existing node after target node.

Request Arguments

| Name | Type | Description | Required |
|--------|--------|---|----------|
| module | String | The name of sugar module that contains a nested set data and implements the NestedSetInterface. | True |
| target | String | The ID of record that will be used as target to move node after | True |

Response Arguments

This endpoint does not return any response arguments.

Response

This endpoint return a record GUID that was moved
"1b7b868d-7357-2e29-7513-54169bdc444d"

Change Log

| Version | Change |
|---------|---|
| v10 | Added /<module>/moveafter/:target PUT endpoint. |

Last Modified: 05/31/2017 05:25pm

/<module>/:record/movebefore/:target PUT

Overview

Move existing node before target node.

Request Arguments

| Name | Type | Description | Required |
|------|------|-------------|----------|
| | | | |

| Name | Type | Description | Required |
|--------|--------|---|----------|
| module | String | The name of sugar module that contains a nested set data and implements the NestedSetInterface. | True |
| target | String | The ID of record that will be used as target to move node before | True |

Response Arguments

This endpoint does not return any response arguments.

Response

This endpoint return a record GUID that was moved
 "1b7b868d-7357-2e29-7513-54169bdc444d"

Change Log

| Version | Change |
|---------|--|
| v10 | Added /<module>/movebefore/:target PUT endpoint. |

Last Modified: 05/31/2017 05:25pm

/<module>/:record/movefirst/:target PUT

Overview

Move existing node as first child of target node.

Request Arguments

| Name | Type | Description | Required |
|--------|--------|---|----------|
| module | String | The name of sugar module that contains a nested set data and implements the NestedSetInterface. | True |
| target | String | The ID of record that will be used as target to move node as first child | True |

Response Arguments

This endpoint does not return any response arguments.

Response

This endpoint return a record GUID that was moved
"1b7b868d-7357-2e29-7513-54169bdc444d"

Change Log

| Version | Change |
|---------|---|
| v10 | Added /<module>/movefirst/:target PUT endpoint. |

Last Modified: 05/31/2017 05:25pm

/<module>/:record/movelast/:target PUT

Overview

Move existing node as last child of target node.

Request Arguments

| Name | Type | Description | Required |
|--------|--------|---|----------|
| module | String | The name of sugar module that contains a nested set data and implements the | True |

| Name | Type | Description | Required |
|--------|--------|---|---------------------|
| | | | NestedSetInterface. |
| target | String | The ID of record that will be used as target to move node as last child | True |

Response Arguments

This endpoint does not return any response arguments.

Response

This endpoint return a record GUID that was moved
 "1b7b868d-7357-2e29-7513-54169bdc444d"

Change Log

| Version | Change |
|---------|--|
| v10 | Added /<module>/movelast/:target PUT endpoint. |

Last Modified: 05/31/2017 05:25pm

/<module>/:record/next GET

Overview

Retrieves next sibling of selected record.

Request Arguments

| Name | Type | Description | Required |
|---------|--------|---|----------|
| module | String | The name of sugar module that contains a nested set data and implements the NestedSetInterface. | True |
| :record | String | The ID of record | True |

Response Arguments

This endpoint does not return any response arguments.

Response

```
{
  id: "e1ae8646-ac90-104a-59d6-5412cf5009b2"
  name: "SugarCategoryExample"
  date_entered: "2014-09-12 10:47:46"
  date_modified: "2014-09-12 10:47:46"
  modified_user_id: "9c9ad14e-1789-3340-f88d-5412cf551d3b"
  created_by: "9c9ad14e-1789-3340-f88d-5412cf551d3b"
```

```
description: null
deleted: "0"
source_id: null
source_type: null
source_meta: null
root: "e1ae8646-ac90-104a-59d6-5412cf5009b2"
lft: "6"
rgt: "11"
level: "1"
}
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added /<module>/:record/next GET endpoint. |

Last Modified: 05/31/2017 05:28pm

/<module>/:record/parent GET

Overview

Retrieves parent node for selected record.

Request Arguments

| Name | Type | Description | Required |
|---------|--------|---|----------|
| module | String | The name of sugar module that contains a nested set data and implements the NestedSetInterface. | True |
| :record | String | The ID of record | True |

Response Arguments

This endpoint does not return any response arguments.

Response

```
{
  id: "e1ae8646-ac90-104a-59d6-5412cf5009b2"
  name: "SugarCategoryExample"
  date_entered: "2014-09-12 10:47:46"
  date_modified: "2014-09-12 10:47:46"
  modified_user_id: "9c9ad14e-1789-3340-f88d-5412cf551d3b"
  created_by: "9c9ad14e-1789-3340-f88d-5412cf551d3b"
  description: null
  deleted: "0"
  source_id: null
  source_type: null
  source_meta: null
  root: "e1ae8646-ac90-104a-59d6-5412cf5009b2"
  lft: "6"
```

```
    rgt: "11"  
    level: "1"  
}
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added /<module>/:record/parent GET endpoint. |

Last Modified: 05/31/2017 05:28pm

/<module>/:record/path GET

Overview

Retrieves all parents of selected record.

Request Arguments

| Name | Type | Description | Required |
|--------|--------|---|----------|
| module | String | The name of sugar module that contains a nested set data and implements the | True |

| Name | Type | Description | Required |
|---------|--------|------------------|---------------------|
| | | | NestedSetInterface. |
| :record | String | The ID of record | True |

Response Arguments

This endpoint does not return any response arguments.

Response

```
[{
  "id": "045c03b9-327b-11e4-818b-5404a67f3363",
  "name": "Audit",
  "date_entered": null,
  "date_modified": null,
  "modified_user_id": null,
  "created_by": null,
  "description": null,
  "deleted": "0",
  "source_id": null,
  "source_type": null,
  "source_meta": null,
  "root": "935d3e07-327a-11e4-818b-5404a67f3363",
  "lft": "10",
  "rgt": "19",
  "level": "1"
}, {
  "id": "045c1de6-327b-11e4-818b-5404a67f3363",
  "name": "Financial",
  "date_entered": null,
```

```
"date_modified": null,
"modified_user_id": null,
"created_by": null,
"description": null,
"deleted": "0",
"source_id": null,
"source_type": null,
"source_meta": null,
"root": "935d3e07-327a-11e4-818b-5404a67f3363",
"lft": "11",
"rgt": "14",
"level": "2"
}]
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added /<module>/:record/path GET endpoint. |

Last Modified: 05/31/2017 05:28pm

/<module>/:record/prev GET

Overview

Retrieves previous sibling of selected record.

Request Arguments

| Name | Type | Description | Required |
|---------|--------|---|----------|
| module | String | The name of sugar module that contains a nested set data and implements the NestedSetInterface. | True |
| :record | String | The ID of record | True |

Response Arguments

This endpoint does not return any response arguments.

Response

```
{
  id: "e1ae8646-ac90-104a-59d6-5412cf5009b2"
  name: "SugarCategoryExample"
  date_entered: "2014-09-12 10:47:46"
  date_modified: "2014-09-12 10:47:46"
  modified_user_id: "9c9ad14e-1789-3340-f88d-5412cf551d3b"
  created_by: "9c9ad14e-1789-3340-f88d-5412cf551d3b"
  description: null
  deleted: "0"
  source_id: null
  source_type: null
  source_meta: null
}
```

```
root: "e1ae8646-ac90-104a-59d6-5412cf5009b2"  
lft: "6"  
rgt: "11"  
level: "1"  
}
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added /<module>/:record/prev GET endpoint. |

Last Modified: 05/31/2017 05:28pm

/<module>/:record/subscribe POST

Summary:

This endpoint creates a subscription record in the Subscriptions table, from a specified record and module. It allows the user to be subscribed to activity stream messages for the record being subscribed to, or "followed".

Query Parameters:

This endpoint does not accept any parameters.

Input Example:

This endpoint does not accept any input.

Output Example:

If successful, the endpoint responds with the GUID of the subscription record.
"96ad733c-df51-dd9d-1888-514112ef0595"

Last Modified: 11/30/2016 09:20pm

/<module>/:record/unfavorite PUT

Overview

Removes a record of a specified type as a favorite for the current user.

Request Arguments

This endpoint does not accept any request arguments.

Response Arguments

| Name | Type | Description |
|----------------|---------------------|---|
| <record field> | <record field type> | Returns the fields for the selected record. |

Response

```
{
  "id": "bdd59d85-687b-1739-b00a-512d09f6db9e",
  "name": "Insight Marketing Inc",
  "date_entered": "2013-02-26T19:12:00+00:00",
  "date_modified": "2013-02-26T19:12:00+00:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
  "description": "",
  "img": "",
  "last_activity_date": "2013-02-26T19:12:00+00:00",
  "deleted": false,
  "assigned_user_id": "seed_max_id",
  "assigned_user_name": "Max Jensen",
  "team_name": [
    {
      "id": "East",
      "name": "East",
      "name_2": "",
      "primary": false
    },
    {
      "id": 1,
      "name": "Global",
      "name_2": "",
      "primary": false
    },
    {
      "id": "West",
```

```
        "name": "West",
        "name_2": "",
        "primary": true
    }
],
"linkedin": "",
"facebook": "",
"twitter": "",
"googleplus": "",
"account_type": "Customer",
"industry": "Electronics",
"annual_revenue": "",
"phone_fax": "",
"billing_address_street": "345 Sugar Blvd.",
"billing_address_street_2": "",
"billing_address_street_3": "",
"billing_address_street_4": "",
"billing_address_city": "San Mateo",
"billing_address_state": "CA",
"billing_address_postalcode": "56019",
"billing_address_country": "USA",
"rating": "",
"phone_office": "(927) 136-9572",
"phone_alternate": "",
"website": "www.sectionvegan.de",
"ownership": "",
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "345 Sugar Blvd.",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "San Mateo",
```

```
"shipping_address_state": "CA",
"shipping_address_postalcode": "56019",
"shipping_address_country": "USA",
"email1": "kid.support.vegan@example.info",
"parent_id": "",
"sic_code": "",
"parent_name": "",
"email_opt_out": false,
"invalid_email": false,
"email": [
  {
    "email_address": "kid.support.vegan@example.info",
    "opt_out": "0",
    "invalid_email": "0",
    "primary_address": "1"
  },
  {
    "email_address": "phone.kid@example.cn",
    "opt_out": "0",
    "invalid_email": "0",
    "primary_address": "0"
  }
],
"campaign_id": "",
"campaign_name": "",
"my_favorite": false,
"_acl": {
  "fields": {
  }
}
}
```

Change Log

| Version | Change |
|---------|---|
| v10 | Added /<module>/:record/favorite DELETE endpoint. |

Last Modified: 05/31/2017 05:26pm

/<module>/:record/unsubscribe DELETE

Summary:

This endpoint deletes a subscription record in the Subscriptions table, from a specified record and module. It allows the user to be unsubscribe from activity stream messages for the record being subscribed to, or "followed".

Query Parameters:

This endpoint does not accept any parameters.

Input Example:

This endpoint does not accept any input.

Output Example:

If successful, the endpoint responds with a bool of true.

Last Modified: 11/30/2016 09:20pm

/<module>/:record/vcard GET

Overview

Downloads a vCard.

Request Arguments

This endpoint does not accept any arguments.

Response Arguments

| Name | Type | Description |
|---------------------|---------|------------------------|
| <vcard information> | String; | Record in vcard format |

Response

```
BEGIN:VCARD
N;CHARSET=utf-8:Leone;Rosemarie;;
FN;CHARSET=utf-8: Rosemarie Leone
BDAY:
TEL;WORK;FAX:
```

```
TEL;HOME;CHARSET=utf-8:(692) 586-8287
TEL;CELL;CHARSET=utf-8:(117) 577-0969
TEL;WORK;CHARSET=utf-8:(844) 325-7679
EMAIL;INTERNET;CHARSET=utf-8:support.im@example.it
ADR;WORK;CHARSET=utf-8;;;777 West Filmore Ln;San Mateo;CA;74984;USA
ORG;CHARSET=utf-8:Q.R.&E. Corp;
TITLE;CHARSET=utf-8:Director Sales
END:VCARD
}
```

Change Log

| Version | Change |
|---------|---|
| v10 | Added /<module>/:record/vcard GET endpoint. |

Last Modified: 05/31/2017 05:28pm

/<module>/:root/tree GET

Overview

Retrieves full tree for selected root record.

Request Arguments

| Name | Type | Description | Required |
|---------|--------|---|----------|
| module | String | The name of sugar module that contains a nested set data and implements the NestedSetInterface. | True |
| :record | String | The ID of record | True |

Response Arguments

This endpoint does not return any response arguments.

Response

```
{
  "next_offset": -1,
  "records": [{
    "id": "ad4ddf76-327a-11e4-818b-5404a67f3363",
    "name": "Documents",
    "date_entered": null,
    "date_modified": null,
    "modified_user_id": null,
    "created_by": null,
    "description": null,
    "deleted": "0",
    "source_id": null,
    "source_type": null,
    "source_meta": null,
    "root": "935d3e07-327a-11e4-818b-5404a67f3363",
```

```
"lft": "2",
"rgt": "9",
"level": "1",
"children": {
  "next_offset": -1,
  "records": [{
    "id": "ad4e03f1-327a-11e4-818b-5404a67f3363",
    "name": "Engeneering",
    "date_entered": null,
    "date_modified": null,
    "modified_user_id": null,
    "created_by": null,
    "description": null,
    "deleted": "0",
    "source_id": null,
    "source_type": null,
    "source_meta": null,
    "root": "935d3e07-327a-11e4-818b-5404a67f3363",
    "lft": "3",
    "rgt": "4",
    "level": "2",
    "children": {
      "next_offset": -1,
      "records": []
    }
  }], {
  "id": "f482dd1b-327a-11e4-818b-5404a67f3363",
  "name": "Testing",
  "date_entered": null,
  "date_modified": null,
  "modified_user_id": null,
  "created_by": null,
  "description": null,
```

```
"deleted": "0",
"source_id": null,
"source_type": null,
"source_meta": null,
"root": "935d3e07-327a-11e4-818b-5404a67f3363",
"lft": "5",
"rgt": "6",
"level": "2",
"children": {
  "next_offset": -1,
  "records": []
}
}, {
  "id": "f482fb7a-327a-11e4-818b-5404a67f3363",
  "name": "Management",
  "date_entered": null,
  "date_modified": null,
  "modified_user_id": null,
  "created_by": null,
  "description": null,
  "deleted": "0",
  "source_id": null,
  "source_type": null,
  "source_meta": null,
  "root": "935d3e07-327a-11e4-818b-5404a67f3363",
  "lft": "7",
  "rgt": "8",
  "level": "2",
  "children": {
    "next_offset": -1,
    "records": []
  }
}
}]
```

```
    }  
  }, {  
    "id": "045c03b9-327b-11e4-818b-5404a67f3363",  
    "name": "Audit",  
    "date_entered": null,  
    "date_modified": null,  
    "modified_user_id": null,  
    "created_by": null,  
    "description": null,  
    "deleted": "0",  
    "source_id": null,  
    "source_type": null,  
    "source_meta": null,  
    "root": "935d3e07-327a-11e4-818b-5404a67f3363",  
    "lft": "10",  
    "rgt": "19",  
    "level": "1",  
    "children": {  
      "next_offset": -1,  
      "records": [{  
        "id": "045c1de6-327b-11e4-818b-5404a67f3363",  
        "name": "Financial",  
        "date_entered": null,  
        "date_modified": null,  
        "modified_user_id": null,  
        "created_by": null,  
        "description": null,  
        "deleted": "0",  
        "source_id": null,  
        "source_type": null,  
        "source_meta": null,  
        "root": "935d3e07-327a-11e4-818b-5404a67f3363",  
        "lft": "11",
```

```
"rgt": "14",
"level": "2",
"children": {
  "next_offset": -1,
  "records": [{
    "id": "0f658847-327b-11e4-818b-5404a67f3363",
    "name": "Invoices",
    "date_entered": null,
    "date_modified": null,
    "modified_user_id": null,
    "created_by": null,
    "description": null,
    "deleted": "0",
    "source_id": null,
    "source_type": null,
    "source_meta": null,
    "root": "935d3e07-327a-11e4-818b-5404a67f3363",
    "lft": "12",
    "rgt": "13",
    "level": "3",
    "children": {
      "next_offset": -1,
      "records": []
    }
  }
]}
}, {
  "id": "0f65a6c7-327b-11e4-818b-5404a67f3363",
  "name": "Agreements",
  "date_entered": null,
  "date_modified": null,
  "modified_user_id": null,
  "created_by": null,
```

```
"description": null,
"deleted": "0",
"source_id": null,
"source_type": null,
"source_meta": null,
"root": "935d3e07-327a-11e4-818b-5404a67f3363",
"lft": "15",
"rgt": "16",
"level": "2",
"children": {
  "next_offset": -1,
  "records": []
}
}, {
  "id": "14d30bf3-327b-11e4-818b-5404a67f3363",
  "name": "Clients",
  "date_entered": null,
  "date_modified": null,
  "modified_user_id": null,
  "created_by": null,
  "description": null,
  "deleted": "0",
  "source_id": null,
  "source_type": null,
  "source_meta": null,
  "root": "935d3e07-327a-11e4-818b-5404a67f3363",
  "lft": "17",
  "rgt": "18",
  "level": "2",
  "children": {
    "next_offset": -1,
    "records": []
  }
}
```

```
    }  
  }  
}]  
}
```

Change Log

| Version | Change |
|---------|--|
| v10 | Added /<module>/:record/tree GET endpoint. |

Last Modified: 05/31/2017 05:28pm

/<module>/temp/file/:field POST

Overview

Saves an image to a temporary folder.

Request Arguments

| Name | Type | Description | Required |
|--------|--------|---|----------|
| format | String | The data format. Currently accepts 'sugar-html-json'. | True |
| | | | |

| Name | Type | Description | Required |
|-----------------|---------|---|--|
| delete_if_fails | Boolean | Indicates whether the API is to mark related record deleted if the file upload fails. | False |
| oauth_token | String | The oauth_token value. | False - Required if only if delete_if_fails is true. |
| <image field> | String | The field and file to populate.
Example:
{ "" : "@\\path\\to\\ExampleImage.png" } | True |

Request

```
{
  "format": "sugar-html-json",
  "delete_if_fails": true,
  "oauth_token": "8d240d9d-04ea-571b-35ea-513037ed5857",
  "<image field>": "@\\path\\to\\ExampleImage.png"
}
```

Response Arguments

| Name | Type | Description |
|------|--------|---------------------------|
| guid | String | The temp ID of the image. |

Response

```
{
  "picture": {
    "guid": "50a8760d-966f-9d7e-f45c-513037fac8fc"
  }
}
```

Change Log

| Version | Change |
|---------|---|
| v10 | Added /<module>/temp/file/:field POST endpoint. |

Last Modified: 05/31/2017 05:25pm

/<module>/temp/file/:field/:temp_id GET

Overview

Reads a temporary image and deletes it.

Request Arguments

This endpoint does not accept any arguments.

Response Arguments

| Name | Type | Description |
|--------------|---------|---|
| <image file> | String; | Returns the image content with the content headers. |

Response

```
result: HTTP/1.1 200 OK
Date: Fri, 01 Mar 2013 05:36:24 GMT
Server: Apache/2.2.22 (Unix) DAV/2 PHP/5.3.17
X-Powered-By: PHP/5.3.17
Expires: Sun, 31 Mar 2013 05:36:24 GMT
Cache-Control: max-age=1, post-check=0, pre-check=0
Pragma: public
X-Content-Type-Options: nosniff
Content-Length: 2072
ETag: d41d8cd98f00b204e9800998ecf8427e
Connection: close
Content-Type: image/png
```

```
%PNG
```

IHDRIIqsÜßIDATxæíæ}hUçç? >ëiðl<Ñ ðë, ™šS'âD,ÉÈ4+^Öv Mö['vl wÉÝ
-mEMŠÔ- jBWDD\$Ý,^dR L\$^^/!^si9çùí ç^st7×crßÚž\î%Ûæû çy y~ ~~~~êA*Ý€
x"÷r! p€p
H ýwEÚ- b'Brj 6` h ýŽç i æ J
%IMX)ÝÈ>?
\$Co÷ àðÝ}ž"WXY\$ÝwK cSÓ Â&éA €ØdÃ l bI%eà\$€àÀF X4fb à
p> "À%À"RVÑ%...RS ÆŠÉZ%-š L Ì è0ÁOWÑ\$...RÓLÂÉY† æ'š à*p+è`€" k^'B ù(v é
•75Q™PÃ > I~_ æ\$)+'%MM'Ð ">5Q1ØuÖàiÀ6]p\dE-JM=v é `Sã6]CÀQ` ÑÓö@I Ô
` äRÓ@u &* HÆŽ]fDLWAI# \$ù
ßXÁ ÷+`Ö- WÍ/Îúæ,IX P ÝX JR<ÛØÊ êÍOâ{IÔ|û IrÝ æW M5|Ìh 'ôX J#BB] Ú 5
ÚÁ ì÷éç-é ß_-€YÄÖ OáýÖ }tÝ)ÛŠòt ù'€ÁÝ ÓãÛS™áiè '6EkA V- "vscŽúíîq}vg3
ßE4 AdÈ+ ÖÀ€ j UNgÐ Ê™7n'-u=è5ÈzÄnP "] > Ài' Û@&Ýfê: >l K.../ ù\$„
e'Ý <èŽG"Ï 0 wIÝqý ngöö ÒM é'vUæL•K~ šÍ~s\ üÜ„îØB+~ð5 n UTäž-5a'
Ñ|x.ŠÉd!j ÒIhWøsdá,S -\p" ~éfê OÆð...P4iö3J,ÈÖK 'ñü~mpä^yù „ÔÔLÀKÿP2™
. >éÆÿ &O_g~ÏtžxTDv8 Û.~ aÑqúzíd ûÛT Ñfî@Ø V încñ 9lrt
!ú@Í s•ÏnlÁóöJàux+ÁÎdùÄD (kq€ r"= îépZ / úÖ/n 7ùÆ
Đž@d |^érö 1Q=EðèÊæasà7u'LlÄÈð N Í.oæ!'GQ† „eÛŃn„ }<
zðÆgðP oMN7ØÀÓÛ™ > jÖ äfÖ>Ày,à^&j†ôç/^ç 5mxPób -dh
>(Šæ%[=gÓŌ*J<f>sÖIžÝ ô#}5 ?•zýqß ûHo lš'án „} 1€ fîŌŌ'Î™ jas65.
2ó %éKÉ *\$Y*Đí nA{añ Áð Nû?z~ðN ÂV]F I -tæ 8 ÉäÛúÉý7ùTk™ð>ñð[
jÛ^"1aJ.)Û... é,-GT xÁ 9 {iÝ% 2øŌ[?7ùf /R+ž?Í"-Sæx?k"'èÆ÷ HO ŌúĐÁJÝô Td
'z NÝ ð /ÚCäĐWvî4]Ý]IRwøøß_>î €bÂTä Ž{ *IiIÀpc6 oÛ@n ^1?ŌD'^TLT×÷AM
Ûqæ:``" • š\$,`v<P5T•\$ jTaf 'XRbI^%E - XRbI^%E - XRbI^%E - XRbI^%E -
XRbI(\$éÃ&pŌpásàîÏÿlM]>
IÁžsp0ð T,/...\$]b<[>È\^ @?óJ~7÷ðoÀXÈ™™"7 ÒLctöoBp" P IYQ>ðG1ÉPìÁ
â%WTø&ðMlçÝ6Að€S%YQÆu;. ÆÏúuc "*Q ð LcÛÿ ji„l%W!99" Ý~"
Ûq[Py[2Û<-È™kÉh - öaKQ/&š~0}>)w nçvöû÷÷ `+ÁvfÂŽ -j~X99æ|^+)×18
=Uÿ;f
'7bÈ-#]['ûÈäç*&LQN ...Æ KÀ!ì ØŽ ùÏ)×Íz "KÍö- À

Change Log

| Version | Change |
|---------|---|
| v10 | Added /<module>/enum/:field GET endpoint. |

Last Modified: 05/31/2017 05:25pm

/mostactiveusers GET

Overview

Fetches the most active users for meetings, calls, inbound emails, and outbound emails.

Request Arguments

| Name | Type | Description | Required |
|------|---------|--|----------|
| days | Integer | Returns most active users for the last specified quantity of days. Defaults to 30 days if not specified. | False |

Request

`http://{site_url}/rest/v10/mostactiveusers?days=30`

Note: GET endpoint parameters are passed in the form of a query string.

Response Arguments

Response

```
{
  "meetings":{
    "user_id":"seed_sarah_id",
    "count":"20",
    "first_name":"Sarah",
    "last_name":"Smith"
  },
  "calls":{
    "user_id":"seed_will_id",
    "count":"7",
    "first_name":"Will",
    "last_name":"Westin"
  },
  "inbound_emails":{
    "user_id":"seed_sarah_id",
    "count":"20",
    "first_name":"Sarah",
    "last_name":"Smith"
  },
  "outbound_emails":{
    "user_id":"seed_max_id",
    "count":"17",
    "first_name":"Max",
```

```
    "last_name": "Jensen"  
  }  
}
```

Change Log

| Version | Change |
|---------|--------------------------------------|
| v10 | Added /mostactiveusers GET endpoint. |

Last Modified: 05/31/2017 05:31pm

/oauth2/bwc/login POST

ATTENTION: FOR INTERNAL USAGE ONLY

This endpoint is subject to change.

Overview

Retrieves a cookie from the OAuth token.

Last Modified: 05/31/2017 05:27pm

/oauth2/logout POST

Overview

Expires the token portion of the OAuth 2.0 specification.

Request Arguments

This endpoint does not accept any arguments.

Response Arguments

| Name | Type | Description |
|---------|---------|----------------------------|
| success | Boolean | The success of the logout. |

Response

```
{  
  "success": true  
}
```

Change Log

| Version | Change |
|---------|-------------------------------------|
| v10 | Added /oauth2/logout POST endpoint. |

Last Modified: 05/31/2017 05:29pm

/oauth2/sudo/:user_name POST

Overview

Get an access token as another user. The current user must be an admin in order to access this endpoint. This method is useful for integrations in order to be able to access the system with the same permission restrictions as a specified user. The calling user does not lose their existing token, this one is granted in addition.

Request Arguments

| Name | Type | Description | Required |
|-----------|--------|--|----------|
| platform | String | Which platform on the session, defaults to "base" | False |
| client_id | String | The client id for the session, defaults to "sugar" | False |

Request

```
{  
  "client_id": "sugar",  
  "platform": "base"  
}
```

Response Arguments

| Name | Type | Description |
|--------------|---------|---|
| access_token | String | The access token needed to authenticate for other methods. |
| expires_in | Integer | The length of time until the access_token expires. |
| token_type | String | Should always be bearer. |
| scope | String | There is no scope implementation in the current release of Sugar. |

Response

```
{
  "access_token": "c19fff9b-b767-233e-ebb4-512e369d3e39",
  "expires_in": 3600,
  "token_type": "bearer",
  "scope": null
}
```

Change Log

| Version | Change |
|---------|---|
| v10 | Added /oauth2/token/sudo/:user POST endpoint. |

Last Modified: 05/31/2017 05:27pm

/oauth2/token POST

Overview

Retrieves the token portion of the OAuth 2.0 specification.

Request Arguments

| Name | Type | Description | Required |
|------------|--------|--|----------|
| grant_type | String | Type of request. Available grant types are "password" and "refresh_token". | True |
| client_id | String | Used to identify the client. A client_id of "sugar" will automatically create an OAuth Key in the system that is used for "password" | True |

| Name | Type | Description | Required |
|----------------|--------|--|----------|
| | | authentication. A client_id of "support_portal" will create an OAuth Key that will allow for portal authentication. Additional client_id's can be created by the administrator in Admin > OAuth Keys to allow for additional grant types. If the client secret is populated, it will be validated against the client id. | |
| client_secret; | String | The clients secret key. | True |
| username | String | The username of the user authenticating to the system. | True |
| password | String | The plaintext password the user authenticating to the system. | True |
| platform | String | The platform type. Available types are | True |

| | | | |
|--|--|------------------------------------|--|
| | | "base", "mobile",
and "portal". | |
|--|--|------------------------------------|--|

Request for Password Grant Types

```
{  
  "grant_type": "password",  
  "client_id": "sugar",  
  "client_secret": "",  
  "username": "admin",  
  "password": "password",  
  "platform": "base"  
}
```

Request for Refresh Token Grant Types

```
{  
  "grant_type": "refresh_token",  
  "refresh_token": "c1be5132-655b-1ca3-fb44-512e36709871",  
  "client_id": "sugar",  
  "client_secret": ""  
}
```

Response Arguments

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

| Name | Type | Description |
|--------------------|---------|---|
| access_token | String | The access token needed to authenticate for other methods. |
| expires_in | Integer | The length of time until access_token expires in seconds. |
| token_type | String | The token type. Currently only "bearer" is supported. |
| null | | The Oauth scope. Normally returned as null. |
| refresh_token | String | The token needed to extend the access_token expiration timeout. |
| refresh_expires_in | Integer | The length of time until refresh_token expires in seconds. |
| download_token | String | The token used to download images and files. |

Response

```
{  
  "access_token": "802b64c0-5eac-8431-a541-5342d38ac527",  
  "expires_in": 3600,  
  "token_type": "bearer",  
  "scope": null,  
  "refresh_token": "85053198-24b1-4521-b1a1-5342d382e0b7",  
  "refresh_expires_in": 1209600,  
}
```

```
    "download_token": "8c9b5461-0d95-8d87-6084-5342d357b39e"  
}
```

Change Log

| Version | Change |
|---------|------------------------------------|
| v10 | Added /oauth2/token POST endpoint. |

Last Modified: 05/31/2017 05:29pm

/password/request GET

Overview

Sends an email request to reset a users password.

Request Arguments

| Name | Type | Description | Required |
|----------|--------|---------------------------|----------|
| email | String | The email of the user. | True |
| username | String | The username of the user. | True |

Request

`http://{site_url}/rest/v10/password/request?email=admin@sugar.crm&username=admin`

Note: GET endpoint parameters are passed in the form of a query string.

Response Arguments

| Name | Type | Description |
|---------|---------|--|
| Success | boolean | Returns the result of sending the email. |

Response

true

Change Log

| Version | Change |
|---------|--|
| v10 | Added /password/request POST endpoint. |

Last Modified: 05/31/2017 05:30pm

/ping GET

Overview

Responds with "pong" if the access_token is valid.

Request Arguments

This endpoint does not accept any arguments.

Response Arguments

| Name | Type | Description |
|------------|--------|--------------------------------|
| <response> | String | Returns pong is authenticated. |

Response

pong

Change Log

| Version | Change |
|---------|--------|
|---------|--------|

| | |
|-----|---------------------------|
| v10 | Added /ping GET endpoint. |
|-----|---------------------------|

Last Modified: 05/31/2017 05:31pm

/ping/whattimeisit GET

Overview

Responds with the current time in server format.

Request Arguments

This endpoint does not accept any arguments.

Response Arguments

| Name | Type | Description |
|---------------|--------|---------------------------|
| <server time> | String | Returns the servers time. |

Response

<time>

Change Log

| Version | Change |
|---------|--|
| v10 | Added /ping/whattimeisit GET endpoint. |

Last Modified: 05/31/2017 05:30pm

/pmse_Business_Rules GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

| Name | Type | Description | Required |
|--------|--------|-------------|----------|
| filter | String | The filter | False |

| Name | Type | Description | Required |
|------|------|-------------|--|
| | | | <p>expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query parameters in one of two ways for GET requests:</p> <ol style="list-style-type: none"> 1. By specifying individual filter arguments as distinct parameters.
Example:
filter[0][id]=1. 2. By specifying the whole filter as a single JSON-encoded string. Note that this syntax is currently not supported on certain modules.
Example:
filter=[{"id": "1 |

| | | | |
|-----------|---------|--|-------|
| | | "}]. | |
| filter_id | String | Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND. | False |
| max_num | Integer | A maximum number of records to return. Default is 20. | False |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |
| fields | String | Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters (applicable to link and collection fields). The fields id and date_modified will always be returned.
Example:
name,account_type, | False |

| | | | |
|----------|--------|---|-------|
| | | <pre>description, {"name": "opportunities", "fields": ["id", "name", "sales_status"], "order_by": "date_closed: desc"}</pre> <p>For more details on additional field parameters, see Relate API and Collection API.</p> | |
| view | String | <p>Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list".
Example: record</p> | False |
| order_by | String | <p>How to sort the returned records, in a comma delimited list with the</p> | False |

| | | | |
|---------|---------|---|-------|
| | | direction appended to the column name after a colon.
Example:
name:DESC,account_type:DESC,date_modified:ASC | |
| q | String | A search expression, will search on this module. Cannot be used at the same time as a filter expression or id. | False |
| deleted | Boolean | Boolean to show deleted records in the result set. | False |

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
```

```
"filter":[
  {
    "name":"Nelson Inc"
  }
]
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
  "filter":[
    {
      "name":{
        "$starts":"Nelson"
      }
    }
  ]
}
```

Below is a list of operation types:

| Operation | Description |
|--------------|--|
| \$equals | Performs an exact match on that field. |
| \$not_equals | Performs an exact match on that field. |
| \$not_equals | Matches on non-matching values. |

| Operation | Description |
|------------|---|
| \$starts | Matches on anything that starts with the value. |
| \$ends | Matches anything that ends with the value. |
| \$contains | Matches anything that contains the value |
| \$in | Finds anything where field matches one of the values as specified as an array. |
| \$not_in | Finds anything where field does not matches any of the values as specified as an array. |
| \$is_null | Checks if the field is null. This operation does not need a value specified. |
| \$not_null | Checks if the field is not null. This operation does not need a value specified. |
| \$lt | Matches when the field is less than the value. |
| \$lte | Matches when the field is less than or equal to the value. |
| \$gt | Matches when the field is greater than the value. |
| \$gte | Matches when the field is greater than or equal to the value. |

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "\$and" and "\$or".

Example

```
{
  "filter":[
    {
      "$or":[
        {
          "name":"Nelson Inc"
        },
        {
          "name":"Nelson LLC"
        }
      ]
    }
  ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "\$favorite" and "\$owner".

Example

```
{
  "filter":[
    {
      "$favorite":"_this"
    }
  ]
}
```

Response Arguments

| Name | Type | Description |
|-------------|---------|---|
| next_offset | Integer | Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records. |
| records | Array | An array of results containing matched records. |

Response

```
{
  "next_offset":-1,
  "records":[
    {
```

```
"id": "fa300a0e-0ad1-b322-9601-512d0983c19a",
"name": "Dale Spivey",
"date_modified": "2013-02-28T05:03:00+00:00",
"description": "",
"opportunities": [
  {
    _module: "Opportunities",
    "id": "b0701501-1fab-8ae7-3942-540da93f5017",
    "name": "360 Vacations - 228 Units",
    "date_modified": "2014-09-08T16:05:00+03:00",
    "sales_status": "New"
  },
],
"_acl": {
  "fields": {
  }
}
},
{
  "id": "95e17367-9b3d-0e26-22dc-512d0961fedf",
  "name": "Florence Haddock",
  "date_modified": "2013-02-26T19:12:00+00:00",
  "description": "",
  "opportunities": [
    {
      _module: "Opportunities"
      date_modified: "2014-09-08T16:05:00+03:00"
      id: "9ce7c088-8ee4-7cd3-18f1-540da944d4c0"
      name: "360 Vacations - 312 Units"
      sales_status: "New"
    },
  ],
  "_acl": {
```

```
    "fields": {  
    }  
  }  
]  
}
```

Change Log

| Version | Change |
|---------|--------------------------------------|
| v10 | Added /<module>/filter GET endpoint. |

Last Modified: 05/31/2017 05:30pm

/pmse_Emails_Templates GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it.

Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate

indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

| Name | Type | Description | Required |
|--------|--------|--|----------|
| filter | String | The filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query parameters in one of two ways for GET requests: <ol style="list-style-type: none">1. By specifying individual filter arguments as distinct parameters.
Example:
filter[0][id]=1.2. By specifying the whole filter as a single | False |

| Name | Type | Description | Required |
|-----------|---------|---|----------|
| | | JSON-encoded string. Note that this syntax is currently not supported on certain modules. Example: filter=[{"id":"1"}]. | |
| filter_id | String | Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND. | False |
| max_num | Integer | A maximum number of records to return. Default is 20. | False |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |
| fields | String | Comma delimited list of fields to return. Each field may be represented either by string, or by map containing | False |

| | | | |
|------|--------|---|-------|
| | | <p>field name and additional field parameters (applicable to link and collection fields). The fields id and date_modified will always be returned.</p> <p>Example:
 name,account_type,description, {"name": "opportunities", "fields": ["id", "name", "sales_status"], "order_by": "date_closed: desc" }</p> <p>For more details on additional field parameters, see Relate API and Collection API.</p> | |
| view | String | <p>Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This</p> | False |

| | | | |
|----------|---------|--|-------|
| | | argument can be used in combination with the fields argument. Common views are "record" and "list".
Example: record | |
| order_by | String | How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon.
Example:
name:DESC,account_type:DESC,date_modified:ASC | False |
| q | String | A search expression, will search on this module. Cannot be used at the same time as a filter expression or id. | False |
| deleted | Boolean | Boolean to show deleted records in the result set. | False |

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
  "filter":[
    {
      "name":"Nelson Inc"
    }
  ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
  "filter":[
    {
      "name":{
        "$starts":"Nelson"
      }
    }
  ]
}
```

```

    }
  ]
}

```

Below is a list of operation types:

| Operation | Description |
|--------------|---|
| \$equals | Performs an exact match on that field. |
| \$not_equals | Performs an exact match on that field. |
| \$not_equals | Matches on non-matching values. |
| \$starts | Matches on anything that starts with the value. |
| \$ends | Matches anything that ends with the value. |
| \$contains | Matches anything that contains the value |
| \$in | Finds anything where field matches one of the values as specified as an array. |
| \$not_in | Finds anything where field does not matches any of the values as specified as an array. |
| \$is_null | Checks if the field is null. This operation does not need a value specified. |
| \$not_null | Checks if the field is not null. This operation does not need a value specified. |
| \$lt | Matches when the field is less than the value. |
| \$lte | Matches when the field is less than or |

| Operation | Description |
|-----------|---|
| | equal to the value. |
| \$gt | Matches when the field is greater than the value. |
| \$gte | Matches when the field is greater than or equal to the value. |

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "\$and" and "\$or".

Example

```
{
  "filter": [
    {
      "$or": [
        {
          "name": "Nelson Inc"
        },
        {
          "name": "Nelson LLC"
        }
      ]
    }
  ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "\$favorite" and "\$owner".

Example

```
{
  "filter":[
    {
      "$favorite":"_this"
    }
  ]
}
```

Response Arguments

| Name | Type | Description |
|-------------|---------|---|
| next_offset | Integer | Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records. |
| records | Array | An array of results |

| Name | Type | Description |
|------|------|-----------------------------|
| | | containing matched records. |

Response

```
{
  "next_offset": -1,
  "records": [
    {
      "id": "fa300a0e-0ad1-b322-9601-512d0983c19a",
      "name": "Dale Spivey",
      "date_modified": "2013-02-28T05:03:00+00:00",
      "description": "",
      "opportunities": [
        {
          _module: "Opportunities",
          "id": "b0701501-1fab-8ae7-3942-540da93f5017",
          "name": "360 Vacations - 228 Units",
          "date_modified": "2014-09-08T16:05:00+03:00",
          "sales_status": "New"
        }
      ],
      "_acl": {
        "fields": {
        }
      }
    }
  ],
  {
    "id": "95e17367-9b3d-0e26-22dc-512d0961fedf",
    "name": "Florence Haddock",
  }
}
```

```

    "date_modified": "2013-02-26T19:12:00+00:00",
    "description": "",
    "opportunities": [
      {
        _module: "Opportunities"
        date_modified: "2014-09-08T16:05:00+03:00"
        id: "9ce7c088-8ee4-7cd3-18f1-540da944d4c0"
        name: "360 Vacations - 312 Units"
        sales_status: "New"
      },
    ],
    "_acl": {
      "fields": {
      }
    }
  }
]
}

```

Change Log

| Version | Change |
|---------|--------------------------------------|
| v10 | Added /<module>/filter GET endpoint. |

Last Modified: 05/31/2017 05:30pm

/pmse_Project GET

Overview

Lists filtered records.

Summary

This endpoint will return a set of records filtered by an expression. The filter can be applied to multiple fields and have multiple and/or conditions in it. Alternatively, you may use an existing filter by specifying its id. If both a filter definition and a filter id are passed, the two filters will be joined with an AND. Care will need to be taken to make sure that any filters used have appropriate indexes on the server side otherwise the runtime of the endpoint will be very long. Related fields can be searched by specifying the field name as: "link_name.remote_field", so if you wished to search the Accounts module by a related member account you would use "members.sic_code".

Request Arguments

| Name | Type | Description | Required |
|--------|--------|---|----------|
| filter | String | The filter expression. Filter expressions are explained below. Note that JSON-encoded filters can be specified as query parameters in one of two ways for GET requests: | False |

| Name | Type | Description | Required |
|-----------|---------|---|----------|
| | | <p>By specifying individual filter arguments as distinct parameters. Example: filter[0][id]=1. By specifying the whole filter as a single JSON-encoded string. Note that this syntax is currently not supported on certain modules. Example: filter=[{"id":"1"}].</p> | |
| filter_id | String | Identifier for a preexisting filter. If filter is also set, the two filters are joined with an AND. | False |
| max_num | Integer | A maximum number of records to return. Default is | False |

| | | | |
|--------|---------|---|-------|
| | | 20. | |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |
| fields | String | Comma delimited list of fields to return. Each field may be represented either by string, or by map containing field name and additional field parameters (applicable to link and collection fields). The fields id and date_modified will always be returned.
Example:
name,account_type,description,{"name":"opportunities","fields":["id","name","sales_status"],"order_by":"date_closed:desc"}
For more details on additional field parameters, see Relate API and | False |

| | | Collection API. | |
|----------|--------|---|-------|
| view | String | Instead of defining the fields argument, the view argument can be used instead. The field list is constructed at the server side based on the view definition which is requested. This argument can be used in combination with the fields argument. Common views are "record" and "list".
Example: record | False |
| order_by | String | How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon.
Example:
name:DESC,account_type:DESC,date_modified:ASC | False |
| q | String | A search expression, will | False |

| | | | |
|---------|---------|--|-------|
| | | search on this module. Cannot be used at the same time as a filter expression or id. | |
| deleted | Boolean | Boolean to show deleted records in the result set. | False |

Filter Expressions

There are four types of filters:

Basic

This will filter the results by checking the field "name" for value "Nelson Inc". This will only find exact matches.

Example

```
{
  "filter":[
    {
      "name":"Nelson Inc"
    }
  ]
}
```

Full

This expression allows you to specify what operation you want to use for filtering on the field. In the example you would match any record where the field "name" starts with the value "Nelson".

Example

```
{
  "filter": [
    {
      "name": {
        "$starts": "Nelson"
      }
    }
  ]
}
```

Below is a list of operation types:

| Operation | Description |
|--------------|--|
| \$equals | Performs an exact match on that field. |
| \$not_equals | Performs an exact match on that field. |
| \$not_equals | Matches on non-matching values. |
| \$starts | Matches on anything that starts with the value. |
| \$ends | Matches anything that ends with the value. |
| \$contains | Matches anything that contains the value |
| \$in | Finds anything where field matches one of the values as specified as an array. |
| | |

| Operation | Description |
|------------|---|
| \$not_in | Finds anything where field does not matches any of the values as specified as an array. |
| \$is_null | Checks if the field is null. This operation does not need a value specified. |
| \$not_null | Checks if the field is not null. This operation does not need a value specified. |
| \$lt | Matches when the field is less than the value. |
| \$lte | Matches when the field is less than or equal to the value. |
| \$gt | Matches when the field is greater than the value. |
| \$gte | Matches when the field is greater than or equal to the value. |

Sub-expressions

This allows you to group filter expressions into or/and groupings. By default all expressions are and'ed together. The example expression would match if the field "name" was either "Nelson Inc" or "Nelson LLC". The only currently accepted sub-expression types are "\$and" and "\$or".

Example

```
{
  "filter": [
```

```
{
  "$or": [
    {
      "name": "Nelson Inc"
    },
    {
      "name": "Nelson LLC"
    }
  ]
}
```

Modules

There are two module expressions, they operate on modules instead of fields. The current module can be specified by either using the module name "_this" or by leaving the module name as a blank string. The example expression would filter the records in the current module to only your favorites. The only currently accepted module expressions are "\$favorite" and "\$owner".

Example

```
{
  "filter": [
    {
      "$favorite": "_this"
    }
  ]
}
```

Response Arguments

| Name | Type | Description |
|-------------|---------|---|
| next_offset | Integer | Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records. |
| records | Array | An array of results containing matched records. |

Response

```
{
  "next_offset":-1,
  "records":[
    {
      "id":"fa300a0e-0ad1-b322-9601-512d0983c19a",
      "name":"Dale Spivey",
      "date_modified":"2013-02-28T05:03:00+00:00",
      "description":"",
      "opportunities": [
        {
          _module: "Opportunities",
          "id": "b0701501-1fab-8ae7-3942-540da93f5017",
          "name": "360 Vacations - 228 Units",
          "date_modified": "2014-09-08T16:05:00+03:00",
```

```
        "sales_status": "New"
    },
],
"_acl": {
    "fields": {
    }
}
},
{
    "id": "95e17367-9b3d-0e26-22dc-512d0961fedf",
    "name": "Florence Haddock",
    "date_modified": "2013-02-26T19:12:00+00:00",
    "description": "",
    "opportunities": [
        {
            _module: "Opportunities"
            date_modified: "2014-09-08T16:05:00+03:00"
            id: "9ce7c088-8ee4-7cd3-18f1-540da944d4c0"
            name: "360 Vacations - 312 Units"
            sales_status: "New"
        },
    ],
    "_acl": {
        "fields": {
        }
    }
}
]
}
```

Change Log

| Version | Change |
|---------|--------------------------------------|
| v10 | Added /<module>/filter GET endpoint. |

Last Modified: 05/31/2017 05:30pm

/recent GET

Overview

Returns all of the current users recently viewed records.

Request Arguments

| Name | Type | Description | Required |
|-------------|--------|--|----------|
| module_list | String | Comma delimited list of modules to return recently viewed records.
Example:
Accounts,Contacts | True |
| fields | String | Comma delimited list of fields to return. The field date_modified will always be returned.
Example: id,name | False |
| date | String | The date | False |

| Name | Type | Description | Required |
|--------|---------|---|----------|
| | | expression. Filter recently viewed records from this date. | |
| limit | Integer | A maximum number of records to return. Default is 20. | False |
| offset | Integer | The number of records to skip over before records are returned. Default is 0. | False |

Request

`http://{site_url}/rest/v10/recent?module_list=Accounts%2CContacts`

Note: GET endpoint parameters are passed in the form of a query string.

Response Arguments

| Name | Type | Description |
|-------------|--------|---|
| next_offset | String | The next offset to start fetching additional records. |
| | | |

| Name | Type | Description |
|---------|-------|--------------------------------------|
| records | Array | An array of recently viewed records. |

Response

```
{
  "next_offset":-1,
  "records":[
    {
      "my_favorite":false,
      "following":false,
      "id":"e4213959-35cd-119b-5cd6-5342e8be16f6",
      "name":"Leila Purifoy",
      "date_entered":"2014-04-07T13:58:50-04:00",
      "date_modified":"2014-04-07T13:58:50-04:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "doc_owner":"",
      "description":"",
      "deleted":false,
      "assigned_user_id":"seed_will_id",
      "assigned_user_name":"Will Westin",
      "team_count":"",
      "team_name":[
        {
          "id":"East",
          "name":"East",

```

```
        "name_2": "",
        "primary": true
    },
    {
        "id": "West",
        "name": "West",
        "name_2": "",
        "primary": false
    }
],
"email": [
    {
        "email_address": "support62@example.biz",
        "invalid_email": false,
        "opt_out": true,
        "primary_address": false,
        "reply_to_address": false
    },
    {
        "email_address": "sales39@example.com",
        "invalid_email": false,
        "opt_out": false,
        "primary_address": true,
        "reply_to_address": true
    }
],
"email1": "sales39@example.com",
"email2": "support62@example.biz",
"invalid_email": false,
"email_opt_out": false,
"email_addresses_non_primary": "",
"salutation": "",
"first_name": "Leila",
```

```
"last_name": "Purifoy",
"full_name": "Leila Purifoy",
"title": "President",
"facebook": "",
"twitter": "",
"googleplus": "",
"department": "",
"do_not_call": false,
"phone_home": "(841) 469-8223",
"phone_mobile": "(286) 010-9553",
"phone_work": "(369) 075-2809",
"phone_other": "",
"phone_fax": "",
"primary_address_street": "345 Sugar Blvd.",
"primary_address_street_2": "",
"primary_address_street_3": "",
"primary_address_city": "Santa Monica",
"primary_address_state": "NY",
"primary_address_postalcode": "52255",
"primary_address_country": "USA",
"alt_address_street": "",
"alt_address_street_2": "",
"alt_address_street_3": "",
"alt_address_city": "",
"alt_address_state": "",
"alt_address_postalcode": "",
"alt_address_country": "",
"assistant": "",
"assistant_phone": "",
"picture": "",
"email_and_name1": "",
"lead_source": "Trade Show",
"account_name": "Sea Region Inc",
```

```
"account_id":"b1cd3a55-7e84-e53b-954e-5342e85b63f1",
"dnb_principal_id":"",
"opportunity_role_fields":"",
"opportunity_role_id":"",
"opportunity_role":"",
"reports_to_id":"",
"report_to_name":"",
"birthdate":"",
"portal_name":"LeilaPurifoy5",
"portal_active":true,
"portal_password":true,
"portal_password1":null,
"portal_app":"",
"preferred_language":"",
"campaign_id":"",
"campaign_name":"",
"c_accept_status_fields":"",
"m_accept_status_fields":"",
"accept_status_id":"",
"accept_status_name":"",
"accept_status_calls":"",
"accept_status_meetings":"",
"sync_contact":false,
"mkto_sync":false,
"mkto_id":null,
"mkto_lead_score":null,
"_acl":{
  "fields":{

  }
},
"_module":"Contacts",
"_last_viewed_date":"2014-04-07T14:43:46-04:00"
```

```
},
{
  "my_favorite":false,
  "following":false,
  "id":"e8f7eb6d-2647-7e57-df30-5342e83c622d",
  "name":"Draft Diversified Energy Inc",
  "date_entered":"2014-04-07T13:58:50-04:00",
  "date_modified":"2014-04-07T13:58:50-04:00",
  "modified_user_id":"1",
  "modified_by_name":"Administrator",
  "created_by":"1",
  "created_by_name":"Administrator",
  "doc_owner":"","
  "description":"","
  "deleted":false,
  "assigned_user_id":"seed_max_id",
  "assigned_user_name":"Max Jensen",
  "team_count":"","
  "team_name":[
    {
      "id":"West",
      "name":"West",
      "name_2":"","
      "primary":true
    }
  ],
  "email":[
    {
      "email_address":"dev.kid.kid@example.de",
      "invalid_email":false,
      "opt_out":false,
      "primary_address":true,
      "reply_to_address":false
    }
  ]
}
```

```
    },
    {
      "email_address": "info.sales@example.co.uk",
      "invalid_email": false,
      "opt_out": false,
      "primary_address": false,
      "reply_to_address": false
    }
  ],
  "email1": "dev.kid.kid@example.de",
  "email2": "info.sales@example.co.uk",
  "invalid_email": false,
  "email_opt_out": false,
  "email_addresses_non_primary": "",
  "facebook": "",
  "twitter": "",
  "googleplus": "",
  "account_type": "Customer",
  "industry": "Education",
  "annual_revenue": "",
  "phone_fax": "",
  "billing_address_street": "111 Silicon Valley Road",
  "billing_address_street_2": "",
  "billing_address_street_3": "",
  "billing_address_street_4": "",
  "billing_address_city": "Los Angeles",
  "billing_address_state": "CA",
  "billing_address_postalcode": "26022",
  "billing_address_country": "USA",
  "rating": "",
  "phone_office": "(790) 406-0049",
  "phone_alternate": "",
  "website": "www.phonesugar.de",
```

```
"ownership": "",
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "111 Silicon Valley Road",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "Los Angeles",
"shipping_address_state": "CA",
"shipping_address_postalcode": "26022",
"shipping_address_country": "USA",
"parent_id": "",
"sic_code": "",
"duns_num": "",
"parent_name": "",
"campaign_id": "",
"campaign_name": "",
"_acl": {
  "fields": {
    }
  },
  "_module": "Accounts",
  "_last_viewed_date": "2014-04-07T14:43:36-04:00"
}
]
```

Change Log

| Version | API | Change |
|---------|-----|-----------------------------|
| 7.2.0 | v10 | Added /recent GET endpoint. |

Last Modified: 05/31/2017 05:31pm

/rssfeed GET

Overview

Consumes an RSS feed as a proxy and returns the feed up to a certain number of entries.

Request Arguments

| Name | Type | Description | Required |
|----------|---------|---|----------|
| feed_url | String | A fully qualified URL to an RSS feed. | True |
| limit | Integer | Maximum number of entries to fetch. If not provided, the API will return up to <code>\$sugar_config['rss_feed_max_entries']</code> or 20 if no config | False |

| Name | Type | Description | Required |
|------|------|-------------|----------------|
| | | | option is set. |

Request

`http://{site_url}/rest/v10/rssfeed?feed_url=http%3A%2F%2Ffqd.rssfeed.url%2F&limit=10`

Note: GET endpoint parameters are passed in the form of a query string.

Response Arguments

| Name | Type | Description |
|------------------|-----------|--|
| title | String | The title of the RSS Feed. This can be blank. |
| link | String | The URL to the source of the RSS Feed. This can be blank. |
| description | String | A description of the feed. This can be blank. |
| publication_date | Timestamp | A timestamp of when the feed was published. |
| entries | Array | An array of entries, each of which will contain the following values: <ul style="list-style-type: none"> • title • description |

| Name | Type | Description |
|------|------|--|
| | | link
publication_date
source
author |

Response

```
{
  "feed":{
    "title":"Sample Feed Title",
    "link":"http://www.samplefeed.com/feed-link-url.htm",
    "description":"A sample of a feed description",
    "publication_date":"Tue, 10 Aug 2014 13:38:55 -0800",
    "entries":[
      {
        "title":"First entry title",
        "description":"A blurb about the first entry. This
will be HTML encoded on return.",
        "link":"http://www.samplefeed.com/feed-entry-1.htm",
        "publication_date":"Tue, 10 Aug 2014 13:38:55 -0800",
        "source":"Reuters",
        "author":"John Doe"
      },
      {
        "title":"Second entry title",
        "description":"A blurb about the Second entry. This
will be HTML encoded on return.",
        "link":"http://www.samplefeed.com/feed-entry-2.htm",
```

```
        "publication_date": "Tue, 10 Aug 2014 13:38:55 -0800",
        "source": "BBC",
        "author": ""
    }
]
}
}
```

Change Log

| Version | Change |
|---------|--------------------------------|
| v10 | Added /<rssfeed> GET endpoint. |

Last Modified: 05/31/2017 05:31pm

/search GET

Overview

List records in a module. Searching, filtering and ordering can be applied to only fetch the records you are interested in. Additionally the set of returned fields can be restricted to speed up processing and reduce download times.

Request Arguments

| Name | Type | Description | Required |
|------|------|-------------|----------|
|------|------|-------------|----------|

| Name | Type | Description | Required |
|----------|---------|--|----------|
| q | String | The search text to match records on. This will search through any fields on the module that has unified_search set to true. | False |
| max_num | Integer | A maximum number of records to return. | False |
| offset | Integer | The number of records to skip over before records are returned. | False |
| fields | String | Comma delimited list of fields to return. The field date_modified will always be returned. Example: name,account_type,description | False |
| order_by | String | How to sort the returned records, in a comma delimited list with the direction appended to the column name after a colon. Example: | False |

| | | | |
|-----------|---------|---|-------|
| | | name:DESC,account_type:DESC,date_modified:ASC | |
| favorites | Boolean | Only fetch the current users favorited records. | False |
| my_items | Boolean | Only fetch items assigned to the current user. | False |

Response Arguments

| Name | Type | Description |
|-------------|---------|---|
| next_offset | Integer | Displays the next offset for retrieval of additional results. -1 will be returned when there are no more records. |
| records | Array | An array of results containing matched records. |

Response

```
{
  "next_offset":2,
  "records":[
```

```
{
  "id":"ecbf2a6c-261e-5fca-fbb6-512d093554b8",
  "name":"Avery Software Co",
  "date_modified":"2013-02-26T19:12:56+00:00",
  "description":"",
  "my_favorite":false,
  "_acl":{
    "fields":{

    }
  },
  "_module":"Accounts",
  "_search":{
    "score":1
  }
},
{
  "id":"af5f8dae-7169-b497-1d77-512d0937ed81",
  "name":"Avery Software Co",
  "date_modified":"2013-02-26T19:12:56+00:00",
  "description":"",
  "my_favorite":false,
  "_acl":{
    "fields":{

    }
  },
  "_module":"Accounts",
  "_search":{
    "score":1
  }
}
]
```

}

Change Log

| Version | Change |
|---------|-------------------------------|
| v10 | Added /<module> GET endpoint. |

Last Modified: 05/31/2017 05:31pm

/theme GET

Overview

Fetches the customizable variables of a theme.

Request Arguments

| Name | Type | Description | Required |
|----------|--------|--|----------|
| platform | String | The theme platform
-
/themes/clients/***P
LATFORM***/them
eName/. Accepted
values are 'base'
and 'portal'. | False |
| | | | |

| Name | Type | Description | Required |
|-----------|--------|---|----------|
| themeName | String | The theme name - /themes/clients/platform/***/THEME NAME***/. | False |

Request

`http://{site_url}/rest/v10/theme?platform=base&themeName=default`

Note: GET endpoint parameters are passed in the form of a query string.

Response Arguments

| Name | Type | Description |
|--------|-------|--|
| mixins | Array | An array of name value pairs detailing mixin colors |
| hex | Array | A array of name value pairs detailing css colors |
| rgba | Array | An array of name value pairs detailing colors |
| rel | Array | An array of name value pairs detailing relationships |

| Name | Type | Description |
|------|-------|---|
| bg | Array | An array of name value pairs detailing backgroup colors |

Response

```
{
  "colors": [
    {
      "name": "BorderColor",
      "value": "#E61718"
    },
    {
      "name": "NavigationBar",
      "value": "#000000"
    },
    {
      "name": "PrimaryButton",
      "value": "#177EE5"
    }
  ]
}
```

Change Log

| Version | Change |
|---------|--------|
| | |

| | |
|-----|------------------------------|
| v10 | Added /<theme> GET endpoint. |
|-----|------------------------------|

Last Modified: 05/31/2017 05:31pm

/theme POST

Overview

Updates the variables.less (less file containing customizable vars in the theme folder) with the set of variables passed as arguments.

Request Arguments

| Name | Type | Description | Required |
|------------------|--------|---|----------|
| platform | String | The theme platform -
/themes/clients/***P
LATFORM***/them
eName/. Accepted
values are 'base'
and 'portal'. | True |
| themeName | String | The theme name -
/themes/clients/plat
form/***THEMENA
ME***/. | True |
| <theme variable> | String | The variables of the
theme | False |

Request

```
{  
  platform: 'portal',  
  themeName: 'default',  
  primary: 'default',  
  secondary: '#aaaaaa',  
  primaryBtn: '#bbbbbb',  
}
```

Response Arguments

| Name | Type | Description |
|------------|--------|---|
| <css path> | String | Returns the path of the new bootstrap.css file. |

Response

```
"http://sugarcrm/cache/themes/clients/base/default/6a031485bf5  
239b9462e4aaba72a4646.css"
```

Change Log

| Version | Change |
|---------|-------------------------------|
| v10 | Added /<theme> POST endpoint. |

Last Modified: 05/31/2017 05:31pm

Extending Endpoints

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. When creating a new endpoint, please note that the file name must match the class name for the endpoint definitions. The example below demonstrates creating a GET endpoint.

./custom/clients/base/api/MyEndpointsApi.php

```
<?php

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

class MyEndpointsApi extends SugarApi
{
    public function registerApiRest()
    {
        return array(
            //GET
            'MyGetEndpoint' => array(
                //request type
                'reqType' => 'GET',

                //set authentication
                'noLoginRequired' => false,

                //endpoint path
                'path' => array('MyEndpoint', 'GetExample', '?'),

                //endpoint variables
                'pathVars' => array('', '', 'data'),

                //method to call
                'method' => 'MyGetMethod',
```

```

        //short help string to be displayed in the help
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;
}

}

?>

```

Note: The file name must match the class name for the endpoint definitions. Given the example above, a class name of MyEndpointsApi must be created as MyEndpointsApi.php.

registerApiRest() Method

Your extended class will contain two methods. The first method is

registerApiRest() which will return an array that defines your REST endpoints. Each endpoint will need the following properties:

| Name | Type | Description |
|-----------------|----------------|--|
| reqType | Array String | An array of one or many request types of the endpoint. Possible values are: GET, PUT, POST and DELETE. |
| noLoginRequired | Boolean | Determines whether the endpoint is authenticated. Setting this value to true will remove the authentication requirement. |
| path | Array | The endpoint path. An endpoint path of:

<pre>'path' =>
array('MyEndpoint',
'GetExample'),</pre>
Will equate to the path of:

<pre>http://{site
url}/rest/v10/MyEndpoi
nt/GetExample</pre>
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 | <p>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.</p> <p>Example:</p> <pre>'path' => array('MyEndpoint', 'GetExample', '?'), ?'pathVars' => array('', '', 'data'),</pre> <p>The above will pass in the following to your selected method as the \$args parameter when calling <code>http://{site url}/rest/v10/MyEndpoint/GetExample/MyData</code></p> <pre>stdClass Object ([__sugar_url] => v10/MyEndpoint/GetExam ple/mydata [data] => MyData</pre> |

| | | |
|------------|---------|--|
| | |) |
| method | String | The method to pass the pathVars to. This can be named anything you choose and will be located in your SugarApi extended class. |
| shortHelp | String | A short help string for developers when looking at the help documentation. |
| longHelp | String | Path to a long help file. This file will be loaded when a user clicks an endpoint from the help documentation. |
| minVersion | Integer | The minimum API Version this endpoint can be used with. See Scoring Section below. |
| maxVersion | Integer | The maximum API Version this endpoint can be used with. See Scoring Section below. |
| extraScore | Integer | Add an extra score value to the Scoring of this endpoint, to place priority on its usage. See Scoring Section below. |

Endpoint Method

The second method can be named anything you choose but it is important to note that it will need to match the name of the method you specified for your endpoint. This method will require the parameters \$api and \$args. The MyGetMethod() in the example above will be used when the endpoint MyEndpoint/GetExample is called. Any path variables, query string parameters or posted data will be available in the \$args parameter for you to work with.

```
public function $endpointMethod($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>

This endpoint does not accept any input parameters.

<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.

Example

```
./custom/clients/base/api/MyEndpointsApi.php
```

```
<?php
```

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

class MyEndpointsApi extends SugarApi
{
    public function registerApiRest()
    {
        return array(
            //GET & POST
            'MyGetEndpoint' => array(
                //request type
                'reqType' => array('GET','POST'),

                //set authentication
                'noLoginRequired' => false,

                //endpoint path
                'path' => array('MyEndpoint', 'GetExample', '?'),

                //endpoint variables
                'pathVars' => array('', '', 'data'),

                //method to call
                'method' => 'MyGetMethod',

                //short help string to be displayed in the help
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;
    }
}

?>
```

Redefining Existing Endpoints

With endpoints, you may have a need to extend an existing endpoint to meet your needs. When doing this it is important that you do not remove anything from the return array of the endpoint. Doing so could result in unexpected behavior due to other functionality using the same endpoint.

For this example, we will extend the ping endpoint to return "Pong <timestamp>". To do this, we will need to extend the existing PingApi class and define our method overrides. When creating a new endpoint, please note that the file name must match the class name for the endpoint definitions. For our example, we will prefix the original class name with "custom" to create our overriding endpoint.

```
./custom/clients/base/api/CustomPingApi.php
```

```
<?php
```

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

require_once("clients/base/api/PingApi.php");

class CustomPingApi extends PingApi
{
    public function registerApiRest()
    {
        //in case we want to add additional endpoints
        return parent::registerApiRest();
    }

    //override to modify the ping function of the ping endpoint
    public function ping($api, $args)
    {
        $result = parent::ping($api, $args);

        //append the current timestamp
        return $result . ' ' . time();
    }
}
```

Note: The file name must match the class name for the endpoint definitions. Given the example above, a class name of CustomPingApi must be created as CustomPingApi.php.

As you can see, we extended registerApiRest to fetch existing endpoint definitions in case we want to add our own. We then added an override for the ping method to return our new value. Once the file is in place you will need to navigate to Admin > Repair > Quick Repair and Rebuild. Once completed, any call made to <url>/rest/v10/ping will result in a response of "ping <timestamp>".

Endpoint Scoring

When generating custom endpoints or overriding existing endpoints, the system can determine which endpoint to use based on the score placed on the registered endpoints. The scoring calculation works in the following manner for registered endpoints.

Route Path	Score
?	0.75
<module>	1.0
Exact Match	1.75
Custom	0.5
Endpoint 'extraScore' property	Defined Extra Score

The defined Path and extraScore properties in the registerApiRest() method, help determine the score for any given Endpoint. The higher the score allows the API to determine which Endpoint is being used for a given request. For example, if you had extended the Accounts/:record GET API Endpoint as follows:

```
/custom/modules/Accounts/clients/base/api/CustomAccountsApi.php
```

```
<?php
```

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

```
require_once("clients/base/api/ModuleApi.php");
```

```
class CustomAccountsApi extends ModuleApi
{
    public function registerApiRest()
```

```

{
  return array(
    //GET & POST
    'getAccount' => array(
      //request type
      'reqType' => array('GET'),

      //set authentication
      'noLoginRequired' => false,

      //endpoint path
      'path' => array('Accounts', '?'),

      //endpoint variables
      'pathVars' => array('module', 'record'),

      //method to call
      'method' => 'retrieveRecord',

      //short help string to be displayed in the help
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 Accounts/:record endpoint
 */

```

```

public function retrieveRecord($api, $args)
{
    //custom logic
    return parent::retrieveRecord($api,$args);
}
}
?>

```

This Endpoint will be used when accessing Accounts/<record_id> since the Exact Match of Accounts in the path property gives a greater score than the <module> match that comes with the standard <module>/:record Endpoint defined in ModuleApi.

Endpoint Versioning

Part of the scoring calculation, allows for versioning of the API Endpoints to accommodate the same endpoint path in the Rest API, for other versions of the API. If you have a custom Endpoint at rest/v10/CustomEndpoint, and would like to alter the logic slightly for another use without deprecating or removing the old Endpoint, you can use the versioning techniques to have the new logic reside at rest/v11/CustomEndpoint. The following table describes how the Versioning properties on the Endpoint definition affect the score of the Endpoint to allow the API to determine which Endpoint should be used for a given request.

Property	Score
minVersion	0.02 + minVersion/1000
maxVersion	0.02
minVersion === maxVersion (i.e. Both minVersion and maxVersion properties)	0.06 + minVersion/1000

are defined on Endpoint to the same value)

To allow for additional Versions in the API, you will first need to customize the 'maxVersion' property in the \$apiSettings array. To do this, create the following file:

```
./custom/include/api/metadata.php
```

Inside the file you can set the \$apiSettings['maxVersion'] property as follows:

```
<?php  
  
$apiSettings['maxVersion'] = 11;
```

Once you have configured the Max Version property you can then set the minVersion and maxVersion properties on the Endpoint definition in the custom registerApiRest() method. For example, the following Endpoint definitions:

```
<?php  
...  
    return array(  
        'foo1' => array(  
            'reqType' => 'GET',  
            'path' => array('Accounts','config',  
            'pathVars' => array('module', 'record'),  
            'method' => 'foo1',  
        ),  
        'foo2' => array(  
            'reqType' => 'GET',  
            'minVersion' => 11,  
            'path' => array('Accounts','config',  
            'pathVars' => array('module', 'record'),  
            'method' => 'foo2',
```

```

    ),
    'foo3' => array(
        'reqType' => 'GET',
        'minVersion' => 11,
        'maxVersion' => 11,
        'path' => array('Accounts','config',
        'pathVars' => array('module', 'record'),
        'method' => 'foo3',
    ),
    'foo4' => array(
        'reqType' => 'GET',
        'minVersion' => 11,
        'maxVersion' => 12,
        'path' => array('Accounts','config',
        'pathVars' => array('module', 'record'),
        'method' => 'foo3',
    ),
    'foo5' => array(
        'reqType' => 'GET',
        'minVersion' => 14,
        'path' => array('Accounts','config',
        'pathVars' => array('module', 'record'),
        'method' => 'foo5',
    ),
);
...

```

With the above definitions the following table provides the Expected Request for each Endpoint.

Request URI	Best Route
/rest/v10/Accounts/config	foo1
/rest/v11/Accounts/config	foo3

/rest/v12/Accounts/config	foo4
/rest/v13/Accounts/config	foo2
/rest/v14/Accounts/config	foo5

Last Modified: 06/26/2017 02:28pm

API Exceptions

Overview

Sugar comes with some predefined API Exceptions, located in `./include/api/`, that can be called from API endpoints. These exceptions return a specific HTTP code and a message.

Stock Exceptions

Exception Class	HTTP Code	Error Label	Language Label Key	Language Label Value
SugarApiExceptionError	500	fatal_error	EXCEPTION_FATAL_ERROR	Your request failed to complete. A fatal error occurred. Check logs for more details.
SugarApiExceptionIncorrectVersion	301	incorrect_version	EXCEPTION_INCORRECT_VERSION	The version of the API you are using is not

				correct for the current request.
SugarApiExceptionNeedLogin	401	need_login	EXCEPTION_NEED_LOGIN	You need to be logged in to perform this action.
SugarApiExceptionInvalidGrant	401	invalid_grant	EXCEPTION_INVALID_TOKEN	Your authentication token is invalid.
SugarApiExceptionNotAuthorized	403	not_authorized	EXCEPTION_NOT_AUTHORIZED	You are not authorized to perform this action. Contact your administrator if you need access.
SugarApiExceptionPortalUserInactive	403	inactive_portal_user	EXCEPTION_INACTIVE_PORTAL_USER	You cannot access Portal because your portal account is inactive. Please contact customer support if you need access.
SugarApiExceptionPortalNotConfigured	403	portal_not_configured	EXCEPTION_PORTAL_NOT_CONFIGURED	Portal is not configured properly. Contact your Portal

				Administrator for assistance.
SugarApiExceptionNoMethod	404	no_method	EXCEPTION_NO_METHOD	Your request was not supported. Could not find the HTTP method of your request for this path.
SugarApiExceptionNotFound	404	not_found	EXCEPTION_NOT_FOUND	Your requested resource was not found. Could not find a handler for the path specified in the request.
SugarApiExceptionEditConflict	409	edit_conflict	EXCEPTION_EDIT_CONFLICT	Edit conflict, please reload the record data.
SugarApiExceptionInvalidHash	412	metadata_out_of_date	EXCEPTION_METADATA_OUT_OF_DATE	Your metadata or user hash did not match the server. Please resync your metadata.
SugarApiExceptionRequestTooLarge	413	request_too_large	EXCEPTION_REQUEST_TOO_LARGE	Your request is too large to process.
SugarApiExceptionMissingParameter	422	missing_parameter	EXCEPTION_MISSING_PARAMETER	A required parameter in

parameter			EXCEPTION_PARAMETER	Your request was missing.
SugarApiExceptionInvalidParameter	422	invalid_parameter	EXCEPTION_INVALID_PARAMETER	A parameter in your request was invalid.
SugarApiExceptionRequestMethodFailure	424	request_failure	EXCEPTION_REQUEST_FAILURE	Your request failed to complete.
SugarApiExceptionClientOutdated	433	client_outdated	EXCEPTION_CLIENT_OUTDATED	Your software is out of date, please update your client before attempting to connect again.
SugarApiExceptionConnectorResponse	502	bad_gateway	EXCEPTION_CONNECTOR_RESPONSE	A connector or an integration request resulted in a failed response.
SugarApiExceptionMaintenance	503	maintenance	EXCEPTION_MAINTENANCE	SugarCRM is in maintenance mode. Only admins can login. Please contact your administrator for details.
SugarApiExceptionServiceUnavailable	503	service_unavailable	EXCEPTION_SERVICE_UNAVAILABLE	The server cannot process your request

				because it is busy or unavailable at this time.
SugarApiExceptionSearchUnavailable	400	search_unavailable	EXCEPTION_SEARCH_UNAVAILABLE	Search engine is temporarily unavailable.
SugarApiExceptionSearchRuntime	400	search_runtime	EXCEPTION_SEARCH_RUNTIME	A search engine runtime error occurred. Please contact your System Administrator.

Using Exceptions

When extending endpoints in Sugar, the stock API exceptions can be used to return errors and messages. The exception classes accept a single parameter on creation for the message value to return. This parameter can accept language label keys or plain text. If no parameter is passed exception, the error will default to the exception's default language label key. To call a stock API exception from custom code add the following snippet :

```
//throwing an api exception using the stock message
throw new SugarApiExceptionError();
```

```
//throwing an api exception using a custom language label key
throw new SugarApiExceptionError('EXCEPTION_CSTM_LABEL_KEY');
```

```
//throwing an api exception with text
throw new SugarApiExceptionError('There has been an error.');
```

This will return an http response code of 500 with the following response data:

```
{
  "error": "fatal_error",
  "error_message": "There has been an error."
}
```

Custom Exceptions

Sugar gives you the ability to create custom API exceptions by creating a new class in `./custom/include/api/` that extends the stock `SugarApiException` class. When extending the class you can provide the following properties in your custom code:

<code>\$httpCode</code>	The http response code to send back in the header. Defaults to 400 if not specified.
<code>\$errorLabel</code>	The string value returned in the 'error' key response.
<code>\$messageLabel</code>	The label to return as a default response if a message is not provided.

Example

The following example will demonstrate how to create a custom exception.

```
./custom/include/api/<name>.php
```

```
<?php
```

```
require_once 'include/api/SugarApiException.php';

/**
 * Custom error.
 */
class cstmSugarApiExceptionError extends SugarApiException
{
    public $httpCode = 404;
    public $errorLabel = 'my_error';
    public $messageLabel = 'EXCEPTION_CSTM_LABEL_KEY';
}
```

Create a custom language label using the extension framework:

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

```
<?php
    $app_strings['EXCEPTION_CSTM_LABEL_KEY'] = 'My Exception
Message.';
```

You can call your new exception by using the following code :

```
require_once 'custom/include/api/<name>.php';

//throwing custom api exception using the default message
throw new cstmSugarApiExceptionError();

//throwing custom api exception using a custom language label key
throw new cstmSugarApiExceptionError('EXCEPTION_CSTM_LABEL_KEY');

//throwing custom api exception with text
throw new cstmSugarApiExceptionError('There has been an error.');
```

You will then need to navigate to Admin > Repair > Quick Repair and Rebuild before using your exception.

The exception error will return a response of:

```
{
  "error": "my_error",
  "error_message": "There has been an error."
}
```

Last Modified: 11/27/2017 12:17pm

Examples

Examples of integrating with Sugar APIs.

Last Modified: 10/14/2016 04:12pm

Bash

Bash cURL examples of integrating with the REST v10 API.

Last Modified: 10/14/2016 04:12pm

How to Export a List of Records

Overview

An example in bash script demonstrating how to export a list of records using the `v10 /<module>/export/:record_list_id` REST GET endpoint.

Exporting Records

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
curl -X POST -H Cache-Control:no-cache -d '{
  "grant_type":"password",
  "client_id":"sugar",
  "client_secret":"",
  "username":"admin",
  "password":"password",
  "platform":"custom_api"
}' https://{site_url}/rest/v10/oauth2/token
```

More information on authenticating can be found in the [How to Authenticate and Log Out](#) example and `/oauth2/logout` endpoint documentation.

Filtering Records

Next, we will need to identify the records we want to export using the `/<module>/filter` endpoint.

```
curl -s -X POST -H OAuth-Token:{access_token} -H
Cache-Control:no-cache -d '{
  "filter":[
    {
      "$or":[
```

```
    {
      "name": {
        "$starts": "A"
      }
    },
    {
      "name": {
        "$starts": "b"
      }
    }
  ]
}
],
"max_num": 2,
"offset": 0,
"fields": "id",
"order_by": "date_entered",
"favorites": false,
"my_items": false
}' https://{site_url}/rest/v10/Accounts/filter
```

More information on the filter API can be found in the /<module>/filter documentation.

Response

The data received from the server is shown below:

```
{
  "next_offset": 2,
  "records": [
    {
```

```
    "id": "f16760a4-3a52-f77d-1522-5703ca28925f",
    "date_modified": "2016-04-05T10:23:28-04:00",
    "_acl": {
      "fields": {
        }
      },
    "_module": "Accounts"
  },
  {
    "id": "ec409fbb-2b22-4f32-7fa1-5703caf78dc3",
    "date_modified": "2016-04-05T10:23:28-04:00",
    "_acl": {
      "fields": {
        }
      },
    "_module": "Accounts"
  }
]
}
```

Creating a Record List

Once we have the list of ids, we then need to create a record list in Sugar that consists of those ids.

```
curl -s -X POST -H OAuth-Token:{access_token} -H
Cache-Control:no-cache -d '{
  "records": [
    "f16760a4-3a52-f77d-1522-5703ca28925f",
    "ec409fbb-2b22-4f32-7fa1-5703caf78dc3"
```

```
    ]  
' https://{site_url}/rest/v10/Accounts/record_list
```

Response

The data received from the server is shown below:

```
{  
  "id": "ef963176-4845-bc55-b03e-570430b4173c",  
  "assigned_user_id": "1",  
  "module_name": "Accounts",  
  "records": [  
    "f16760a4-3a52-f77d-1522-5703ca28925f",  
    "ec409fbb-2b22-4f32-7fa1-5703caf78dc3"  
  ],  
  "date_modified": "2016-04-05 21:39:19"  
}
```

Exporting Records

Once we have the record list created, we can then export the CSV file.

```
curl -i -s -X GET -H OAuth-Token:{access_token} -H  
Cache-Control:no-cache  
https://{site_url}/rest/v10/Accounts/export/ef963176-4845-bc55-b03e-57  
0430b4173c
```

More information on exporting records can be found in the
/<module>/export/:record_list_id documentation.

Response

The data received from the server is shown below:

```
HTTP/1.1 200 OK
Date: Tue, 05 Apr 2016 21:50:32
GMT Server: Apache/2.2.29 (Unix)
DAV/2 PHP/5.3.29 mod_ssl/2.2.29
OpenSSL/0.9.8zg X-Powered-By:
PHP/5.3.29 Expires:
Cache-Control: max-age=10,
private Pragma:
Content-Disposition: attachment; filename=Accounts.csv
Content-transfer-encoding: binary
Last-Modified: Tue, 05 Apr 2016 21:50:32
GMT ETag: 9b34f5d74e0298aaf7fd1f27d02e14f2
Content-Length: 1703
Connection: close
Content-Type: application/octet-stream; charset=ISO-8859-1
"Name","ID","Website","Office Phone","Alternate Phone","Fax","Billing
Street","Billing City","Billing State","Billing Postal Code","Billing
Country","Shipping Street","Shipping City","Shipping State","Shipping
Postal Code","Shipping
Country","Description","Type","Industry","Annual
Revenue","Employees","SIC Code","Ticker Symbol","Parent Account
ID","Ownership","Campaign ID","Rating","Assigned User Name","Assigned
User ID","Team ID","Teams","Team Set ID","Date Created","Date
Modified","Modified By Name","Modified By ID","Created By","Created By
ID","Deleted","test","Facebook Account","Twitter Account","Google Plus
ID","DUNS","Email","Invalid Email","Email Opt Out","Non-primary
emails"
"Arts & Crafts
Inc","ec409fbb-2b22-4f32-7fa1-5703caf78dc3","www.hrinfo.tw","(252)
```

```
456-8602", "", "", "777 West Filmore Ln", "Los
Angeles", "CA", "77076", "USA", "777 West Filmore Ln", "Los
Angeles", "CA", "77076", "USA", "", "Customer", "Hospitality", "", "", "", "", ""
, "", "", "", "Max Jensen", "seed_max_id", "West", "West, East,
Global", "dec43cb2-5273-8be2-968a-5703cadee75f", "2016-04-05
10:23", "2016-04-05
10:23", "Administrator", "1", "Administrator", "1", "0", "", "", "", "", "", "sug
ar.sugar.section@example.org", "0", "0", "dev.phone@example.biz,0,0"
"B.H. Edwards
Inc", "f16760a4-3a52-f77d-1522-5703ca28925f", "www.sectiondev.edu", "(361
) 765-0216", "", "", "111 Silicon Valley
Road", "Persistence", "CA", "29709", "USA", "111 Silicon Valley
Road", "Persistence", "CA", "29709", "USA", "", "Customer", "Apparel", "", "", "
", "", "", "", "", "", "Sally
Bronsen", "seed_sally_id", "West", "West", "West", "2016-04-05
10:23", "2016-04-05
10:23", "Administrator", "1", "Administrator", "1", "0", "", "", "", "", "", "inf
o.sugar@example.de", "0", "1", "phone.sales.section@example.tv,0,0"
```

You can also output the results directly to a csv file by ommiting the header data and output the results directly to a new CSV file.

```
curl -s -X GET -H OAuth-Token:{access_token} -H Cache-Control:no-cache
https://{site_url}/rest/v10/Accounts/export/ef963176-4845-bc55-b03e-57
0430b4173c > Output.csv
```

Last Modified: 10/14/2016 04:12pm

How to Filter a List of Records

Overview

An example in bash script demonstrating how to filter records using the v10 /<module>/filter REST POST endpoints.

Filtering Records

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
curl -X POST -H Cache-Control:no-cache -d '{
  "grant_type":"password",
  "client_id":"sugar",
  "client_secret":"",
  "username":"admin",
  "password":"password",
  "platform":"custom_api"
}' https://{site_url}/rest/v10/oauth2/token
```

More information on authenticating can be found in the [How to Authenticate and Log Out](#) example and [/oauth2/logout](#) endpoint documentation.

Filtering Records

Next we can filter the records we want to return using the /<module>/filter endpoint with a POST request.

```
curl -s -X POST -H OAuth-Token:{access_token} -H
Cache-Control:no-cache -d '{
  "filter":[
    {
      "$or":[
```

```
    {
      "name": {
        "$starts": "A"
      }
    },
    {
      "name": {
        "$starts": "b"
      }
    }
  ]
}
],
"max_num": 2,
"offset": 0,
"fields": "id",
"order_by": "date_entered",
"favorites": false,
"my_items": false
}' https://{site_url}/rest/v10/Accounts/filter
```

More information on the filter API can be found in the [/<module>/filter](#) documentation.

Note : The /<module>/filter endpoint can be called using a GET request as well, though long URL requests can have issues so the POST request is recommended.

Response

The data received from the server is shown below:

```
{
```

```
"next_offset":2,
"records":[
  {
    "id":"f16760a4-3a52-f77d-1522-5703ca28925f",
    "date_modified":"2016-04-05T10:23:28-04:00",
    "_acl":{"
      "fields":{

      }
    },
    "_module":"Accounts"
  },
  {
    "id":"ec409fbb-2b22-4f32-7fa1-5703caf78dc3",
    "date_modified":"2016-04-05T10:23:28-04:00",
    "_acl":{"
      "fields":{

      }
    },
    "_module":"Accounts"
  }
]
}
```

Last Modified: 11/27/2017 12:17pm

How to Favorite a Record

Overview

An example in bash script demonstrating how to favorite a record using the v10 `<module>/:record/favorite` REST PUT API endpoint.

Favoriting a Record

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
curl -X POST -H Cache-Control:no-cache -d '{
  "grant_type":"password",
  "client_id":"sugar",
  "client_secret":"",
  "username":"admin",
  "password":"password",
  "platform":"custom_api"
}' https://{site_url}/rest/v10/oauth2/token
```

More information on authenticating can be found in the [How to Authenticate and Log Out example and /oauth2/logout endpoint documentation](#).

Favoriting a Record

Next, we can favorite a specific record using the `/<module>/:record/favorite` endpoint.

```
curl -s -X PUT -H OAuth-Token:{access_token} -H Cache-Control:no-cache
https://{site_url}/rest/v10/Accounts/ae8b1783-404a-fcb8-1e1e-56f1cc52c
d1a/favorite
```

More information on the unfavorite API can be found in the

[/record/favorite](#) PUT documentation.

Response

The data received from the server is shown below:

```
{
  "id": "ae8b1783-404a-fcb8-1e1e-56f1cc52cd1a",
  "name": "Union Bank",
  "date_entered": "2016-03-22T17:49:50-05:00",
  "date_modified": "2016-03-30T17:44:20-05:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "modified_user_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
      "fields": [],
      "delete": "no",
      "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
    }
  },
  "created_by": "1",
  "created_by_name": "Administrator",
  "created_by_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
      "fields": [],
      "delete": "no",
      "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
    }
  }
}
```

```
},
"description": "",
"deleted": false,
"facebook": "",
"twitter": "",
"googleplus": "",
"account_type": "Customer",
"industry": "Banking",
"annual_revenue": "",
"phone_fax": "",
"billing_address_street": "67321 West Siam St.",
"billing_address_street_2": "",
"billing_address_street_3": "",
"billing_address_street_4": "",
"billing_address_city": "Ohio",
"billing_address_state": "CA",
"billing_address_postalcode": "25159",
"billing_address_country": "USA",
"rating": "",
"phone_office": "(065) 489-6104",
"phone_alternate": "",
"website": "www.qahr.edu",
"ownership": "",
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "67321 West Siam St.",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "Ohio",
"shipping_address_state": "CA",
"shipping_address_postalcode": "25159",
"shipping_address_country": "USA",
```

```
"parent_id": "",
"sic_code": "",
"duns_num": "",
"parent_name": "",
"member_of": {
  "name": "",
  "id": "",
  "_acl": {
    "fields": [],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"campaign_id": "",
"campaign_name": "",
"campaign_accounts": {
  "name": "",
  "id": "",
  "_acl": {
    "fields": [],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"following": true,
"my_favorite": true,
"tag": [],
"assigned_user_id": "seed_sarah_id",
"assigned_user_name": "Sarah Smith",
"assigned_user_link": {
  "full_name": "Sarah Smith",
  "id": "seed_sarah_id",
  "_acl": {
    "fields": [],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
}
```

```
    }
  },
  "team_count": "",
  "team_count_link": {
    "team_count": "",
    "id": "West",
    "_acl": {
      "fields": [],
      "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
  },
  "team_name": [{
    "id": "East",
    "name": "East",
    "name_2": "",
    "primary": false
  }, {
    "id": 1,
    "name": "Global",
    "name_2": "",
    "primary": false
  }, {
    "id": "West",
    "name": "West",
    "name_2": "",
    "primary": true
  }],
  "email": [{
    "email_address": "hr.support.kid@example.info",
    "invalid_email": false,
    "opt_out": false,
    "primary_address": true,
    "reply_to_address": false
  }]
```

```
}, {
  "email_address": "info.support.the@example.com",
  "invalid_email": false,
  "opt_out": false,
  "primary_address": false,
  "reply_to_address": false
}],
"email1": "hr.support.kid@example.info",
"email2": "info.support.the@example.com",
"invalid_email": false,
"email_opt_out": false,
"email_addresses_non_primary": "",
"_acl": {
  "fields": {}
},
"_module": "Accounts"
}
```

Last Modified: 11/27/2017 12:17pm

How to Manipulate File Attachments

Overview

An example in bash script demonstrating how to attach a file to a record using the v10 <module>/:record/file/:field REST POST API endpoint, then retrieve it with the GET endpoint.

Manipulating File Attachments

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
curl -X POST -H Cache-Control:no-cache -d '{
  "grant_type":"password",
  "client_id":"sugar",
  "client_secret":"",
  "username":"admin",
  "password":"password",
  "platform":"custom_api"
}' https://{site_url}/rest/v10/oauth2/token
```

More information on authenticating can be found in the [How to Authenticate and Log Out](#) example and [/oauth2/logout](#) endpoint documentation.

Submitting a File Attachment

Next, we can create a Note record using the [/<module>](#) endpoint, and then submit a File to the Note record using the [/<module>/:record/file/:field](#) endpoint.

Create Note

```
curl -s -X POST -H OAuth-Token:{access_token} -H
Cache-Control:no-cache -d '{
  "name":"Test Note"
}' https://{site_url}/rest/v10/Notes
```

Add An Attachment to the Note

```
curl -X POST -H OAuth-Token:{access_token} -H Cache-Control:no-cache
-F "filename=@/path/to/file.txt"
```

https://{site_url}/rest/v10/Notes/7b49aebd-8734-9773-8ef1-53553fa369c7
/file/filename

Response

```
{
  "filename":{
    "content-type":"text/plain",
    "content-length":13,
    "name":"file.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":"file.txt",
"parent_type":"",
"parent_id":"",
"contact_id":"",
"portal_flag":false,
"embed_flag":false,
"parent_name":"",
"contact_name":"",
"contact_phone":"",
"contact_email":"",
"account_id":"",
"opportunity_id":"",
"acase_id":"",
"lead_id":"",
"product_id":"",
"quote_id":"",
"_acl":{
  "fields":{

  }
},
"_module":"Notes"
}
```

}

Getting a File Attachment

Next, we can retrieve the file attachment stored in Sugar by utilizing the `/<module>/:record/file/:fieldGET` endpoint.

```
curl -i -s -X GET -H OAuth-Token:{access_token} -H
Cache-Control:no-cache
https://{site_url}/rest/v10/Notes/7b49aebd-8734-9773-8ef1-53553fa369c7
/file/filename
```

Response

HTTP/1.1 200 OK

Date: Wed, 12 Mar 2014 15:15:03 GMT

Server: Apache/2.2.22 (Unix) PHP/5.3.14 mod_ssl/2.2.22 OpenSSL/0.9.8o

X-Powered-By: PHP/5.3.14 ZendServer/5.0

Set-Cookie: ZDEDebuggerPresent=php,php3; path=

Expires:

Cache-Control: max-age=0, private

Pragma:

Content-Disposition: attachment; filename="file.txt"

```
X-Content-Type-Options: nosniff
ETag: d41d8cd98f00b204e9800998ecf8427e
Content-Length: 16
Connection: close
Content-Type: application/octet-stream
```

This is the file contents.

You can also output the results directly to a file by omitting the header data and output the results directly to a new file.

```
curl -s -X GET -H OAuth-Token:{access_token} -H Cache-Control:no-cache
https://{site_url}/rest/v10/Notes/7b49aebd-8734-9773-8ef1-53553fa369c7
/file/filename > file.txt
```

Last Modified: 11/27/2017 12:17pm

How to Fetch Related Records

Overview

An example in bash script demonstrating how to fetch related records using the v10 /<module>/:record/link/:link REST GET endpoints.

Fetching Related Records

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
curl -X POST -H Cache-Control:no-cache -d '{
  "grant_type":"password",
  "client_id":"sugar",
  "client_secret":"",
  "username":"admin",
  "password":"password",
  "platform":"custom_api"
}' https://{site_url}/rest/v10/oauth2/token
```

More information on authenticating can be found in the [How to Authenticate and Log Out](#) example and [/oauth2/logout](#) endpoint documentation.

Fetching Related Records

Next, we can fetch the related records we want to return using the `/<module>/:record/link/:link` endpoint with a GET request where

Element	Meaning
<code><module></code>	The parent module name
<code>:record</code>	The parent records ID
<code>link</code>	the actual word "link"
<code>:link</code>	The name of the relationship to fetch

In this example we will fetch the related Contacts for an Account :

```
curl -s -X GET -H OAuth-Token:{access_token} -H Cache-Control:no-cache
```

`https://{site_url}/rest/v10/Accounts/e8c641ca-1b8c-74c1-d08d-56fedbdd3187/link/contacts`

Response

The data received from the server is shown below:

```
{
  "next_offset":-1,
  "records":[
    {
      "my_favorite":false,
      "following":false,
      "id":"819f4149-b007-a6da-a5fa-56fedbf2de77",
      "name":"Florine Marcus",
      "date_entered":"2016-04-01T15:34:00-05:00",
      "date_modified":"2016-04-01T15:34:00-05:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "doc_owner":"","",
      "user_favorites":"","",
      "description":"","",
      "deleted":false,
      "assigned_user_id":"seed_will_id",
      "assigned_user_name":"Will Westin",
      "team_count":"","",
      "team_name":[
        {
          "id":"East",
```

```
        "name": "East",
        "name_2": "",
        "primary": true
    },
    {
        "id": "West",
        "name": "West",
        "name_2": "",
        "primary": false
    }
],
"email": [
    {
        "email_address": "support27@example.tv",
        "primary_address": true,
        "reply_to_address": false,
        "invalid_email": false,
        "opt_out": false
    },
    {
        "email_address": "support.support.kid@example.net",
        "primary_address": false,
        "reply_to_address": false,
        "invalid_email": false,
        "opt_out": true
    }
],
"email1": "support27@example.tv",
"email2": "",
"invalid_email": false,
"email_opt_out": false,
"email_addresses_non_primary": "",
"salutation": "",
```

```
"first_name":"Florine",
"last_name":"Marcus",
"full_name":"Florine Marcus",
"title":"President",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(746) 162-2314",
"phone_mobile":"(941) 088-2874",
"phone_work":"(827) 541-9614",
"phone_other":"",
"phone_fax":"",
"primary_address_street":"1715 Scott Dr",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Alabama",
"primary_address_state":"NY",
"primary_address_postalcode":"70187",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
"assistant":"",
"assistant_phone":"",
"picture":"",
"email_and_name1":"",
"lead_source":"Employee",
```

```
"account_name": "MTM Investment Bank F S B",
"account_id": "e8c641ca-1b8c-74c1-d08d-56fedbdd3187",
"dnb_principal_id": "",
"opportunity_role_fields": "",
"opportunity_role_id": "",
"opportunity_role": "",
"reports_to_id": "",
"report_to_name": "",
"birthdate": "",
"portal_name": "FlorineMarcus119",
"portal_active": true,
"portal_password": true,
"portal_password1": null,
"portal_app": "",
"preferred_language": "en_us",
"campaign_id": "",
"campaign_name": "",
"c_accept_status_fields": "",
"m_accept_status_fields": "",
"accept_status_id": "",
"accept_status_name": "",
"accept_status_calls": "",
"accept_status_meetings": "",
"sync_contact": false,
"mkto_sync": false,
"mkto_id": null,
"mkto_lead_score": null,
"_acl": {
  "fields": {

  }
},
"_module": "Contacts"
```

```
},
{
  "my_favorite":false,
  "following":false,
  "id":"527cc1a9-7984-91fe-4148-56fedbc356aa",
  "name":"Shaneka Aceto",
  "date_entered":"2016-04-01T15:34:00-05:00",
  "date_modified":"2016-04-01T15:34:00-05:00",
  "modified_user_id":"1",
  "modified_by_name":"Administrator",
  "created_by":"1",
  "created_by_name":"Administrator",
  "doc_owner":"",
  "user_favorites":"",
  "description":"",
  "deleted":false,
  "assigned_user_id":"seed_will_id",
  "assigned_user_name":"Will Westin",
  "team_count":"",
  "team_name":[
    {
      "id":"East",
      "name":"East",
      "name_2":"",
      "primary":true
    },
    {
      "id":"West",
      "name":"West",
      "name_2":"",
      "primary":false
    }
  ],
}
```

```
"email":[
  {
    "email_address":"support17@example.cn",
    "primary_address":true,
    "reply_to_address":false,
    "invalid_email":false,
    "opt_out":false
  },
  {
    "email_address":"section.sugar.the@example.tv",
    "primary_address":false,
    "reply_to_address":false,
    "invalid_email":false,
    "opt_out":true
  }
],
"email1":"support17@example.cn",
"email2":"","
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"","
"salutation":"","
"first_name":"Shaneka",
"last_name":"Aceto",
"full_name":"Shaneka Aceto",
"title":"IT Developer",
"facebook":"","
"twitter":"","
"googleplus":"","
"department":"","
"do_not_call":false,
"phone_home":"(502) 528-5151",
"phone_mobile":"(816) 719-3739",
```

```
"phone_work": "(994) 769-5855",
"phone_other": "",
"phone_fax": "",
"primary_address_street": "123 Anywhere Street",
"primary_address_street_2": "",
"primary_address_street_3": "",
"primary_address_city": "Denver",
"primary_address_state": "NY",
"primary_address_postalcode": "15128",
"primary_address_country": "USA",
"alt_address_street": "",
"alt_address_street_2": "",
"alt_address_street_3": "",
"alt_address_city": "",
"alt_address_state": "",
"alt_address_postalcode": "",
"alt_address_country": "",
"assistant": "",
"assistant_phone": "",
"picture": "",
"email_and_name1": "",
"lead_source": "Email",
"account_name": "MTM Investment Bank F S B",
"account_id": "e8c641ca-1b8c-74c1-d08d-56fedbdd3187",
"dnb_principal_id": "",
"opportunity_role_fields": "",
"opportunity_role_id": "",
"opportunity_role": "",
"reports_to_id": "",
"report_to_name": "",
"birthdate": "",
"portal_name": "ShanekaAceto151",
"portal_active": true,
```

```
"portal_password":true,
"portal_password1":null,
"portal_app":"","
"preferred_language":"en_us",
"campaign_id":"","
"campaign_name":"","
"c_accept_status_fields":"","
"m_accept_status_fields":"","
"accept_status_id":"","
"accept_status_name":"","
"accept_status_calls":"","
"accept_status_meetings":"","
"sync_contact":false,
"mkto_sync":false,
"mkto_id":null,
"mkto_lead_score":null,
"_acl":{
  "fields":{
    }
  },
  "_module":"Contacts"
},
{
  "my_favorite":false,
  "following":false,
  "id":"42d703ed-f834-f87c-967d-56fedb044051",
  "name":"Johnnie Pina",
  "date_entered":"2016-04-01T15:34:00-05:00",
  "date_modified":"2016-04-01T15:34:00-05:00",
  "modified_user_id":"1",
  "modified_by_name":"Administrator",
  "created_by":"1",
```

```
"created_by_name": "Administrator",
"doc_owner": "",
"user_favorites": "",
"description": "",
"deleted": false,
"assigned_user_id": "seed_will_id",
"assigned_user_name": "Will Westin",
"team_count": "",
"team_name": [
  {
    "id": "East",
    "name": "East",
    "name_2": "",
    "primary": true
  },
  {
    "id": "West",
    "name": "West",
    "name_2": "",
    "primary": false
  }
],
"email": [
  {
    "email_address": "sugar.support.hr@example.co.uk",
    "primary_address": true,
    "reply_to_address": false,
    "invalid_email": false,
    "opt_out": false
  },
  {
    "email_address": "support.im@example.tw",
    "primary_address": false,
```

```
        "reply_to_address":false,
        "invalid_email":false,
        "opt_out":true
    }
],
"email1":"sugar.support.hr@example.co.uk",
"email2":"","
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"","
"salutation":"","
"first_name":"Johnnie",
"last_name":"Pina",
"full_name":"Johnnie Pina",
"title":"VP Operations",
"facebook":"","
"twitter":"","
"googleplus":"","
"department":"","
"do_not_call":false,
"phone_home":"(159) 335-1423",
"phone_mobile":"(580) 140-4050",
"phone_work":"(418) 792-9611",
"phone_other":"","
"phone_fax":"","
"primary_address_street":"345 Sugar Blvd.",
"primary_address_street_2":"","
"primary_address_street_3":"","
"primary_address_city":"Denver",
"primary_address_state":"NY",
"primary_address_postalcode":"70648",
"primary_address_country":"USA",
"alt_address_street":"","
```

```
"alt_address_street_2": "",
"alt_address_street_3": "",
"alt_address_city": "",
"alt_address_state": "",
"alt_address_postalcode": "",
"alt_address_country": "",
"assistant": "",
"assistant_phone": "",
"picture": "",
"email_and_name1": "",
"lead_source": "Direct Mail",
"account_name": "MTM Investment Bank F S B",
"account_id": "e8c641ca-1b8c-74c1-d08d-56fedbdd3187",
"dnb_principal_id": "",
"opportunity_role_fields": "",
"opportunity_role_id": "",
"opportunity_role": "",
"reports_to_id": "",
"report_to_name": "",
"birthdate": "",
"portal_name": "JohnniePinal94",
"portal_active": true,
"portal_password": true,
"portal_password1": null,
"portal_app": "",
"preferred_language": "en_us",
"campaign_id": "",
"campaign_name": "",
"c_accept_status_fields": "",
"m_accept_status_fields": "",
"accept_status_id": "",
"accept_status_name": "",
"accept_status_calls": "",
```

```
    "accept_status_meetings": "",
    "sync_contact": false,
    "mkto_sync": false,
    "mkto_id": null,
    "mkto_lead_score": null,
    "_acl": {
      "fields": {

      }
    },
    "_module": "Contacts"
  }
]
}
```

Last Modified: 10/14/2016 04:12pm

How to Manipulate Records (CRUD)

Overview

The following page will provide examples in bash script demonstrating how to use the CRUD (Create, Read, Update, Delete) endpoints in the REST v10 API.

CRUD Operations

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
curl -X POST -H Cache-Control:no-cache -d '{
  "grant_type":"password",
  "client_id":"sugar",
  "client_secret":"",
  "username":"admin",
  "password":"password",
  "platform":"custom_api"
}' http://<site_url>/rest/v10/oauth2/token
```

More information on authenticating can be found in the [How to Authenticate and Log Out example](#) and [/oauth2/logout endpoint documentation](#).

Creating a Record

Next, we need to submit the record to the Sugar instance using the [/<module>](#) endpoint. In this example we are going to create an Account record with a Name of 'Test Record' and an email of 'test@sugar.com'.

```
curl -X POST -H OAuth-Token:<access_token> -H Cache-Control:no-cache
-d '{
  "name":"Test Record",
  "email1":"test@sugar.com"
}' http://<site_url>/rest/v10/Accounts
```

More information on this API endpoint can be found in the [/<module> - POST documentation](#).

Response

The data received from the server is shown below:

```
{
  "id": "ab2222df-73da-0e92-6887-5705428f4d68",
  "name": "Test Record",
  "date_entered": "2016-04-06T13:07:41-04:00",
  "date_modified": "2016-04-06T13:07:41-04:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "modified_user_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
      "fields": [

    ],
    "delete": "no",
    "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
    }
  },
  "created_by": "1",
  "created_by_name": "Administrator",
  "created_by_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
      "fields": [

    ],
    "delete": "no",
    "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
    }
  },
  "description": "",
  "deleted": false,
```

```
"facebook": "",
"twitter": "",
"googleplus": "",
"account_type": "",
"industry": "",
"annual_revenue": "",
"phone_fax": "",
"billing_address_street": "",
"billing_address_street_2": "",
"billing_address_street_3": "",
"billing_address_street_4": "",
"billing_address_city": "",
"billing_address_state": "",
"billing_address_postalcode": "",
"billing_address_country": "",
"rating": "",
"phone_office": "",
"phone_alternate": "",
"website": "",
"ownership": "",
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "",
"shipping_address_state": "",
"shipping_address_postalcode": "",
"shipping_address_country": "",
"parent_id": "",
"sic_code": "",
"duns_num": "",
```

```
"parent_name": "",
"member_of": {
  "name": "",
  "id": "",
  "_acl": {
    "fields": [

      ],
      "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
  },
"campaign_id": "",
"campaign_name": "",
"campaign_accounts": {
  "name": "",
  "id": "",
  "_acl": {
    "fields": [

      ],
      "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
  },
"following": true,
"my_favorite": false,
"tag": [

],
"assigned_user_id": "",
"assigned_user_name": "",
"assigned_user_link": {
  "full_name": "",
  "id": "",
```

```
"_acl":{
  "fields":[

  ],
  "_hash":"654d337e0e912edaa00dbb0fb3dc3c17"
},
"team_count":"","
"team_count_link":{
  "team_count":"","
  "id":"1",
  "_acl":{
    "fields":[

    ],
    "_hash":"654d337e0e912edaa00dbb0fb3dc3c17"
  },
"team_name":[
  {
    "id":1,
    "name":"Global",
    "name_2":"","
    "primary":true
  }
],
"email":[
  {
    "email_address":"test@sugar.com",
    "invalid_email":false,
    "opt_out":false,
    "primary_address":true,
    "reply_to_address":false
```

```
    }
  ],
  "email1": "test@sugar.com",
  "email2": "",
  "invalid_email": false,
  "email_opt_out": false,
  "email_addresses_non_primary": "",
  "_acl": {
    "fields": {

    }
  },
  "_module": "Accounts"
}
```

Getting a Record

Next we can get the created record from the Sugar instance using the `/<module>/:record` endpoint. In this example we are going to get an Account record by its ID, but only request the Name, Email, and Industry fields.

```
curl -H OAuth-Token:<access_token> -H Cache-Control:no-cache
http://<site_url>/rest/v10/Accounts/<record_id>?fields=name,email1,industry
```

More information on this API endpoint can be found in the `/<module>/:record - GET` documentation.

Response

The data received from the server is shown below:

```
{
  "id": "ab2222df-73da-0e92-6887-5705428f4d68",
  "name": "Test Record",
  "date_modified": "2016-04-06T15:03:21-04:00",
  "industry": "",
  "email1": "test@sugar.com",
  "_acl": {
    "fields": {

    }
  },
  "_module": "Accounts"
}
```

Updating a Record

Next we can update the record in the Sugar instance using the `/<module>/:record` endpoint, and the PUT Http method. In this example we are going to update the Account record and change it's name to "Updated Test Record".

```
curl -X PUT -H OAuth-Token:<access_token> -H Cache-Control:no-cache -d
'{
  "name": "Updated Record"
}' http://<site_url>/rest/v10/Accounts/<record_id>
```

More information on this API endpoint can be found in the `/<module>/:record - PUT` documentation.

Response

The data received from the server is shown below:

```
{
  "id": "ab2222df-73da-0e92-6887-5705428f4d68",
  "name": "Updated Test Record",
  "date_entered": "2016-04-06T15:03:21-04:00",
  "date_modified": "2016-04-06T15:03:22-04:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "modified_user_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
      "fields": [

    ],
    "delete": "no",
    "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
  }
},
  "created_by": "1",
  "created_by_name": "Administrator",
  "created_by_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
      "fields": [

    ],
    "delete": "no",
    "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
  }
},
}
```

```
"description": "",
"deleted": false,
"facebook": "",
"twitter": "",
"googleplus": "",
"account_type": "",
"industry": "",
"annual_revenue": "",
"phone_fax": "",
"billing_address_street": "",
"billing_address_street_2": "",
"billing_address_street_3": "",
"billing_address_street_4": "",
"billing_address_city": "",
"billing_address_state": "",
"billing_address_postalcode": "",
"billing_address_country": "",
"rating": "",
"phone_office": "",
"phone_alternate": "",
"website": "",
"ownership": "",
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "",
"shipping_address_state": "",
"shipping_address_postalcode": "",
"shipping_address_country": "",
"parent_id": "",
```

```
"sic_code": "",
"duns_num": "",
"parent_name": "",
"member_of": {
  "name": "",
  "id": "",
  "_acl": {
    "fields": [

    ],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"campaign_id": "",
"campaign_name": "",
"campaign_accounts": {
  "name": "",
  "id": "",
  "_acl": {
    "fields": [

    ],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"following": true,
"my_favorite": false,
"tag": [

],
"assigned_user_id": "",
"assigned_user_name": "",
"assigned_user_link": {
```

```
"full_name": "",
"id": "",
"_acl": {
  "fields": [

  ],
  "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
}
},
"team_count": "",
"team_count_link": {
  "team_count": "",
  "id": "1",
  "_acl": {
    "fields": [

    ],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"team_name": [
  {
    "id": 1,
    "name": "Global",
    "name_2": "",
    "primary": true
  }
],
"email": [
  {
    "email_address": "test@sugar.com",
    "invalid_email": false,
    "opt_out": false,
```

```
        "primary_address":true,
        "reply_to_address":false
    }
],
"email1":"test@sugar.com",
"email2":"",
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"",
"_acl":{
    "fields":{

    }
},
"_module":"Accounts"
}
```

Deleting a Record

Next, we can delete the record from the Sugar instance using the `/<module>/:record` endpoint, by using the DELETE Http Method.

```
curl -X DELETE -H OAuth-Token:<access_token> -H Cache-Control:no-cache
http://<site_url>/rest/v10/Accounts/<record_id>
```

More information on this API endpoint can be found in the `/<module>/:record - DELETE` documentation.

Response

The data received from the server is shown below:

```
{
  "id": "ab2222df-73da-0e92-6887-5705428f4d68"
}
```

Last Modified: 10/14/2016 04:12pm

How to Follow a Record

Overview

An example in bash script demonstrating how to follow a record using the v10 /<module>/record/subscribe REST POST endpoint.

Following a Record

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
curl -X POST -H Cache-Control:no-cache -d '{
  "grant_type": "password",
  "client_id": "sugar",
  "client_secret": "",
  "username": "admin",
  "password": "password",
  "platform": "custom_api"
}' https://{site_url}/rest/v10/oauth2/token
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Following a Record

Next, we can follow a specific record using the `/<module>/:record/subscribe` endpoint.

```
curl -s -X POST -H OAuth-Token:{access_token} -H
Cache-Control:no-cache
https://{site_url}/rest/v10/Accounts/e3a59dcd-ff5e-8a78-cd7f-570811366
792/subscribe
```

More information on the subscribe API can be found in the [/<module>/:record/subscribe POST](#) documentation.

Response

The data received from the server is shown below:

```
"58f96315-9e75-6562-42e9-5705917d2cdc"
```

Last Modified: 11/27/2017 12:17pm

How to Unfavorite a Record

Overview

An example in bash script demonstrating how to unfavorite a record using the `v10 /<module>/:record/unfavorite` REST PUT endpoint.

Unfavoriting a Record

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
curl -X POST -H Cache-Control:no-cache -d '{
  "grant_type":"password",
  "client_id":"sugar",
  "client_secret":"",
  "username":"admin",
  "password":"password",
  "platform":"custom_api"
}' https://{site_url}/rest/v10/oauth2/token
```

More information on authenticating can be found in the [How to Authenticate and Log Out](#) example and [/oauth2/logout](#) endpoint documentation.

Unfavoriting a Record

Next, we can unfavorite a specific record using the [/<module>/:record/unfavorite](#) endpoint.

```
curl -s -X PUT -H OAuth-Token:{access_token} -H Cache-Control:no-cache
https://{site_url}/rest/v10/Accounts/ae8b1783-404a-fcb8-1e1e-56f1cc52c
d1a/unfavorite
```

More information on the unfavorite API can be found in the [/<module>/:record/unfavorite PUT](#) documentation.

Response

The data received from the server is shown below:

```
{
  "id": "ae8b1783-404a-fcb8-1e1e-56f1cc52cd1a",
  "name": "Union Bank",
  "date_entered": "2016-03-22T17:49:50-05:00",
  "date_modified": "2016-03-30T17:44:20-05:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "modified_user_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
      "fields": [],
      "delete": "no",
      "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
    }
  },
  "created_by": "1",
  "created_by_name": "Administrator",
  "created_by_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
      "fields": [],
      "delete": "no",
      "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
    }
  },
  "description": "",
  "deleted": false,
  "facebook": "",
  "twitter": ""
}
```

```
"googleplus": "",
"account_type": "Customer",
"industry": "Banking",
"annual_revenue": "",
"phone_fax": "",
"billing_address_street": "67321 West Siam St.",
"billing_address_street_2": "",
"billing_address_street_3": "",
"billing_address_street_4": "",
"billing_address_city": "Ohio",
"billing_address_state": "CA",
"billing_address_postalcode": "25159",
"billing_address_country": "USA",
"rating": "",
"phone_office": "(065) 489-6104",
"phone_alternate": "",
"website": "www.qahr.edu",
"ownership": "",
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "67321 West Siam St.",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "Ohio",
"shipping_address_state": "CA",
"shipping_address_postalcode": "25159",
"shipping_address_country": "USA",
"parent_id": "",
"sic_code": "",
"duns_num": "",
"parent_name": "",
"member_of": {
```

```
"name": "",
"id": "",
"_acl": {
  "fields": [],
  "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
}
},
"campaign_id": "",
"campaign_name": "",
"campaign_accounts": {
  "name": "",
  "id": "",
  "_acl": {
    "fields": [],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"following": true,
"my_favorite": false,
"tag": [],
"assigned_user_id": "seed_sarah_id",
"assigned_user_name": "Sarah Smith",
"assigned_user_link": {
  "full_name": "Sarah Smith",
  "id": "seed_sarah_id",
  "_acl": {
    "fields": [],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"team_count": "",
"team_count_link": {
  "team_count": "",
```

```
    "id": "West",
    "_acl": {
      "fields": [],
      "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
  },
  "team_name": [{
    "id": "East",
    "name": "East",
    "name_2": "",
    "primary": false
  }, {
    "id": 1,
    "name": "Global",
    "name_2": "",
    "primary": false
  }, {
    "id": "West",
    "name": "West",
    "name_2": "",
    "primary": true
  }
  ],
  "email": [{
    "email_address": "hr.support.kid@example.info",
    "invalid_email": false,
    "opt_out": false,
    "primary_address": true,
    "reply_to_address": false
  }, {
    "email_address": "info.support.the@example.com",
    "invalid_email": false,
    "opt_out": false,
    "primary_address": false,
```

```
    "reply_to_address": false
  }],
  "email1": "hr.support.kid@example.info",
  "email2": "info.support.the@example.com",
  "invalid_email": false,
  "email_opt_out": false,
  "email_addresses_non_primary": "",
  "_acl": {
    "fields": {}
  },
  "_module": "Accounts"
}
```

Last Modified: 11/27/2017 12:17pm

How to Unfollow a Record

Overview

An example in bash script demonstrating how to unfollow a record using the v10 /<module>/:record/unsubscribe REST DELETE endpoint.

Unfollowing a Record

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
curl -X POST -H Cache-Control:no-cache -d '{
  "grant_type": "password",
```

```
"client_id": "sugar",
"client_secret": "",
"username": "admin",
"password": "password",
"platform": "custom_api"
}' https://{site_url}/rest/v10/oauth2/token
```

More information on authenticating can be found in the [How to Authenticate and Log Out](#) example and [/oauth2/logout](#) endpoint documentation.

Unfollowing a Record

Next, we can unfollow a specific record using the `/<module>/:record/unsubscribe` endpoint.

```
curl -s -X DELETE -H OAuth-Token:{access_token} -H
Cache-Control:no-cache
https://{site_url}/rest/v10/Accounts/e7b0d745-0a3e-c896-5358-570811378
6d7/unsubscribe
```

More information on the unsubscribe API can be found in the [/:record/unsubscribe DELETE](#) documentation.

Response

The data received from the server is shown below:

```
true
```

Last Modified: 10/14/2016 04:12pm

How to Get the Most Active Users

Overview

An example in bash script demonstrating how to fetch the most active users for meetings, calls, inbound emails, and outbound emails using the v10 /mostactiveusers REST GET endpoint.

Get Most Active Users

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
curl -X POST -H Cache-Control:no-cache -d '{
  "grant_type":"password",
  "client_id":"sugar",
  "client_secret":"",
  "username":"admin",
  "password":"password",
  "platform":"custom_api"
}' https://{site_url}/rest/v10/oauth2/token
```

More information on authenticating can be found in the [How to Authenticate and Log Out](#) example and [/oauth2/logout](#) endpoint documentation.

Active Users

Next, we can retrieve the most active users using the [/mostactiveusers](#) endpoint.

```
curl -s -X GET -H OAuth-Token:{access_token} -H Cache-Control:no-cache
https://{site_url}/rest/v10/mostactiveusers?days=30
```

More information on the mostactiveusers API can be found in the [/mostactiveusers](#)

Response

The data received from the server is shown below:

```
{
  "meetings": {
    "user_id": "seed_max_id",
    "count": "21",
    "first_name": "Max",
    "last_name": "Jensen"
  },
  "calls": {
    "user_id": "seed_chris_id",
    "count": "4",
    "first_name": "Chris",
    "last_name": "Olliver"
  },
  "inbound_emails": {
    "user_id": "seed_chris_id",
    "count": "23",
    "first_name": "Chris",
    "last_name": "Olliver"
  },
  "outbound_emails": {
    "user_id": "seed_sarah_id",
    "count": "20",
    "first_name": "Sarah",
```

```
        "last_name": "Smith"
    }
}
```

Last Modified: 10/14/2016 04:12pm

How to Authenticate and Log Out

Overview

An example in bash script on how to authenticate and logout of the v10 REST API using the /oauth2/token and /oauth2/logout POST endpoints.

Authenticating

The example below demonstrates how to authenticate to the REST v10 API. It is important to note that the oauth2 token arguments takes in several parameters that you should be aware of. The platform argument tends to cause confusion in that it is used to authenticate a user to a specific platform. Since Sugar only allows 1 user in the system at a time per platform, authenticating an integration script with a platform type of "base" will logout any current users in the system using those credentials. To work around this, your custom scripts should have a new platform type specified such as "custom_api" or any other static text you prefer. The client_id and client_secret parameters can be used for additional security based on client types. You can create your own client type in Admin > OAuth Keys. More information can be found in the /oauth2/token documentation. An example script is shown below:

```
curl -X POST -H Cache-Control:no-cache -d '{
    "grant_type": "password",
```

```
"client_id": "sugar",
"client_secret": "",
"username": "admin",
"password": "password",
"platform": "custom_api"
}' https://{site_url}/rest/v10/oauth2/token
```

Response

The data received from the server is shown below:

```
{
  "access_token": "c6d495c9-bb25-81d2-5f81-533ef6479f9b",
  "expires_in": 3600,
  "token_type": "bearer",
  "scope": null,
  "refresh_token": "cbc40e67-12bc-4b56-a1d9-533ef62f2601",
  "refresh_expires_in": 1209600,
  "download_token": "cc5d1a9f-6627-3349-96e5-533ef6b1a493"
}
```

Logout

To log out of a session, a request can be made to the `/oauth2/logout` POST endpoint. More information can be found in the `/oauth2/logout` documentation. An example extending off of the above authentication example is shown below:

```
curl -s -X POST -H OAuth-Token:<access_token> -H
Cache-Control:no-cache https://{site_url}/rest/v10/oauth2/logout
```

Response

The data received from the server is shown below:

```
{
  "success": true
}
```

Last Modified: 10/14/2016 04:12pm

How to Ping

Overview

An example in bash script demonstrating how to ping using the REST v10 /ping GET endpoint.

Pinging

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
curl -X POST -H Cache-Control:no-cache -d '{
  "grant_type": "password",
  "client_id": "sugar",
  "client_secret": "",
  "username": "admin",
  "password": "password",
  "platform": "custom_api"
}
```

```
}' https://{site_url}/rest/v10/oauth2/token
```

More information on authenticating can be found in the [How to Authenticate and Log Out example](#) and [/oauth2/logout endpoint documentation](#).

Pinging

Once we have authenticated, we can now ping the system as shown below.

```
curl -s -X GET -H OAuth-Token:{access_token} -H Cache-Control:no-cache  
https://{site_url}/rest/v10/ping
```

More information on pinging can be found in the [/ping](#) documentation.

Response

```
"pong"
```

Last Modified: 10/14/2016 04:12pm

How to Fetch the Current Users Time

Overview

An example in bash script demonstrating how to fetch the current users time with the v10 [/ping/whattimeisit](#) REST GET endpoint.

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
curl -X POST -H Cache-Control:no-cache -d '{
  "grant_type":"password",
  "client_id":"sugar",
  "client_secret":"",
  "username":"admin",
  "password":"password",
  "platform":"custom_api"
}' https://{site_url}/rest/v10/oauth2/token
```

More information on authenticating can be found in the [How to Authenticate and Log Out example](#) and [/oauth2/logout endpoint documentation](#).

Getting the Current Time and Date for the User

Next, we can get the current time and date using the [/ping/whattimeisit](#) endpoint.

```
curl -s -X GET -H OAuth-Token:{access_token} -H Cache-Control:no-cache
https://{site_url}/rest/v10/ping/whattimeisit
```

Response

```
"2014-04-08T14:59:13-04:00"
```

Last Modified: 10/14/2016 04:12pm

How to Fetch Recently Viewed Records

Overview

An example in bash script demonstrating how to retrieve recently viewed records using the v10 /recent REST GET endpoint.

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
curl -X POST -H Cache-Control:no-cache -d '{
  "grant_type":"password",
  "client_id":"sugar",
  "client_secret":"",
  "username":"admin",
  "password":"password",
  "platform":"custom_api"
}' https://{site_url}/rest/v10/oauth2/token
```

More information on authenticating can be found in the [How to Authenticate and Log Out](#) example and [/oauth2/logout](#) endpoint documentation.

Recently Viewed Records

Next we will need to identify the records we want to see using the /recent endpoint. In this case, we are going to request Accounts, Contacts and Leads.

```
curl -s -X GET -H OAuth-Token:{access_token} -H Cache-Control:no-cache
https://{site_url}/rest/v10/recent?module_list=Accounts%2CContacts%2CL
eads
```

More information on the search API can be found in the [/recent](#) documentation.

Response

The data received from the server is shown below:

```
{
  "next_offset":-1,
  "records":[
    {
      "my_favorite":false,
      "following":false,
      "id":"f3951f4d-2d17-7939-c5ec-56fedbb9e92f",
      "name":"Talia Knupp",
      "date_entered":"2016-04-01T15:34:00-05:00",
      "date_modified":"2016-04-06T10:33:24-05:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "doc_owner":"","
      "user_favorites":"","
      "description":"","
      "deleted":false,
      "assigned_user_id":"seed_will_id",
      "assigned_user_name":"Will Westin",
      "team_count":"","
      "team_name":[
        {
          "id":"East",
          "name":"East",
          "name_2":"","
          "primary":true
        },
      ],
    }
  ]
}
```

```
    {
      "id": "West",
      "name": "West",
      "name_2": "",
      "primary": false
    }
  ],
  "email": [
    {
      "email_address": "jsmith@sugar.com",
      "invalid_email": false,
      "opt_out": false,
      "primary_address": true,
      "reply_to_address": false
    },
    {
      "email_address": "cjsmith@sugar.com",
      "invalid_email": false,
      "opt_out": false,
      "primary_address": false,
      "reply_to_address": false
    }
  ],
  "email1": "jsmith@sugar.com",
  "email2": "cjsmith@sugar.com",
  "invalid_email": false,
  "email_opt_out": false,
  "email_addresses_non_primary": "",
  "salutation": "",
  "first_name": "Talía",
  "last_name": "Knupp",
  "full_name": "Talía Knupp",
  "title": "Senior Product Manager",
```

```
"facebook": "",
"twitter": "",
"googleplus": "",
"department": "",
"do_not_call": false,
"phone_home": "(963) 741-3689",
"phone_mobile": "(600) 831-9872",
"phone_work": "(680) 991-2837",
"phone_other": "",
"phone_fax": "",
"primary_address_street": "111 Silicon Valley Road",
"primary_address_street_2": "",
"primary_address_street_3": "",
"primary_address_city": "Sunnyvale",
"primary_address_state": "NY",
"primary_address_postalcode": "99452",
"primary_address_country": "USA",
"alt_address_street": "",
"alt_address_street_2": "",
"alt_address_street_3": "",
"alt_address_city": "",
"alt_address_state": "",
"alt_address_postalcode": "",
"alt_address_country": "",
"assistant": "",
"assistant_phone": "",
"picture": "",
"converted": false,
"referred_by": "",
"lead_source": "Word of mouth",
"lead_source_description": "",
"status": "In Process",
"status_description": "",
```

```
"reports_to_id":"","  
"report_to_name":"","  
"dnb_principal_id":"","  
"account_name":"First National S\B",  
"account_description":"","  
"contact_id":"","  
"contact_name":"","  
"account_id":"","  
"opportunity_id":"","  
"opportunity_name":"","  
"opportunity_amount":"","  
"campaign_id":"","  
"campaign_name":"","  
"c_accept_status_fields":"","  
"m_accept_status_fields":"","  
"accept_status_id":"","  
"accept_status_name":"","  
"accept_status_calls":"","  
"accept_status_meetings":"","  
"webtolead_email1":[  
  {  
    "email_address":"jsmith@sugar.com",  
    "invalid_email":false,  
    "opt_out":false,  
    "primary_address":true,  
    "reply_to_address":false  
  },  
  {  
    "email_address":"cjsmith@sugar.com",  
    "invalid_email":false,  
    "opt_out":false,  
    "primary_address":false,  
    "reply_to_address":false  
  }  
]
```

```
    }
  ],
  "webtolead_email2": [
    {
      "email_address": "jsmith@sugar.com",
      "invalid_email": false,
      "opt_out": false,
      "primary_address": true,
      "reply_to_address": false
    },
    {
      "email_address": "cjsmith@sugar.com",
      "invalid_email": false,
      "opt_out": false,
      "primary_address": false,
      "reply_to_address": false
    }
  ],
  "webtolead_email_opt_out": "",
  "webtolead_invalid_email": "",
  "birthdate": "",
  "portal_name": "",
  "portal_app": "",
  "website": "",
  "preferred_language": "",
  "mkto_sync": false,
  "mkto_id": null,
  "mkto_lead_score": null,
  "fruits_c": "Apples",
  "_acl": {
    "fields": {

```

```
    },
    "_module": "Leads",
    "_last_viewed_date": "2016-04-06T10:33:24-05:00"
  },
  {
    "my_favorite": false,
    "following": false,
    "id": "e8c641ca-1b8c-74c1-d08d-56fedbdd3187",
    "name": "MTM Investment Bank F S B",
    "date_entered": "2016-04-01T15:34:00-05:00",
    "date_modified": "2016-04-06T10:16:52-05:00",
    "modified_user_id": "1",
    "modified_by_name": "Administrator",
    "created_by": "1",
    "created_by_name": "Administrator",
    "doc_owner": "",
    "user_favorites": "",
    "description": "",
    "deleted": false,
    "assigned_user_id": "seed_will_id",
    "assigned_user_name": "Will Westin",
    "team_count": "",
    "team_name": [
      {
        "id": "East",
        "name": "East",
        "name_2": "",
        "primary": true
      },
      {
        "id": "West",
        "name": "West",
        "name_2": ""
      }
    ]
  }
}
```

```
        "primary":false
    }
],
"email":[
    {
        "email_address":"jsmith@sugar.com",
        "invalid_email":false,
        "opt_out":false,
        "primary_address":true,
        "reply_to_address":false
    },
    {
        "email_address":"the60@example.us",
        "invalid_email":false,
        "opt_out":false,
        "primary_address":false,
        "reply_to_address":false
    }
],
"email1":"jsmith@sugar.com",
"email2":"the60@example.us",
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"","
"facebook":"","
"twitter":"","
"googleplus":"","
"account_type":"Customer",
"industry":"a",
"annual_revenue":"","
"phone_fax":"","
"billing_address_street":"67321 West Siam St.",
"billing_address_street_2":"","
```

```
"billing_address_street_3": "",
"billing_address_street_4": "",
"billing_address_city": "Alabama",
"billing_address_state": "NY",
"billing_address_postalcode": "52272",
"billing_address_country": "USA",
"rating": "",
"phone_office": "(012) 704-8075",
"phone_alternate": "",
"website": "www.salesqa.it",
"ownership": "",
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "67321 West Siam St.",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "Alabama",
"shipping_address_state": "NY",
"shipping_address_postalcode": "52272",
"shipping_address_country": "USA",
"parent_id": "",
"sic_code": "",
"duns_num": "",
"parent_name": "",
"campaign_id": "",
"campaign_name": "",
"_acl": {
  "fields": {

  }
},
"_module": "Accounts",
```

```
"_last_viewed_date": "2016-04-06T10:16:52-05:00"
},
{
  "my_favorite": false,
  "following": false,
  "id": "f31b2f00-468c-3d35-1e88-56fedbd3921d",
  "name": "Kaycee Gibney",
  "date_entered": "2016-04-01T15:34:00-05:00",
  "date_modified": "2016-04-06T10:16:24-05:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
  "doc_owner": "",
  "user_favorites": "",
  "description": "",
  "deleted": false,
  "assigned_user_id": "seed_jim_id",
  "assigned_user_name": "Jim Brennan",
  "team_count": "",
  "team_name": [
    {
      "id": "East",
      "name": "East",
      "name_2": "",
      "primary": true
    }
  ],
  "email": [
    {
      "email_address": "jsmith@sugar.com",
      "invalid_email": false,
      "opt_out": false,

```

```
        "primary_address":true,
        "reply_to_address":false
    },
    {
        "email_address":"sales.kid.dev@example.info",
        "invalid_email":false,
        "opt_out":true,
        "primary_address":false,
        "reply_to_address":false
    }
],
"email1":"jsmith@sugar.com",
"email2":"sales.kid.dev@example.info",
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"","",
"salutation":"","",
"first_name":"Kaycee",
"last_name":"Gibney",
"full_name":"Kaycee Gibney",
"title":"Mgr Operations",
"facebook":"","",
"twitter":"","",
"googleplus":"","",
"department":"","",
"do_not_call":false,
"phone_home":"(599) 165-2396",
"phone_mobile":"(215) 591-9574",
"phone_work":"(771) 945-3648",
"phone_other":"","",
"phone_fax":"","",
"primary_address_street":"321 University Ave.",
"primary_address_street_2":"","",
```

```
"primary_address_street_3": "",
"primary_address_city": "Santa Monica",
"primary_address_state": "NY",
"primary_address_postalcode": "96154",
"primary_address_country": "USA",
"alt_address_street": "",
"alt_address_street_2": "",
"alt_address_street_3": "",
"alt_address_city": "",
"alt_address_state": "",
"alt_address_postalcode": "",
"alt_address_country": "",
"assistant": "",
"assistant_phone": "",
"picture": "",
"email_and_name1": "",
"lead_source": "Existing Customer",
"account_name": "Tracker Com LP",
"account_id": "72ad6f00-e345-1cab-b370-56fedbd23deb",
"dnb_principal_id": "",
"opportunity_role_fields": "",
"opportunity_role_id": "",
"opportunity_role": "",
"reports_to_id": "",
"report_to_name": "",
"birthdate": "",
"portal_name": "KayceeGibney33",
"portal_active": true,
"portal_password": true,
"portal_password1": null,
"portal_app": "",
"preferred_language": "en_us",
"campaign_id": "",
```

```
    "campaign_name": "",
    "c_accept_status_fields": "",
    "m_accept_status_fields": "",
    "accept_status_id": "",
    "accept_status_name": "",
    "accept_status_calls": "",
    "accept_status_meetings": "",
    "sync_contact": false,
    "mkto_sync": false,
    "mkto_id": null,
    "mkto_lead_score": null,
    "_acl": {
      "fields": {

      }
    },
    "_module": "Contacts",
    "_last_viewed_date": "2016-04-06T10:16:24-05:00"
  }
]
}
```

There are 3 records shown above, the `_module` field tells you if the record is from Accounts, Contacts or Leads. The `_last_viewed_date` tells you when the record was last seen.

Last Modified: 10/14/2016 04:12pm

How to Use the Global Search

Overview

An example in bash script demonstrating how to globally search for records using the REST v10 /search GET endpoint.

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
curl -X POST -H Cache-Control:no-cache -d '{
  "grant_type":"password",
  "client_id":"sugar",
  "client_secret":"",
  "username":"admin",
  "password":"password",
  "platform":"custom_api"
}' https://{site_url}/rest/v10/oauth2/token
```

More information on authenticating can be found in the [How to Authenticate and Log Out example](#) and [/oauth2/logout endpoint documentation](#).

Searching Records

Next we will need to identify the records we want to see using the /search endpoint. In this case, we are going to search for all records that have the email address of 'jsmith@sugar.com'. In this example, there are 3 records, an Account, a Contact and a Lead.

```
curl -s -X GET -H OAuth-Token:{access_token} -H Cache-Control:no-cache
https://{site_url}/rest/v10/search?
q=jsmith@sugar.com&
max_num=3&
offset=0&
```

```
fields=&
order_by=&
favorites=false&
my_items=false
```

More information on the search API can be found in the [/search](#) documentation.

Response

The data received from the server is shown below:

```
{
  "next_offset":-1,
  "records":[
    {
      "my_favorite":false,
      "following":false,
      "id":"f31b2f00-468c-3d35-1e88-56fedbd3921d",
      "name":"Kaycee Gibney",
      "date_entered":"2016-04-01T20:34:00+00:00",
      "date_modified":"2016-04-06T15:16:24+00:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "doc_owner":"","
      "user_favorites":"","
      "description":"","
      "deleted":false,
      "assigned_user_id":"seed_jim_id",
      "assigned_user_name":"Jim Brennan",
      "team_count":"","
```

```
"team_name":[
  {
    "id":"East",
    "name":"East",
    "name_2":"",
    "primary":true
  }
],
"email":[
  {
    "email_address":"jsmith@sugar.com",
    "invalid_email":false,
    "opt_out":false,
    "primary_address":true,
    "reply_to_address":false
  },
  {
    "email_address":"sales.kid.dev@example.info",
    "invalid_email":false,
    "opt_out":true,
    "primary_address":false,
    "reply_to_address":false
  }
],
"email1":"jsmith@sugar.com",
"email2":"sales.kid.dev@example.info",
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"",
"salutation":"",
"first_name":"Kaycee",
"last_name":"Gibney",
"full_name":"Kaycee Gibney",
```

```
"title":"Mgr Operations",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(599) 165-2396",
"phone_mobile":"(215) 591-9574",
"phone_work":"(771) 945-3648",
"phone_other":"",
"phone_fax":"",
"primary_address_street":"321 University Ave.",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Santa Monica",
"primary_address_state":"NY",
"primary_address_postalcode":"96154",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
"assistant":"",
"assistant_phone":"",
"picture":"",
"email_and_name1":"",
"lead_source":"Existing Customer",
"account_name":"Tracker Com LP",
"account_id":"72ad6f00-e345-1cab-b370-56fedbd23deb",
"dnb_principal_id":"",
```

```
"opportunity_role_fields":"","  
"opportunity_role_id":"","  
"opportunity_role":"","  
"reports_to_id":"","  
"report_to_name":"","  
"birthdate":"","  
"portal_name":"KayceeGibney33",  
"portal_active":true,  
"portal_password":true,  
"portal_password1":null,  
"portal_app":"","  
"preferred_language":"en_us",  
"campaign_id":"","  
"campaign_name":"","  
"c_accept_status_fields":"","  
"m_accept_status_fields":"","  
"accept_status_id":"","  
"accept_status_name":"","  
"accept_status_calls":"","  
"accept_status_meetings":"","  
"sync_contact":false,  
"mkto_sync":false,  
"mkto_id":null,  
"mkto_lead_score":null,  
"_acl":{  
  "fields":{  
  
  }  
},  
"_module":"Contacts",  
"_search":{  
  "score":0.70710677,  
  "highlighted":{
```

```
    "email1":{
"text": "\u003Cstrong\u003Ejsmith@sugar.com\u003C\/strong\u003E",
      "module": "Contacts",
      "label": "LBL_EMAIL_ADDRESS"
    }
  }
},
{
  "my_favorite": false,
  "following": false,
  "id": "e8c641ca-1b8c-74c1-d08d-56fedbdd3187",
  "name": "MTM Investment Bank F S B",
  "date_entered": "2016-04-01T20:34:00+00:00",
  "date_modified": "2016-04-06T15:16:52+00:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
  "doc_owner": "",
  "user_favorites": "",
  "description": "",
  "deleted": false,
  "assigned_user_id": "seed_will_id",
  "assigned_user_name": "Will Westin",
  "team_count": "",
  "team_name": [
    {
      "id": "East",
      "name": "East",
      "name_2": "",
      "primary": true
    }
  ]
}
```

```
    },
    {
      "id": "West",
      "name": "West",
      "name_2": "",
      "primary": false
    }
  ],
  "email": [
    {
      "email_address": "jsmith@sugar.com",
      "invalid_email": false,
      "opt_out": false,
      "primary_address": true,
      "reply_to_address": false
    },
    {
      "email_address": "the60@example.us",
      "invalid_email": false,
      "opt_out": false,
      "primary_address": false,
      "reply_to_address": false
    }
  ],
  "email1": "jsmith@sugar.com",
  "email2": "the60@example.us",
  "invalid_email": false,
  "email_opt_out": false,
  "email_addresses_non_primary": "",
  "facebook": "",
  "twitter": "",
  "googleplus": "",
  "account_type": "Customer",
```

```
"industry": "a",
"annual_revenue": "",
"phone_fax": "",
"billing_address_street": "67321 West Siam St.",
"billing_address_street_2": "",
"billing_address_street_3": "",
"billing_address_street_4": "",
"billing_address_city": "Alabama",
"billing_address_state": "NY",
"billing_address_postalcode": "52272",
"billing_address_country": "USA",
"rating": "",
"phone_office": "(012) 704-8075",
"phone_alternate": "",
"website": "www.salesqa.it",
"ownership": "",
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "67321 West Siam St.",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "Alabama",
"shipping_address_state": "NY",
"shipping_address_postalcode": "52272",
"shipping_address_country": "USA",
"parent_id": "",
"sic_code": "",
"duns_num": "",
"parent_name": "",
"campaign_id": "",
"campaign_name": "",
"_acl": {
```

```
    "fields":{
      }
    },
    "_module": "Accounts",
    "_search": {
      "score": 0.70710677,
      "highlighted": {
        "email1": {
"text": "\u003Cstrong\u003Ejsmith@sugar.com\u003C\/strong\u003E",
          "module": "Accounts",
          "label": "LBL_EMAIL_ADDRESS"
        }
      }
    }
  },
  {
    "my_favorite": false,
    "following": false,
    "id": "f3951f4d-2d17-7939-c5ec-56fedbb9e92f",
    "name": "Talia Knupp",
    "date_entered": "2016-04-01T20:34:00+00:00",
    "date_modified": "2016-04-06T15:33:24+00:00",
    "modified_user_id": "1",
    "modified_by_name": "Administrator",
    "created_by": "1",
    "created_by_name": "Administrator",
    "doc_owner": "",
    "user_favorites": "",
    "description": "",
    "deleted": false,
    "assigned_user_id": "seed_will_id",
```

```
"assigned_user_name": "Will Westin",
"team_count": "",
"team_name": [
  {
    "id": "East",
    "name": "East",
    "name_2": "",
    "primary": true
  },
  {
    "id": "West",
    "name": "West",
    "name_2": "",
    "primary": false
  }
],
"email": [
  {
    "email_address": "jsmith@sugar.com",
    "invalid_email": false,
    "opt_out": false,
    "primary_address": true,
    "reply_to_address": false
  },
  {
    "email_address": "cjsmith@sugar.com",
    "invalid_email": false,
    "opt_out": false,
    "primary_address": false,
    "reply_to_address": false
  }
],
"email1": "jsmith@sugar.com",
```

```
"email2":"cjsmith@sugar.com",
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"","
"salutation":"","
"first_name":"Talial",
"last_name":"Knupp",
"full_name":"Talial Knupp",
"title":"Senior Product Manager",
"facebook":"","
"twitter":"","
"googleplus":"","
"department":"","
"do_not_call":false,
"phone_home":"(963) 741-3689",
"phone_mobile":"(600) 831-9872",
"phone_work":"(680) 991-2837",
"phone_other":"","
"phone_fax":"","
"primary_address_street":"111 Silicon Valley Road",
"primary_address_street_2":"","
"primary_address_street_3":"","
"primary_address_city":"Sunnyvale",
"primary_address_state":"NY",
"primary_address_postalcode":"99452",
"primary_address_country":"USA",
"alt_address_street":"","
"alt_address_street_2":"","
"alt_address_street_3":"","
"alt_address_city":"","
"alt_address_state":"","
"alt_address_postalcode":"","
"alt_address_country":"","
```

```
"assistant": "",
"assistant_phone": "",
"picture": "",
"converted": false,
"referred_by": "",
"lead_source": "Word of mouth",
"lead_source_description": "",
"status": "In Process",
"status_description": "",
"reports_to_id": "",
"report_to_name": "",
"dnb_principal_id": "",
"account_name": "First National S\B",
"account_description": "",
"contact_id": "",
"contact_name": "",
"account_id": "",
"opportunity_id": "",
"opportunity_name": "",
"opportunity_amount": "",
"campaign_id": "",
"campaign_name": "",
"c_accept_status_fields": "",
"m_accept_status_fields": "",
"accept_status_id": "",
"accept_status_name": "",
"accept_status_calls": "",
"accept_status_meetings": "",
"webtolead_email1": [
  {
    "email_address": "jsmith@sugar.com",
    "invalid_email": false,
    "opt_out": false,
```

```
        "primary_address":true,
        "reply_to_address":false
    },
    {
        "email_address":"cjsmith@sugar.com",
        "invalid_email":false,
        "opt_out":false,
        "primary_address":false,
        "reply_to_address":false
    }
],
"webtolead_email2":[
    {
        "email_address":"jsmith@sugar.com",
        "invalid_email":false,
        "opt_out":false,
        "primary_address":true,
        "reply_to_address":false
    },
    {
        "email_address":"cjsmith@sugar.com",
        "invalid_email":false,
        "opt_out":false,
        "primary_address":false,
        "reply_to_address":false
    }
],
"webtolead_email_opt_out":"","
"webtolead_invalid_email":"","
"birthdate":"","
"portal_name":"","
"portal_app":"","
"website":"","
```

```

    "preferred_language": "",
    "mkto_sync": false,
    "mkto_id": null,
    "mkto_lead_score": null,
    "fruits_c": "Apples",
    "_acl": {
      "fields": {

      }
    },
    "_module": "Leads",
    "_search": {
      "score": 0.70710677,
      "highlighted": {
        "email1": {

        }
      }
    }
  }
]
}

```

There are 3 records shown above, the `_module` field tells you if the record is from Accounts, Contacts or Leads. The `_search` field is an array that tells you where in the record it found the search string. It gives you back a highlighted string that you can use for display.

Last Modified: 10/14/2016 04:12pm

How to Check for Duplicate Records

Overview

An example in bash script demonstrating how to check for duplicate records using the v10 /<module>/duplicateCheck REST POST endpoint.

Duplicate Records

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
curl -X POST -H Cache-Control:no-cache -d '{
  "grant_type":"password",
  "client_id":"sugar",
  "client_secret":"",
  "username":"admin",
  "password":"password",
  "platform":"custom_api"
}' https://{site_url}/rest/v10/oauth2/token
```

More information on authenticating can be found in the How to Authenticate and Log Out example and /oauth2/logout endpoint documentation.

Retrieving Duplicates

Next, we will need to identify the records that are duplicates using the /<module>/duplicateCheck endpoint.

```
curl -s -X POST -H OAuth-Token:{access_token} -H
Cache-Control:no-cache -d '{
  "name":"Test Record"
}' https://{site_url}/rest/v10/Accounts/duplicateCheck
```

More information on the duplicate check API can be found in the [/module/duplicateCheck](#) documentation.

Response

The data received from the server is shown below:

```
{
  "next_offset": -1,
  "records": [{
    "id": "7f6ea7be-60d6-11e6-8885-a0999b033b33",
    "name": "Test Record",
    "date_entered": "2016-08-12T14:48:25-07:00",
    "date_modified": "2016-08-12T14:48:25-07:00",
    "modified_user_id": "1",
    "modified_by_name": "Administrator",
    "modified_user_link": {
      "full_name": "Administrator",
      "id": "1",
      "_acl": {
        "fields": [],
        "delete": "no",
        "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
      }
    },
    "created_by": "1",
    "created_by_name": "Administrator",
```

```
"created_by_link": {
  "full_name": "Administrator",
  "id": "1",
  "_acl": {
    "fields": [],
    "delete": "no",
    "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
  }
},
"description": "Test Data 1",
"deleted": false,
"facebook": "",
"twitter": "",
"googleplus": "",
"account_type": "",
"industry": "",
"annual_revenue": "",
"phone_fax": "",
"billing_address_street": "",
"billing_address_street_2": "",
"billing_address_street_3": "",
"billing_address_street_4": "",
"billing_address_city": "",
"billing_address_state": "",
"billing_address_postalcode": "",
"billing_address_country": "",
"rating": "",
"phone_office": "",
"phone_alternate": "",
"website": "",
"ownership": "",
"employees": "",
"ticker_symbol": "",
```

```
"shipping_address_street": "",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "",
"shipping_address_state": "",
"shipping_address_postalcode": "",
"shipping_address_country": "",
"parent_id": "",
"sic_code": "",
"duns_num": "",
"parent_name": "",
"member_of": {
  "name": "",
  "id": "",
  "_acl": {
    "fields": [],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"campaign_id": "",
"campaign_name": "",
"campaign_accounts": {
  "name": "",
  "id": "",
  "_acl": {
    "fields": [],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"following": true,
"my_favorite": false,
"tag": [],
```

```
"assigned_user_id": "1",
"assigned_user_name": "Administrator",
"assigned_user_link": {
  "full_name": "Administrator",
  "id": "1",
  "_acl": {
    "fields": [],
    "delete": "no",
    "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
  }
},
"team_count": "",
"team_count_link": {
  "team_count": "",
  "id": "1",
  "_acl": {
    "fields": [],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"team_name": [{
  "id": "1",
  "name": "Global",
  "name_2": "",
  "primary": true
}],
"email": [],
"email1": "",
"email2": "",
"invalid_email": "",
"email_opt_out": "",
"email_addresses_non_primary": "",
"_acl": {
```

```
        "fields": {}
    },
    "_module": "Accounts",
    "duplicate_check_rank": 8
}, {
    "id": "868b4f16-60d6-11e6-bdfc-a0999b033b33",
    "name": "Test Record",
    "date_entered": "2016-08-12T14:48:37-07:00",
    "date_modified": "2016-08-12T14:48:37-07:00",
    "modified_user_id": "1",
    "modified_by_name": "Administrator",
    "modified_user_link": {
        "full_name": "Administrator",
        "id": "1",
        "_acl": {
            "fields": [],
            "delete": "no",
            "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
        }
    },
    "created_by": "1",
    "created_by_name": "Administrator",
    "created_by_link": {
        "full_name": "Administrator",
        "id": "1",
        "_acl": {
            "fields": [],
            "delete": "no",
            "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
        }
    },
    "description": "Test Data 2",
    "deleted": false,
```

```
"facebook": "",
"twitter": "",
"googleplus": "",
"account_type": "",
"industry": "",
"annual_revenue": "",
"phone_fax": "",
"billing_address_street": "",
"billing_address_street_2": "",
"billing_address_street_3": "",
"billing_address_street_4": "",
"billing_address_city": "",
"billing_address_state": "",
"billing_address_postalcode": "",
"billing_address_country": "",
"rating": "",
"phone_office": "",
"phone_alternate": "",
"website": "",
"ownership": "",
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "",
"shipping_address_state": "",
"shipping_address_postalcode": "",
"shipping_address_country": "",
"parent_id": "",
"sic_code": "",
"duns_num": "",
```

```
"parent_name": "",
"member_of": {
  "name": "",
  "id": "",
  "_acl": {
    "fields": [],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"campaign_id": "",
"campaign_name": "",
"campaign_accounts": {
  "name": "",
  "id": "",
  "_acl": {
    "fields": [],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"following": true,
"my_favorite": false,
"tag": [],
"assigned_user_id": "1",
"assigned_user_name": "Administrator",
"assigned_user_link": {
  "full_name": "Administrator",
  "id": "1",
  "_acl": {
    "fields": [],
    "delete": "no",
    "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
  }
},
```

```
"team_count": "",
"team_count_link": {
  "team_count": "",
  "id": "1",
  "_acl": {
    "fields": [],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"team_name": [{
  "id": "1",
  "name": "Global",
  "name_2": "",
  "primary": true
}],
"email": [],
"email1": "",
"email2": "",
"invalid_email": "",
"email_opt_out": "",
"email_addresses_non_primary": "",
"_acl": {
  "fields": {}
},
"_module": "Accounts",
"duplicate_check_rank": 8
}]
}
```

Last Modified: 11/27/2017 12:17pm

PHP

PHP Examples interacting with the v10 REST API.

Last Modified: 06/26/2017 02:47pm

How to Export a List of Records

Overview

An PHP example demonstrating how to export a list of records using the v10 `/<module>/export/:record_list_id` REST GET endpoint.

Exporting Records

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
<?php

$instance_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";
```

```
$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
    //It is recommended to create your own in Admin > OAuth Keys
    "client_id" => "sugar",
    "client_secret" => "",
    "username" => $username,
    "password" => $password,
    //platform type - default is base.
    //It is recommend to change the platform to a custom name such as
    "custom_api" to avoid authentication conflicts.
    "platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION,
CURLOPT_HTTP_VERSION_1_0);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
```

```
$oauth2_token_response_obj = json_decode($oauth2_token_response);  
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the [How to Authenticate and Log Out](#) example and [/oauth2/logout](#) endpoint documentation.

Filtering Records

Next, we will need to identify the records we want to export using the `/<module>/filter` endpoint.

```
//Identify records to export - POST /<module>/filter  
  
$filter_url = $instance_url . "/Accounts/filter";  
  
$filter_arguments = array(  
    "filter" => array(  
        array(  
            '$or' => array(  
                array(  
                    //name starts with 'a'  
                    "name" => array(  
                        '$starts'=>"A",  
                    )  
                ),  
                array(  
                    //name starts with 'b'  
                    "name" => array(  
                        '$starts'=>"b",  
                    )  
                )  
            )  
        ),  
    ),  
);
```

```
        ),
    ),
    "max_num" => 2,
    "offset" => 0,
    "fields" => "id",
    "order_by" => "date_entered",
    "favorites" => false,
    "my_items" => false,
);

$filter_request = curl_init($filter_url);
curl_setopt($filter_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
curl_setopt($filter_request, CURLOPT_HEADER, false);
curl_setopt($filter_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($filter_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($filter_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($filter_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//convert arguments to json
$json_arguments = json_encode($filter_arguments);
curl_setopt($filter_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$filter_response = curl_exec($filter_request);

//decode json
$filter_response_obj = json_decode($filter_response);

//store ids of records to export
```

```
$export_ids = array();
foreach ($filter_response_obj->records as $record)
{
    $export_ids[] = $record->id;
}
```

More information on the filter API can be found in the `/<module>/filter` documentation.

Request Payload

The data sent to the server is shown below:

```
{
  "filter": [
    {
      "$or": [
        {
          "name": {
            "$starts": "A"
          }
        },
        {
          "name": {
            "$starts": "b"
          }
        }
      ]
    }
  ],
  "max_num": 2,
  "offset": 0,
```

```
"fields": "id",
"order_by": "date_entered",
"favorites": false,
"my_items": false
}
```

Response

The data received from the server is shown below:

```
{
  "next_offset": 2,
  "records": [
    {
      "id": "f16760a4-3a52-f77d-1522-5703ca28925f",
      "date_modified": "2016-04-05T10:23:28-04:00",
      "_acl": {
        "fields": {
          }
        }
      },
      "_module": "Accounts"
    },
    {
      "id": "ec409fbb-2b22-4f32-7fa1-5703caf78dc3",
      "date_modified": "2016-04-05T10:23:28-04:00",
      "_acl": {
        "fields": {
          }
        }
      },
      "_module": "Accounts"
    }
  ]
}
```

```
    }  
  ]  
}
```

Creating a Record List

Once we have the list of ids, we then need to create a record list in Sugar that consists of those ids.

```
//Create a record list - POST /<module>/record_list  
  
$record_list_url = $instance_url . "/Accounts/record_list";  
  
$record_list_arguments = array(  
    "records" => $export_ids,  
);  
  
$record_list_request = curl_init($record_list_url);  
curl_setopt($record_list_request, CURLOPT_HTTP_VERSION,  
CURLOPT_HTTP_VERSION_1_0);  
curl_setopt($record_list_request, CURLOPT_HEADER, false);  
curl_setopt($record_list_request, CURLOPT_SSL_VERIFYPEER, 0);  
curl_setopt($record_list_request, CURLOPT_RETURNTRANSFER, 1);  
curl_setopt($record_list_request, CURLOPT_FOLLOWLOCATION, 0);  
curl_setopt($record_list_request, CURLOPT_HTTPHEADER, array(  
    "Content-Type: application/json",  
    "oauth-token: {$oauth_token}"  
));  
  
//convert arguments to json  
$json_arguments = json_encode($record_list_arguments);  
curl_setopt($record_list_request, CURLOPT_POSTFIELDS,
```

```
$json_arguments);
```

```
//execute request
```

```
$record_list_response = curl_exec($record_list_request);
```

Request Payload

The data sent to the server is shown below:

```
{
  "records": [
    "f16760a4-3a52-f77d-1522-5703ca28925f",
    "ec409fbb-2b22-4f32-7fa1-5703caf78dc3"
  ]
}
```

Response

The data received from the server is shown below:

```
{
  "id": "ef963176-4845-bc55-b03e-570430b4173c",
  "assigned_user_id": "1",
  "module_name": "Accounts",
  "records": [
    "f16760a4-3a52-f77d-1522-5703ca28925f",
    "ec409fbb-2b22-4f32-7fa1-5703caf78dc3"
  ],
  "date_modified": "2016-04-05 21:39:19"
}
```

Exporting Records

Once we have the record list created, we can then export the CSV file.

```
//Export Records - GET /<module>/export/:record_list_id

$export_url = $instance_url . "/Accounts/export/" .
$record_list_response_obj->id;

$export_request = curl_init($export_url);
curl_setopt($export_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
curl_setopt($export_request, CURLOPT_HEADER, true); //needed to return
file headers
curl_setopt($export_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($export_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($export_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($export_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

$export_response = curl_exec($export_request);

//set headers from response
list($headers, $content) = explode("\r\n\r\n", $export_response ,2);
foreach (explode("\r\n",$headers) as $header) {
    header($header);
}

$content = trim($content);
echo $content;
```

More information on exporting records can be found in the
/<module>/export/:record_list_id documentation.

Response

The data received from the server is shown below:

```
HTTP/1.1 200 OK
Date: Tue, 05 Apr 2016 21:50:32
GMT Server: Apache/2.2.29 (Unix)
DAV/2 PHP/5.3.29 mod_ssl/2.2.29
OpenSSL/0.9.8zg X-Powered-By:
PHP/5.3.29 Expires:
Cache-Control: max-age=10,
private Pragma:
Content-Disposition: attachment; filename=Accounts.csv
Content-transfer-encoding: binary
Last-Modified: Tue, 05 Apr 2016 21:50:32
GMT ETag: 9b34f5d74e0298aaf7fd1f27d02e14f2
Content-Length: 1703
Connection: close
Content-Type: application/octet-stream; charset=ISO-8859-1
"Name","ID","Website","Office Phone","Alternate Phone","Fax","Billing
Street","Billing City","Billing State","Billing Postal Code","Billing
Country","Shipping Street","Shipping City","Shipping State","Shipping
Postal Code","Shipping
Country","Description","Type","Industry","Annual
Revenue","Employees","SIC Code","Ticker Symbol","Parent Account
ID","Ownership","Campaign ID","Rating","Assigned User Name","Assigned
User ID","Team ID","Teams","Team Set ID","Date Created","Date
Modified","Modified By Name","Modified By ID","Created By","Created By
ID","Deleted","test","Facebook Account","Twitter Account","Google Plus
```

ID", "DUNS", "Email", "Invalid Email", "Email Opt Out", "Non-primary emails"
"Arts & Crafts
Inc", "ec409fbb-2b22-4f32-7fa1-5703caf78dc3", "www.hrinfo.tw", "(252)
456-8602", "", "", "777 West Filmore Ln", "Los
Angeles", "CA", "77076", "USA", "777 West Filmore Ln", "Los
Angeles", "CA", "77076", "USA", "", "Customer", "Hospitality", "", "", "", "", "",
", "", "", "", "Max Jensen", "seed_max_id", "West", "West, East,
Global", "dec43cb2-5273-8be2-968a-5703cadee75f", "2016-04-05
10:23", "2016-04-05
10:23", "Administrator", "1", "Administrator", "1", "0", "", "", "", "", "", "sug
ar.sugar.section@example.org", "0", "0", "dev.phone@example.biz,0,0"
"B.H. Edwards
Inc", "f16760a4-3a52-f77d-1522-5703ca28925f", "www.sectiondev.edu", "(361
) 765-0216", "", "", "111 Silicon Valley
Road", "Persistence", "CA", "29709", "USA", "111 Silicon Valley
Road", "Persistence", "CA", "29709", "USA", "", "Customer", "Apparel", "", "", "
", "", "", "", "", "", "Sally
Bronsen", "seed_sally_id", "West", "West", "West", "2016-04-05
10:23", "2016-04-05
10:23", "Administrator", "1", "Administrator", "1", "0", "", "", "", "", "", "inf
o.sugar@example.de", "0", "1", "phone.sales.section@example.tv,0,0"

Download

You can download the full API example [here](#).

Last Modified: 10/14/2016 04:12pm

How to Filter a List of Records

Overview

A PHP example demonstrating how to filter records using the v10 /<module>/filter REST POST endpoints.

Filtering Records

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
<?php

$instance_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
    //It is recommended to create your own in Admin > OAuth Keys
    "client_id" => "sugar",
    "client_secret" => "",
    "username" => $username,
    "password" => $password,
    //platform type - default is base.
    //It is recommend to change the platform to a custom name such as
    "custom_api" to avoid authentication conflicts.
```

```
    "platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION,
CURLOPT_HTTP_VERSION_1_0);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the [How to Authenticate and Log Out example](#) and [/oauth2/logout endpoint documentation](#).

Filtering Records

Next, we can filter the records we want to return using the `/<module>/filter` endpoint with a POST request.

```
//Identify records to export - POST /<module>/filter

$filter_url = $instance_url . "/Accounts/filter";

$filter_arguments = array(
    "filter" => array(
        array(
            '$or' => array(
                array(
                    //name starts with 'a'
                    "name" => array(
                        '$starts'=>"A",
                    )
                ),
                array(
                    //name starts with 'b'
                    "name" => array(
                        '$starts'=>"b",
                    )
                )
            )
        ),
    ),
    "max_num" => 2,
    "offset" => 0,
    "fields" => "id",
    "order_by" => "date_entered",
    "favorites" => false,
    "my_items" => false,
);

$filter_request = curl_init($filter_url);
curl_setopt($filter_request, CURLOPT_HTTP_VERSION,
```

```
CURL_HTTP_VERSION_1_0);
curl_setopt($filter_request, CURLOPT_HEADER, false);
curl_setopt($filter_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($filter_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($filter_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($filter_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//convert arguments to json
$json_arguments = json_encode($filter_arguments);
curl_setopt($filter_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$filter_response = curl_exec($filter_request);

//decode json
$filter_response_obj = json_decode($filter_response);
```

More information on the filter API can be found in the [/<module>/filter](#) documentation.

Note : The /<module>/filter endpoint can be called using a GET request as well, though long URL requests can have issues so the POST request is recommended.

Request Payload

The data sent to the server is shown below:

```
{
  "filter":[
```

```
{
  "$or": [
    {
      "name": {
        "$starts": "A"
      }
    },
    {
      "name": {
        "$starts": "b"
      }
    }
  ]
},
"max_num": 2,
"offset": 0,
"fields": "id",
"order_by": "date_entered",
"favorites": false,
"my_items": false
}
```

Response

The data received from the server is shown below:

```
{
  "next_offset": 2,
  "records": [
    {
      "id": "f16760a4-3a52-f77d-1522-5703ca28925f",
```

```
"date_modified": "2016-04-05T10:23:28-04:00",
  "_acl": {
    "fields": {
      }
    },
  "_module": "Accounts"
},
{
  "id": "ec409fbb-2b22-4f32-7fa1-5703caf78dc3",
  "date_modified": "2016-04-05T10:23:28-04:00",
  "_acl": {
    "fields": {
      }
    },
  "_module": "Accounts"
}
]
}
```

Download

You can download the full API example using [POSThere](#) and using [GEThere](#).

Last Modified: 11/27/2017 12:17pm

How to Favorite a Record

Overview

A PHP example demonstrating how to favorite a record using the v10 <module>/:record/favorite REST PUT API endpoint.

Favoriting a Record

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
<?php

$instance_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
    //It is recommended to create your own in Admin > OAuth Keys
    "client_id" => "sugar",
    "client_secret" => "",
    "username" => $username,
    "password" => $password,
    //platform type - default is base.
    //It is recommend to change the platform to a custom name such as
    "custom_api" to avoid authentication conflicts.
    "platform" => "custom_api"
```

```
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the [How to Authenticate and Log Out](#) example and [/oauth2/logout](#) endpoint documentation.

Favoriting a Record

Next, we can favorite a specific record using the `/<module>/:record/favorite` endpoint.

```
//Favorite record - PUT //:record/favorite

$favorite_url = $instance_url .
"/Accounts/ae8b1783-404a-fcb8-1e1e-56f1cc52cd1a/favorite";

$favorite_request = curl_init($favorite_url);

curl_setopt($favorite_request, CURLOPT_CUSTOMREQUEST, "PUT");
curl_setopt($favorite_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
curl_setopt($favorite_request, CURLOPT_HEADER, false);
curl_setopt($favorite_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($favorite_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($favorite_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($favorite_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//execute request
$favorite_response = curl_exec($favorite_request);
```

More information on the `unfavorite` API can be found in the [/<module>/:record/favorite PUT](#) documentation.

Request Payload

The data sent to the server is shown below:

This endpoint does not accept any request arguments.

Response

The data received from the server is shown below:

```
{
  "id": "ae8b1783-404a-fcb8-1e1e-56f1cc52cd1a",
  "name": "Union Bank",
  "date_entered": "2016-03-22T17:49:50-05:00",
  "date_modified": "2016-03-30T17:44:20-05:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "modified_user_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
      "fields": [],
      "delete": "no",
      "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
    }
  },
  "created_by": "1",
  "created_by_name": "Administrator",
  "created_by_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
      "fields": [],
      "delete": "no",
      "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
    }
  },
  "description": "",
  "deleted": false,
```

```
"facebook": "",
"twitter": "",
"googleplus": "",
"account_type": "Customer",
"industry": "Banking",
"annual_revenue": "",
"phone_fax": "",
"billing_address_street": "67321 West Siam St.",
"billing_address_street_2": "",
"billing_address_street_3": "",
"billing_address_street_4": "",
"billing_address_city": "Ohio",
"billing_address_state": "CA",
"billing_address_postalcode": "25159",
"billing_address_country": "USA",
"rating": "",
"phone_office": "(065) 489-6104",
"phone_alternate": "",
"website": "www.qahr.edu",
"ownership": "",
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "67321 West Siam St.",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "Ohio",
"shipping_address_state": "CA",
"shipping_address_postalcode": "25159",
"shipping_address_country": "USA",
"parent_id": "",
"sic_code": "",
"duns_num": "",
```

```
"parent_name": "",
"member_of": {
  "name": "",
  "id": "",
  "_acl": {
    "fields": [],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"campaign_id": "",
"campaign_name": "",
"campaign_accounts": {
  "name": "",
  "id": "",
  "_acl": {
    "fields": [],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"following": true,
"my_favorite": true,
"tag": [],
"assigned_user_id": "seed_sarah_id",
"assigned_user_name": "Sarah Smith",
"assigned_user_link": {
  "full_name": "Sarah Smith",
  "id": "seed_sarah_id",
  "_acl": {
    "fields": [],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"team_count": "",
```

```
"team_count_link": {
  "team_count": "",
  "id": "West",
  "_acl": {
    "fields": [],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"team_name": [{
  "id": "East",
  "name": "East",
  "name_2": "",
  "primary": false
}, {
  "id": 1,
  "name": "Global",
  "name_2": "",
  "primary": false
}, {
  "id": "West",
  "name": "West",
  "name_2": "",
  "primary": true
}],
"email": [{
  "email_address": "hr.support.kid@example.info",
  "invalid_email": false,
  "opt_out": false,
  "primary_address": true,
  "reply_to_address": false
}, {
  "email_address": "info.support.the@example.com",
  "invalid_email": false,
```

```
        "opt_out": false,
        "primary_address": false,
        "reply_to_address": false
    }],
    "email1": "hr.support.kid@example.info",
    "email2": "info.support.the@example.com",
    "invalid_email": false,
    "email_opt_out": false,
    "email_addresses_non_primary": "",
    "_acl": {
        "fields": {}
    },
    "_module": "Accounts"
}
```

Download

You can download the full API example [here](#).

Last Modified: 11/27/2017 12:17pm

How to Manipulate File Attachments

Overview

A PHP example demonstrating how to attach a file to a record using the v10 `<module>/:record/file/:field` REST POST API endpoint, then retrieve it with the GET endpoint.

Manipulating File Attachments

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
<?php

$instance_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
    //It is recommended to create your own in Admin > OAuth Keys
    "client_id" => "sugar",
    "client_secret" => "",
    "username" => $username,
    "password" => $password,
    //platform type - default is base.
    //It is recommend to change the platform to a custom name such as
    "custom_api" to avoid authentication conflicts.
    "platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
curl_setopt($auth_request, CURLOPT_HEADER, false);
```

```
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the [How to Authenticate and Log Out example](#) and [/oauth2/logout endpoint documentation](#).

Submitting a File Attachment

Next, we can create a Note record using the `/<module>` endpoint, and then submit a File to the Note record using the `/<module>/:record/file/:field` endpoint.

```
//Create Note - POST /
$url = $instance_url . "/Notes";
//Set up the Record details
$record = array(
    'name' => 'Test Note'
);
```

```
$curl_request = curl_init($url);
curl_setopt($curl_request, CURLOPT_POST, 1);
curl_setopt($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
curl_setopt($curl_request, CURLOPT_HEADER, false);
curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($curl_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//convert arguments to json
$json_arguments = json_encode($record);
curl_setopt($curl_request, CURLOPT_POSTFIELDS, $json_arguments);
//execute request
$curl_response = curl_exec($curl_request);
//decode json
$noteRecord = json_decode($curl_response);

//display the created record
echo "Created Record: ". $noteRecord->id;

curl_close($curl_request);

//Add An Attachment to the Note
$url = $instance_url . "/Notes/{$noteRecord->id}/file/filename";

$file_arguments = array(
    "format" => "sugar-html-json",
    "delete_if_fails" => true,
```

```

        "oauth_token" => $oauth_token,
    );

    if ((version_compare(PHP_VERSION, '5.5') >= 0)) {
        $file_arguments['filename'] = new CURLFile($path);
    } else {
        $file_arguments['filename'] = '@'.$path;
    }

    $curl_request = curl_init($url);
    curl_setopt($curl_request, CURLOPT_POST, 1);
    curl_setopt($curl_request, CURLOPT_HTTP_VERSION,
    CURL_HTTP_VERSION_1_0);
    curl_setopt($curl_request, CURLOPT_HEADER, false);
    curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);
    //Do NOT set Content Type Header to JSON
    curl_setopt($curl_request, CURLOPT_HTTPHEADER, array(
        "oauth-token: {$oauth_token}"
    ));

    curl_setopt($curl_request, CURLOPT_POSTFIELDS, $file_arguments);
    //execute request
    $curl_response = curl_exec($curl_request);
    //decode json
    $noteRecord = json_decode($curl_response);
    //print Note with attachment details
    print_r($noteRecord);

    curl_close($curl_request);

```

Note: As of PHP 5.6, the '@' upload modifier is disabled for security reasons by the

CURLOPT_SAFE_UPLOAD option. We recommending using CURLOPTFILE if using PHP >= 5.5, If you would prefer to use the upload modifier, you can set the CURLOPT_SAFE_UPLOAD option to false.

```
curl_setopt($ch, CURLOPT_SAFE_UPLOAD, false);
```

Request

```
//Raw Post - not json encoded
```

```
Array
```

```
(
  [format] => sugar-html-json
  [delete_if_fails] => 1
  [oauth_token] => 09eac950-c99e-4786-8b10-a3670f38fb3f
  [filename] => CURLFile Object
    (
      [name] =>
/Library/WebServer/Documents/file_attachment_manipulation/testfile.txt
      [mime] =>
      [postname] =>
    )
)
```

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"  
}  
}
```

Getting a File Attachment

Next, we can retrieve the file attachment stored in Sugar by utilizing the `/<module>/:record/file/:fieldGET` endpoint. The following code example, works when being accessed via a web browser, as it receives the response from Sugar, and sets the Headers received from Sugar on itself, so that the browser knows to download a file.

```
//Get An Attachment on a Note  
$url = $instance_url . "/Notes/{$noteRecord->id}/file/filename";
```

```
$curl_request = curl_init($url);
curl_setopt($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
//Return Headers
curl_setopt($curl_request, CURLOPT_HEADER, true);
curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);
//DO NOT set Content Type Header to JSON
curl_setopt($curl_request, CURLOPT_HTTPHEADER, array(
    "oauth-token: {$oauth_token}"
));

//execute request
$curl_response = curl_exec($curl_request);

//Get Return Headers
$header_size = curl_getinfo($curl_request,CURLINFO_HEADER_SIZE);
$headers = substr($curl_response, 0, $header_size);

//Outputting the file contents
echo 'File saved to download.txt';
$file = substr($curl_response, $header_size);
file_put_contents('download.txt', $file);

curl_close($curl_request);
```

Request

```
http://{site_url}/rest/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,php3; path=

Expires:

Cache-Control: max-age=0, private

Pragma:

Content-Disposition: attachment; filename="upload.txt"

X-Content-Type-Options: nosniff

ETag: d41d8cd98f00b204e9800998ecf8427e

Content-Length: 16

Connection: close

Content-Type: application/octet-stream

This is the file contents.

Download

You can download the full API example here.

Last Modified: 11/27/2017 12:17pm

How to Fetch Related Records

Overview

A PHP example demonstrating how to fetch related records using the v10 /<module>/:record/link/:link REST GET endpoints.

Fetching Related Records

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
<?php
```

```
$instance_url = "http://{site_url}/rest/v10";  
$username = "admin";  
$password = "password";
```

```
//Login - POST /oauth2/token
```

```
$auth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
    //It is recommended to create your own in Admin > OAuth Keys
    "client_id" => "sugar",
    "client_secret" => "",
    "username" => $username,
    "password" => $password,
    //platform type - default is base.
    //It is recommend to change the platform to a custom name such as
    "custom_api" to avoid authentication conflicts.
    "platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);
```

```
//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the [How to Authenticate and Log Out example](#) and [/oauth2/logout endpoint documentation](#).

Fetching Related Records

Next, we can fetch the related records we want to return using the `/<module>/:record/link/:link` endpoint with a GET request where

Element	Meaning
<code><module></code>	The parent module name
<code>:record</code>	The parent records ID
<code>link</code>	the actual word "link"
<code>:link</code>	The name of the relationship to fetch

In this example, we will fetch the related Contacts for an Account

```
//Identify records to fetch - POST /<module>/:record/link/:link

$fetch_url = $instance_url .
"/Accounts/d8f05e67-dee3-553d-0040-5342e88f2fd1/link/contacts";

$fetch_request = curl_init($fetch_url);
curl_setopt($fetch_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
curl_setopt($fetch_request, CURLOPT_HEADER, false);
curl_setopt($fetch_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($fetch_request, CURLOPT_RETURNTRANSFER, 1);
```

```
curl_setopt($fetch_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($fetch_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//execute request
$fetch_response = curl_exec($fetch_request);

//decode json
$fetch_response_obj = json_decode($fetch_response);
```

Request

```
http://{site_url}/rest/v10/Accounts/d8f05e67-dee3-553d-0040-5342e88f2fd1/link/contacts
```

Response

The data received from the server is shown below:

```
{
  "next_offset":-1,
  "records":[
    {
      "my_favorite":false,
      "following":false,
      "id":"819f4149-b007-a6da-a5fa-56fedbf2de77",
      "name":"Florine Marcus",
      "date_entered":"2016-04-01T15:34:00-05:00",
      "date_modified":"2016-04-01T15:34:00-05:00",
```

```
"modified_user_id": "1",
"modified_by_name": "Administrator",
"created_by": "1",
"created_by_name": "Administrator",
"doc_owner": "",
"user_favorites": "",
"description": "",
"deleted": false,
"assigned_user_id": "seed_will_id",
"assigned_user_name": "Will Westin",
"team_count": "",
"team_name": [
  {
    "id": "East",
    "name": "East",
    "name_2": "",
    "primary": true
  },
  {
    "id": "West",
    "name": "West",
    "name_2": "",
    "primary": false
  }
],
"email": [
  {
    "email_address": "support27@example.tv",
    "primary_address": true,
    "reply_to_address": false,
    "invalid_email": false,
    "opt_out": false
  },

```

```
    {
      "email_address": "support.support.kid@example.net",
      "primary_address": false,
      "reply_to_address": false,
      "invalid_email": false,
      "opt_out": true
    }
  ],
  "email1": "support27@example.tv",
  "email2": "",
  "invalid_email": false,
  "email_opt_out": false,
  "email_addresses_non_primary": "",
  "salutation": "",
  "first_name": "Florine",
  "last_name": "Marcus",
  "full_name": "Florine Marcus",
  "title": "President",
  "facebook": "",
  "twitter": "",
  "googleplus": "",
  "department": "",
  "do_not_call": false,
  "phone_home": "(746) 162-2314",
  "phone_mobile": "(941) 088-2874",
  "phone_work": "(827) 541-9614",
  "phone_other": "",
  "phone_fax": "",
  "primary_address_street": "1715 Scott Dr",
  "primary_address_street_2": "",
  "primary_address_street_3": "",
  "primary_address_city": "Alabama",
  "primary_address_state": "NY",
```

```
"primary_address_postalcode": "70187",
"primary_address_country": "USA",
"alt_address_street": "",
"alt_address_street_2": "",
"alt_address_street_3": "",
"alt_address_city": "",
"alt_address_state": "",
"alt_address_postalcode": "",
"alt_address_country": "",
"assistant": "",
"assistant_phone": "",
"picture": "",
"email_and_name1": "",
"lead_source": "Employee",
"account_name": "MTM Investment Bank F S B",
"account_id": "e8c641ca-1b8c-74c1-d08d-56fedbdd3187",
"dnb_principal_id": "",
"opportunity_role_fields": "",
"opportunity_role_id": "",
"opportunity_role": "",
"reports_to_id": "",
"report_to_name": "",
"birthdate": "",
"portal_name": "FlorineMarcus119",
"portal_active": true,
"portal_password": true,
"portal_password1": null,
"portal_app": "",
"preferred_language": "en_us",
"campaign_id": "",
"campaign_name": "",
"c_accept_status_fields": "",
"m_accept_status_fields": "",
```

```
"accept_status_id":"","
"accept_status_name":"","
"accept_status_calls":"","
"accept_status_meetings":"","
"sync_contact":false,
"mkto_sync":false,
"mkto_id":null,
"mkto_lead_score":null,
"_acl":{
  "fields":{

  }
},
"_module":"Contacts"
},
{
  "my_favorite":false,
  "following":false,
  "id":"527ccla9-7984-91fe-4148-56fedbc356aa",
  "name":"Shaneka Aceto",
  "date_entered":"2016-04-01T15:34:00-05:00",
  "date_modified":"2016-04-01T15:34:00-05:00",
  "modified_user_id":"1",
  "modified_by_name":"Administrator",
  "created_by":"1",
  "created_by_name":"Administrator",
  "doc_owner":"","
  "user_favorites":"","
  "description":"","
  "deleted":false,
  "assigned_user_id":"seed_will_id",
  "assigned_user_name":"Will Westin",
  "team_count":"","
```

```
"team_name":[
  {
    "id":"East",
    "name":"East",
    "name_2":"",
    "primary":true
  },
  {
    "id":"West",
    "name":"West",
    "name_2":"",
    "primary":false
  }
],
"email":[
  {
    "email_address":"support17@example.cn",
    "primary_address":true,
    "reply_to_address":false,
    "invalid_email":false,
    "opt_out":false
  },
  {
    "email_address":"section.sugar.the@example.tv",
    "primary_address":false,
    "reply_to_address":false,
    "invalid_email":false,
    "opt_out":true
  }
],
"email1":"support17@example.cn",
"email2":"",
"invalid_email":false,
```

```
"email_opt_out":false,
"email_addresses_non_primary":"","
"salutation":"","
"first_name":"Shaneka",
"last_name":"Aceto",
"full_name":"Shaneka Aceto",
"title":"IT Developer",
"facebook":"","
"twitter":"","
"googleplus":"","
"department":"","
"do_not_call":false,
"phone_home":"(502) 528-5151",
"phone_mobile":"(816) 719-3739",
"phone_work":"(994) 769-5855",
"phone_other":"","
"phone_fax":"","
"primary_address_street":"123 Anywhere Street",
"primary_address_street_2":"","
"primary_address_street_3":"","
"primary_address_city":"Denver",
"primary_address_state":"NY",
"primary_address_postalcode":"15128",
"primary_address_country":"USA",
"alt_address_street":"","
"alt_address_street_2":"","
"alt_address_street_3":"","
"alt_address_city":"","
"alt_address_state":"","
"alt_address_postalcode":"","
"alt_address_country":"","
"assistant":"","
"assistant_phone":"","
```

```
"picture": "",
"email_and_name1": "",
"lead_source": "Email",
"account_name": "MTM Investment Bank F S B",
"account_id": "e8c641ca-1b8c-74c1-d08d-56fedbdd3187",
"dnb_principal_id": "",
"opportunity_role_fields": "",
"opportunity_role_id": "",
"opportunity_role": "",
"reports_to_id": "",
"report_to_name": "",
"birthdate": "",
"portal_name": "ShanekaAceto151",
"portal_active": true,
"portal_password": true,
"portal_password1": null,
"portal_app": "",
"preferred_language": "en_us",
"campaign_id": "",
"campaign_name": "",
"c_accept_status_fields": "",
"m_accept_status_fields": "",
"accept_status_id": "",
"accept_status_name": "",
"accept_status_calls": "",
"accept_status_meetings": "",
"sync_contact": false,
"mkto_sync": false,
"mkto_id": null,
"mkto_lead_score": null,
"_acl": {
  "fields": {
```

```
    }
  },
  "_module": "Contacts"
},
{
  "my_favorite": false,
  "following": false,
  "id": "42d703ed-f834-f87c-967d-56fedb044051",
  "name": "Johnnie Pina",
  "date_entered": "2016-04-01T15:34:00-05:00",
  "date_modified": "2016-04-01T15:34:00-05:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
  "doc_owner": "",
  "user_favorites": "",
  "description": "",
  "deleted": false,
  "assigned_user_id": "seed_will_id",
  "assigned_user_name": "Will Westin",
  "team_count": "",
  "team_name": [
    {
      "id": "East",
      "name": "East",
      "name_2": "",
      "primary": true
    },
    {
      "id": "West",
      "name": "West",
      "name_2": ""
    }
  ]
}
```

```
        "primary":false
    }
],
"email":[
    {
        "email_address":"sugar.support.hr@example.co.uk",
        "primary_address":true,
        "reply_to_address":false,
        "invalid_email":false,
        "opt_out":false
    },
    {
        "email_address":"support.im@example.tw",
        "primary_address":false,
        "reply_to_address":false,
        "invalid_email":false,
        "opt_out":true
    }
],
"email1":"sugar.support.hr@example.co.uk",
"email2":"","
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"","
"salutation":"","
"first_name":"Johnnie",
"last_name":"Pina",
"full_name":"Johnnie Pina",
"title":"VP Operations",
"facebook":"","
"twitter":"","
"googleplus":"","
"department":"","
```

```
"do_not_call":false,
"phone_home":"(159) 335-1423",
"phone_mobile":"(580) 140-4050",
"phone_work":"(418) 792-9611",
"phone_other":"","
"phone_fax":"","
"primary_address_street":"345 Sugar Blvd.",
"primary_address_street_2":"","
"primary_address_street_3":"","
"primary_address_city":"Denver",
"primary_address_state":"NY",
"primary_address_postalcode":"70648",
"primary_address_country":"USA",
"alt_address_street":"","
"alt_address_street_2":"","
"alt_address_street_3":"","
"alt_address_city":"","
"alt_address_state":"","
"alt_address_postalcode":"","
"alt_address_country":"","
"assistant":"","
"assistant_phone":"","
"picture":"","
"email_and_name1":"","
"lead_source":"Direct Mail",
"account_name":"MTM Investment Bank F S B",
"account_id":"e8c641ca-1b8c-74c1-d08d-56fedbdd3187",
"dnb_principal_id":"","
"opportunity_role_fields":"","
"opportunity_role_id":"","
"opportunity_role":"","
"reports_to_id":"","
"report_to_name":"","
```

```
"birthdate": "",
"portal_name": "JohnniePinal94",
"portal_active": true,
"portal_password": true,
"portal_password1": null,
"portal_app": "",
"preferred_language": "en_us",
"campaign_id": "",
"campaign_name": "",
"c_accept_status_fields": "",
"m_accept_status_fields": "",
"accept_status_id": "",
"accept_status_name": "",
"accept_status_calls": "",
"accept_status_meetings": "",
"sync_contact": false,
"mkto_sync": false,
"mkto_id": null,
"mkto_lead_score": null,
"_acl": {
  "fields": {

  }
},
"_module": "Contacts"
}
]
}
```

Download

You can download the full API example [here](#)

Last Modified: 11/27/2017 12:17pm

How to Manipulate Records (CRUD)

Overview

A PHP example demonstrating how to use the CRUD (Create, Read, Update, Delete) endpoints in the REST v10 API.

CRUD Operations

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
<?php

$instance_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
    //It is recommended to create your own in Admin > OAuth Keys
    "client_id" => "sugar",
    "client_secret" => "",
```

```
    "username" => $username,
    "password" => $password,
    //platform type - default is base.
    //It is recommend to change the platform to a custom name such as
"custom_api" to avoid authentication conflicts.
    "platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the [How to Authenticate and Log Out example](#) and [/oauth2/logout endpoint documentation](#).

Creating a Record

Next, we need to submit the record to the Sugar instance using the `/<module>` endpoint. In this example we are going to create an Account record with a Name of 'Test Record' and an email of 'test@sugar.com'.

```
//Create Records - POST /<module>
$url = $instance_url . "/Accounts";
//Set up the Record details
$record = array(
    'name' => 'Test Record',
    'email1' => 'test@sugar.com'
);

$curl_request = curl_init($url);
curl_setopt($curl_request, CURLOPT_HTTP_VERSION,
    CURL_HTTP_VERSION_1_0);
curl_setopt($curl_request, CURLOPT_HEADER, false);
curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($curl_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//convert arguments to json
$json_arguments = json_encode($record);
curl_setopt($curl_request, CURLOPT_POSTFIELDS, $json_arguments);
//execute request
$curl_response = curl_exec($curl_request);
//decode json
```

```
$createdRecord = json_decode($curl_response);

//display the created record
print_r($createdRecord);
curl_close($curl_request);
```

More information on this API endpoint can be found in the [/<module> - POST documentation](#).

Request Payload

The data sent to the server is shown below:

```
{
  "name": "Test Record",
  "email": "test@sugar.com"
}
```

Response

The data received from the server is shown below:

```
{
  "id": "ab2222df-73da-0e92-6887-5705428f4d68",
  "name": "Test Record",
  "date_entered": "2016-04-06T13:07:41-04:00",
  "date_modified": "2016-04-06T13:07:41-04:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "modified_user_link": {
    "full_name": "Administrator",
```

```
"id": "1",
"_acl": {
  "fields": [
    ],
    "delete": "no",
    "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
  }
},
"created_by": "1",
"created_by_name": "Administrator",
"created_by_link": {
  "full_name": "Administrator",
  "id": "1",
  "_acl": {
    "fields": [
      ],
      "delete": "no",
      "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
    }
  },
"description": "",
"deleted": false,
"facebook": "",
"twitter": "",
"googleplus": "",
"account_type": "",
"industry": "",
"annual_revenue": "",
"phone_fax": "",
"billing_address_street": "",
"billing_address_street_2": "",
```

```
"billing_address_street_3": "",
"billing_address_street_4": "",
"billing_address_city": "",
"billing_address_state": "",
"billing_address_postalcode": "",
"billing_address_country": "",
"rating": "",
"phone_office": "",
"phone_alternate": "",
"website": "",
"ownership": "",
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "",
"shipping_address_state": "",
"shipping_address_postalcode": "",
"shipping_address_country": "",
"parent_id": "",
"sic_code": "",
"duns_num": "",
"parent_name": "",
"member_of": {
  "name": "",
  "id": "",
  "_acl": {
    "fields": [
      ],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
```

```
    }
  },
  "campaign_id": "",
  "campaign_name": "",
  "campaign_accounts": {
    "name": "",
    "id": "",
    "_acl": {
      "fields": [

    ],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"following": true,
"my_favorite": false,
"tag": [

],
"assigned_user_id": "",
"assigned_user_name": "",
"assigned_user_link": {
  "full_name": "",
  "id": "",
  "_acl": {
    "fields": [

  ],
  "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
}
},
"team_count": "",
"team_count_link": {
```

```
"team_count":"","  
"id":"1",  
"_acl":{  
  "fields":[  
  
  ],  
  "_hash":"654d337e0e912edaa00dbb0fb3dc3c17"  
}  
},  
"team_name":[  
  {  
    "id":1,  
    "name":"Global",  
    "name_2":"","  
    "primary":true  
  }  
],  
"email":[  
  {  
    "email_address":"test@sugar.com",  
    "invalid_email":false,  
    "opt_out":false,  
    "primary_address":true,  
    "reply_to_address":false  
  }  
],  
"email1":"test@sugar.com",  
"email2":"","  
"invalid_email":false,  
"email_opt_out":false,  
"email_addresses_non_primary":"","  
"_acl":{  
  "fields":{
```

```
    }
  },
  "_module": "Accounts"
}
```

Getting a Record

Next, we can get the created record from the Sugar instance using the `/<module>/:record` endpoint. In this example, we are going to get an Account record by its ID, but only request the Name, Email, and Industry fields.

```
$id = $createdRecord->id;

//Get Record - GET //:record
$url = $instance_url . "/Accounts/$id";

//Setup request to only return some fields on module
$data = array(
    'fields' => 'name,email,industry'
);

//Add data to the URL
$url = $url."?".http_build_query($data);

$curl_request = curl_init($url);
curl_setopt($curl_request, CURLOPT_HTTP_VERSION,
    CURL_HTTP_VERSION_1_0);
curl_setopt($curl_request, CURLOPT_HEADER, false);
curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);
```

```
curl_setopt($curl_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//execute request
$curl_response = curl_exec($curl_request);
//decode json
$record = json_decode($curl_response);

//display the created record
print_r($record);
curl_close($curl_request);
```

Updating a Record

Next, we can update the record in the Sugar instance using the `/<module>/:record` endpoint, and the PUT Http method. In this example we are going to update the Account record and change it's name to "Updated Test Record".

```
$id = $record->id;
//Update Record - PUT /<module>/:record
$url = $instance_url . "/Accounts/$id";
//Set up the Record details
$record->name = 'Updated Test Record';

$curl_request = curl_init($url);
curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "PUT");
curl_setopt($curl_request, CURLOPT_HTTP_VERSION,
    CURL_HTTP_VERSION_1_0);
curl_setopt($curl_request, CURLOPT_HEADER, false);
curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
```

```
curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($curl_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//convert arguments to json
$json_arguments = json_encode($record);
curl_setopt($curl_request, CURLOPT_POSTFIELDS, $json_arguments);
//execute request
$curl_response = curl_exec($curl_request);
//decode json
$updatedRecord = json_decode($curl_response);

//display the created record
echo "Updated Record Name:". $updatedRecord->name;
curl_close($curl_request);
```

More information on this API endpoint can be found in the [/<module>/:record - PUT](#) documentation.

Request Payload

The URL sent to the server is shown below:

```
{
  "id": "ab2222df-73da-0e92-6887-5705428f4d68",
  "name": "Updated Test Record",
  "date_modified": "2016-04-06T15:03:21-04:00",
  "industry": "",
  "email1": "test@sugar.com",
```

```
  "_acl":{
    "fields":{

    }
  },
  "_module": "Accounts"
}
```

Response

The data received from the server is shown below:

```
{
  "id": "ab2222df-73da-0e92-6887-5705428f4d68",
  "name": "Updated Test Record",
  "date_entered": "2016-04-06T15:03:21-04:00",
  "date_modified": "2016-04-06T15:03:22-04:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "modified_user_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
      "fields": [

      ],
      "delete": "no",
      "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
    }
  },
  "created_by": "1",
  "created_by_name": "Administrator",
}
```

```
"created_by_link":{
  "full_name":"Administrator",
  "id":"1",
  "_acl":{
    "fields":[

    ],
    "delete":"no",
    "_hash":"8e11bf9be8f04daddee9d08d44ea891e"
  }
},
"description":"",
"deleted":false,
"facebook":"",
"twitter":"",
"googleplus":"",
"account_type":"",
"industry":"",
"annual_revenue":"",
"phone_fax":"",
"billing_address_street":"",
"billing_address_street_2":"",
"billing_address_street_3":"",
"billing_address_street_4":"",
"billing_address_city":"",
"billing_address_state":"",
"billing_address_postalcode":"",
"billing_address_country":"",
"rating":"",
"phone_office":"",
"phone_alternate":"",
"website":"",
"ownership":"",
```

```
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "",
"shipping_address_state": "",
"shipping_address_postalcode": "",
"shipping_address_country": "",
"parent_id": "",
"sic_code": "",
"duns_num": "",
"parent_name": "",
"member_of": {
  "name": "",
  "id": "",
  "_acl": {
    "fields": [

      ],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"campaign_id": "",
"campaign_name": "",
"campaign_accounts": {
  "name": "",
  "id": "",
  "_acl": {
    "fields": [

      ],

```

```
        "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
},
"following": true,
"my_favorite": false,
"tag": [

],
"assigned_user_id": "",
"assigned_user_name": "",
"assigned_user_link": {
    "full_name": "",
    "id": "",
    "_acl": {
        "fields": [

            ],
            "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
        }
    },
"team_count": "",
"team_count_link": {
    "team_count": "",
    "id": "1",
    "_acl": {
        "fields": [

            ],
            "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
        }
    },
"team_name": [
    {
```

```
        "id":1,
        "name":"Global",
        "name_2":"","
        "primary":true
    }
],
"email":[
    {
        "email_address":"test@sugar.com",
        "invalid_email":false,
        "opt_out":false,
        "primary_address":true,
        "reply_to_address":false
    }
],
"email1":"test@sugar.com",
"email2":"","
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"","
"_acl":{
    "fields":{

    }
},
"_module":"Accounts"
}
```

Deleting a Record

Next, we can delete the record from the Sugar instance using the `/<module>/:record` endpoint, by using the DELETE Http Method.

```
$id = $updatedRecord->id;
//Delete Record - DELETE /<module>/:record
$url = $instance_url . "/Accounts/$id";

$curl_request = curl_init($url);
curl_setopt($curl_request, CURLOPT_CUSTOMREQUEST, "DELETE");
curl_setopt($curl_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
curl_setopt($curl_request, CURLOPT_HEADER, false);
curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($curl_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//execute request
$curl_response = curl_exec($curl_request);
//decode json
$deletedRecord = json_decode($curl_response);

//display the created record
echo "Deleted Record:". $deletedRecord->id;
curl_close($curl_request);
```

More information on this API endpoint can be found in the [/<module>/:record - DELETE](#) documentation.

Request Payload

The URL sent to the server is shown below:

No payload is sent for this request.

Response

The data received from the server is shown below:

```
{  
  "id": "ab2222df-73da-0e92-6887-5705428f4d68"  
}
```

Download

You can download the full API example [here](#).

Last Modified: 11/27/2017 12:17pm

How to Follow a Record

Overview

A PHP example demonstrating how to follow a record using the v10 `/<module>/:record/subscribe` REST POST endpoint.

Following a Record

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
<?php

$instance_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
    //It is recommended to create your own in Admin > OAuth Keys
    "client_id" => "sugar",
    "client_secret" => "",
    "username" => $username,
    "password" => $password,
    //platform type - default is base.
    //It is recommend to change the platform to a custom name such as
    "custom_api" to avoid authentication conflicts.
    "platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION,
CURLOPT_HTTP_VERSION_1_0);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));
```

```
//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the [How to Authenticate and Log Out example](#) and [/oauth2/logout endpoint documentation](#).

Following a Record

Next, we can follow a specific record using the `/<module>/:record/subscribe` endpoint.

```
//Subscribe to record - POST //:record/subscribe

$subscribe_url = $instance_url .
"/Accounts/ae8b1783-404a-fcb8-1ele-56f1cc52cd1a/subscribe";

$subscribe_request = curl_init($subscribe_url);

curl_setopt($subscribe_request, CURLOPT_POST, 1);
curl_setopt($subscribe_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
curl_setopt($subscribe_request, CURLOPT_HEADER, false);
curl_setopt($subscribe_request, CURLOPT_SSL_VERIFYPEER, 0);
```

```
curl_setopt($subscribe_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($subscribe_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($subscribe_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));
```

```
//execute request
$subscribe_response = curl_exec($subscribe_request);
```

More information on the subscribe API can be found in the [/record/subscribe POST](#) documentation.

Request Payload

The data sent to the server is shown below:

This endpoint does not accept any request arguments.

Response

The data received from the server is shown below:

```
"58f96315-9e75-6562-42e9-5705917d2cdc"
```

Download

You can download the full API example [here](#).

Last Modified: 11/27/2017 12:17pm

How to Unfavorite a Record

Overview

A PHP example demonstrating how to unfavorite a record using the v10 /<module>/:record/unfavorite REST PUT endpoint.

Unfavoriting a Record

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
<?php

$instance_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
    //It is recommended to create your own in Admin > OAuth Keys
```

```
"client_id" => "sugar",
"client_secret" => "",
"username" => $username,
"password" => $password,
//platform type - default is base.
//It is recommend to change the platform to a custom name such as
"custom_api" to avoid authentication conflicts.
"platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION,
CURLOPT_HTTP_VERSION_1_0);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the [How to Authenticate and](#)

Log Out example and `/oauth2/logout` endpoint documentation.

Unfavoriting a Record

Next, we can unfavorite a specific record using the `/<module>/:record/unfavorite` endpoint.

```
//Unfavorite record - PUT //:record/unfavorite

$unfavorite_url = $instance_url .
"/Accounts/ae8b1783-404a-fcb8-1e1e-56f1cc52cd1a/unfavorite";

$unfavorite_request = curl_init($unfavorite_url);

curl_setopt($unfavorite_request, CURLOPT_CUSTOMREQUEST, "PUT");
curl_setopt($unfavorite_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
curl_setopt($unfavorite_request, CURLOPT_HEADER, false);
curl_setopt($unfavorite_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($unfavorite_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($unfavorite_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($unfavorite_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//execute request
$unfavorite_response = curl_exec($unfavorite_request);
```

More information on the unfavorite API can be found in the [/<module>/:record/unfavorite PUT](#) documentation.

Request Payload

The data sent to the server is shown below:

This endpoint does not accept any request arguments.

Response

The data received from the server is shown below:

```
{
  "id": "ae8b1783-404a-fcb8-1e1e-56f1cc52cd1a",
  "name": "Union Bank",
  "date_entered": "2016-03-22T17:49:50-05:00",
  "date_modified": "2016-03-30T17:44:20-05:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "modified_user_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
      "fields": [],
      "delete": "no",
      "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
    }
  },
  "created_by": "1",
  "created_by_name": "Administrator",
  "created_by_link": {
    "full_name": "Administrator",
    "id": "1",
```

```
    "_acl": {
      "fields": [],
      "delete": "no",
      "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
    }
  },
  "description": "",
  "deleted": false,
  "facebook": "",
  "twitter": "",
  "googleplus": "",
  "account_type": "Customer",
  "industry": "Banking",
  "annual_revenue": "",
  "phone_fax": "",
  "billing_address_street": "67321 West Siam St.",
  "billing_address_street_2": "",
  "billing_address_street_3": "",
  "billing_address_street_4": "",
  "billing_address_city": "Ohio",
  "billing_address_state": "CA",
  "billing_address_postalcode": "25159",
  "billing_address_country": "USA",
  "rating": "",
  "phone_office": "(065) 489-6104",
  "phone_alternate": "",
  "website": "www.qahr.edu",
  "ownership": "",
  "employees": "",
  "ticker_symbol": "",
  "shipping_address_street": "67321 West Siam St.",
  "shipping_address_street_2": "",
  "shipping_address_street_3": "",
```

```
"shipping_address_street_4": "",
"shipping_address_city": "Ohio",
"shipping_address_state": "CA",
"shipping_address_postalcode": "25159",
"shipping_address_country": "USA",
"parent_id": "",
"sic_code": "",
"duns_num": "",
"parent_name": "",
"member_of": {
  "name": "",
  "id": "",
  "_acl": {
    "fields": [],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"campaign_id": "",
"campaign_name": "",
"campaign_accounts": {
  "name": "",
  "id": "",
  "_acl": {
    "fields": [],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"following": true,
"my_favorite": false,
"tag": [],
"assigned_user_id": "seed_sarah_id",
"assigned_user_name": "Sarah Smith",
"assigned_user_link": {
```

```
    "full_name": "Sarah Smith",
    "id": "seed_sarah_id",
    "_acl": {
      "fields": [],
      "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
  },
  "team_count": "",
  "team_count_link": {
    "team_count": "",
    "id": "West",
    "_acl": {
      "fields": [],
      "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
  },
  "team_name": [{
    "id": "East",
    "name": "East",
    "name_2": "",
    "primary": false
  }, {
    "id": 1,
    "name": "Global",
    "name_2": "",
    "primary": false
  }, {
    "id": "West",
    "name": "West",
    "name_2": "",
    "primary": true
  }],
  "email": [{
```

```
    "email_address": "hr.support.kid@example.info",
    "invalid_email": false,
    "opt_out": false,
    "primary_address": true,
    "reply_to_address": false
  }, {
    "email_address": "info.support.the@example.com",
    "invalid_email": false,
    "opt_out": false,
    "primary_address": false,
    "reply_to_address": false
  }],
  "email1": "hr.support.kid@example.info",
  "email2": "info.support.the@example.com",
  "invalid_email": false,
  "email_opt_out": false,
  "email_addresses_non_primary": "",
  "_acl": {
    "fields": {}
  },
  "_module": "Accounts"
}
```

Download

You can download the full API example [here](#).

Last Modified: 11/27/2017 12:17pm

How to Unfollow a Record

Overview

A PHP example demonstrating how to unfollow a record using the v10 /<module>/:record/unsubscribe REST DELETE endpoint.

Unfollowing a Record

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
<?php

$instance_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
    //It is recommended to create your own in Admin > OAuth Keys
    "client_id" => "sugar",
    "client_secret" => "",
    "username" => $username,
    "password" => $password,
    //platform type - default is base.
    //It is recommend to change the platform to a custom name such as
    "custom_api" to avoid authentication conflicts.
```

```
    "platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION,
CURLOPT_HTTP_VERSION_1_0);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the [How to Authenticate and Log Out example](#) and [/oauth2/logout endpoint documentation](#).

Unfollowing a Record

Next, we can unfollow a specific record using the `/<module>/:record/unsubscribe` endpoint.

```
//Unfollow a record - DELETE /<module>/:record/unsubscribe

$unsubscribe_url = $instance_url .
"/Accounts/ae8b1783-404a-fcb8-1e1e-56f1cc52cd1a/unsubscribe";

$unsubscribe_request = curl_init($unsubscribe_url);

curl_setopt($unsubscribe_request, CURLOPT_CUSTOMREQUEST, "DELETE");
curl_setopt($unsubscribe_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
curl_setopt($unsubscribe_request, CURLOPT_HEADER, false);
curl_setopt($unsubscribe_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($unsubscribe_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($unsubscribe_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($unsubscribe_request, CURLOPT_HTTPHEADER, array(
  "Content-Type: application/json",
  "oauth-token: {$oauth_token}"
));

//execute request
$unsubscribe_response = curl_exec($unsubscribe_request);
```

More information on the unsubscribe API can be found in the [/:record/unsubscribe DELETE](#) documentation.

Request Payload

The data sent to the server is shown below:

This endpoint does not accept any request arguments.

Response

The data received from the server is shown below:

```
true
```

Download

You can download the full API example [here](#).

Last Modified: 10/14/2016 04:12pm

How to Get the Most Active Users

Overview

A PHP example demonstrating how to fetch the most active users for meetings, calls, inbound emails, and outbound emails using the v10 /mostactiveusers REST GET endpoint.

Get Most Active Users

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
<?php
```

```
$instance_url = "http://{site_url}/rest/v10";
```

```
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
    //It is recommended to create your own in Admin > OAuth Keys
    "client_id" => "sugar",
    "client_secret" => "",
    "username" => $username,
    "password" => $password,
    //platform type - default is base.
    //It is recommend to change the platform to a custom name such as
    "custom_api" to avoid authentication conflicts.
    "platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
```

```
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the [How to Authenticate and Log Out example](#) and [/oauth2/logout endpoint documentation](#).

Active Users

Next, we can retrieve the most active users using the `/mostactiveusers` endpoint.

```
//Fetch users - GET /mostactiveusers

$most_active_arguments = array(
    "days" => 30,
);

$most_active_url = $instance_url . "/mostactiveusers";

$most_active_url .= "?" . http_build_query($most_active_arguments);

$most_active_request = curl_init($most_active_url);

curl_setopt($most_active_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
curl_setopt($most_active_request, CURLOPT_HEADER, false);
curl_setopt($most_active_request, CURLOPT_SSL_VERIFYPEER, 0);
```

```
curl_setopt($most_active_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($most_active_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($most_active_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//execute request
$most_active_response = curl_exec($most_active_request);

//decode json
$most_active_response_obj = json_decode($most_active_response);
```

More information on the mostactiveusers API can be found in the [/mostactiveusers](#)

Request

The URL sent to the server is shown below:

```
http://{site_url}/rest/v10/mostactiveusers?days=30
```

Response

The data received from the server is shown below:

```
{
  "meetings": {
    "user_id": "seed_max_id",
    "count": "21",
    "first_name": "Max",
    "last_name": "Jensen"
```

```
},
"calls": {
  "user_id": "seed_chris_id",
  "count": "4",
  "first_name": "Chris",
  "last_name": "Olliver"
},
"inbound_emails": {
  "user_id": "seed_chris_id",
  "count": "23",
  "first_name": "Chris",
  "last_name": "Olliver"
},
"outbound_emails": {
  "user_id": "seed_sarah_id",
  "count": "20",
  "first_name": "Sarah",
  "last_name": "Smith"
}
}
```

Download

You can download the full API example [here](#).

Last Modified: 10/14/2016 04:12pm

How to Authenticate and Log Out

Overview

A PHP example on how to authenticate and logout of the v10 REST API using the `/oauth2/token` and `/oauth2/logout` POST endpoints.

Authenticating

The example below demonstrates how to authenticate to the REST v10 API. It is important to note that the `oauth2` token arguments takes in several parameters that you should be aware of. The `platform` argument tends to cause confusion in that it is used to authenticate a user to a specific platform. Since Sugar only allows 1 user in the system at a time per platform, authenticating an integration script with a platform type of "base" will logout any current users in the system using those credentials. To work around this, your custom scripts should have a new platform type specified such as "custom_api" or any other static text you prefer. The `client_id` and `client_secret` parameters can be used for additional security based on client types. You can create your own client type in Admin > OAuth Keys. More information can be found in the `/oauth2/token` documentation. An example script is shown below:

```
<?php

$instance_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
$sauth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
```

```
//It is recommended to create your own in Admin > OAuth Keys
"client_id" => "sugar",
"client_secret" => "",
"username" => $username,
"password" => $password,
//platform type - default is base.
//It is recommend to change the platform to a custom name such as
"custom_api" to avoid authentication conflicts.
"platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);
```

Request Payload

The data sent to the server is shown below:

```
{
  "grant_type": "password",
  "client_id": "sugar",
  "client_secret": "",
  "username": "admin",
  "password": "password",
  "platform": "custom_api"
}
```

Response

The data received from the server is shown below:

```
{
  "access_token": "c6d495c9-bb25-81d2-5f81-533ef6479f9b",
  "expires_in": 3600,
  "token_type": "bearer",
  "scope": null,
  "refresh_token": "cbc40e67-12bc-4b56-a1d9-533ef62f2601",
  "refresh_expires_in": 1209600,
  "download_token": "cc5d1a9f-6627-3349-96e5-533ef6b1a493"
}
```

Logout

To log out of a session, a request can be made to the `/oauth2/logout` POST endpoint. More information can be found in the `/oauth2/logout` documentation. An example extending off of the above authentication example is shown below:

```
//Logout - POST /oauth2/logout
```

```
$logout_url = $instance_url . "/oauth2/logout";

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;

$logout_request = curl_init($logout_url);
curl_setopt($logout_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
curl_setopt($logout_request, CURLOPT_HEADER, false);
curl_setopt($logout_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($logout_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($logout_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($logout_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//set empty post fields
curl_setopt($logout_request, CURLOPT_POSTFIELDS, array());

//execute request
$oauth2_logout_response = curl_exec($logout_request);
```

Response

The data received from the server is shown below:

```
{
  "success":true
}
```

Download

You can download the full API example here.

Last Modified: 10/14/2016 04:12pm

How to Ping

Overview

A PHP example demonstrating how to ping using the REST v10 /ping GET endpoint.

Pinging

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
<?php

$instance_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
```

```
//client id - default is sugar.
//It is recommended to create your own in Admin > OAuth Keys
"client_id" => "sugar",
"client_secret" => "",
"username" => $username,
"password" => $password,
//platform type - default is base.
//It is recommend to change the platform to a custom name such as
"custom_api" to avoid authentication conflicts.
"platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the [How to Authenticate and Log Out](#) example and [/oauth2/logout](#) endpoint documentation.

Pinging

Once we have authenticated, we can now ping the system as shown below.

```
//Ping - GET /ping

$ping_url = $instance_url . "/ping";

$ping_request = curl_init($ping_url);
curl_setopt($ping_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
curl_setopt($ping_request, CURLOPT_HEADER, false); //needed to return
file headers
curl_setopt($ping_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($ping_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ping_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($ping_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

$ping_response = curl_exec($ping_request);
```

More information on pinging can be found in the [/ping](#) documentation.

Response

```
"pong"
```

Download

You can download the full API example here.

Last Modified: 10/14/2016 04:12pm

How to Fetch the Current Users Time

Overview

A PHP example demonstrating how to fetch the current users time with the v10 /ping/whattimeisit REST GET endpoint.

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
<?php

$instance_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
```

```
"grant_type" => "password",
//client id - default is sugar.
//It is recommended to create your own in Admin > OAuth Keys
"client_id" => "sugar",
"client_secret" => "",
"username" => $username,
"password" => $password,
//platform type - default is base.
//It is recommend to change the platform to a custom name such as
"custom_api" to avoid authentication conflicts.
"platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
```

```
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the [How to Authenticate and Log Out example](#) and [/oauth2/logout endpoint documentation](#).

Getting the Current Time and Date for the User

```
//Set up the time parameters - GET /ping/whattimeisit

$time_url = $instance_url . "/ping/whattimeisit";

$time_request = curl_init($search_url);
curl_setopt($time_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
curl_setopt($time_request, CURLOPT_HEADER, false);
curl_setopt($time_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($time_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($time_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($time_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//execute request
$time_response = curl_exec($time_request);

//decode json
$time_response_obj = json_decode($time_response);
```

Request

```
http://{site_url}/rest/v10/ping/whattimeisit
```

Response

"2014-04-08T14:59:13-04:00"

Downloads

You can download the full API example [here](#)

Last Modified: 10/14/2016 04:12pm

How to Fetch Recently Viewed Records

Overview

A PHP example demonstrating how to retrieve recently viewed records using the v10 /recent REST GET endpoint.

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
<?php
```

```
$instance_url = "http://{site_url}/rest/v10";  
$username = "admin";  
$password = "password";
```

```
//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
    //It is recommended to create your own in Admin > OAuth Keys
    "client_id" => "sugar",
    "client_secret" => "",
    "username" => $username,
    "password" => $password,
    //platform type - default is base.
    //It is recommend to change the platform to a custom name such as
    "custom_api" to avoid authentication conflicts.
    "platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION,
CURLOPT_HTTP_VERSION_1_0);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);
```

```
//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the [How to Authenticate and Log Out](#) example and [/oauth2/logout](#) endpoint documentation.

Recently Viewed Records

Next, we will need to identify the records we want to see using the `/recent` endpoint. In this case we are going to request Accounts, Contacts and Leads.

```
//Set up the search parameters - GET /recent

$recent_url = $instance_url . "/recent";

$recent_arguments = array(
    "module_list" => 'Accounts,Contacts,Leads',
);

//As this request is a GET we will add the arguments to the URL
$recent_url .= "?" . http_build_query($recent_arguments);

$recent_request = curl_init($recent_url);
curl_setopt($recent_request, CURLOPT_HTTP_VERSION,
CURLOPT_HTTP_VERSION_1_0);
curl_setopt($recent_request, CURLOPT_HEADER, false);
curl_setopt($recent_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($recent_request, CURLOPT_RETURNTRANSFER, 1);
```

```
curl_setopt($recent_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($recent_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//execute request
$recent_response = curl_exec($recent_request);

//decode json
$recent_response_obj = json_decode($recent_response);
```

More information on the search API can be found in the [/recent](#) documentation.

Request

```
http://{site_url}/rest/v10/recent?module_list=Accounts%2CContacts%2CLEads
```

Response

The data received from the server is shown below:

```
{
  "next_offset": -1,
  "records": [
    {
      "my_favorite": false,
      "following": false,
      "id": "f3951f4d-2d17-7939-c5ec-56fedbb9e92f",
      "name": "Talia Knupp",
    }
  ]
}
```

```
"date_entered": "2016-04-01T15:34:00-05:00",
"date_modified": "2016-04-06T10:33:24-05:00",
"modified_user_id": "1",
"modified_by_name": "Administrator",
"created_by": "1",
"created_by_name": "Administrator",
"doc_owner": "",
"user_favorites": "",
"description": "",
"deleted": false,
"assigned_user_id": "seed_will_id",
"assigned_user_name": "Will Westin",
"team_count": "",
"team_name": [
  {
    "id": "East",
    "name": "East",
    "name_2": "",
    "primary": true
  },
  {
    "id": "West",
    "name": "West",
    "name_2": "",
    "primary": false
  }
],
"email": [
  {
    "email_address": "jsmith@sugar.com",
    "invalid_email": false,
    "opt_out": false,
    "primary_address": true,
```

```
        "reply_to_address":false
    },
    {
        "email_address":"cjsmith@sugar.com",
        "invalid_email":false,
        "opt_out":false,
        "primary_address":false,
        "reply_to_address":false
    }
],
"email1":"jsmith@sugar.com",
"email2":"cjsmith@sugar.com",
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"","
"salutation":"","
"first_name":"Talia",
"last_name":"Knupp",
"full_name":"Talia Knupp",
"title":"Senior Product Manager",
"facebook":"","
"twitter":"","
"googleplus":"","
"department":"","
"do_not_call":false,
"phone_home":"(963) 741-3689",
"phone_mobile":"(600) 831-9872",
"phone_work":"(680) 991-2837",
"phone_other":"","
"phone_fax":"","
"primary_address_street":"111 Silicon Valley Road",
"primary_address_street_2":"","
"primary_address_street_3":"","
```

```
"primary_address_city": "Sunnyvale",
"primary_address_state": "NY",
"primary_address_postalcode": "99452",
"primary_address_country": "USA",
"alt_address_street": "",
"alt_address_street_2": "",
"alt_address_street_3": "",
"alt_address_city": "",
"alt_address_state": "",
"alt_address_postalcode": "",
"alt_address_country": "",
"assistant": "",
"assistant_phone": "",
"picture": "",
"converted": false,
"referred_by": "",
"lead_source": "Word of mouth",
"lead_source_description": "",
"status": "In Process",
"status_description": "",
"reports_to_id": "",
"report_to_name": "",
"dnb_principal_id": "",
"account_name": "First National S\B",
"account_description": "",
"contact_id": "",
"contact_name": "",
"account_id": "",
"opportunity_id": "",
"opportunity_name": "",
"opportunity_amount": "",
"campaign_id": "",
"campaign_name": ""
```

```
"c_accept_status_fields":"","  
"m_accept_status_fields":"","  
"accept_status_id":"","  
"accept_status_name":"","  
"accept_status_calls":"","  
"accept_status_meetings":"","  
"webtolead_email1":[  
  {  
    "email_address":"jsmith@sugar.com",  
    "invalid_email":false,  
    "opt_out":false,  
    "primary_address":true,  
    "reply_to_address":false  
  },  
  {  
    "email_address":"cjsmith@sugar.com",  
    "invalid_email":false,  
    "opt_out":false,  
    "primary_address":false,  
    "reply_to_address":false  
  }  
],  
"webtolead_email2":[  
  {  
    "email_address":"jsmith@sugar.com",  
    "invalid_email":false,  
    "opt_out":false,  
    "primary_address":true,  
    "reply_to_address":false  
  },  
  {  
    "email_address":"cjsmith@sugar.com",  
    "invalid_email":false,
```

```
        "opt_out":false,
        "primary_address":false,
        "reply_to_address":false
    }
],
"webtolead_email_opt_out":"","
"webtolead_invalid_email":"","
"birthdate":"","
"portal_name":"","
"portal_app":"","
"website":"","
"preferred_language":"","
"mkto_sync":false,
"mkto_id":null,
"mkto_lead_score":null,
"fruits_c":"Apples",
"_acl":{
    "fields":{

    }
},
"_module":"Leads",
"_last_viewed_date":"2016-04-06T10:33:24-05:00"
},
{
    "my_favorite":false,
    "following":false,
    "id":"e8c641ca-1b8c-74c1-d08d-56fedbdd3187",
    "name":"MTM Investment Bank F S B",
    "date_entered":"2016-04-01T15:34:00-05:00",
    "date_modified":"2016-04-06T10:16:52-05:00",
    "modified_user_id":"1",
    "modified_by_name":"Administrator",
```

```
"created_by": "1",
"created_by_name": "Administrator",
"doc_owner": "",
"user_favorites": "",
"description": "",
"deleted": false,
"assigned_user_id": "seed_will_id",
"assigned_user_name": "Will Westin",
"team_count": "",
"team_name": [
  {
    "id": "East",
    "name": "East",
    "name_2": "",
    "primary": true
  },
  {
    "id": "West",
    "name": "West",
    "name_2": "",
    "primary": false
  }
],
"email": [
  {
    "email_address": "jsmith@sugar.com",
    "invalid_email": false,
    "opt_out": false,
    "primary_address": true,
    "reply_to_address": false
  },
  {
    "email_address": "the60@example.us",
```

```
        "invalid_email":false,
        "opt_out":false,
        "primary_address":false,
        "reply_to_address":false
    }
],
"email1":"jsmith@sugar.com",
"email2":"the60@example.us",
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"","
"facebook":"","
"twitter":"","
"googleplus":"","
"account_type":"Customer",
"industry":"a",
"annual_revenue":"","
"phone_fax":"","
"billing_address_street":"67321 West Siam St.",
"billing_address_street_2":"","
"billing_address_street_3":"","
"billing_address_street_4":"","
"billing_address_city":"Alabama",
"billing_address_state":"NY",
"billing_address_postalcode":"52272",
"billing_address_country":"USA",
"rating":"","
"phone_office":"(012) 704-8075",
"phone_alternate":"","
"website":"www.salesqa.it",
"ownership":"","
"employees":"","
"ticker_symbol":"","
```

```
"shipping_address_street": "67321 West Siam St.",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "Alabama",
"shipping_address_state": "NY",
"shipping_address_postalcode": "52272",
"shipping_address_country": "USA",
"parent_id": "",
"sic_code": "",
"duns_num": "",
"parent_name": "",
"campaign_id": "",
"campaign_name": "",
"_acl": {
  "fields": {

  }
},
"_module": "Accounts",
"_last_viewed_date": "2016-04-06T10:16:52-05:00"
},
{
  "my_favorite": false,
  "following": false,
  "id": "f31b2f00-468c-3d35-1e88-56fedbd3921d",
  "name": "Kaycee Gibney",
  "date_entered": "2016-04-01T15:34:00-05:00",
  "date_modified": "2016-04-06T10:16:24-05:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "created_by": "1",
  "created_by_name": "Administrator",
```

```
"doc_owner": "",
"user_favorites": "",
"description": "",
"deleted": false,
"assigned_user_id": "seed_jim_id",
"assigned_user_name": "Jim Brennan",
"team_count": "",
"team_name": [
  {
    "id": "East",
    "name": "East",
    "name_2": "",
    "primary": true
  }
],
"email": [
  {
    "email_address": "jsmith@sugar.com",
    "invalid_email": false,
    "opt_out": false,
    "primary_address": true,
    "reply_to_address": false
  },
  {
    "email_address": "sales.kid.dev@example.info",
    "invalid_email": false,
    "opt_out": true,
    "primary_address": false,
    "reply_to_address": false
  }
],
"email1": "jsmith@sugar.com",
"email2": "sales.kid.dev@example.info",
```

```
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"","
"salutation":"","
"first_name":"Kaycee",
"last_name":"Gibney",
"full_name":"Kaycee Gibney",
"title":"Mgr Operations",
"facebook":"","
"twitter":"","
"googleplus":"","
"department":"","
"do_not_call":false,
"phone_home":"(599) 165-2396",
"phone_mobile":"(215) 591-9574",
"phone_work":"(771) 945-3648",
"phone_other":"","
"phone_fax":"","
"primary_address_street":"321 University Ave.",
"primary_address_street_2":"","
"primary_address_street_3":"","
"primary_address_city":"Santa Monica",
"primary_address_state":"NY",
"primary_address_postalcode":"96154",
"primary_address_country":"USA",
"alt_address_street":"","
"alt_address_street_2":"","
"alt_address_street_3":"","
"alt_address_city":"","
"alt_address_state":"","
"alt_address_postalcode":"","
"alt_address_country":"","
"assistant":"","
```

```
"assistant_phone": "",
"picture": "",
"email_and_name1": "",
"lead_source": "Existing Customer",
"account_name": "Tracker Com LP",
"account_id": "72ad6f00-e345-1cab-b370-56fedbd23deb",
"dnb_principal_id": "",
"opportunity_role_fields": "",
"opportunity_role_id": "",
"opportunity_role": "",
"reports_to_id": "",
"report_to_name": "",
"birthdate": "",
"portal_name": "KayceeGibney33",
"portal_active": true,
"portal_password": true,
"portal_password1": null,
"portal_app": "",
"preferred_language": "en_us",
"campaign_id": "",
"campaign_name": "",
"c_accept_status_fields": "",
"m_accept_status_fields": "",
"accept_status_id": "",
"accept_status_name": "",
"accept_status_calls": "",
"accept_status_meetings": "",
"sync_contact": false,
"mkto_sync": false,
"mkto_id": null,
"mkto_lead_score": null,
"_acl": {
  "fields": {
```

```
    }
  },
  "_module": "Contacts",
  "_last_viewed_date": "2016-04-06T10:16:24-05:00"
}
]
}
```

There are 3 records shown above, the `_module` field tells you if the record is from Accounts, Contacts or Leads. The `_last_viewed_date` will tell you when the record was last seen.

Downloads

You can download the full API example [here](#)

Last Modified: 10/14/2016 04:12pm

How to Use the Global Search

Overview

A PHP example demonstrating how to globally search for records using the REST v10 `/search` GET endpoint.

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
<?php

$instance_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
    //It is recommended to create your own in Admin > OAuth Keys
    "client_id" => "sugar",
    "client_secret" => "",
    "username" => $username,
    "password" => $password,
    //platform type - default is base.
    //It is recommend to change the platform to a custom name such as
    "custom_api" to avoid authentication conflicts.
    "platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
```

```
        "Content-Type: application/json"
    ));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the [How to Authenticate and Log Out example](#) and [/oauth2/logout endpoint documentation](#).

Searching Records

So, we will need to identify the records we want to see using the `/search` endpoint. In this case, we are going to search for all records that have the email address of `'jsmith@sugar.com'`. In this example, there are 3 records, an Account, a Contact and a Lead.

```
//Set up the search parameters - GET /search

$search_url = $instance_url . "/search";

$search_arguments = array(
    "q" => 'jsmith@sugar.com',
    "max_num" => 3,
    "offset" => 0,
```

```
    "fields" => "",
    "order_by" => "",
    "favorites" => false,
    "my_items" => false,
);

//As this request is a GET we will add the arguments to the URL
$search_url .= "?" . http_build_query($search_arguments);

$search_request = curl_init($search_url);
curl_setopt($search_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
curl_setopt($search_request, CURLOPT_HEADER, false);
curl_setopt($search_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($search_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($search_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($search_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//execute request
$search_response = curl_exec($search_request);

//decode json
$search_response_obj = json_decode($search_response);
```

More information on the search API can be found in the [/search](#) documentation.

Request

```
http://{site_url}/rest/v10/search?q=jsmith%40sugar.com&max_num=3&offset=0&fields=&order_by=&favorites=0&my_items=0
```

Response

The data received from the server is shown below:

```
{
  "next_offset":-1,
  "records":[
    {
      "my_favorite":false,
      "following":false,
      "id":"f31b2f00-468c-3d35-1e88-56fedbd3921d",
      "name":"Kaycee Gibney",
      "date_entered":"2016-04-01T20:34:00+00:00",
      "date_modified":"2016-04-06T15:16:24+00:00",
      "modified_user_id":"1",
      "modified_by_name":"Administrator",
      "created_by":"1",
      "created_by_name":"Administrator",
      "doc_owner":"",
      "user_favorites":"",
      "description":"",
      "deleted":false,
      "assigned_user_id":"seed_jim_id",
      "assigned_user_name":"Jim Brennan",
      "team_count":"",
      "team_name":[
        {
          "id":"East",
          "name":"East",
          "name_2":""
        }
      ]
    }
  ]
}
```

```
        "primary":true
    }
],
"email":[
    {
        "email_address":"jsmith@sugar.com",
        "invalid_email":false,
        "opt_out":false,
        "primary_address":true,
        "reply_to_address":false
    },
    {
        "email_address":"sales.kid.dev@example.info",
        "invalid_email":false,
        "opt_out":true,
        "primary_address":false,
        "reply_to_address":false
    }
],
"email1":"jsmith@sugar.com",
"email2":"sales.kid.dev@example.info",
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"","
"salutation":"","
"first_name":"Kaycee",
"last_name":"Gibney",
"full_name":"Kaycee Gibney",
"title":"Mgr Operations",
"facebook":"","
"twitter":"","
"googleplus":"","
"department":"","
```

```
"do_not_call":false,
"phone_home":"(599) 165-2396",
"phone_mobile":"(215) 591-9574",
"phone_work":"(771) 945-3648",
"phone_other":"","
"phone_fax":"","
"primary_address_street":"321 University Ave.",
"primary_address_street_2":"","
"primary_address_street_3":"","
"primary_address_city":"Santa Monica",
"primary_address_state":"NY",
"primary_address_postalcode":"96154",
"primary_address_country":"USA",
"alt_address_street":"","
"alt_address_street_2":"","
"alt_address_street_3":"","
"alt_address_city":"","
"alt_address_state":"","
"alt_address_postalcode":"","
"alt_address_country":"","
"assistant":"","
"assistant_phone":"","
"picture":"","
"email_and_name1":"","
"lead_source":"Existing Customer",
"account_name":"Tracker Com LP",
"account_id":"72ad6f00-e345-1cab-b370-56fedbd23deb",
"dnb_principal_id":"","
"opportunity_role_fields":"","
"opportunity_role_id":"","
"opportunity_role":"","
"reports_to_id":"","
"report_to_name":"","
```

```
"birthdate": "",
"portal_name": "KayceeGibney33",
"portal_active": true,
"portal_password": true,
"portal_password1": null,
"portal_app": "",
"preferred_language": "en_us",
"campaign_id": "",
"campaign_name": "",
"c_accept_status_fields": "",
"m_accept_status_fields": "",
"accept_status_id": "",
"accept_status_name": "",
"accept_status_calls": "",
"accept_status_meetings": "",
"sync_contact": false,
"mkto_sync": false,
"mkto_id": null,
"mkto_lead_score": null,
"_acl": {
  "fields": {

  }
},
"_module": "Contacts",
"_search": {
  "score": 0.70710677,
  "highlighted": {
    "email1": {

"text": "\u003Cstrong\u003Ejsmith@sugar.com\u003C/strong\u003E",
      "module": "Contacts",
      "label": "LBL_EMAIL_ADDRESS"
```

```
    }
  }
},
{
  "my_favorite":false,
  "following":false,
  "id":"e8c641ca-1b8c-74c1-d08d-56fedbdd3187",
  "name":"MTM Investment Bank F S B",
  "date_entered":"2016-04-01T20:34:00+00:00",
  "date_modified":"2016-04-06T15:16:52+00:00",
  "modified_user_id":"1",
  "modified_by_name":"Administrator",
  "created_by":"1",
  "created_by_name":"Administrator",
  "doc_owner":"","
  "user_favorites":"","
  "description":"","
  "deleted":false,
  "assigned_user_id":"seed_will_id",
  "assigned_user_name":"Will Westin",
  "team_count":"","
  "team_name":[
    {
      "id":"East",
      "name":"East",
      "name_2":"","
      "primary":true
    },
    {
      "id":"West",
      "name":"West",
      "name_2":"","
```

```
        "primary":false
    }
],
"email":[
    {
        "email_address":"jsmith@sugar.com",
        "invalid_email":false,
        "opt_out":false,
        "primary_address":true,
        "reply_to_address":false
    },
    {
        "email_address":"the60@example.us",
        "invalid_email":false,
        "opt_out":false,
        "primary_address":false,
        "reply_to_address":false
    }
],
"email1":"jsmith@sugar.com",
"email2":"the60@example.us",
"invalid_email":false,
"email_opt_out":false,
"email_addresses_non_primary":"","
"facebook":"","
"twitter":"","
"googleplus":"","
"account_type":"Customer",
"industry":"a",
"annual_revenue":"","
"phone_fax":"","
"billing_address_street":"67321 West Siam St.",
"billing_address_street_2":"","
```

```
"billing_address_street_3": "",
"billing_address_street_4": "",
"billing_address_city": "Alabama",
"billing_address_state": "NY",
"billing_address_postalcode": "52272",
"billing_address_country": "USA",
"rating": "",
"phone_office": "(012) 704-8075",
"phone_alternate": "",
"website": "www.salesqa.it",
"ownership": "",
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "67321 West Siam St.",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "Alabama",
"shipping_address_state": "NY",
"shipping_address_postalcode": "52272",
"shipping_address_country": "USA",
"parent_id": "",
"sic_code": "",
"duns_num": "",
"parent_name": "",
"campaign_id": "",
"campaign_name": "",
"_acl": {
  "fields": {

  }
},
"_module": "Accounts",
```

```
    "_search":{
      "score":0.70710677,
      "highlighted":{
        "email1":{
"text":"\u003Cstrong\u003Ejsmith@sugar.com\u003C\/strong\u003E",
          "module":"Accounts",
          "label":"LBL_EMAIL_ADDRESS"
        }
      }
    },
  {
    "my_favorite":false,
    "following":false,
    "id":"f3951f4d-2d17-7939-c5ec-56fedbb9e92f",
    "name":"Talia Knupp",
    "date_entered":"2016-04-01T20:34:00+00:00",
    "date_modified":"2016-04-06T15:33:24+00:00",
    "modified_user_id":"1",
    "modified_by_name":"Administrator",
    "created_by":"1",
    "created_by_name":"Administrator",
    "doc_owner":"","",
    "user_favorites":"","",
    "description":"","",
    "deleted":false,
    "assigned_user_id":"seed_will_id",
    "assigned_user_name":"Will Westin",
    "team_count":"","",
    "team_name":[
      {
        "id":"East",
```

```
        "name": "East",
        "name_2": "",
        "primary": true
    },
    {
        "id": "West",
        "name": "West",
        "name_2": "",
        "primary": false
    }
],
"email": [
    {
        "email_address": "jsmith@sugar.com",
        "invalid_email": false,
        "opt_out": false,
        "primary_address": true,
        "reply_to_address": false
    },
    {
        "email_address": "cjsmith@sugar.com",
        "invalid_email": false,
        "opt_out": false,
        "primary_address": false,
        "reply_to_address": false
    }
],
"email1": "jsmith@sugar.com",
"email2": "cjsmith@sugar.com",
"invalid_email": false,
"email_opt_out": false,
"email_addresses_non_primary": "",
"salutation": "",
```

```
"first_name":"Talia",
"last_name":"Knupp",
"full_name":"Talia Knupp",
"title":"Senior Product Manager",
"facebook":"",
"twitter":"",
"googleplus":"",
"department":"",
"do_not_call":false,
"phone_home":"(963) 741-3689",
"phone_mobile":"(600) 831-9872",
"phone_work":"(680) 991-2837",
"phone_other":"",
"phone_fax":"",
"primary_address_street":"111 Silicon Valley Road",
"primary_address_street_2":"",
"primary_address_street_3":"",
"primary_address_city":"Sunnyvale",
"primary_address_state":"NY",
"primary_address_postalcode":"99452",
"primary_address_country":"USA",
"alt_address_street":"",
"alt_address_street_2":"",
"alt_address_street_3":"",
"alt_address_city":"",
"alt_address_state":"",
"alt_address_postalcode":"",
"alt_address_country":"",
"assistant":"",
"assistant_phone":"",
"picture":"",
"converted":false,
"referred_by":"",
```

```
"lead_source": "Word of mouth",
"lead_source_description": "",
"status": "In Process",
"status_description": "",
"reports_to_id": "",
"report_to_name": "",
"dnb_principal_id": "",
"account_name": "First National S\B",
"account_description": "",
"contact_id": "",
"contact_name": "",
"account_id": "",
"opportunity_id": "",
"opportunity_name": "",
"opportunity_amount": "",
"campaign_id": "",
"campaign_name": "",
"c_accept_status_fields": "",
"m_accept_status_fields": "",
"accept_status_id": "",
"accept_status_name": "",
"accept_status_calls": "",
"accept_status_meetings": "",
"webtolead_email1": [
  {
    "email_address": "jsmith@sugar.com",
    "invalid_email": false,
    "opt_out": false,
    "primary_address": true,
    "reply_to_address": false
  },
  {
    "email_address": "cjsmith@sugar.com",
```

```
        "invalid_email":false,
        "opt_out":false,
        "primary_address":false,
        "reply_to_address":false
    }
],
"webtolead_email2":[
    {
        "email_address":"jsmith@sugar.com",
        "invalid_email":false,
        "opt_out":false,
        "primary_address":true,
        "reply_to_address":false
    },
    {
        "email_address":"cjsmith@sugar.com",
        "invalid_email":false,
        "opt_out":false,
        "primary_address":false,
        "reply_to_address":false
    }
],
"webtolead_email_opt_out":"","
"webtolead_invalid_email":"","
"birthdate":"","
"portal_name":"","
"portal_app":"","
"website":"","
"preferred_language":"","
"mkto_sync":false,
"mkto_id":null,
"mkto_lead_score":null,
"fruits_c":"Apples",
```

```
    "_acl":{
      "fields":{

      }
    },
    "_module": "Leads",
    "_search":{
      "score":0.70710677,
      "highlighted":{
        "email1":{

        }
      }
    }
  }
]
}
```

There are 3 records shown above, the `_module` field tells you if the record is from Accounts, Contacts or Leads. The `_search` field is an array that tells you where in the record it found the search string. It gives you back a highlighted string that you can use for display.

Downloads

You can download the full API example [here](#)

Last Modified: 11/27/2017 12:17pm

How to Check for Duplicate Records

Overview

An PHP example demonstrating how to check for duplicate records using the v10 /<module>/duplicateCheck REST POST endpoint.

Duplicate Records

Authenticating

First, you will need to authenticate to the Sugar API. An example is shown below:

```
<?php

$instance_url = "http://{site_url}/rest/v10";
$username = "admin";
$password = "password";

//Login - POST /oauth2/token
$auth_url = $instance_url . "/oauth2/token";

$oauth2_token_arguments = array(
    "grant_type" => "password",
    //client id - default is sugar.
    //It is recommended to create your own in Admin > OAuth Keys
    "client_id" => "sugar",
```

```
"client_secret" => "",
"username" => $username,
"password" => $password,
//platform type - default is base.
//It is recommend to change the platform to a custom name such as
"custom_api" to avoid authentication conflicts.
"platform" => "custom_api"
);

$auth_request = curl_init($auth_url);
curl_setopt($auth_request, CURLOPT_HTTP_VERSION,
CURL_HTTP_VERSION_1_0);
curl_setopt($auth_request, CURLOPT_HEADER, false);
curl_setopt($auth_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($auth_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($auth_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($auth_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json"
));

//convert arguments to json
$json_arguments = json_encode($oauth2_token_arguments);
curl_setopt($auth_request, CURLOPT_POSTFIELDS, $json_arguments);

//execute request
$oauth2_token_response = curl_exec($auth_request);

//decode oauth2 response to get token
$oauth2_token_response_obj = json_decode($oauth2_token_response);
$oauth_token = $oauth2_token_response_obj->access_token;
```

More information on authenticating can be found in the [How to Authenticate and Log Out](#) example and [/oauth2/logout](#) endpoint documentation.

Retrieving Duplicates

Next, we will need to identify the records that are duplicates using the `/<module>/duplicateCheck` endpoint.

```
//Check for duplicate records - POST /<module>/duplicateCheck

$url = $instance_url . "/Accounts/duplicateCheck";
//Set up the Record details
$record = array(
    'name' => 'Test Record',
);

$curl_request = curl_init($url);
curl_setopt($curl_request, CURLOPT_HTTP_VERSION,
    CURL_HTTP_VERSION_1_0);
curl_setopt($curl_request, CURLOPT_HEADER, false);
curl_setopt($curl_request, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($curl_request, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($curl_request, CURLOPT_FOLLOWLOCATION, 0);
curl_setopt($curl_request, CURLOPT_HTTPHEADER, array(
    "Content-Type: application/json",
    "oauth-token: {$oauth_token}"
));

//convert arguments to json
$json_arguments = json_encode($record);
curl_setopt($curl_request, CURLOPT_POSTFIELDS, $json_arguments);
//execute request
$curl_response = curl_exec($curl_request);
//decode json
$createdRecord = json_decode($curl_response);
```

```
//display the created record
print_r($createdRecord);
curl_close($curl_request);
```

More information on the filter API can be found in the [/<module>/duplicateCheck](#) documentation.

Request Payload

The data sent to the server is shown below:

```
{
  "name": "Test Record"
}
```

Response

The data received from the server is shown below:

```
{
  "next_offset": -1,
  "records": [{
    "id": "7f6ea7be-60d6-11e6-8885-a0999b033b33",
    "name": "Test Record",
    "date_entered": "2016-08-12T14:48:25-07:00",
    "date_modified": "2016-08-12T14:48:25-07:00",
    "modified_user_id": "1",
    "modified_by_name": "Administrator",
    "modified_user_link": {
      "full_name": "Administrator",
```

```
    "id": "1",
    "_acl": {
      "fields": [],
      "delete": "no",
      "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
    }
  },
  "created_by": "1",
  "created_by_name": "Administrator",
  "created_by_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
      "fields": [],
      "delete": "no",
      "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
    }
  },
  "description": "Test Data 1",
  "deleted": false,
  "facebook": "",
  "twitter": "",
  "googleplus": "",
  "account_type": "",
  "industry": "",
  "annual_revenue": "",
  "phone_fax": "",
  "billing_address_street": "",
  "billing_address_street_2": "",
  "billing_address_street_3": "",
  "billing_address_street_4": "",
  "billing_address_city": "",
  "billing_address_state": "",
```

```
"billing_address_postalcode": "",
"billing_address_country": "",
"rating": "",
"phone_office": "",
"phone_alternate": "",
"website": "",
"ownership": "",
"employees": "",
"ticker_symbol": "",
"shipping_address_street": "",
"shipping_address_street_2": "",
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "",
"shipping_address_state": "",
"shipping_address_postalcode": "",
"shipping_address_country": "",
"parent_id": "",
"sic_code": "",
"duns_num": "",
"parent_name": "",
"member_of": {
  "name": "",
  "id": "",
  "_acl": {
    "fields": [],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"campaign_id": "",
"campaign_name": "",
"campaign_accounts": {
  "name": "",
```

```
    "id": "",
    "_acl": {
      "fields": [],
      "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
  },
  "following": true,
  "my_favorite": false,
  "tag": [],
  "assigned_user_id": "1",
  "assigned_user_name": "Administrator",
  "assigned_user_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
      "fields": [],
      "delete": "no",
      "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
    }
  },
  "team_count": "",
  "team_count_link": {
    "team_count": "",
    "id": "1",
    "_acl": {
      "fields": [],
      "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
    }
  },
  "team_name": [{
    "id": "1",
    "name": "Global",
    "name_2": "",
```

```
    "primary": true
  }],
  "email": [],
  "email1": "",
  "email2": "",
  "invalid_email": "",
  "email_opt_out": "",
  "email_addresses_non_primary": "",
  "_acl": {
    "fields": {}
  },
  "_module": "Accounts",
  "duplicate_check_rank": 8
}, {
  "id": "868b4f16-60d6-11e6-bdfc-a0999b033b33",
  "name": "Test Record",
  "date_entered": "2016-08-12T14:48:37-07:00",
  "date_modified": "2016-08-12T14:48:37-07:00",
  "modified_user_id": "1",
  "modified_by_name": "Administrator",
  "modified_user_link": {
    "full_name": "Administrator",
    "id": "1",
    "_acl": {
      "fields": [],
      "delete": "no",
      "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
    }
  },
  "created_by": "1",
  "created_by_name": "Administrator",
  "created_by_link": {
    "full_name": "Administrator",
```

```
    "id": "1",
    "_acl": {
      "fields": [],
      "delete": "no",
      "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
    }
  },
  "description": "Test Data 2",
  "deleted": false,
  "facebook": "",
  "twitter": "",
  "googleplus": "",
  "account_type": "",
  "industry": "",
  "annual_revenue": "",
  "phone_fax": "",
  "billing_address_street": "",
  "billing_address_street_2": "",
  "billing_address_street_3": "",
  "billing_address_street_4": "",
  "billing_address_city": "",
  "billing_address_state": "",
  "billing_address_postalcode": "",
  "billing_address_country": "",
  "rating": "",
  "phone_office": "",
  "phone_alternate": "",
  "website": "",
  "ownership": "",
  "employees": "",
  "ticker_symbol": "",
  "shipping_address_street": "",
  "shipping_address_street_2": "",
```

```
"shipping_address_street_3": "",
"shipping_address_street_4": "",
"shipping_address_city": "",
"shipping_address_state": "",
"shipping_address_postalcode": "",
"shipping_address_country": "",
"parent_id": "",
"sic_code": "",
"duns_num": "",
"parent_name": "",
"member_of": {
  "name": "",
  "id": "",
  "_acl": {
    "fields": [],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"campaign_id": "",
"campaign_name": "",
"campaign_accounts": {
  "name": "",
  "id": "",
  "_acl": {
    "fields": [],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"following": true,
"my_favorite": false,
"tag": [],
"assigned_user_id": "1",
"assigned_user_name": "Administrator",
```

```
"assigned_user_link": {
  "full_name": "Administrator",
  "id": "1",
  "_acl": {
    "fields": [],
    "delete": "no",
    "_hash": "8e11bf9be8f04daddee9d08d44ea891e"
  }
},
"team_count": "",
"team_count_link": {
  "team_count": "",
  "id": "1",
  "_acl": {
    "fields": [],
    "_hash": "654d337e0e912edaa00dbb0fb3dc3c17"
  }
},
"team_name": [{
  "id": "1",
  "name": "Global",
  "name_2": "",
  "primary": true
}],
"email": [],
"email1": "",
"email2": "",
"invalid_email": "",
"email_opt_out": "",
"email_addresses_non_primary": "",
"_acl": {
  "fields": {}
},
```

```
        "_module": "Accounts",
        "duplicate_check_rank": 8
    }
}
```

Download

You can download the full API example [here](#).

Last Modified: 10/14/2016 04:12pm

v1 - v4.1

Overview

v1 - v4.1 API documentation.

SOAP VS REST

There are significant differences between how the legacy REST and SOAP protocols function on an implementation level (e.g. Performance, response size, etc). Deciding which protocol to use is up to the individual developer and is beyond the scope of this guide. Starting in SugarCRM version 6.2.0, there are some deviations between the protocols with the v4 API. There are additional core calls that are only made available through the REST protocol. They are listed below:

- `get_module_layout`

-
- `get_module_layout_md5`
 - `get_quotes_pdf` method
 - `get_report_pdf` method
 - `snip_import_emails`
 - `snip_update_contacts`
 - `job_queue_cycle`
 - `job_queue_next`
 - `job_queue_run`
 - `oauth_access` method
 - `oauth_access_token`
 - `oauth_request_token`

REST

REST stands for 'Representational State Transfer'. This protocol is used by Sugar to exchange information both internally and externally.

How do I access the REST service?

The legacy REST services in SugarCRM can be found by navigating to:

```
http://{site url}/service/{version}/rest.php
```

Where 'site url' is the URL of your Sugar instance and 'version' is the latest version of the API specific to your release of Sugar. You can find out more about versioning in the Web Services documentation.

Input / Output Datatypes

The default input / output datatype for REST is JSON / PHP serialize.

These datatype files, SugarRestJSON.php and SugarRestSerialize.php, are in:

```
./service/core/REST/
```

Defining your own Datatypes

You can also define your own datatype. To do this, you need to create a new file such as:

```
./service/core/REST/SugarRest<CustomDataType>.php
```

Next, you will need to override generateResponse() and serve() functions. The serve function decodes or unserializes the appropriate datatype based on the input type; the generateResponse function encodes or serializes it based on the output type.

See service/test.html for more examples on usage. In this file, the getRequestData function, which generates URL with json, is both the input_type and the response_type. That is, the request data from the JavaScript to the server is JSON and response data from server is also JSON. You can mix and match any datatype as input and output. For example, you can have JSON as the input_type and serialize as the response_type based on your application's requirements.

REST Failure Response

If a call failure should occur, the result will be as shown below:

Name	Type	Description
name	String	Error message.

Name	Type	Description
number	Integer	Error number.
description	String	Description of error.

SOAP

SOAP stands for 'Simple Object Access Protocol'. SOAP is a simple XML-based protocol that is used to allow applications to exchange information.

How do I access the SOAP service?

The legacy SOAP service in SugarCRM can 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-parameter. 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.

Name	Type	Description
detail	String	Description of fault.

Last Modified: 11/27/2017 12:17pm

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: 10/14/2016 04:12pm

Methods

Web Service Method Calls.

Last Modified: 10/14/2016 04:12pm

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 available 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

Name	Type	Description
		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: 10/14/2016 04:12pm

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	<code>new_return_document_revision</code> Array	The call result.

Name	Type	Description
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 <code>return_document_revision</code> to <code>new_return_document_revision</code> .

PHP

```
$get_document_revision_parameters = array(
    //Session id
    "session" => $session_id,
```

```
//The attachment details
    "i" => '723b7dcb-27b3-e53d-b348-50bd283f8e48',
);
```

Last Modified: 10/14/2016 04:12pm

get_entries

Overview

Retrieves a list of beans based on specified record IDs.

Available APIs

- SOAP
- REST

Definition

```
get_entries(session, module_name, ids, select_fields, link_name_to_fields_array,
track_view)
```

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
module_name	String	The name of the module from which to retrieve records. Note: This is the modules key which may not be the same as the modules display name.
ids	String	The list of record IDs to retrieve.
select_fields	select_fields Array	The list of fields to be returned in the results. Specifying an empty array will return all fields.
link_name_to_fields_array	link_names_to_fields_array Array	A list of link names and the fields to be returned for each link.
track_view	Boolean	Flag the record as a recently viewed item.

Result

Name	Type	Description
result	get_entry_result_version2 Array	The call result.

Name	Type	Description
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: 10/14/2016 04:12pm

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

Name	Type	Description
		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: 10/14/2016 04:12pm

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: 10/14/2016 04:12pm

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

Name	Type	Description
		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.
result.next_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 <code>get_entry_list_result</code> to <code>get_entry_list_result_version2</code> .
--

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: 10/14/2016 04:12pm

get_language_definition

Overview

Retrieves the language label strings for the specified modules.

Available APIs

-
- REST

Definition

`get_available_modules(session, modules, md5)`

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
modules	Array	The list of modules to retrieve language definitions for.
md5	Boolean	Setting this to true will return an MD5 hash of the modules labels strings instead of the label strings themselves.

Result

Name	Type	Description
result	Array	The call result. Contains a list of modules.

Name	Type	Description
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: 10/14/2016 04:12pm

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 recently viewed records.
result[].id	Integer	The result ID.
result[].item_id	String	The recently viewed record ID.
result[].item_summary	String	The name of the recently viewed record.
result[].module_name	String	The name of the module.
result[].monitor_id	String	The monitor ID from the tracker table.
result[].date_modified	String	The date the record was viewed.

Change Log

Version	Change

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: 10/14/2016 04:12pm

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

Name	Type	Description
		compatibility purposes.
select_fields	select_fields	List of fields to select and return as name/value pairs.
relationship_name	String	The name of the relationship name to search on.
deletion_date	String	Date value in YYYY-MM-DD HH:MM:SS format for filtering on deleted records whose date_modified falls within range.

Result

Name	Type	Description
result	modified_relationship_result Array	The call result.
result.result_count	Integer	The result count.
result.next_offset	Integer	The next offset to retrieve from.
result.entry_list	Array	List of found records.
result.error	Array	Error Message.
result.error.number	Integer	The error number.
result.error.name	String	The name of the error.

Name	Type	Description
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.
'relationship_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: 10/14/2016 04:12pm

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

Name	Type	Description
		modules display name.
fields	select_fields Array	The list of fields to retrieve. An empty parameter will return all.

Result

Name	Type	Description
result	new_module_fields Array	The call result.
result.module_name	String	The name of the module.
result.table_name	String	The name of the modules primary table.
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: 10/14/2016 04:12pm

get_module_fields_md5

Overview

Retrieves the MD5 hash of the vardefs for the specified modules.

Available APIs

- SOAP
- REST

Definition

get_module_fields_md5(session, module_names)

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
module_names	select_fields Array	The list of modules to retrieve MD5 hashes for.

Result

Name	Type	Description
result	md5_results Array	The call result. Contains a list of module field hashes. Order is based on the module_names parameter.

Change Log

Version	Change

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: 10/14/2016 04:12pm

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

Name	Type	Description
		list of modules.
result[[\$type]	Array	The list of types requested.
result[[\$view]	Array or String	The list of layout views requested or MD5 hashes. This is dependent on the '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: 10/14/2016 04:12pm

get_module_layout_md5

Overview

Retrieves the MD5 hash value for a layout given a specific module, type and view.

Available APIs

- REST

Definition

`get_module_layout_md5(session, modules, types, views, acl_check)`

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
modules	Array	The list of modules to retrieve layouts for.
types	Array	The types of views requested. Current supported types are 'default' (for application) and 'wireless'.
views	Array	The views requested. Current supported types are 'edit', 'detail', 'list', and 'subpanel'.
acl_check	Boolean	Whether or not to check for ACL access.

Result

Name	Type	Description

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: 10/14/2016 04:12pm

get_note_attachment

Overview

Retrieves an attachment associated with a specific note record.

Available APIs

- SOAP
- REST

Definition

`get_note_attachment(session, id)`

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
id	String	The ID of the note record to associate the attachment to.

Result

Name	Type	Description
result	<code>new_return_note_attachment</code> 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	String	The related parent ID.

Name	Type	Description
ated_module_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: 10/14/2016 04:12pm

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: 10/14/2016 04:12pm

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

Name	Type	Description
		not be the same as the modules display name.
module_id	String	The ID of the specified module record.
link_field_name	String	The name of the link field for the related module.
related_module_query	String	The list of related record IDs you are relating
related_fields	select_fields Array	An array specifying relationship fields to populate. An example of this is contact_role between Opportunities and Contacts.
related_module_link_name_to_fields_array	link_names_to_fields_array Array	For every related record returned, specify link field names to field information.
deleted	Integer	
order_by	String	The SQL ORDER BY clause without the phrase "order by".
offset	Integer	The record offset from which to start.
limit	Integer	The maximum number of results to return.

Result

Name	Type	Description
result	get_entry_result_version2 Array	The call result.
result.entry_list	Array	The record's name-value pair for the simple datatypes excluding the link field data. If you do not have access to the object, 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: 10/14/2016 04:12pm

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.

Name	Type	Description
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 <code>get_report_entries</code> 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: 10/14/2016 04:12pm

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: 10/14/2016 04:12pm

get_server_info

Overview

Retrieves info about the SugarCRM instance.

Available APIs

- SOAP
- REST

Definition

get_server_info()

Parameters

Name	Type	Description
null	null	No parameters available.

Result

Name	Type	Description
result	get_server_info_result Array	The call result.
result.flavor	String	The flavor of the instance.
result.version	String	The version of the instance.
result.gmt_time	String	The GMT time of the server.

Change Log

Version	Change
v2	Method get_server_info was added to replace get_server_time, get_server_version and get_sugar_flavor.
v2	Method get_server_time was removed.
v2	Method get_server_version was removed.

v2	Method get_sugar_flavor was removed.
----	--------------------------------------

PHP

```
//this method does not have any parameters  
$get_server_info_parameters = array();
```

Last Modified: 10/14/2016 04:12pm

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: 10/14/2016 04:12pm

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: 10/14/2016 04:12pm

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 team

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: 10/14/2016 04:12pm

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

Name	Type	Description
		'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: 10/14/2016 04:12pm

job_queue_next

Overview

Retrieves the next job from the job queue and marks it as 'In Progress'.

Available APIs

- REST

Definition

`job_queue_next(session, clientid)`

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
clientid	String	The client id calling the application. This parameter is of your choosing for the calling application.

Result

Name	Type	Description
result	Array	The call result.
result.results	String	The next job ID.

Name	Type	Description
------	------	-------------

Change Log

Version	Change
v4	Added <code>job_queue_next</code> method.

PHP

```
$job_queue_next_parameters = array(  
    //Session id  
    'session' => $session_id,  
  
    //The ID of the calling application.  
    'clientid' => 'MyAppID',  
);
```

Last Modified: 10/14/2016 04:12pm

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.

Name	Type	Description
result.results	Boolean	The result of the job run.
result.message	String	This is only returned if a failure occurs.

Change Log

Version	Change
v4	Added job_queue_run method.

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: 10/14/2016 04:12pm

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
<code>user_auth</code>	<code>user_auth Array</code>	Contains the parameters to authenticate a user.
<code>user_auth.user_name</code>	String	The user name of your user
<code>user_auth.password</code>	String	The MD5 hash of the users password.

Name	Type	Description
application name	String	The name of the application logging in.
name_value_list	name_value_list Array	Sets the name_value pair. The parameter is used to set values for the 'language' and 'notifyonsave' user settings.
name_value_list.language	String	The language for the user.
name_value_list.notifyonsave	Boolean	Alerts users on new record creations when set to true.

Result

Name	Type	Description
result	entry_value Array	The call result
result.id	String	This is the session id required to make other method calls.
result.module_name	String	Returns the 'Users' module.
result.name_value_list	Array	Authenticated user properties.
result.name_value_list.user_id	String	ID of the authenticated user.

Name	Type	Description
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: 10/14/2016 04:12pm

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: 10/14/2016 04:12pm

oauth_access

Overview

Retrieves the OAuth access token.

Available APIs

- REST

Definition

oauth_access()

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.

Result

Name	Type	Description
result	Array	The call result.
result.id	String	The OAuth access token.

Change Log

Version	Change
v4	Added oauth_access method.

PHP

```
$oauth_access_parameters = array(  
    //Session id  
    'session' => $session_id,
```

) ;

Last Modified: 10/14/2016 04:12pm

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

Name	Type	Description
		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: 10/14/2016 04:12pm

search_by_module

Overview

Searches modules for a string and returns matched records.

Available APIs

- SOAP
- REST

Definition

```
search_by_module(session, search_string, modules, offset, max_results,  
assigned_user_id, select_fields, unified_search_only, favorites)
```

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
search_string	String	The string to search for.
modules	Integer	The list of modules to query.
offset	Integer	The record offset from which to start.
max_results	Integer	The maximum number of records to return.
assigned_user_id	String	Filters records by the assigned user ID. Leave this empty if no filter should be applied.
select_fields	select_fields Array	An array of fields to return. If empty the default return fields will be from the active listviewdefs.
unified_search_only	Boolean	If the search is only against modules participating in the unified search.
favorites	Boolean	If only records marked as favorites should be returned.

Name	Type	Description
------	------	-------------

Result

Name	Type	Description
result	return_search_result Array	Call result.
result.entry_list	Array	The count of records in paged result.
result.entry_list[].name	String	The .name of the module
result.entry_list[].records	Array	A list of name_value lists for each record matched.

Change Log

Version	Change
v3_1	Added unified_search_only parameter.

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: 10/14/2016 04:12pm

set_campaign_merge

Overview

Handles campaign log entry creation for mail-merge activity given a specified campaign.

Available APIs

- SOAP
- REST

Definition

```
set_campaign_merge(session, targets, campaign_id)
```

Parameters

Name	Type	Description
session	String	Session ID returned by a previous call to login.
targets	select_fields Array	A string array of IDs identifying the targets used in the merge. The IDs used in this parameter come from the column 'prospect_lists_prospepects.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: 10/14/2016 04:12pm

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
<code>session</code>	String	Session ID returned by a previous login call.
<code>note</code>	<code>document_revision</code> Array	The file attachment details
<code>note.id</code>	String	The ID of the note record to associate the attachment to.
<code>note.file</code>	String	The binary contents of the file.
<code>note.filename</code>	String	The file name of the file attachment.
<code>note.revision</code>	String	The revision number

Name	Type	Description
------	------	-------------

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: 10/14/2016 04:12pm

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
<code>session</code>	String	Session ID returned by a previous login call.
<code>module_name</code>	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.
<code>name_value_lists</code>	<code>name_value_lists</code> Array	The an array of name/value lists containing the record attributes.

Name	Type	Description
------	------	-------------

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: 10/14/2016 04:12pm

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.

Name	Type	Description
------	------	-------------

Change Log

Version	Change
v2	Return type was changed from <code>set_entry_result</code> to <code>new_set_entry_result</code> .

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: 10/14/2016 04:12pm

set_note_attachment

Overview

Creates an attachment and associated it to a specific note record.

Available APIs

- SOAP
- REST

Definition

`set_note_attachment(session, note)`

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
note	new_note_attachment Array	The file attachment details
note.id	String	The ID of the note record to associate the attachment to.
note.filename	String	The file name of the file attachment.
note.file	String	The binary contents of the file.

Result

Name	Type	Description
result	new_set_entry_result Array	The call result.
result.id	String	The ID of the note record / attachment

Name	Type	Description
------	------	-------------

Change Log

Version	Change
v2	Return type was changed from <code>set_entry_result</code> to <code>new_set_entry_result</code> .
v2	Parameter <code>note</code> type change from <code>note_attachment</code> 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: 10/14/2016 04:12pm

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.

Name	Type	Description
result.failed	Integer	Determines whether or not the relationship failed. This is normally thrown when the parameters <code>module_name</code> or <code>link_field_name</code> are incorrect.
result.deleted	Integer	The number of relationships deleted.

Change Log

Version	Change
v2	Removed <code>set_relationship_value</code> parameter.
v2	Added <code>module_name</code> parameter.
v2	Added <code>module_id</code> parameter.
v2	Added <code>link_field_name</code> parameter.
v2	Added <code>related_ids</code> parameter.
v2	Added <code>name_value_list</code> 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: 10/14/2016 04:12pm

set_relationships

Overview

Sets multiple relationships between multiple record sets.

Available APIs

- SOAP
- REST

Definition

`set_relationships(session, module_names, module_ids, link_field_names, related_ids, name_value_lists, delete_array)`

Parameters

Name	Type	Description
session	String	Session ID returned by a previous login call.
module_names	select_fields Array	The list of modules from which to retrieve records. Note: This is the modules key which may not be the same as the modules

Name	Type	Description
		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_relationship_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_r esult 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 <code>module_name</code> or <code>link_field_name</code> are incorrect.
result.deleted	Integer	The number of relationships deleted.

Change Log

Version	Change
v2	Removed <code>set_relationship_value</code> 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: 10/14/2016 04:12pm

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
------	------	-------------

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.
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: 10/14/2016 04:12pm

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
------	------	-------------

Name	Type	Description
session	String	Session ID returned by a previous login call.
report_id	String	The ID of the report record to generate the PDF for.

Result

Name	Type	Description
result	Array	The call result.
result.results	Boolean	The new emails.
result.count	Integer	The count of results.
result.message	String	The return message.

Change Log

Version	Change
v4	Added snip_update_contacts method.

Last Modified: 10/14/2016 04:12pm

Extending v1 - v4.1 Web Services

Overview

The guide will demonstrate how to add your own custom methods to the REST and SOAP API or extend existing ones. It is important to note that this customization is not supported on Sugar's cloud service and it is recommended to extend the v10 endpoints instead.

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 `./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: 03/16/2018 02:30pm

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.

Last Modified: 10/14/2016 04:12pm

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: 10/14/2016 04:12pm

Examples

Examples of v4_1 Web Service API Calls.

Last Modified: 10/14/2016 04:12pm

REST

Examples of v4_1 REST API Calls.

Last Modified: 10/14/2016 04:12pm

PHP

PHP v4_1 REST Examples.

Last Modified: 10/14/2016 04:12pm

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: 10/14/2016 04:12pm

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);

        $jsonData = json_encode($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 need 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: 10/14/2016 04:12pm

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 need 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: 10/14/2016 04:12pm

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 need 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 need 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: 10/14/2016 04:12pm

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);

    $jsonData = json_encode($parameters);

    $post = array(
        "method" => $method,
        "input_type" => "JSON",
        "response_type" => "JSON",
        "rest_data" => $jsonData
    );

    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: 10/14/2016 04:12pm

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);

    $jsonData = json_encode($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: 10/14/2016 04:12pm

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: 10/14/2016 04:12pm

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);

    $jsonData = json_encode($parameters);

    $post = array(
        "method" => $method,
        "input_type" => "JSON",
        "response_type" => "JSON",
        "rest_data" => $jsonData
    );
}
```

```
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: 10/14/2016 04:12pm

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);

    $jsonData = json_encode($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: 10/14/2016 04:12pm

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
```

```
[email_address] => stdClass Object
                                                    (
[name] => email_address
[value] => qa.section@example.it
                                                    )
=> stdClass Object
                                                    [opt_out]
                                                    (
[name] => opt_out
[value] => 1
                                                    )
[primary_address] => stdClass Object
                                                    (
[name] => primary_address
[value] =>
```

```
stdClass Object                                     [link_value] =>
                                                    (
stdClass Object                                     [id] =>
                                                    (
[name] => id
[value] => 16aeaf8b-b31f-943d-3844-4f7cb6ac63f2
                                                    )
[email_address] => stdClass Object
                                                    (
[name] => email_address
[value] => vegan.sales.im@example.cn
                                                    )
=> stdClass Object                                     [opt_out]
                                                    (
```

[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
                                                                    )
```

Last Modified: 10/14/2016 04:12pm

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);

    $jsonData = json_encode($parameters);

    $post = array(
        "method" => $method,
        "input_type" => "JSON",
        "response_type" => "JSON",
        "rest_data" => $jsonData
    );

    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
```

new items

```
(
    [name] => intent
    [value] => pick
)

[mailbox_id] => stdClass Object
(
    [name] => mailbox_id
    [value] =>
)

[parent_name] => stdClass Object
(
    [name] => parent_name
    [value] => Having trouble adding
)

[parent_type] => stdClass Object
(
    [name] => parent_type
    [value] => Cases
)

[parent_id] => stdClass Object
```

```
Example.zip      [name] => stdClass Object
                 (
                 [name] => name
                 [value] =>
                 )

stdClass Object  [file_mime_type] =>
                 (
                 [name] =>
                 [value] =>
                 )

file_mime_type   application/zip

Object           [filename] => stdClass
                 (
                 [name] => filename
                 [value] =>
                 )

Example2.zip
```

```
Object [description] => stdClass
      (
        [name] =>
description [value] =>
      )

Object [_empty_] => stdClass
      (
        [name] =>
        [value] =>
      )
)

[1] => stdClass Object
  (
    [id] => stdClass Object
      (
        [name] => id
        [value] =>
592f382d-b633-fd5f-803e-5205423a6d0b
      )
  )
```

```
Example2.zip [name] => stdClass Object
              (
                [name] => name
                [value] =>
              )

stdClass Object [file_mime_type] =>
                (
                  [name] =>
                  [value] =>
                )
file_mime_type application/zip

Object [filename] => stdClass
        (
          [name] => filename
          [value] =>
        )
Example2.zip
```

```
)  
  
//Attachment 1 Result  
stdClass Object  
(  
  [note_attachment] => stdClass Object  
  (  
    [id] => 1b63a8f9-ce67-6aad-b5a4-52054af18c47  
    [filename] => Example.zip  
    [file] =>  
UESDBAoAAAAIAOujCEOkMbvsNa0AAMCFAgAXAAAAARmlsZXMvaW  
    [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] =>
AEUoaAAARmlsIsZXMvaWZxujCEOkMbvsNa0AAMCFAgAXAAAARm
        [related_module_id] =>
5826bd75-527a-a736-edf5-5205421467bf
        [related_module_name] => Emails
    )
)
```

Last Modified: 10/14/2016 04:12pm

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);

    $jsonData = json_encode($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
                                (
```

Object

```
        [name] => billing_address_state
        [value] => NY
    )

    [billing_address_country] => stdClass

    (
        [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: 10/14/2016 04:12pm

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);

    $jsonData = json_encode($parameters);

    $post = array(
        "method" => $method,
        "input_type" => "JSON",
        "response_type" => "JSON",
        "rest_data" => $jsonData
    );
};
```

```
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] =>

)

)

```
Object                                     )
                                           [1] => stdClass
                                           (
[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] =>  
)  
  
)  
  
)  
  
)
```

```
                [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] =>
```

```
)
```

```
)
```

```
)
```

```

    )
)
[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

 (

```
Object                                     [0] => stdClass
                                           (
[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: 10/14/2016 04:12pm

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);

    $jsonData = json_encode($parameters);

    $post = array(
        "method" => $method,
        "input_type" => "JSON",
```

```
        "response_type" => "JSON",
        "rest_data" => $jsonData
    );

    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: 10/14/2016 04:12pm

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: 10/14/2016 04:12pm

SOAP

Examples of v4_1 SOAP API Calls.

Last Modified: 10/14/2016 04:12pm

C#

C# SOAP v4_1 Examples.

Last Modified: 10/14/2016 04:12pm

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 need 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: 10/14/2016 04:12pm

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().ToLower();
}
}
```

Result

b2uv0vrjfiiov41d03sk578ufq6

Last Modified: 11/18/2017 10:25am

PHP

PHP SOAP v4_1 Examples.

Last Modified: 10/14/2016 04:12pm

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
$error = $client->getError();
if ($error)
{
    echo '<h2>Constructor error</h2><pre>' . $error . '</pre>';
    echo '<h2>Debug</h2><pre>' .
htmlspecialchars($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 need 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: 10/14/2016 04:12pm

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
$error = $client->getError();
if ($error)
{
    echo '<h2>Constructor error</h2><pre>' . $error . '</pre>';
    echo '<h2>Debug</h2><pre>' .
htmlspecialchars($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 need 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: 10/14/2016 04:12pm

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 need 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: 10/14/2016 04:12pm

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
$error = $client->getError();
if ($error)
{
    echo '<h2>Constructor error</h2><pre>' . $error . '</pre>';
    echo '<h2>Debug</h2><pre>' .
htmlspecialchars($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 need 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 need 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: 10/14/2016 04:12pm

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
$error = $client->getError();
if ($error)
{
    echo '<h2>Constructor error</h2><pre>' . $error . '</pre>';
    echo '<h2>Debug</h2><pre>' .
htmlspecialchars($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: 10/14/2016 04:12pm

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
$error = $client->getError();
if ($error)
{
    echo '<h2>Constructor error</h2><pre>' . $error . '</pre>';
    echo '<h2>Debug</h2><pre>' .
htmlspecialchars($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] => 9uukc92a61b9v620reqv34mmg1
  [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: 10/14/2016 04:12pm

Relating Quotes and Products

Overview

A PHP example demonstrating how to create and relate Products to Quotes 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 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: 10/14/2016 04:12pm

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
$error = $client->getError();
if ($error)
{
    echo '<h2>Constructor error</h2><pre>' . $error . '</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: 10/14/2016 04:12pm

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
$error = $client->getError();
if ($error)
{
    echo '<h2>Constructor error</h2><pre>' . $error . '</pre>';
    echo '<h2>Debug</h2><pre>' .
htmlspecialchars($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: 10/14/2016 04:12pm

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
$error = $client->getError();
if ($error)
{
    echo '<h2>Constructor error</h2><pre>' . $error . '</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
(
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
)
```

```
Array [3] =>
      (
[name] => primary_address
[value] =>
      )
    )
  )
[1] => Array
    (
      [link_value] =>
Array
      (
Array
      [0] =>
      (
[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
```

```
[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
(
```

```
Array [0] =>
      (
[name] => id
[value] => 13ee5142-17c1-76fd-3cc4-5282a32cf864
      )
Array [1] =>
      (
[name] => email_address
[value] => kid.hr.sugar@example.biz
      )
Array [2] =>
      (
[name] => opt_out
[value] => 1
      )
```

```
Array
[3] =>
(name) => primary_address
(value) =>
)
```

)

Last Modified: 10/14/2016 04:12pm

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
$error = $client->getError();
if ($error)
{
    echo '<h2>Constructor error</h2><pre>' . $error . '</pre>';
    echo '<h2>Debug</h2><pre>' .
htmlspecialchars($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: 10/14/2016 04:12pm

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
$error = $client->getError();
if ($error)
{
    echo '<h2>Constructor error</h2><pre>' . $error . '</pre>';
    echo '<h2>Debug</h2><pre>' .
htmlspecialchars($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
                                (
```

Mining Inc.

```
[0] => Array
(
    [name] => name
    [value] => Underwater
)

)

)

[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] =>

)

)

)

```
        (
            [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] =>  
)  
  
)  
  
)  
  
)
```

```
                [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] =>
```

```
)
```

[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: 10/14/2016 04:12pm

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
$error = $client->getError();
if ($error)
{
    echo '<h2>Constructor error</h2><pre>' . $error . '</pre>';
    echo '<h2>Debug</h2><pre>' .
htmlspecialchars($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: 10/14/2016 04:12pm

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
$error = $client->getError();
if ($error)
{
```

```
        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: 10/14/2016 04:12pm

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: 10/14/2016 04:12pm

Migration

Migrating Sugar instances and external data.

Last Modified: 11/30/2016 01:37pm

Importing Records

Covers the Sugar structure when migrating data into the application.

Last Modified: 11/30/2016 01:38pm

Importing Email Addresses

Overview

Recommended approaches when Importing email addresses.

Importing via API

Sugar comes out of the box with an API that can be called from custom applications utilizing the REST interface. The API can be used to mass create and update records in Sugar with external data. For more information on the REST API in Sugar, please refer to the Web Services documentation.

Importing New Records

When importing new records into Sugar through the API, modules with relationships to email addresses can utilize the email1 field or the email link field to specify email addresses for a record.

Email1 Field

When using the email1 field, the default functionality is to import the email address specified as the primary address. Assuming the email address does not already exist in the database, the email address is then flagged as being valid and is not opted out. Using the /<module> POST endpoint, you can send the following JSON payload to create a contact record with a primary email address using the email1 field:

POST URL: `http://<site url>/rest/v10/Contacts`

```
{
  "first_name": "Rob",
  "last_name": "Robertson",
  "email1": "rob.robertson@sugar.crm"
}
```

Note : For importing multiple email addresses, you will need to use the email link field described below.

Email Link Field

When using the email link field, you can specify multiple email addresses to assign to the record. You may specify the following additional information regarding each

email address being added:

- `invalid_email` : Specify this email address as being invalid
- `opt_out` : Specify this email address as being opted out
- `primary_address` : Specify this email address as the primary

Using the `/<module>` POST endpoint, you can send the following JSON payload to create a contact record with multiple email addresses using the email link field:
POST URL: `http://<site url>/rest/v10/Contacts`

```
{
  "first_name": "Rob",
  "last_name": "Robertson",
  "email": [
    {
      "email_address": "rob.robertson@sugar.crm",
      "primary_address": "1",
      "invalid_email": "0",
      "opt_out": "0"
    },
    {
      "email_address": "rob@sugar.crm",
      "primary_address": "0",
      "invalid_email": "0",

```

```
        "opt_out": "1"
    }
]
}
```

For more information on the /<module>/:record POST endpoint, you can refer to your instance's help documentation found at:

<http://<site url>/rest/v10/help>

Or you can reference the <module> POST PHP example.

Updating Existing Records

When updating existing records in Sugar through the API, modules with relationships to email addresses can also utilize the email1 field or the email link field to specify email addresses for a record.

Email1 Field

When using the email1 field, the default functionality is to replace the existing email primary address. Assuming the email does not already exist in the database, the new email address is flagged as being valid and is not opted out. Using the

/<module>/:record PUT endpoint, you can send the following JSON payload to update a contact records primary email address:

PUT URL: `http://<site url>/rest/v10/Contacts/<record id>`

```
{
  "email1": "rob.robertson@sugar.crm"
}
```

Note : This will replace the current email address on the record with the new data. The old email address will no longer be associated with the record.

Email Link Field

When using the email link field, you can specify multiple email addresses to update the record with. You may specify the following additional information regarding each email address being added:

- `invalid_email` : Specify this email address as being invalid
- `opt_out` : Specify this email address as being opted out
- `primary_address` : Specify this email address as the primary

Using the /<module>/:record PUT endpoint, you can send the following JSON

payload to update a contact record with multiple email addresses:
PUT URL: <http://<site url>/rest/v10/Contacts/<record id>>

```
{
  "email": [
    {
      "email_address": "rob.robertson@sugar.crm",
      "primary_address": "1",
      "invalid_email": "0",
      "opt_out": "0"
    },
    {
      "email_address": "rob@sugar.crm",
      "primary_address": "0",
      "invalid_email": "0",
      "opt_out": "1"
    }
  ]
}
```

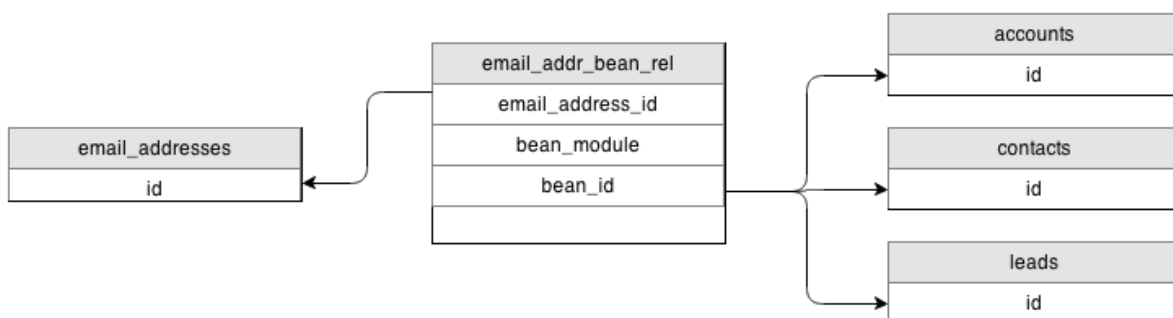
For more information on the [/<module>/record](#) PUT endpoint, you can refer to your instance's help documentation found at:

<http://<site url>/rest/v10/help>

Or you can reference the <module>/:record PUT PHP example.

Importing via Database

When importing records into Sugar directly via the database, it is important that you understand the data structure involved before loading data. Email addresses are not stored directly on the table for the module being imported in, but are related via the `email_addr Bean Rel` table.



The table structure for email addresses can be seen from the database via the following SQL statement:

```
SELECT
  email_addr Bean Rel.bean_id,
```

```
    email_addr_bean_rel.bean_module,  
    email_addresses.email_address  
FROM email_addr_bean_rel  
INNER JOIN email_addresses  
    ON email_addresses.id = email_addr_bean_rel.email_address_id  
    AND email_addr_bean_rel.deleted = 0  
WHERE email_addresses.deleted = 0;
```

Checking for Duplicates

Email addresses can become duplicated in Sugar from a variety of sources including API calls, imports, and from data entry. There are a few ways to have the system check for duplicate contact records, but not many of those methods work for checking email addresses for duplicates. The following section will demonstrate how to find and clean up duplicate email addresses using SQL.

The following SQL query can be used to locate if any email addresses are being used against more than one bean record within Sugar:

```
SELECT  
    email_addresses.email_address,  
    COUNT(*) AS email_address_count  
FROM email_addr_bean_rel  
INNER JOIN email_addresses  
    ON email_addresses.id = email_addr_bean_rel.email_address_id
```

```
AND email_addr_bean_rel.deleted = 0
WHERE email_addresses.deleted = 0
GROUP BY email_addresses.email_address
HAVING COUNT(*) > 1;
```

Note : If you convert a Lead record to a Contact record, both the Lead and the Contact will be related to the same Email Address and will return as having duplicates in this query. You can add the following line to the WHERE clause to filter the duplicate check down to only one bean type:

```
AND email_addr_bean_rel.bean_module = 'Contacts'
```

Email addresses can not only be duplicated in the system but can occasionally be missing critical data. Each bean record with an email address assigned to it should have an email address designated the primary. The following query will locate any bean records that have at least one email address, where there is not an email address designated as the primary:

```
SELECT
  email_addr_bean_rel.bean_module,
  email_addr_bean_rel.bean_id,
  COUNT(*) AS email_count,
  COUNT(primary_email_addr_bean_rel.id) AS primary_email_count
FROM email_addr_bean_rel
LEFT JOIN email_addr_bean_rel primary_email_addr_bean_rel
```

```
ON primary_email_addr_bean_rel.bean_module =
email_addr_bean_rel.bean_module
AND primary_email_addr_bean_rel.bean_id = email_addr_bean_rel.bean_id
AND primary_email_addr_bean_rel.primary_address = '1'
AND primary_email_addr_bean_rel.deleted = '0'
WHERE email_addr_bean_rel.deleted = '0'
GROUP BY email_addr_bean_rel.bean_module,
email_addr_bean_rel.bean_id
HAVING primary_email_count < 1;
```

Removing Duplicates

If it is determined you have duplicate email addresses being used in your system, you can use the following query to 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: 11/30/2016 01:48pm

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: 10/14/2016 04:12pm

Security

Security documents for securing Sugars infrastructure.

Last Modified: 10/14/2016 04:12pm

Endpoints and Entry Points

Overview

This document describes how to disable out of the box REST API endpoints and legacy MVC entry points.

Advisory ID: sugarcrm-sr-2015-001
Revision: 1.1
Last updated: 2015-10-01

Description

SugarCRM has both legacy entry points and REST API endpoints which are shipped out of the box. Not every customer uses all capabilities of the SugarCRM product. To harden the configuration both entry points can be disabled based on the customer's requirements.

Legacy Entry Points

All stock entry points are defined in `include/MVC/Controller/entry_point_registry.php`. Using the SugarCRM Extension framework the configuration directives can be overridden in an upgrade safe manner. As an example consider the endpoint "removeme". To disable this endpoint use the following procedure.

Create a new php file `./custom/Extension/application/Ext/EntryPointRegistry/disable_removeme.php` and add the following content:

```
<?php

if (isset($entry_point_registry['removeme'])) {
    unset($entry_point_registry['removeme']);
}
```

Execute a quick repair and rebuild as SugarCRM administrator. The entry point is now fully disabled and no longer accessible to respond to any calls. Note that when trying to hit a non-existing (or disabled) entry point, the application will redirect you to the homepage (or login screen if the user has no session).

REST API Endpoints

To disable the HelpAPI which is located at `clients/base/api/HelpApi.php` use the following procedure.

Create a new php file `custom/clients/base/api/CustomHelpApi.php` and add the following content:

```
<?php

require_once 'clients/base/api/HelpApi.php';

class CustomHelpApi extends HelpApi
```

```
{
  public function getHelp($api, $args)
  {
    throw new SugarApiExceptionNotFound();
  }
}
```

Execute a quick repair and rebuild as SugarCRM administrator. The entry point is now fully disabled. When making a REST API call to /rest/v10/help the following will be returned:

```
HTTP 404 Not Found
{
  "error": "not_found",
  "error_message": "Your requested resource was not found."
}
```

Publication History

2015-10-01	Adding REST API endpoint example
2015-06-16	Initial publication

A stand-alone copy of this document that omits the distribution URL is an

uncontrolled copy, and may lack important information or contain factual errors. SugarCRM reserves the right to change or update this document at any time.

Last Modified: 10/14/2016 04:12pm

Web Server Configuration

Overview

This document serves as a guideline to harden the web server configuration with regard to running SugarCRM. Note that this is a guideline and certain settings will depend on your specific environment and setup. This guideline focuses on Apache web server as this is SugarCRM's primary supported web server. However the recommendations in this document apply to all web servers in general.

Advisory ID: sugarcrm-sr-2015-003

Revision: 1.1

Last updated: 2015-10-01

SugarCRM .htaccess file

During installation, SugarCRM deploys an `.htaccess` file in SugarCRM's root folder. The content of this file may change during upgrades as well. The primary configuration directives managed by SugarCRM are focused around disabling direct access to certain directories and files. Secondly the configuration contains specific URL rewrite rules which are required for SugarCRM.

Although this file can be used to ship most of the settings which are explained in this document to harden the web server, SugarCRM has chosen not to do so as most of them depend on the customer's setup which cannot be fully controlled by the SugarCRM application itself.

All directives which are managed by SugarCRM are placed between the following markers inside the `.htaccess` file:

```
# BEGIN SUGARCRM RESTRICTIONS
RedirectMatch 403 .*\.log$
...
# END SUGARCRM RESTRICTIONS
```

Customers can add additional directives if need. Make sure to put those directives outside of the above mentioned markers. SugarCRM can rebuild during upgrades all directives enclosed in the above markers.

Securing `.htaccess`

Apache allows admins to drop in .htaccess files in any given directory. When a request comes in, Apache will scan the directory recursively for any .htaccess file and apply its configuration directives. This ability is orchestrated by the AllowOverride directive which is enabled by default on most Apache configurations.

To disable this dynamic Apache configuration behavior, disable AllowOverride and make sure to copy paste the .htaccess file content as well.

```
<Directory "/var/www/html">
    AllowOverride None
    # BEGIN SUGARCRM RESTRICTIONS
    RedirectMatch 403 .*\.log$
    ...
    # END SUGARCRM RESTRICTIONS
</Directory>
```

When choosing for this security measure, make sure to verify if any changes have happened in the .htaccess after an upgrade. Those changes need to be applied manually back in the Apache configuration to ensure SugarCRM functions properly. Disabling the .htaccess file support also has a positive impact on the performance as Apache will no longer need to scan for the presence of .htaccess files.

Although Apache does not allow to download .htaccess files, to tighten the security

around `.htaccess` even more the following directives can be added:

```
<Files ".ht*">  
    Order deny,allow  
    Deny from all  
</Files>
```

Prevent version exposure

By default Apache web server exposes its version and OS details in most *nix distributions in both the HTTP response headers as well as on the default error pages. To prevent this behavior use the following configuration directives:

```
ServerSignature Off  
ServerTokens Prod
```

The exposure of the PHP version can be disabled in your `php.ini` file. When exposed an X-Powered-By response header will be added containing the PHP version information.

```
expose_php = Off
```

Another option is to unset the header explicitly in the Apache configuration:

```
<IfModule mod_headers.c>
    Header unset X-Powered-By
</IfModule>
```

Note that unsetting the Server header is not possible using this way. The web server will always respond with a Server header containing the string "Apache". If it is desired to remove this header as well, additional modules need to be used for this (i.e. mod_security can do this).

Directory listing and index page

Make sure to configure a default index page which should only be index.php for SugarCRM. This is mostly configured but worth checking. Additional SugarCRM does not require the directory browsing support and this is recommended to be disabled. This can be accomplished by removing Options Indexes and/or completely disable the mod_autoindex module.

Optionally additional Allow/Deny directives can be used to apply access lists to certain directories. It is not recommended to add additional authentication through Apache as SugarCRM already fully manages user authentication. The Allow/Deny directives can still be used to apply any IP access list if required by the customer.

The following example secures the root directory (default configuration), disable directory browsing and an additional IP access list is applied.

```
DocumentRoot "/var/www/html"
```

```
<Directory />  
    AllowOverride none  
    Order allow,deny  
    Deny from all  
</Directory>
```

```
<Directory "/var/www/html">  
    DirectoryIndex index.php  
    Options -Indexes  
    Order deny,allow  
    Allow from 10.0.0.0/8  
    Deny from all  
</Directory>
```

Additional Options directives

SugarCRM does not rely on both CGI and SSI and it is recommended to disable this functionality explicitly. Optionally the FollowSymLinks options can also be disabled depending on your directory setup.

```
<Directory "/var/www/html">  
    Options -ExecCGI -Includes -FollowSymLinks
```

</Directory>

Web server permissions

Ensure your web server is configured to run as a dedicated non-privileged user. Certain *nix distributions use out of the box the user/group nobody. It is recommended to use a dedicated user/group instead.

```
User wwwrun  
Group www
```

Ensure the configured user/group have the proper file and directory permissions for SugarCRM. See the installation/upgrade guide for more detailed instructions regarding the required permissions.

Disable unnecessary loadable modules

Apache is pluggable by means of loadable modules. To minimize the chances becoming a victim of an attack it is recommended to disable all loadable modules which are not needed. SugarCRM recommends the following base Apache modules:

-
- `mod_authz_host`
 - `mod_dir`
 - `mod_expires`
 - `mod_rewrite`
 - `mod_headers`
 - `mod_mime`
 - `mod_alias`

From an Apache perspective to disable a loadable module, it is sufficient to comment out the `LoadModule` directives. However certain *nix distribution have additional configuration scripts to manage the loaded Apache modules. Please refer to your *nix distribution for this.

Based on your setup additional Apache modules can be used. When using MPM prefork you will also need `mod_php5` and when terminating secure HTTP connection on the web server directly also `mod_ssl` (see below).

HTTP methods

HTTP uses different methods (verbs) in requests. As per RFC-7231 for HTTP/1.1 different request methods are defined which serve the primary source of the request semantics. The following methods are used by SugarCRM:

For new REST API:

-
- GET
 - POST
 - PUT
 - DELETE
 - HEAD (*)
 - OPTIONS (*)

Bootstrap, BWC access, old SOAP/REST API:

- GET
- POST

(*) Currently not in use, but may be implemented in the future.

The new REST API is accessed using the URL `"/rest/vxx/..."` where `xxx` represents the API version number. To harden the usage of the additional HTTP methods the following directives can be used for the new REST API endpoint while only allowing GET and POST on the legacy access.

```
<Directory "/var/www/html">  
  <LimitExcept GET POST>  
    Order deny,allow  
    Deny from all
```

```
        </LimitExcept>
</Directory>

<Location "/rest">
    <LimitExcept GET POST PUT DELETE>
        Order deny,allow
        Deny from all
    </LimitExcept>
</Location>

<Location "/api/rest.php">
    <LimitExcept GET POST PUT DELETE>
        Order deny,allow
        Deny from all
    </LimitExcept>
</Location>
```

Instead of using the Limit directive, an alternative is using mod_allowmethods which is available since Apache 2.4. However this module is flagged as experimental.

Secure HTTP traffic

SugarCRM requires the usage of HTTPS transport without any exception.

Depending on your setup you can terminate the secure connection on a separate end point (load balancer, dedicated HTTPS termination point) or directly on the web server(s). You will also need a valid private key and X.509 certificate. In most cases the customer needs to acquire an X.509 certificate through a public certification authority of their choice. For internal deployments this may not be necessary when a local PKI environment is available.

When terminating the secure connection directly on the Apache web server, `mod_ssl` is required. A dedicated `VirtualHost` needs to be configured on port 443 (standard HTTPS port). The following is a basic SSL/TLS configuration as a reference. More details are covered in the next paragraphs.

```
<VirtualHost 1.2.3.4:443>
    SSLEngine on
    SSLCertificateFile /etc/pki/tls/certs/example.com.crt
    SSLCertificateKeyFile /etc/pki/tls/certs/example.com.key
    SSLCertificateChainFile /etc/pki/tls/certs/example_bundle.crt
    ServerName crm.example.com
</VirtualHost>
```

Make sure to have your server name properly configured as it should match the X.509 certificate's `CommonName` (or one of the SNA's, subject alternative names). More information regarding the certificate requests including the certificate chain can be obtained from your certification authority.

Ensure to redirect all non-secure HTTP (port 80) traffic to the secure virtual host.

```
<VirtualHost *:80>  
    Redirect "/" "https://crm.example.com/"  
</VirtualHost >
```

Cipher suite hardening

One of the crucial points of HTTPS is to force only secure cryptographic algorithms. Based on new future weaknesses which may be discovered in the future the recommended cipher suite settings can change. As per publish date of this document the usage of SSLv2 and SSLv3 is recommended to be disabled. The current safe secure transport protocols are TLSv1.0, TLSv1.1 and TLSv1.2 with properly configured cipher suites.

For proper deployment of Diffie-Hellman (DH) key exchange algorithms the following guidelines apply:

- No longer use out-dated export cipher suites
- Ensure the usage of Elliptic Curve Diffie-Hellman (ECDHE) based algorithms
- Use of non-common Diffie-Hellman groups

More detail around the DH configuration and a detailed list of recommended

cipher suites including web server configuration examples, can be found at <https://weakdh.org/sysadmin.html>. Additionally, an online SSL/TLS checker can be used to generate a full report of your current configuration in case your web server is publically reachable. A recommended tool is <https://www.ssllabs.com/ssltest/analyze.html>. The output of this SSL/TLS test suite will guide you into further (optional) details which can be tweaked regarding your configuration (i.e. OCSP stapling, certificate pinning, ...).

Please note that using the above instruction "perfect forward secrecy" will be enabled. This basically means that captured TLS traffic can only be decoded using the session key between a specific client and server. Non perfect forwarding secrecy connections can always be decrypted by having access to only the server's private key. If the customer's setup has intermediate IDS/WAF systems to inspect traffic looking inside the secure traffic, having perfect forwarding secrecy configured may not be desirable.

Public trusted root CA's

It is recommended that the *nix system where Apache is running always has the latest public root CA (certification authority) files available to authenticate server certificates during the SSL/TLS handshake. Most *nix distributions take care of this through their online update process. If this is not the case, the system administrator is responsible to ensure the latest trusted root CA's are up-to-date. Out of the box the SugarCRM application contacts the SugarCRM heartbeat and

licensing server using a secure connection.

HSTS header

As all traffic requires HTTPS for SugarCRM, it is recommended to configure a global HSTS (HTTP Strict Transport Security) header. This is an additional opt-in security enhancement supported by most modern browsers. The HSTS will instruct the browser to always use HTTPS in the future for the given FQDN. More details regarding HSTS can be found here:

https://www.owasp.org/index.php/HTTP_Strict_Transport_Security.

Before enabling the HSTS header, ensure that no other non-HTTPS website is running for the given FQDN and any subdomains.

```
Header always set Strict-Transport-Security \  
"max-age=10886400; includeSubdomains; preload"
```

Optionally when your SugarCRM application is publically reachable, you can submit your FQDN on the preload which is maintained by Chrome and included by Firefox, Safari, Internet Explorer 11 and Edge browser. Understand the consequences and requirements to end up on the preload list. Once registered, this cannot be undone easily. Submissions for the HSTS preload list can be found at <https://hstspreload.appspot.com>.

CSP header

A Content Security Policy header can be added to offer additional protection against XSS (cross site scripting). Most modern browser support CSP and will apply the limitation as directed by the CSP header. The CSP header defines the allowed sources through different directives which are allowed regarding JScript, CSS, images, fonts, form actions, plugins, ... More information can be found at https://www.owasp.org/index.php/Content_Security_Policy.

The management of CSP headers can become complex and may depend on different types of customizations which can be deployed on the SugarCRM platform. Therefor SugarCRM will ship in the future with the ability of controlling CSP headers directly from within SugarCRM itself.

For now CSP headers can be configured directly on the web server. A basic configuration looks like this:

```
Header always set Content-Security-Policy \  
"default-src: 'self'; script-src: 'self'"
```

Note that older browsers used the header X-Content-Security-Policy or X-WebKit-CSP instead of the standardized one. The CSP header can also safely be added first in "report only" mode before deploying it in production using Content-Security-Policy-Report-Only.

Additional security headers

To prevent clickjacking an X-Frame-Options header can be added. This header can also be used to be able specifically iframe the SugarCRM application inside another web site if required. This can be better controlled with the above mentioned CSP header. Both can coexist together but it is encouraged to used the CSP approach.

```
Header always set X-Frame-Options "SAMEORIGIN"  
Header always set X-Frame-Options "ALLOW-FROM  
https://someothersite.com"
```

To protect against so called drive-by download attacks, it is encouraged to disable Content-Type sniffing. This is particularly targeted at Chrome and IE.

```
Header always set X-Content-Type-Options "nosniff"
```

Ensure XSS browser protection is enabled by adding the following header. By default this feature should be enabled in every browser. In case it has been disabled, sending this header will re-enabled it.

```
Header always set X-XSS-Protection "1; mode=block"
```

Secure cookie settings

Ensure the following settings are applied in `php.ini` to ensure legacy cookies are properly protected against XSS attacks and are only transmitted over a secure HTTPS connection.

```
session.cookie_httponly = 1
session.cookie_secure = 1
```

Additionally, the `session.cookie_path` can be tweaked if other applications are running on the same webserver.

Security modules

Additional Apache loadable security modules exist which can optionally be installed to better protect your system from malicious requests and/or DDOS attacks. The following modules are available for Apache 2.4:

- `mod_security` - This module adds WAF (web application firewall) capabilities to your Apache setup. In combination with the OWASP CRS (core rule set) gives you a solid starting point for a WAF implementation.
- `mod_evasive` - This module helps preventing (D)DOS attacks

Covering both modules in details is out of scope of this Security Response. Refer to the module documentation for additional details. In the future SugarCRM may publish specific settings and fine tuning for both modules.

Additional resources

Generic Apache security tips

http://httpd.apache.org/docs/2.4/misc/security_tips.html

Online SSL/TLS tester

<https://www.ssllabs.com/ssltest/analyze.html>

Useful HTTP headers

https://www.owasp.org/index.php/List_of_useful_HTTP_headers

Publication History

2015-10-01	Adding HTTP method restrictions
2015-09-08	Initial publication

A stand-alone copy of this document that omits the distribution URL is an uncontrolled copy, and may lack important information or contain factual errors.

SugarCRM reserves the right to change or update this document at any time.

Last Modified: 10/14/2016 04:12pm

XSS Prevention

Overview

This document describes how to prevent cross-site scripting (XSS) attacks in Sugar customizations. XSS attacks occur when malicious entities are able to inject client-side scripts into web pages.

Best Practices

When creating custom code for Sugar, be sure to respect the following best practices. These rules serve to protect the elements of your customizations that are most susceptible to XSS vulnerabilities:

- Create safe variables when injecting HTML via Handlebars templates or JavaScript methods such as: `$(selector).html(variable)`,

`$(selector).before(variable)`, or `$(selector).after(variable)`.

- Avoid using triple-bracket (e.g. `{{{variable}}}`) syntax to allow unescaped HTML content in Handlebars templates.
- Never use input from an unknown source as it may contain unsafe data.
- Protect dynamic content, which may also contain unsafe data.

Please refer to the following sections for more information and examples that demonstrate these best practices.

Creating Safe Variables

You must always encode a variable before you inject or display it on a webpage. The piece that is not encoded should be isolated and marked as a safe string very carefully. The the following sections outline protecting your system when using JavaScript and Handlebars templates.

JavaScript

To ensure displayed JavaScript variables are safe, respect these two rules:

- If you are displaying plain text without any HTML, use `.text()` instead of `.html()` with your selectors.

-
- If your text contains HTML formatting, you can use `.html()` as long as it does not contain any dynamic data coming from a variable.

Vulnerable Setup

This is an example of bad code because it utilizes the `.html()` method to display a dynamic-value variable, leaving the JavaScript vulnerable to injection:

```
var inputData = $('#myInput').val(); // "<script>alert(0)</script>"
$(selector).html('This is the value: ' + inputData); // "This is the
value: <script>alert(0)</script>"
```

Protected Setup

This is an example of good code, which uses the safe `.text()` method to protect the JavaScript from potential injection:

```
var status = 'safe';
$(selector).text('This is very ' + status); // "This is very safe"
```

This is also an acceptable approach, which uses the `Handlebars.Utils.escapeExpression` to safely escape the content:

```
var inputData = $('#myinput').val(); // "<script>alert(0)</script>"
var safeData = Handlebars.Utils.escapeExpression(inputData); //
"<script>alert(0)</script>"
$(selector).html('This is the value: ' + safeData); // "This is the
value: <script>alert(0)</script>"
```

Handlebars Templates

Handlebars, by default, handles the encoding of data passed to the template in double brackets (e.g. `{{variable}}`), however, it also allows you to bypass this encoding by using triple brackets (e.g. `{{{variable}}}`). Text passed via triple brackets will not be encoded. As a rule, do not use triple brackets. If you don't want Handlebars to encode a piece of a string, use double brackets and execute `Handlebars.SafeString()` in your JavaScript controller.

As an example, we will apply best practices to a complex use case with dynamic values that need to be encoded. For this example, we want to display a message similar to:

```
Congrats! You just created the record <a
href="/#Accounts/abcd">ABCD</a>
```

The values `ABCD` and `/#Accounts/abcd` are dynamic values and, therefore, must be encoded. But the message must also be displayed as HTML, so the overall element

cannot be entirely encoded. To address this use case, we must properly script the JavaScript controller and the Handlebars template.

The following JavaScript encodes the message's dynamic values, marks the displayed hyperlink as a safe string, and then outputs the safe message:

```
var record = {
  id: 'abcd',
  name: 'ABCD'
};

// escape `ABCD`
var safeName = Handlebars.Utils.escapeExpression(record.name);

// escape `abcd`
var safeId = Handlebars.Utils.escapeExpression(record.id);

// Mark the link as a SafeString now that the unsafe pieces are
// encoded and the content has safe HTML.
var link = new Handlebars.SafeString('<a href="/#Accounts/' + safeId +
' ">' + safeName . '</a>');

// This can be displayed with double brackets in the template because
// html parts will not be encoded.
this.safeMessage = new Handlebars.SafeString('Congrats! You just
```

```
created the record ' + link + '.'');
```

To display the safe message in the application, use the following syntax in your Handlebars template:

```
<div class="success">{{safeMessage}}</div>
```

Please refer to the [HTML Escaping documentation](#) on the Handlebars website for more information.

Last Modified: 11/27/2017 12:00pm