SugarCRM Adapter Guide

For SAP Integration Suite and SAP Cloud Integration

Version 1.0.3 – May 2024

Contents

1 Introduction

This is the official guide for the SugarCRM Adapter for SAP Integration Suite and SAP Cloud Integration. This guide covers all relevant information for integration developers to start working with the SugarCRM adapter. Please read this guide carefully before using the adapter.

1.1 Coding Samples

Any software coding and/or code lines/strings ("Code") included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules of certain coding. We do not warrant the correctness and completeness of the Code given herein.

1.2 Internet Hyperlinks

The documentation may contain hyperlinks to the Internet. These hyperlinks are intended to serve as a hint about where to find related information. We do not warrant the availability and correctness of this related information or the ability of this information to serve a particular purpose.

2 SugarCRM Integration

2.1 Introduction

SugarCRM provides the web application named Sugar which is a customer relationship management system. It is a category of integrated, data-driven software solutions that improve how businesses interact and do business with their customers. Sugar includes modules Marketing Automation, Sales Automation and Customer Service. Refer to the links below to read more about the SugarCRM:

https://www.sugarcrm.com/solutions/

2.2 SugarCRM Adapter

The SugarCRM receiver adapter enables an SAP Cloud Integration tenant to accelerate the implementation time and reduce the complexity of connecting to SugarCRM based upon its API.

2.2.1 Features

From a high-level perspective, the following key features are provided by the SugarCRM adapter:

- Support for multiple versions: The SugarCRM adapter lets you integrate with SugarCRM using your preferred API version. At the moment of publishing this documentation, the adapter supports SugarCRM API V10, V11 and V11.1.
- Support multiple operations, including:
    - Create Record
    - Delete Record
    - Get Record by ID
    - Query Records
    - Update Record
    - Upload a File
    - Global Search
    - Bulk
- Support for OData Query parameters like filter, top, count, select and search.
- Ability to Link or Unlink records.
- Ability to retrieve all linked Objects of a Module with Query Records.
- Support outbound headers.
- Support pagination.
- Secure authentication with OAuth 2.0: Every REST call between SugarCRM and SAP CPI is secured by the OAuth 2.0 industry-standard for authorization.
- Dynamic configuration with headers and properties: It is possible to set up your scenarios using SAP CPI dynamic values from the header and properties stored in the exchange of your SAP CPI tenant.
- Support for XML and JSON formats: The adapter can handle both XML and JSON input and response messages.
- XSD generation: The adapter comes equipped with an Eclipse Plug-in which generates XSDs representing the object in SugarCRM.

2.3 Architecture Overview

From a technical perspective, the SugarCRM adapter can only be used on the receiver side. This means that the SAP Cloud Integration is the initiator of the calls. In case the calls toward SugarCRM need to be scheduled, use the Scheduler within the integration flow.

Figure 2.1 shows how the SugarCRM adapter can be used in a simple request-reply scenario. Various operations can be used in the adapter – such as query data, delete or create an entity etc. – which are explained in detail in Section 3. To learn more about the different adapter operations and their configuration, please refer to Section 9.

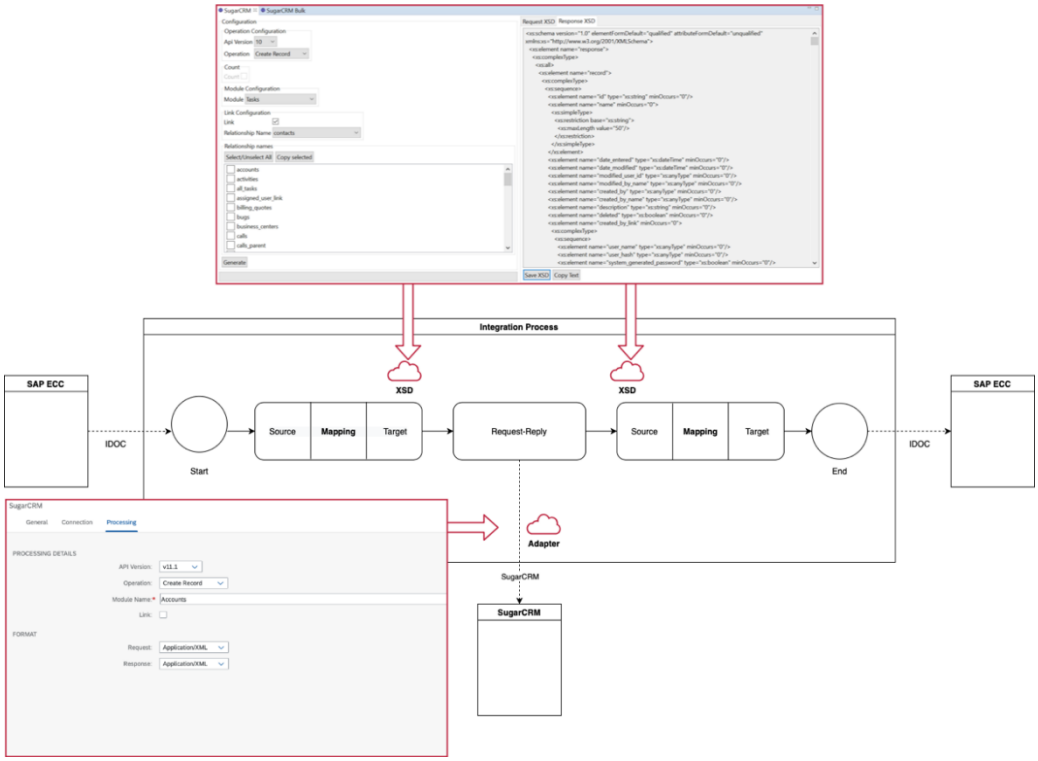

Figure 2.1 SugarCRM Adapter solutions for SAP CPI

The mapping step makes mapping to other system's message types easy as there are several XSDs created for your SugarCRM adapter operations. For the SugarCRM plug-in and XSD Generator configuration please refer to Section 7.

3 Supported Operations and Versions

Currently, the API versions V10, V11 and V11.1 are supported by the adapter. **Error! Reference source not found.** lists the different operations that are supported.

Operation	Description
Create Record	Create a record specific to a module in SugarCRM. For example: Account. This operation can also be used to create a link to an existing record. For example: You can create a new Contact and add it to an existing Account.
Delete Record	Delete a record or a link between two records.
Get Record by ID	Get one record or linked entity of a record by providing the ID.
Query Records	Retrieve records or linked relationships of records by using various query options.
Update Record	Update a record or the linked relationship of a record.
Upload a File	Upload a file/attachment to a field of record.
Global Search	Execute a free text search on the specified module.
Bulk	Perform multiple operations in one call to SugarCRM. Each operation will have its dedicated request and response payload.

Table 1 Description of the supported Operations

For a more detailed explanation of each of the above operations and how to configure them, refer to Section 9.2.

4 SugarCRM Configuration

To connect SAP CPI to SugarCRM, we need to create a user and an OAuth Consumer key in SugarCRM. Please follow this guide to generate these.

4.1 Create User

To create a user in SugarCRM follow the below steps:

1. Go to the Administration page within the SugarCRM portal. Click User Management.

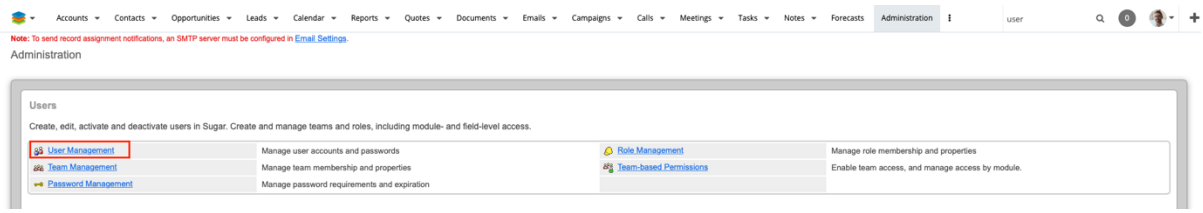

Figure 4.1 SugarCRM Dashboard

2. To create a user click Create New User from the Users menu.

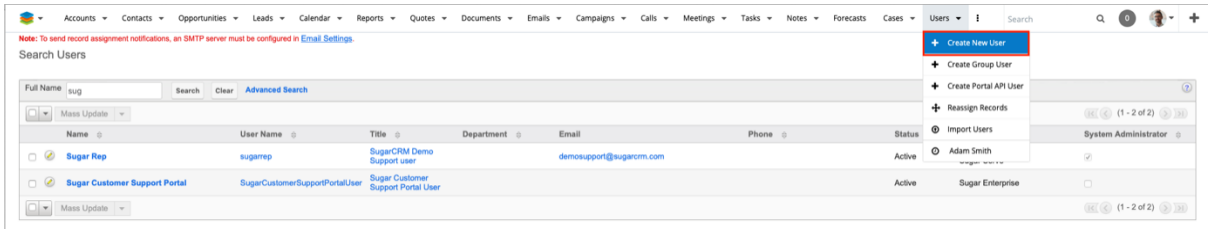

Figure 4.2 SugarCRM User Management

3. On the next page provide a User Name, Status, Last Name, License Type and email address and click Save:

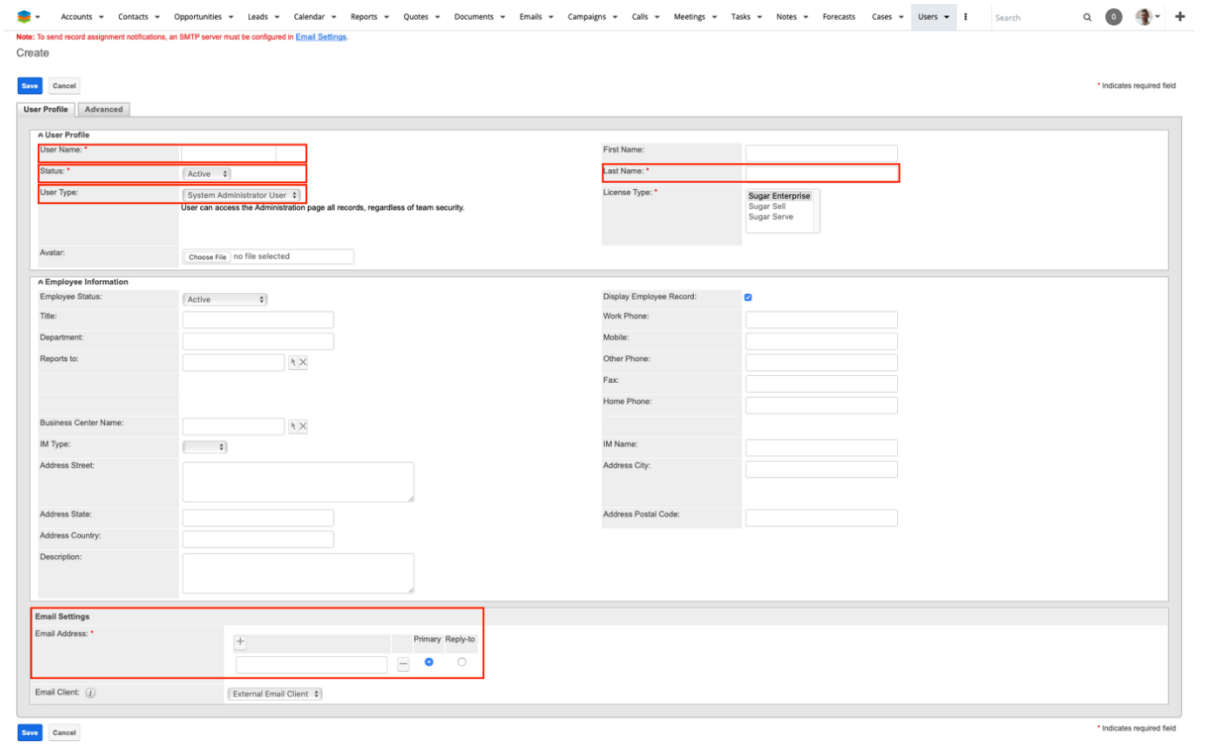

Figure 4.3 SugarCRM User creation

Section	Parameter name	Description
User creation	User Name*	Provide a username. Example: Administrator
	Status*	The status of a user can be Active or Inactive. Select Active for the new user.
	Last Name*	Provide a last name for the user.
	User Type	The user type can be either Regular User or System Administator User. Select the type System Administrator User for the new user.
	License Type*	Select the appropriate License Type.
	Email Addresses*	Specify an email address. Via this email address the password for this user can be set. Example: admin@example.com

Note the Username and Password as you will need these to create credentials in SAP CPI.

4.2 Generate OAuth Keys

After the user is created, create an Oath Consumer Key by following the below steps.

1. Navigate to the Administration page and click OAuth Keys to open the OAuth Key Management console.

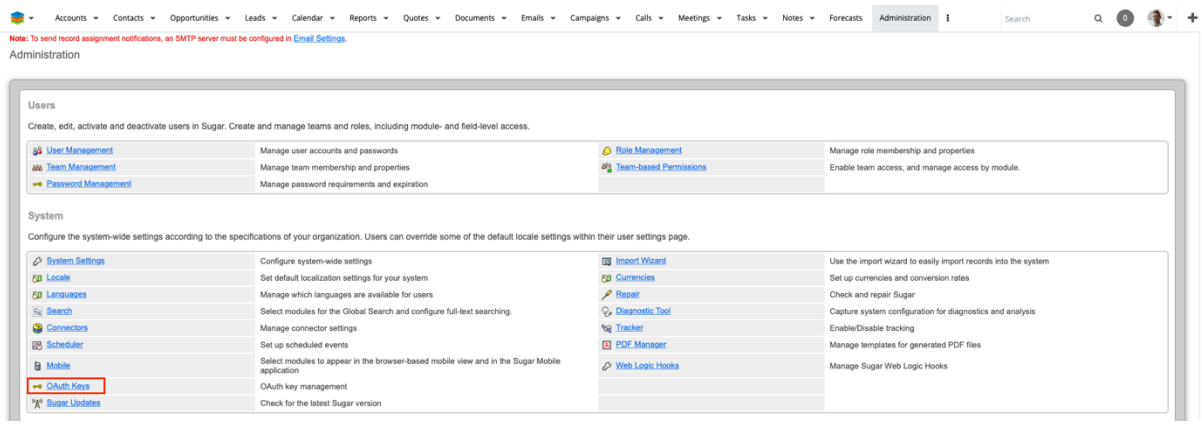

Figure 4.4 SugarCRM Dashboard

2. To create a new OAuth key click Create OAuth Key from the OAuth Keys drop-down menu:

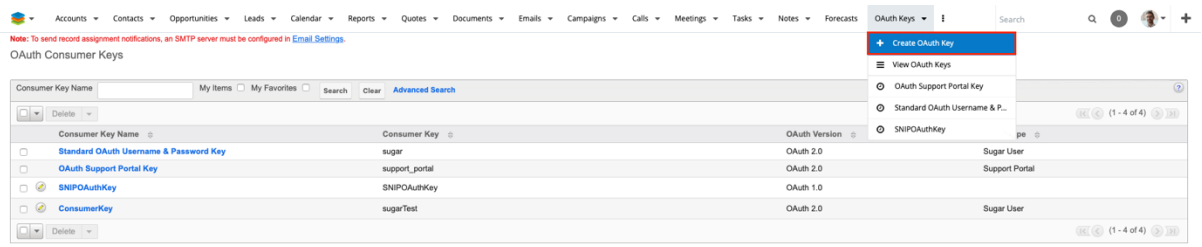

Figure 4.5 SugarCRM OAuth creation

3. On the next page provide a Consumer Key Name, Consumer Key, Consumer Secret, OAuth Version and a Client Type:

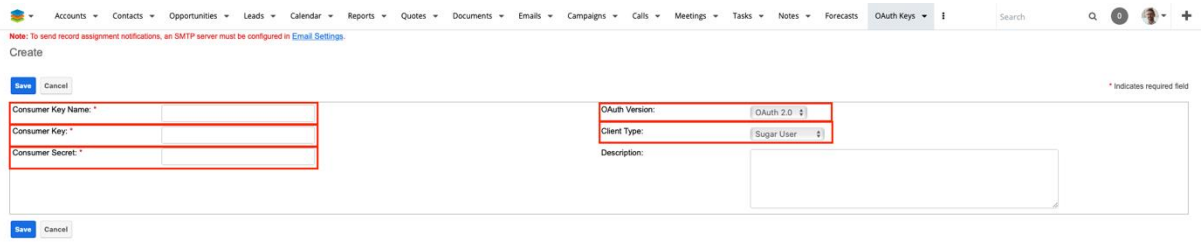

Figure 4.6 SugarCRM OAuth configuration

For more details on the above properties, refer to the following table.

Section	Parameter name	Description
OAuth creation	Consumer Key Name*	Provide a Consumer Key Name, this is the name used in SugarCRM to store this OAuth key. Example: SAP_CPI
	Consumer Key*	Provide a Consumer Key. Example: SAPCPIKey
	Consumer Secret*	Provide a Consumer Secret, make sure this is long and complex.
	OAuth Version	Select OAuth 2.0. This is the OAuth version used by the adapter.
	Client Type	Select Sugar User. This is the type of user created in the previous step.

Table 2 OAuth Keys

Do not lose the values of the Consumer Key and Consumer Secret as you will need to create secure parameters in SAP CPI for these values.

5 Adapter Installation

Refer to the **SugarCRM Adapter Installation Guide** which is included in the package for detailed steps on installing the SugarCRM Adapter.

6 Eclipse Plug-in Installation

Refer to the **Eclipse Plug-in Installation Guide** which is included in the package for detailed steps on installing the Eclipse Plug-in.

7 Use of the Eclipse Plug-in

The Eclipse plug-in or workbench can be used to extract and generate XSDs of the different SugarCRM entities. These XSDs can then be used in mappings for the request or response messages in your integration scenarios. The next sections will explain how to use it.

7.1 Configuration of SugarCRM Tenant Login Information

The Eclipse plug-in generates the XSDs according to the metadata of the business objects or entities that are part of a given SugarCRM tenant. To retrieve the metadata, the users need to configure the connection and authentication information for the target tenant. To configure the connections and authentications, click on the **Window** option at the Eclipse menu bar and select the **Preferences** menu item. On the left side of the dialogue box, navigate to the **SugarCRM** preference page. You get a page that looks like Figure 7.1.

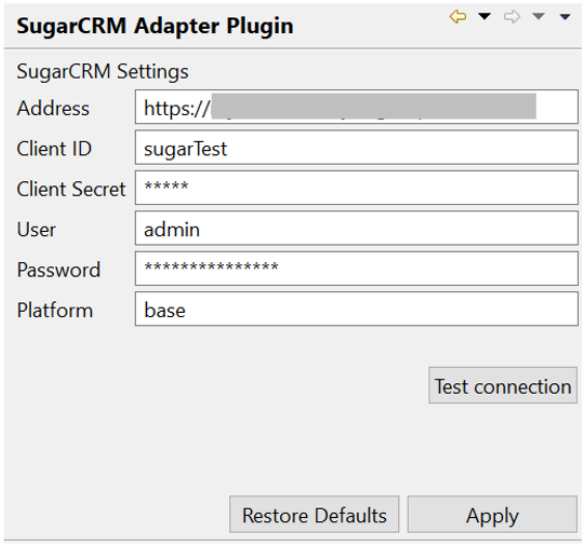

Figure 7.1 Configuring the connection details of the plug-in

On the preference page, the **Address** field refers to the address of the target SugarCRM tenant. Example: https://yoursugarcrm.tenant.eu. The **Client ID** and the **Client Secret** are the **Consumer Key** and **Consumer Secret** which are configured in SugarCRM. Please follow the guide in Section 4 for extracting this information. Next, a **User** and a **Password** should be provided. Note that these four fields correspond to the same information required in the adapter's **Connection** tab – See Section 9.1. After providing these details, click **Apply and Close**.

7.2 Generate XSD

The SugarCRM Adapter Eclipse Plug-in is divided into two tabs, namely the SugarCRM and SugarCRM Bulk. The SugarCRM tab allows the creation of schemas for single operations while the Bulk tab allows multiple operations per schema through the bulk process. In the next sections, we will discuss each of these tabs.

7.2.1 Single Operation

The Single Operation tab configuration pane is situated in the left side of the window. The configuration has five sections; **Operation**, **Count**, **Module**, **Link**, **Relationships** and **Response fields**, as shown in Figure 7.2.

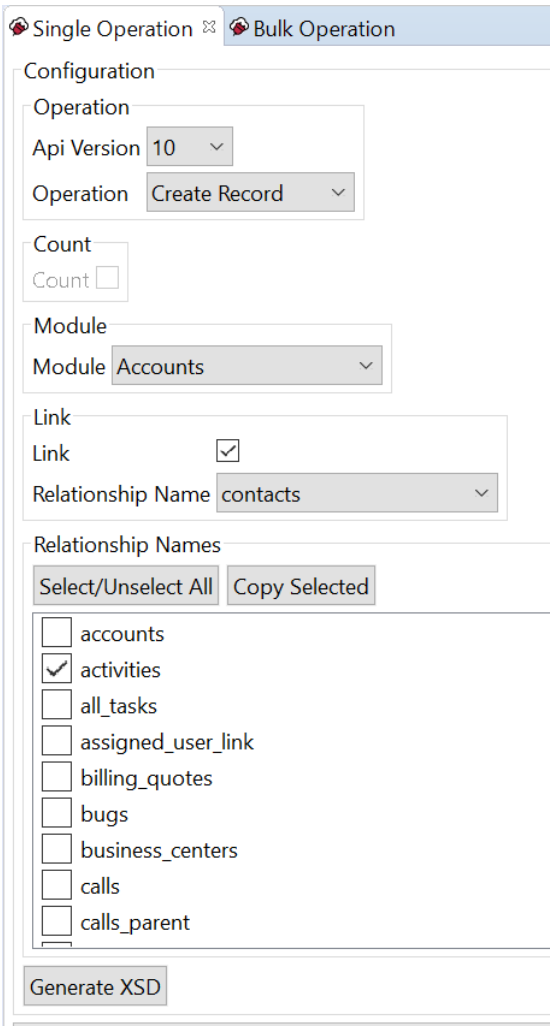

Figure 7.2 SugarCRM plug-in configurations

Table 3 provides details for each of these configuration settings.

Section	Property	Description
Operation	API Version	Version of the SugarCRM REST Api. The workbench supports V10, V11 and V11.1

	Operation	Select the operation for which you want to generate the XSD. • Create Record • Delete Record • Get Record by ID • Query Records • Update Record • Upload a File
Count	Count	Select if you want to get the total number of records for the operation Query Records.
Module	Module Name	Select the module name from the dropdown. For example: Accounts.
Link	Link	Select in case the operation needs to be executed for a relationship of the selected Module.
	Relationship Name	This dropdown populates all the possible relationships of the selected Module. For example: contacts.
Response Fields		Check all the fields which you want to be a part of the XSD. The set of fields depends upon the selected module or linked relationship.

Table 3 SugarCRM plug-in configuration setting details

You can explore the functionalities of the Eclipse plug-in by selecting a version and an operation that fits your needs.

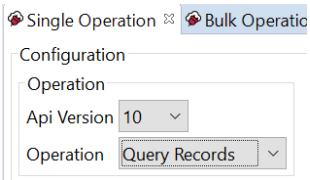

Figure 7.3 SugarCRM Processing Details

Depending on the selected Operation, the Count configuration is enabled. In case you select Query then Count is applicable, otherwise the checkbox stays inactive as shown in Figure 7.4.

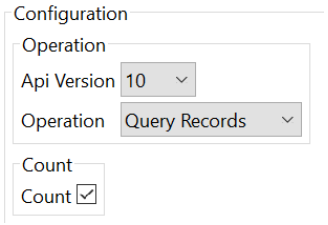

Figure 7.4 SugarCRM count configuration

Please note the if the count property is checked, the plug-in will only generate a Response XSD.

Based on your requirements, select the Module on which the operation should to carry out for. In case you are trying to interact with the linked relationships, then check the Link checkbox and select the Relationship you want to operate upon. For example, if you want to create a contact for an Account then select the Module as Account and the Relationship Name as contact. This setting is shown in Figure 7.5.

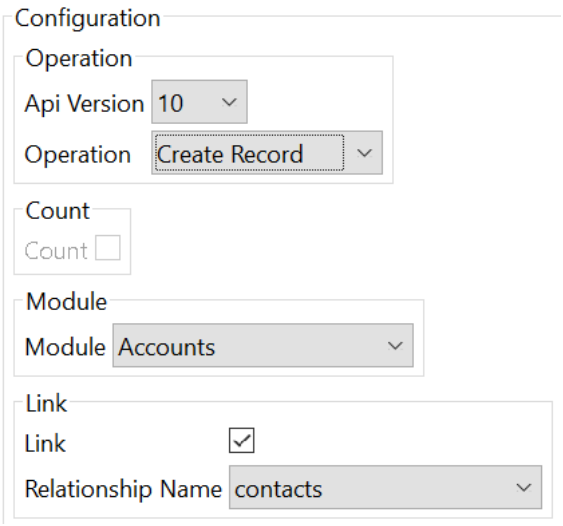

Figure 7.5 SugarCRM module and linked relationship configuration

In SugarCRM, it is also possible to interact with relationships of a specific Linked module. For example, while creating a contact for an Account, if you want to also create a call for this contact, then you should select the Relationship Name calls from the Relationship names configuration as shown in Figure 7.6.

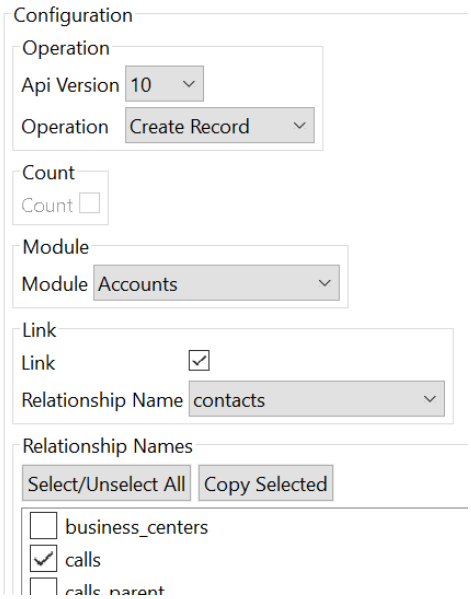

Figure 7.6 SugarCRM relationship name settings

Once you have configured the settings as per your scenario, click the Generate button to generate XSDs. On the right side of the screen, you will find three tabs: **Request XSD** and **Response XSD**. The Request XSD tab contains the XSD that represents the properties of the selected entity in SugarCRM. Note that this tab is only filled in for the Create Record and Update Record operations. This XSD can be copied or saved on your local machine. It can later be used in a mapping of your integration flow; see Figure 7.7.

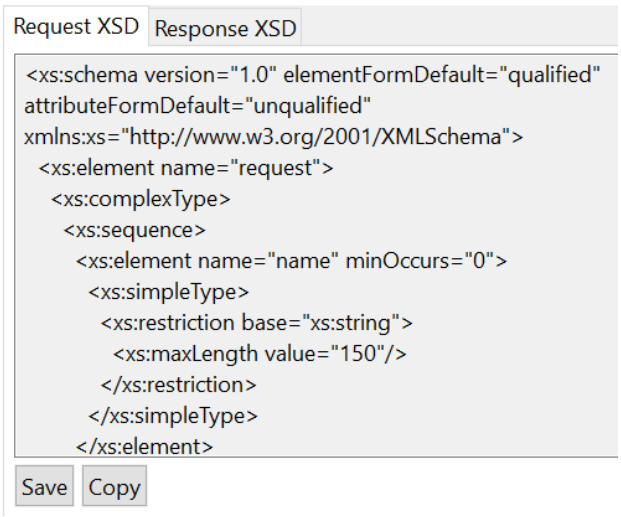

Figure 7.7 Example Request XSD

Similarly, the Response XSD tab contains the XSD that represents the properties of the selected module. But, this XSD only includes properties that are returned in the response of the specified operation. As an illustration, if we are using the create operation, the primary key property might not be present in the input (Request XSD), but will be present in the response XSD; see Figure 7.8. In contrast with the request XSD tab, the Response XSD tab is always filled for all the operations.

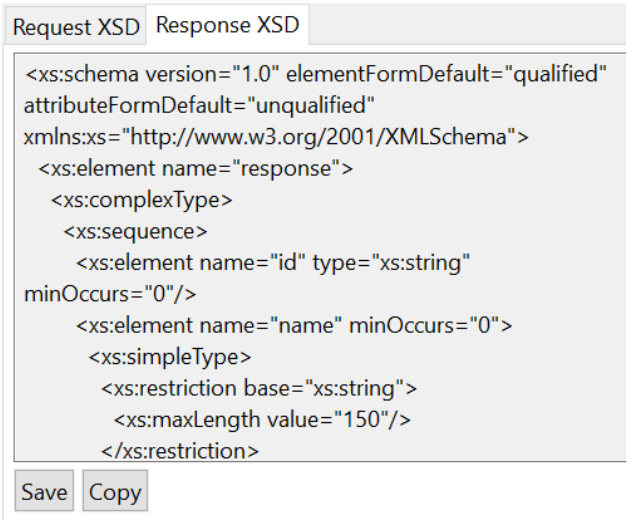

Figure 7.8 Example Response XSD

The generated XSDs can be copied or saved to the local computer using the Save and Copy buttons; see bottom left corner of Figure 7.8.

7.2.2 Bulk Operation

The Bulk Operation view includes a table that allows to add more entities to the XSD. Besides the entities, you also need to select an operation for each entity and whether to return a response or not.

To populate the table, you need to click on the **Add operation** button at the top of the screen (see Figure 7.9). Clicking the button will open a configuration window, which looks exactly like the configuration pane of the SugarCRM Single Operation tab. Fill in the information based on your requirements and click the **Add operation** button on the bottom of the window. You can add as many entities as required for your scenario. Lines can be removed by clicking the **Delete selected** button.

After adding all the required entities, click on the **Generate** button to create an XSD schema.

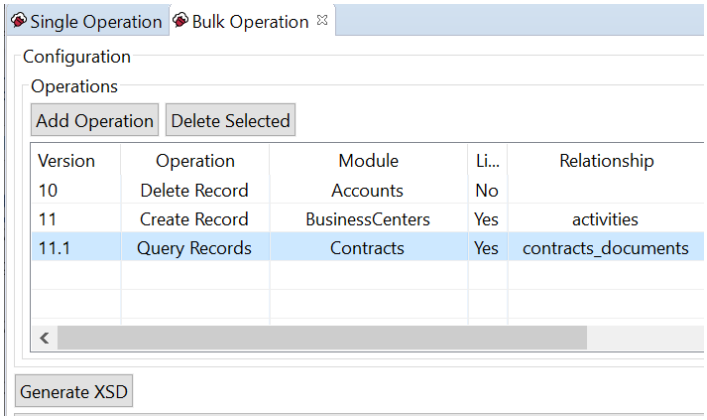

Figure 7.9 Batch Processing Mode Options

After clicking on the Generate button, the generated XSDs can be copied from the right side of the plug-in. These are the same three tabs that were discussed in Section 7.2.1.

8 Creating Credential and Client ID/Secret as Security Material

To follow the best practices and the security policy of SAP Cloud Integration, the adapter works with the standard security artifacts such as Secure Parameter and User Credentials that are stored in the SAP secure store. This way, the user password and secrets can be safely provided to the adapter using aliases. The next sections provide details on how to create a Security Parameter and user credential artifact that will be needed to configure the adapter later on. In SAP Cloud Integration, go to the **Operations View** and select **Security Material** under the **Manage Security** section.

8.1 Create User Credential

In Security Material click on the **Create** dropdown at the top right and select **User Credential.** Provide a name, select **User Credential** in the Type dropdown, fill with SugarCRM username and password. Then click on deploy.

8.2 Create Client ID and Client Secret

In Security Material, click on the **Create** dropdown at the top right and select **Secure Parameter**. Provide a name for the secure artifact and fill in your Client ID in the Secure Parameter. Note that the value for the Client ID should be retrieved from SugarCRM OAuth key setup which was created in Section 4.2. Then Deploy the artifact. Repeat the same steps for Client Secret.

9 Adapter Configuration

This section describes different parameters that can be configured for the Receiver Adapter. The adapter has the Connection and Processing tabs. Each one of these will be discussed in the next sections. Please note that when the (*) is present in front of a property, it indicates that the property is mandatory.

9.1 Connection Tab

The Connection tab contains parameters defining how to connect and authenticates with SugarCRM. See an example in Figure 9.1.

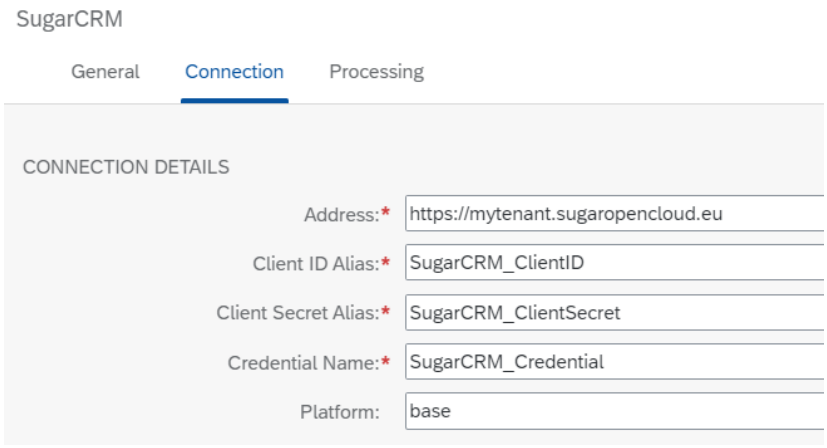

Figure 9.1 Example of filled Connection Tab

The details of each field included in the Connection tab are explained in Table 4.

Tab	Parameter name	Description
Connection	Address*	Specify the recipient's endpoint URL. This is the URL to which you connect to your SugarCRM tenant. Value can also be read dynamically. To learn how to find this address, refer to Section 11.1. Example: https://mytenant.sugaropencloud.eu
	Client ID Alias*	Specify the Client ID Alias that you created as a security material in your Cloud Integration (see section 8.2). Note that it is also possible to specify this value using property or header. Example: ${header.<name>}or ${property.<name>}
	Client Secret Alias*	Specify the Client Secret Alias that you created as a security material in your Cloud Integration (see section 8.2). Note that it is also possible to specify this value using property or header. Example: ${header.<name>}or ${property.<name>}
	Credential Name*	Specify the credential name used to authenticate against the server. This represents the SugarCRM credential name (username-password) pair stored as a Security Material. You created this User Credential in your Cloud Integration (see section 8.1).

	Platform	Specify the SugarCRM platform name for the connection. A platform provides the possibility to define custom metadata and views in SugarCRM, hence supporting customers to customize their experience. The default platform used for connection is base, some other platforms are Mobile and Portal. Register the platform using SugarCRM Platform Extensions or via the Administration Panel. Note that it is also possible to specify this value using property or header.

Table 4 Connection tab description

9.2 Processing Tab

In the **Processing tab,** you can specify the type of operation to be performed. Then select the version of SugarCRM API to be used from the **API Version** dropdown. Note that currently versions 10, 11 and 11.1 are supported by the adapter.

Operation refers to the activity that you want to perform; for example, getting data from SugarCRM or creating records in SugarCRM. Select the required operation from the Operation dropdown. The adapter supports eight operations; namely: Create, Update, Query, Delete, Get Record by ID, Upload a File, Global Search and Bulk. Figure 9.2 illustrates the basic options of the Processing tab.

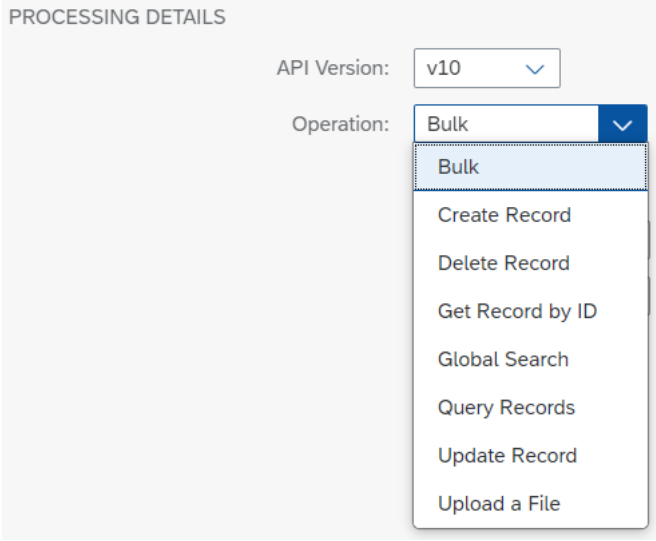

Figure 9.2 Processing tab SugarCRM adapter

Depending on the selected Operation, you need to specify different options. In the later sections we will cover each operation in detail.

9.2.1 Get Record by ID

The operation Get Record by ID will help you retrieve all information on an entity by providing its id to the adapter. The configurations for this operation is illustrated in the table below.

Parameter name	Description
Module Name*	Provide a module name. Example: Accounts, Contacts, Tasks For full list of modules, refer to the plug-in.
Module Record ID*	Provide the unique record ID for which you want to extract the data.
Response	This field specifies the format of the response message to be returned by SugarCRM. Possible values include Application/XML and Application/JSON. Note that the default value is Application/XML.

Table 5 Get Object by ID configuration

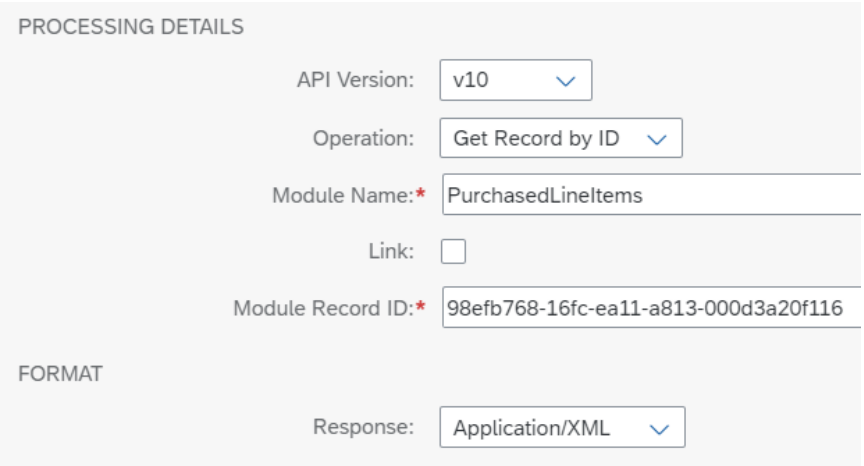

Figure 9.3 Example Get Object By ID

Figure 9.3 illustrates an example scenario to retrieve a *PurchasedLineItem* from SugarCRM with the ID "98efb768-16fc-ea11-a813-000d3a20f116" in XML format. Note that instead of using the value directly as illustrated above, it is also possible to use a property or a header value.

In SugarCRM it is also possible to extract a record for a Linked relationship.

9.2.1.1 Get Record by ID for a Linked Relationship

Records can be linked to each other, in SugarCRM these relationships between records are referred as Linked Relationships. For example, an Account can have multiple Contacts or Calls. To get the information of a specific Contact of a specific Account we use the Link checkbox in the Get Record by ID Operation. In addition to Module Name and Module ID you also need to specify the properties listed in Table 6.

Parameter name	Description
Link	Select to perform operation on the relationship of the specified SugarCRM Module.
Relationship Name*	Provide the Relationship name of the specified Module. Example: Contacts, Documents, Email For full list of Relationships of a specific Module, refer to the plug-in.
Relationship Record ID*	Provide the unique record ID for which you want to extract the data.

Table 6 Get Record by ID for Linked Relationships

Figure 9.4 illustrates the scenario where you want to extract a specific contract associated with the Account "98efb768-16fc-ea11-a813-000d3a20f116".

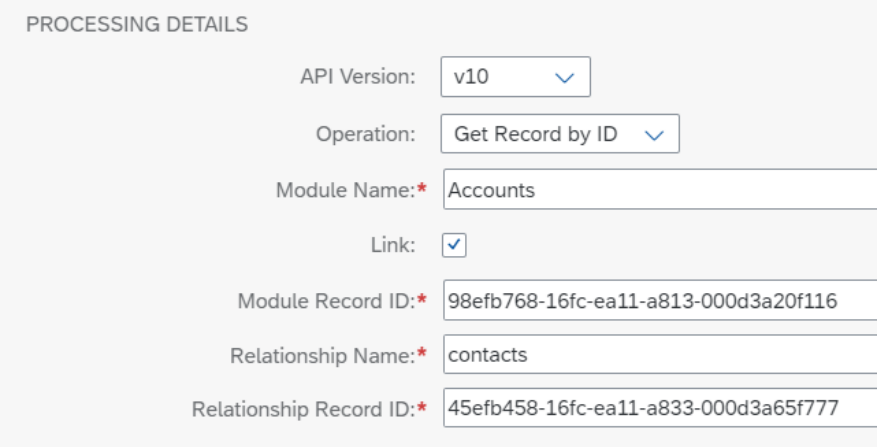

Figure 9.4 Get Record By ID for Linked Relationship

Please note that the set of linked relationships depend on the selected Module. Use the Plug-in to get the list of relations based on the module.

9.2.2 Query Records

The Operation Query Records can be used to extract data from your SugarCRM tenant. The difference between Get Object by ID and Query Objects is that you can get multiple instances when you query for entities. Whereas the Get Object by ID operation only retrieves one instance. Table 7 lists the options available for Query Records operation.

Parameter name	Description
Module Name*	Provide a module name. Example: Accounts, Contacts, Tasks
Limit	Limit the number of records retrieved. This value can also be set dynamically via the exchange properties by setting the property named **SugarCRM_limit**. The Limit defaults to 20. Note that the value set dynamically is leading. Example static: 5
Offset	Skip the first number of records retrieved. This value can also be set dynamically via the exchange properties by using setting the property named **SugarCRM_offset**. Note that the value set dynamically is leading Example static: 5
Response fields	List the fields required in the response. It is also possible to copy the list of response fields from the plug-in Example: Name, ID, Date
Count	Check this checkbox if only the number of records should be retrieved.
Filter Type	A filter can be created to filter the records. Example: None, Simple, And, Or
Filter ID	Refers to the identifier for a pre-existing filter. Custom filters can be created in SugarCRM WebUI. If filter is also set, the two filters are joined with an AND.
View	Instead of defining the fields argument, the view argument can be used. The field list is constructed at the server side based on the

	view definition which is requested. Common views are "record" and "list".
Search	Refers to 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.
Deleted	Boolean to show deleted records in the result set.
Nulls Last	Boolean to return records with null values in order_by fields last in the result set.
Order By	The order of the records in the response can be specified. To create a rule click add and provide a field name and an order. Example: Field name: Name, ID Order: Ascending, Descending

Table 7 Query Records configuration

Let us illustrate the above settings by an example. If you want to extract the name, ID and lastModifiedDate of top 20 Accounts from SugarCRM, then the adapter configuration would look similar to Figure 9.5.

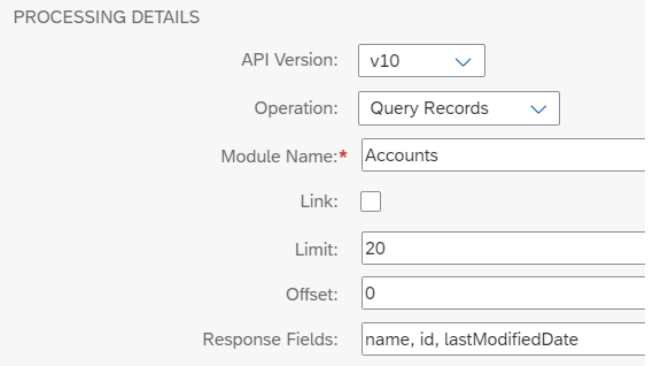

Figure 9.5 Example Query Objects

If you want to sort the response records by the lastModifiedDate then you can set the Order By table as shown in Figure 9.6.

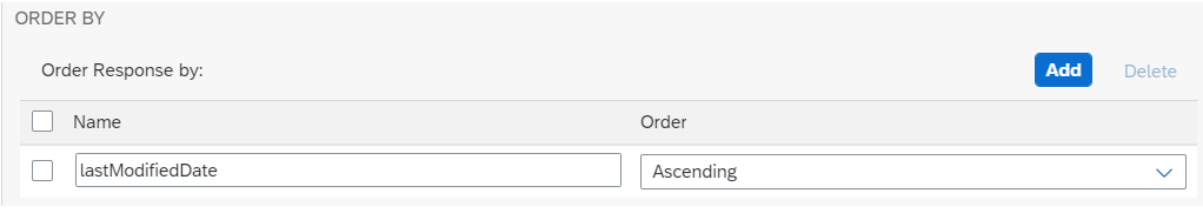

Figure 9.6 Example Query Objects with Order By

There are many more possibilities to manipulate the expected response. In the following sections we will discuss more about these.

9.2.2.1 Pagination

Pagination is helpful if you want to retrieve large number of records. Such operations can take long time and can even result in timeouts. Using pagination,you will configure the adapter in such a way that x number of records are extracted in each call to SugarCRM. If you have 1000 records, you can break this up into 10 pages of 100 records each. To achieve this, you need to set the limit to 100 which will limit the returned result to 100 records for each call you.

In general, if there are more records than the ones returned in the response, SugarCRM will return a field called **next_offset** in response. To get to first 100 records the value of Offset would be 0, after that for each subsequent call you will increment the value of Offset by 100 or use the value from the next_Offset field of the response message. You can use this value to dynamically set the **SugarCRM_offset** exchange property of the adapter to get the next set of results. The value of **next_offset** is -1 when all the records are returned. Keep in mind if you set the exchange property **SugarCRM_offset** then the value mentioned in the adapter will ignored.

Figure 9.7 Pagination options for SugarCRM

To extract all records effectively in SAP Cloud Integration you can make use of looping subprocesses.

9.2.2.2 Filters

It is possible to Filter records while executing the Query Operation in SugarCRM. The adapter supports three types of Filters namely: Simple, And and Or.

9.2.2.2.1 Simple Filter

When you select the Simple filter from the Filter Type dropdown, you will be presented with three additional options as listed in Table 8.

Parameter Name	Description
Name	Specify the Field name on which the records should be filtered. For example: last_name.
Operator	Select the operator to be determine the operation to be performed between name and value. For example: Equals, In, Ends etc.
Value	Specify the value on which the filtering need to happen.

Table 8 Simple Filter options

Figure illustrates a simple filter to get all contacts where the last name is Smith.

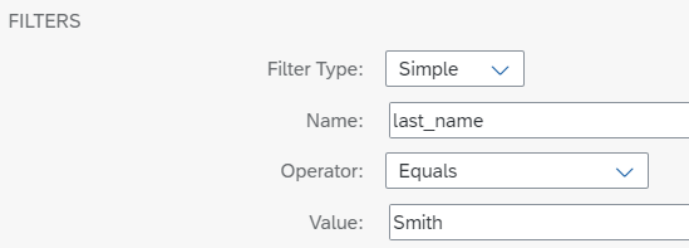

Figure 9.8 Simple Filter

9.2.2.2.2 And Filter

When you select the And filter, you will get a Filter Table with three columns; name, operator and value. The configuration is same as you do for the simple filter, but here you are able to add rows to the table. **AND** operation would be executed between the conditions mentioned in each row.

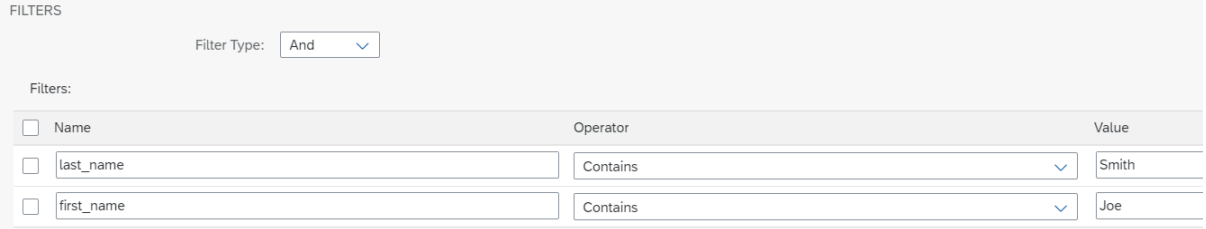

Figure 9.9 And Filter

Figure 9.9 illustrates the filtering of records where last_name contains the value Smith and first_name contains the value Joe.

9.2.2.2.3 Or Filter

When you select the Or filter, you will get a Filter Table with three columns; name, operator and value. The configuration is same as you do for the simple filter, but here you are able to add rows to the table. **OR** operation would be executed between the conditions mentioned in each row.

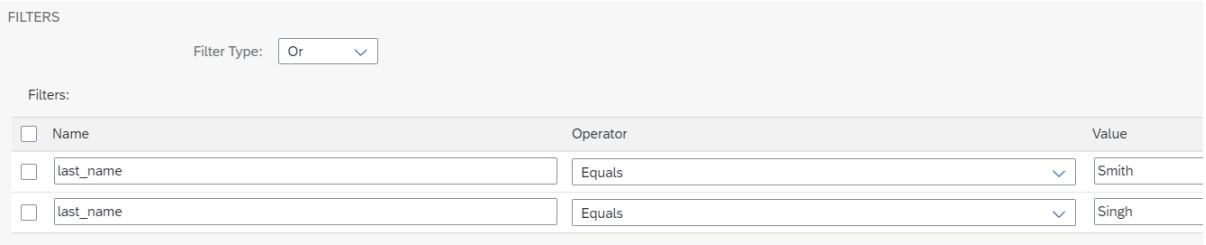

Figure 9.10 Or Filter

Figure 9.10 illustrates setup of Filter for or condition.

9.2.2.3 Query Records for a Linked Relationship

Linked Relationships option allows you to query not only the Records of a specific Module, but all the associated relations. For example, you can query all the contacts of a specific account. To do so, enable the **Link** check box.

Parameter name	Description
Relationship Name*	Provide the Relationship name of the specified Module. Example: Contacts, Documents, Email For full list of Relationships of a specific Module, refer to the plug-in.
Relationship Record ID*	Provide the unique record ID for which you want to extract the data.

Table 9 Query Records for a Linked Relationship

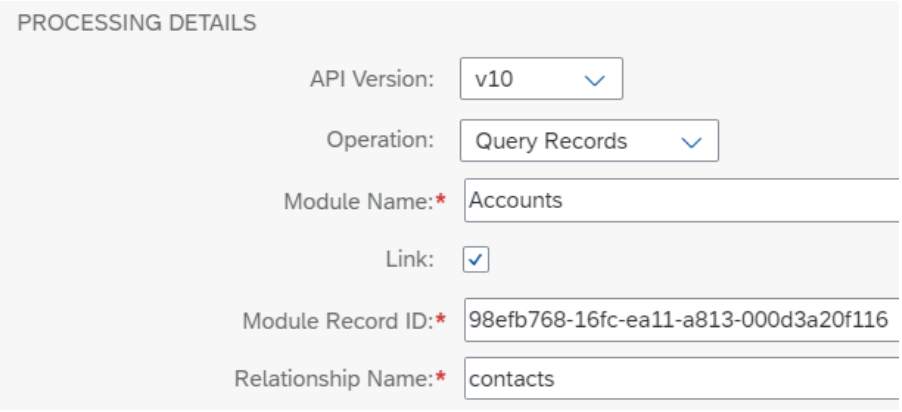

Figure 9.11 Query Records example

Figure 9.11 illustrates an example scenario to retrieve a PurchasedLineItem from SugarCRM with ID "98efb768-16fc-ea11-a813-000d3a20f116" in XML.

9.2.3 Create Record

The operation Create Record helps you create a new record of an entity or a business object. To configure the create record operation, the properties specified in Table 10.

Parameter name	Description
Module Name*	Provide a module name. Example: Accounts, Contacts, Tasks For full list of modules, refer to the plug-in.
Request	This field specifies the format of the request message to be sent to SugarCRM. Possible values include Application/XML and Application/JSON. Note that the default value is Application/XML
Response	This field specifies the format of the response message to be returned by SugarCRM. Possible values include Application/XML and Application/JSON. Note that the default value is Application/XML.

Table 10 Parameters for configuring Create Record

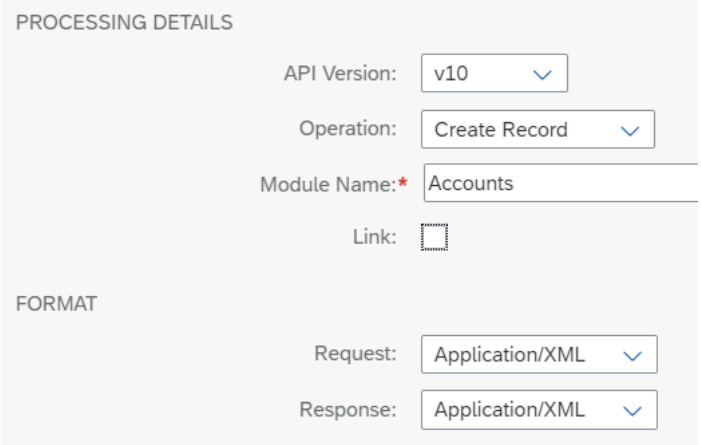

Figure 9.12 Example Create Object

For illustration purposes, in the example in Figure 9.12, we create an account in SugarCRM.

In addition to the adapter configuration, you also need to send a Payload in form of JSON or XML (using the Request format). You can use the Eclipse Plug-in to generate the XSDs based on your format choice, which can then be used to create the message mapping in front of the adapter.

9.2.3.1 Create Record for Linked Relationship

Linked Relationships option allows you to create not only the Records of a specific Module, but all the associated relations. For example, you can create a call for a specific Account. To do so, enable the **Link** check box.

Parameter name	Description
Relationship Name*	Provide the Relationship name of the specified Module. Example: Contacts, Documents, Email For full list of Relationships of a specific Module, refer to the plug-in.

Table 11 Parameters required for Create Records for Linked Relationship

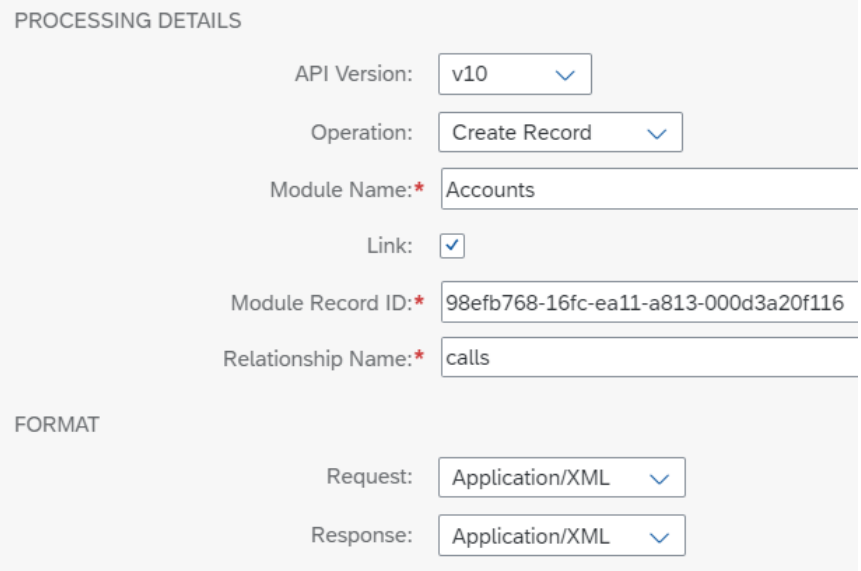

Figure 9.13 Create Record for a Linked Relationship

Figure 9.13 illustrates an example scenario to create a call for the Account with ID "98efb768-16fc-ea11-a813-000d3a20f116" by passing data in XML format.

Note that also for this operation you need to pass either XML or JSON data related to the Linked relation you want to create.

9.2.4 Update Record

The Update Record operation allows you to update the properties of a specific record of an entity or a business object. For this operation, the ID of the concerned entity needs to be provided. Table 12 describes various parameters required to configure the Update Record.

Parameter name	Description
Module Name*	Provide a module name. Example: Accounts, Contacts, Tasks For full list of modules, refer to the plug-in.
Module Record ID*	Provide the Record ID of the SugarCRM module for which the update operation should be executed.
Request	This field specifies the format of the request message to be sent to SugarCRM. Possible values include Application/XML and Application/JSON. Note that the default value is Application/XML

Response	This field specifies the format of the response message to be returned by SugarCRM. Possible values include Application/XML and Application/JSON. Note that the default value is Application/XML.

Table 12 Parameters for configuring Update Record

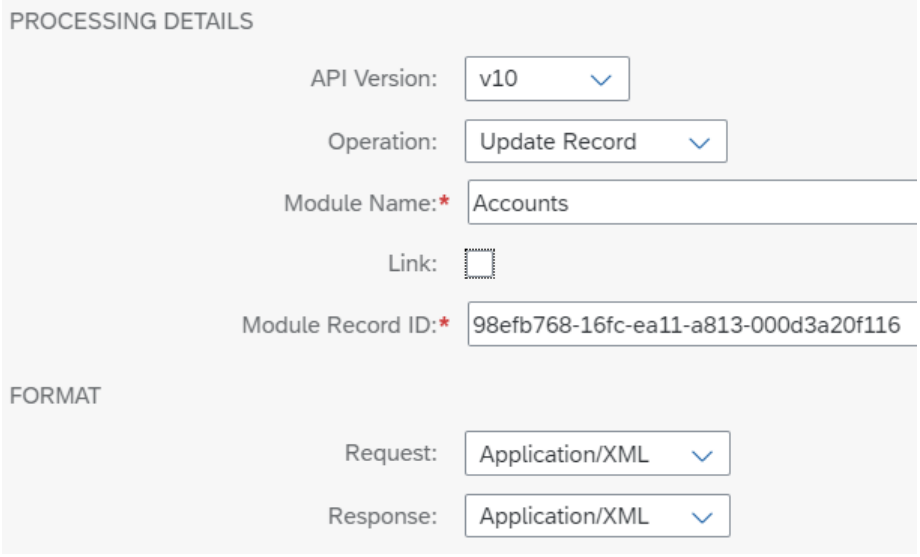

Figure 9.14 Update Record operation parameters

The example in Figure 9.14 updates an Account in SugarCRM with the ID "98efb768-16fc-ea11-a813-000d3a20f116".

In addition to the adapter configuration, you also need to send a Payload in form of JSON or XML (using the Request format). Depending on your choice of format, you can use the Eclipse Plug-in to generate the XSDs that can then later be used to create the message mapping in front of the adapter.

9.2.4.1 Update Record for Linked Relationships

Linked Relationships option allows you to update not only the Records of a specific Module, but all the associated relations. To do so, enable the **Link** check box.

Parameter name	Description
Relationship Name*	Provide the Relationship name of the specified Module. Example: Contacts, Documents, Email

	For full list of Relationships of a specific Module, refer to the plug-in.
Relationship Record ID*	Provide the unique record ID for which you want to extract the data.

Table 13 Update Records for a Linked Relationship

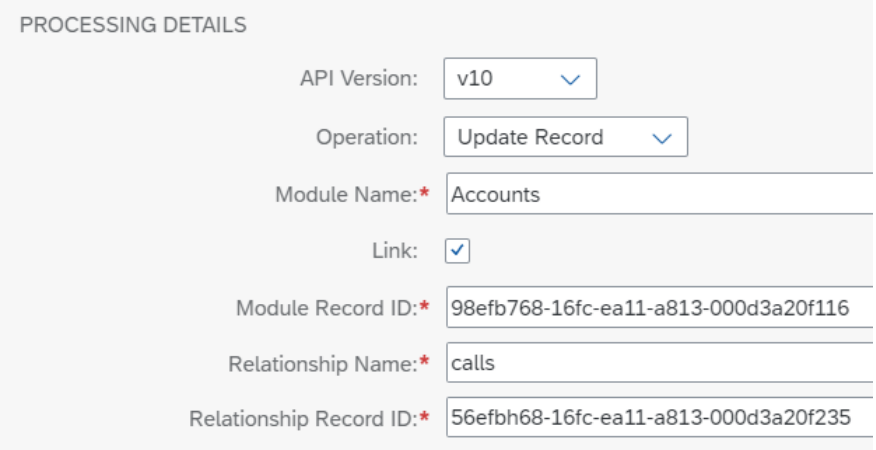

Figure 9.15 Update Record for Linked Relationship

The example in Figure 9.15, illustrates how you can update the call with ID "56efbh68-16fc-ea11-a813-000d3a20f235" which is accosiated with the Account with ID "98efb768-16fc-ea11-a813-000d3a20f116". Note that you can also use this operation to create a link between an existing records.

In addition to the adapter configuration, you also need to send a Payload in form of JSON or XML (using the Request format). Depending on your choice of format, you can use the Eclipse Plug-in to generate the XSDs that can then later be used to create the message mapping in front of the adapter.

9.2.5 Delete Record

The Delete Object operation allows you to delete a record of an entity or a business object. For this operation, the ID of the concerned entity needs to be provided.

Parameter name	Description
Module Name*	Provide a module name. Example: Accounts, Contacts, Tasks For full list of modules, refer to the plug-in.

Module Record ID*	Provide the Record ID of the SugarCRM module for which the update operation should be executed.
Response	This field specifies the format of the response message to be returned by SugarCRM. Possible values include Application/XML and Application/JSON. Note that the default value is Application/XML.

Table 14 Parameters for configuring Delete Record

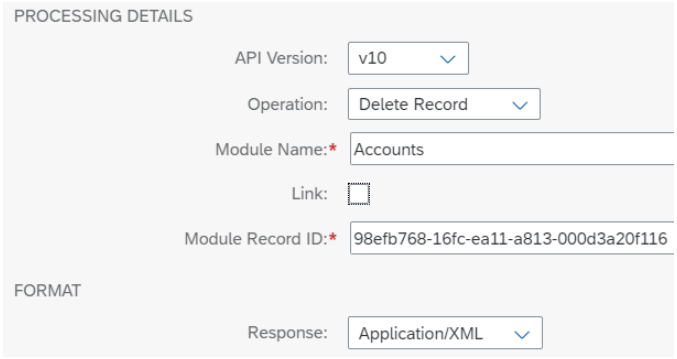

Figure 9.16 Delete record

The example in Figure 9.14 updates an Account in SugarCRM with ID "98efb768-16fc-ea11-a813-000d3a20f116".

9.2.5.1 Delete the Linked Relationship

Linked Relationships option allows you to delete the link between records. To do so enable the **Link** check box.

Parameter name	Description
Relationship Name*	Provide the Relationship name of the specified Module. Example: Contacts, Documents, Email For full list of Relationships of a specific Module, refer to the plug-in.
Relationship Record ID*	Provide the unique record ID for which you want to extract the data.

Table 15 Delete Records for a Linked Relationship

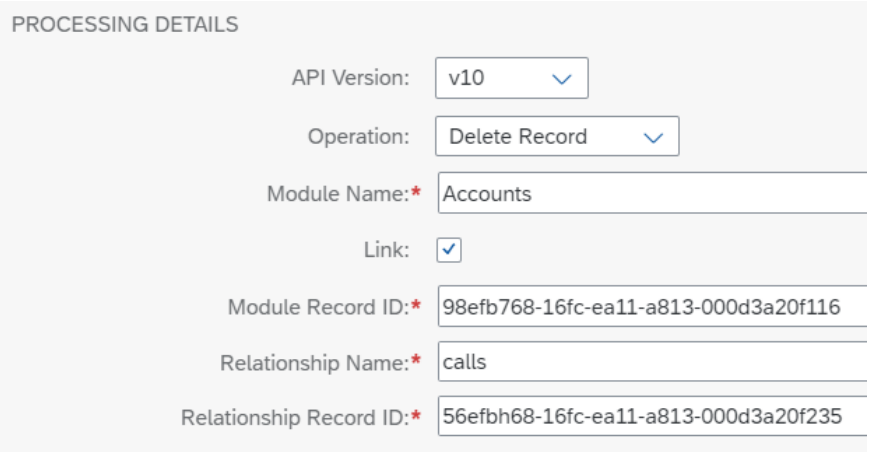

Figure 9.17 Delete Linked Relationship between records

The example in Figure 9.17 illustrates how you can delete the link between call with ID "56efbh68-16fc-ea11-a813-000d3a20f235" and the Account with ID "98efb768-16fc-ea11-a813-000d3a20f116". Note that this doesn not delete the call itself.

9.2.6 Upload a File

In SugarCRM it is possible to upload a file as an attachment to a Record. Take an example where you create a Contact. This contact can have a photo as well. The **Upload a File** operation of the adapter makes it possible for you to upload a picture/document to an existing record.

Parameter name	Description
Module Name*	Provide a module name Example: Accounts, Contacts, Tasks For full list of modules, refer to the plug-in.
Module Record ID*	Provide the Record ID of the SugarCRM module for which the update operation should be executed.
Field Name*	Specifies the target field name in the Record where the file should be uploaded. Example: Under the module name Contacts a photo of the contact can be uploaded in the field picture.
Attachment Name*	Refers to the exchange attachment name that contains the file to be uploaded to SugarCRM.

File Name*	Specifies the name of the file that you want to be visible in SugarCRM.
Content-Type	Refers to the content type of the attached file. Example: image/png, image/jpg
Response	This field specifies the format of the response message to be returned by SugarCRM. Possible values include Application/XML and Application/JSON. Note that the default value is Application/XML

Table 16 Parameters for Upload a File Operation

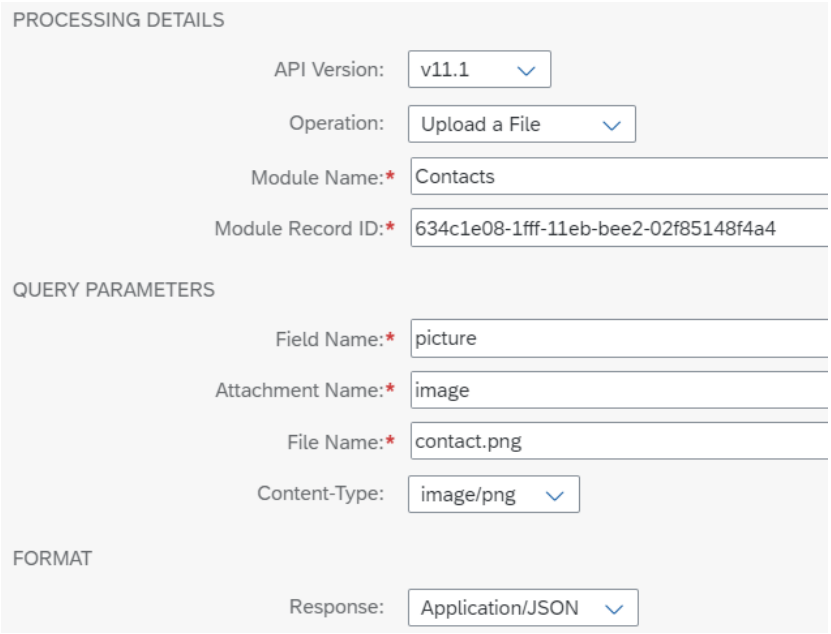

Figure 9.18 Upload a File configuration

The Figure 9.18 illustrates the example where you want to Upload an image to the **picture** field of the module called **Contact** with the ID "634c1e08-1fff-11eb-bee2-02f85148f4a4". In this case, the picture itself will be picked up from the camel attachment named **image**.

To use this operation you do not need any request payload. Just ensure that the attachment with the specified name is available in the camel attachment.

9.2.7 Bulk

SugarCRM supports multiple operations in one call using the Bulk API endpoint. In the adapter this is supported by the Bulk operation. The Bulk operation only requires the users to configure the parameters listed in Table 17.

Parameter name	Description
Request	This field specifies the format of the request message to be sent to SugarCRM. Possible values include Application/XML and Application/JSON. Note that the default value is Application/XML.
Response	This field specifies the format of the response message to be returned by SugarCRM. Possible values include Application/XML and Application/JSON. Note that the default value is Application/XML.

Table 17 Parameters for configuring Bulk operation

The Figure 9.19 illustrates the configuration of the Bulk operation.

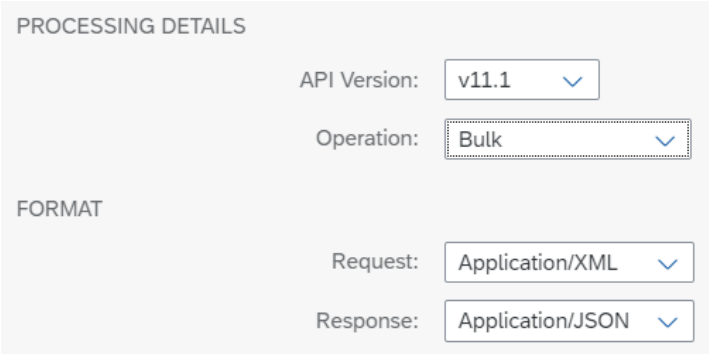

Figure 9.19 Bulk Operation configuration

All the configuration for executing a Bulk operation needs to be provided via the request payload. To generate the XSD for this operation, refer to Section 7.2.2.

9.2.8 Global Search

Global Search referes to the possibility to execute a free-text search over multiple modules of SugarCRM. Table 18 describes all the supported parameters for the Global Searh Operation.

Parameter name	Description
Limit	Limit the number of records retrieved. This value can also be set dynamically via the exchange properties by setting the property named **SugarCRM_limit**. The Limit defaults to 20. Note that the value set dynamically is leading. Example static: 5
Offset	Skip the first number of records retrieved. This value can also be set dynamically via the exchange properties by using setting the property named **SugarCRM_offset**. Note that the value set dynamically is leading Example static: 5
Module List	Provide a comma separated list of module names. Example: Accounts, Contacts, Tasks
Highlights	Select to get a tag named **highlighted** in the response. This tag contains a string where the match was found.
Search	Refers to 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.
Order By	The order of the records in the response can be specified. To create a rule click add and provide a field name and an order. Example: Field name: Name, ID Order: Ascending, Descending
Response	This field specifies the format of the response message to be returned by SugarCRM. Possible values include Application/XML and Application/JSON. Note that the default value is Application/XML.

Table 18 Query Records configuration for Global Search operation

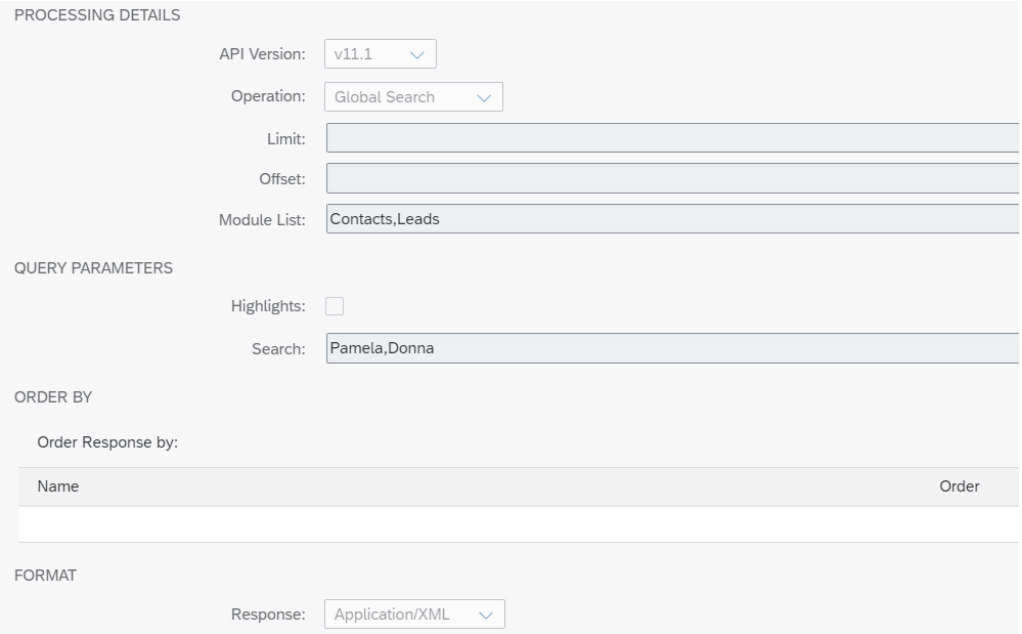

Figure 9.20 Global Search configuration

Figure 9.20 illustrates settings for Global Search where a search is performed over the Contacts and Leads and all the records which have the strings Pamela or Donna will be returned in the response.

Note that Glabal Search is not enabled by default on the SugarCRM platform. Contact your system administrator to configure it in your SugarCRM tenant.

10 Sample Scenario Explained

In this chapter, with the help of a business scenario we will explain how the SugarCRM adapter and plug-in can be used. The Figure 10.1 shows a CPI flow for the creation of an Account in SugarCRM. This iflow can be called from any external system (example: SAP S/4HANA).

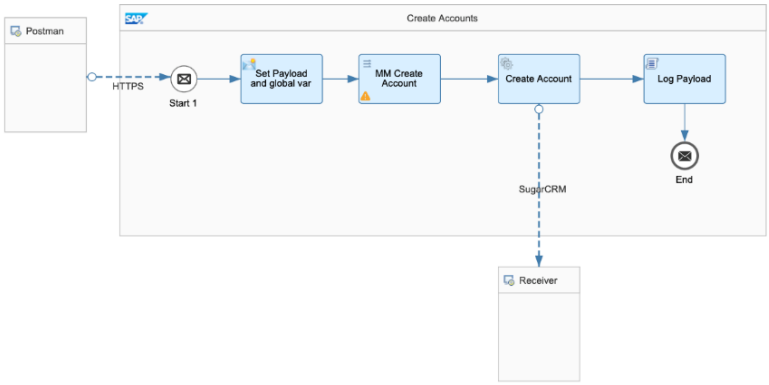

Figure 10.1 SAP CPI Sample Scenario

For simplicity reasons, the integration flow exposes an HTTPS endpoint. This means that the integration flow needs to be triggered externally via an HTTPS call. You can for instance use Postman to test it.

To create the Account, you will need an XSD that contains details about all the fields that describes the structure of an account in SugarCRM.

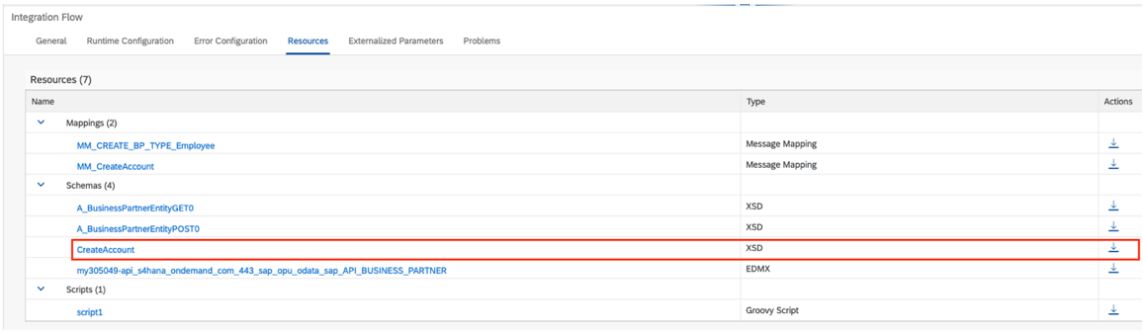

Figure 10.2 SAP CPI CreateAccount XSD

To generate the XSD needed on the SugarCRM side to create an account, we need to use the Eclipse Plug-in, which is delivered along with the Adapter. See Figure 10.3 for an example setup of the plug-in.

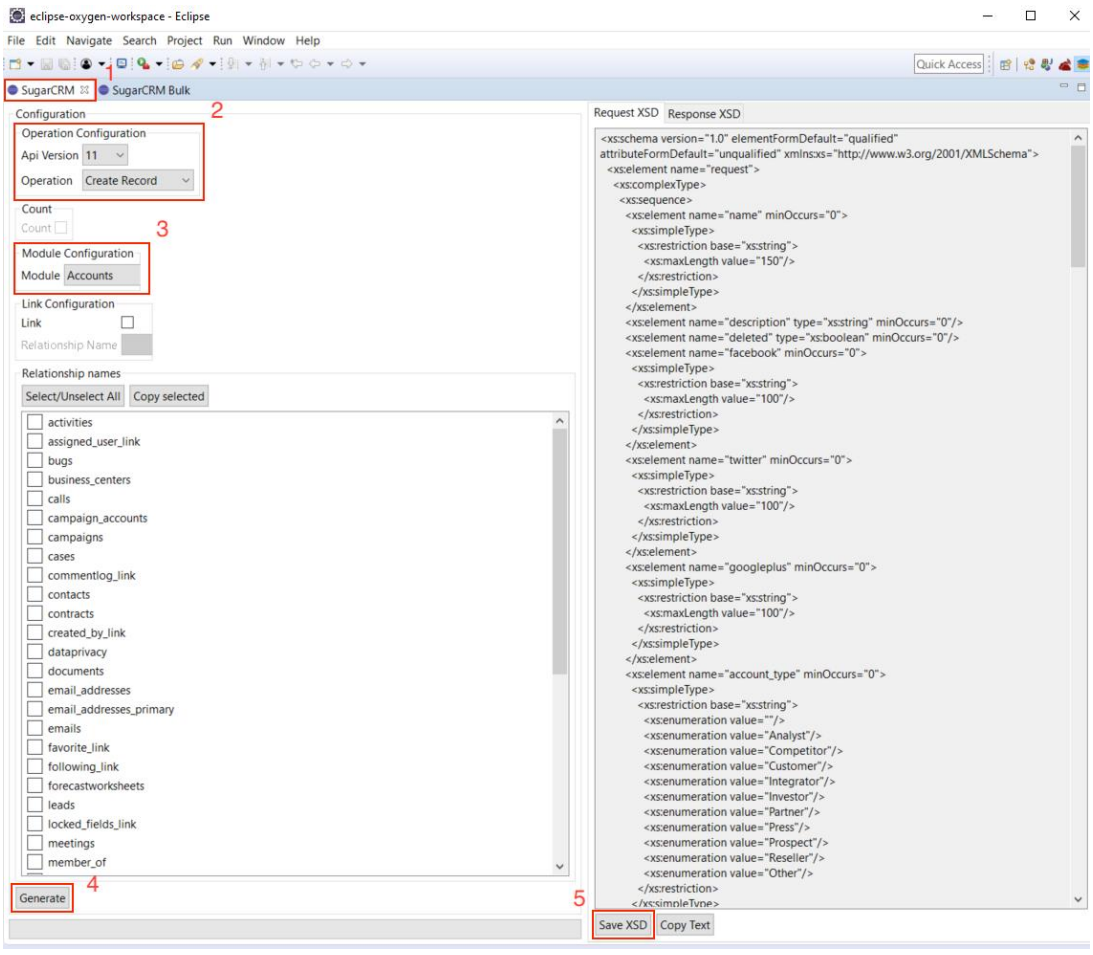

Figure 10.3 SugarCRM plug-in

To create the XSD, the first step is to select the SugarCRM tab (Figure 10.3, step 1). In the second step we select the API version 11 and the Operation Accounts. In the example, as we want all the fields to be in the XSD, we don't select any specific fields - by default all fields will be included. The next step is to click Generate (Figure 10.3, step 4). Then the XSD will be generated and can be saved (Figure 10.3, step 5).

Now we can create a simple 1-to-1 mapping using the XSD we have just created; see Figure 10.4.

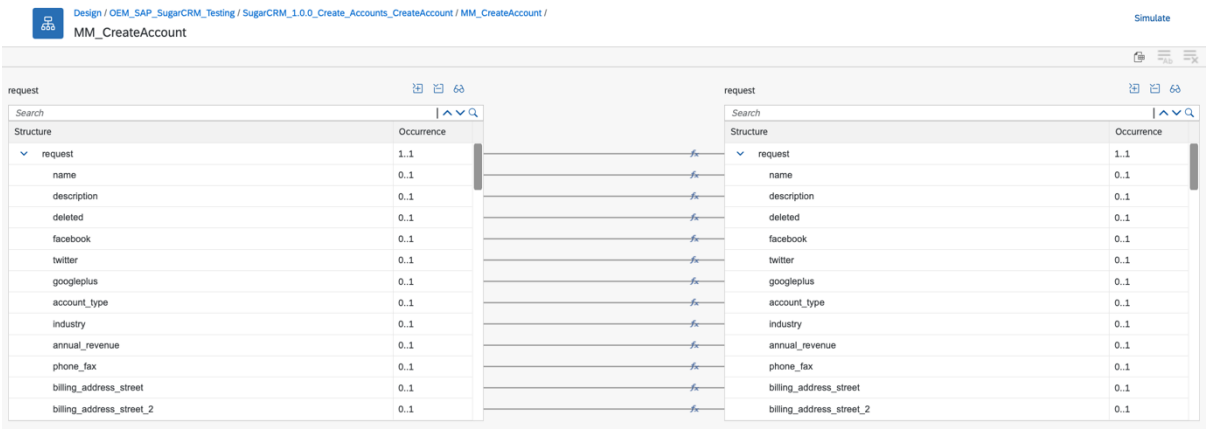

Figure 10.4 SugarCRM plug-in

After the mapping is completed the SugarCRM adapter needs to be configured. See Figure 10.5 for an example.

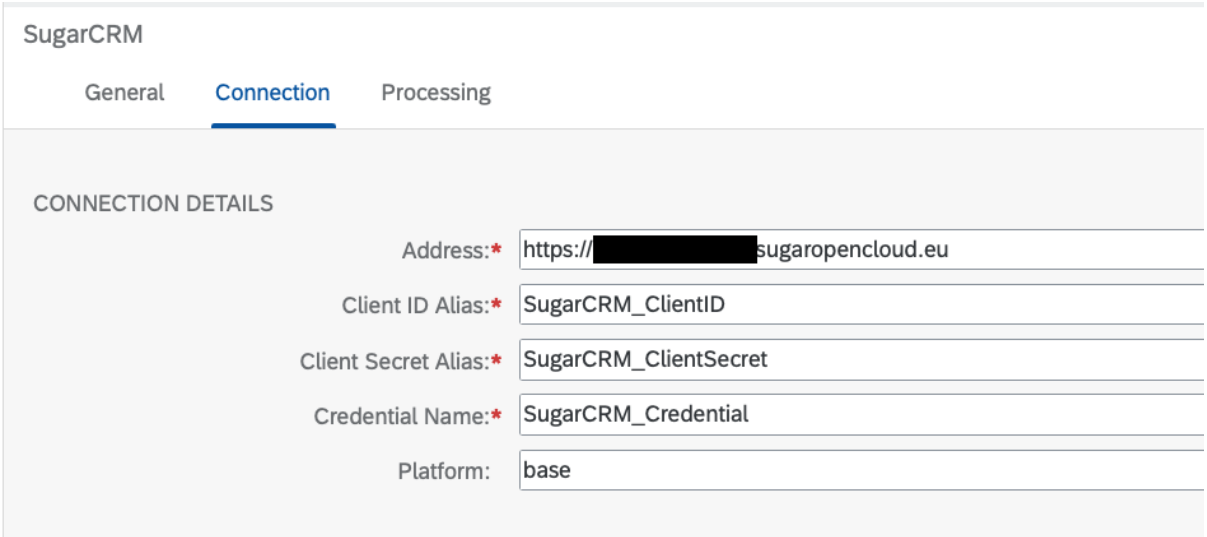

Figure 10.5 SugarCRM Connection details

In the Connection tab shown in Figure 10.5, you need to provide your SugarCRM details. Credentials previously deployed for Client ID Alias, Client Secret Alias and Credential Name are used in the connection details. Refer to Section 7 for further instructions on how to

deploy these credentials. Each field of the Connection tab was previously explained in Section 8.1.

Following the Connection tab, the next step is to configure the Process Tab. An example of the needed configuration to achieve the integration of our example is shown in Figure 10.6.

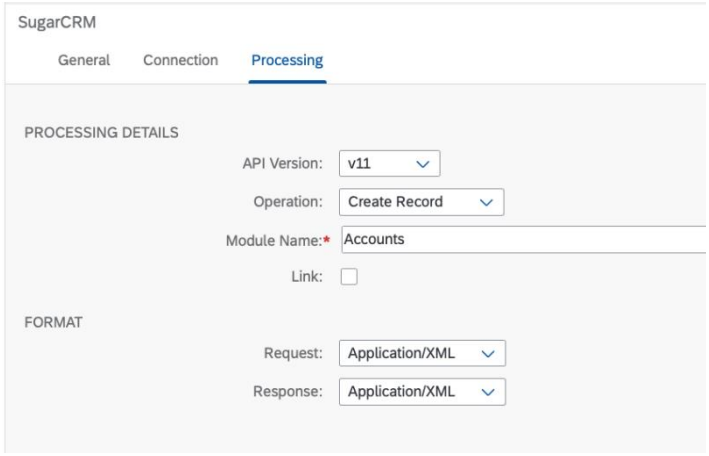

Figure 10.6 SugarCRM Processing details

In the Processing tab, select the API Version (which needs to match the version we used to generate the XSD in the Eclipse Plug-in). In our case this is Version 11. Select the Operation "Create Record" and Module Name "Accounts". For simplicity, we do not use the Link functionality in this example.

Now that all the necessary configurations are made, the integration flow can be deployed and then called from Postman. Figure 10.7 shows an example of the input message.

```
<?xml version="1.0" encoding="utf-8"?>
<request>
    <name>CaixaBank BCN XML</name>
    <description>LaCaixa</description>
    <deleted>false</deleted>
    <facebook>https://www.caixabank.es/particular/home/particulares_es.html</facebook>
    <twitter>@caixabank</twitter>
    <googleplus>string</googleplus>
    <account_type>Customer</account_type>
    <industry>Electronics</industry>
    <annual_revenue>100.000</annual_revenue>
    <phone_fax>00314578671</phone_fax>
    <billing_address_street>Carrer de Balmes</billing_address_street>
    <billing_address_street_2>347</billing_address_street_2>
    <billing_address_street_3>12</billing_address_street_3>
    <billing_address_street_4>b</billing_address_street_4>
    <billing_address_city>Barcelona</billing_address_city>
    <billing_address_postalcode>12345ERT</billing_address_postalcode>
    <billing_address_country>Spain</billing_address_country>
    <rating>1</rating>
    <phone_office>934058520</phone_office>
    <phone_alternate>934058520</phone_alternate>
    <website>example.com</website>
</request>
```

Figure 10.7 SugarCRM create account request

As a result now a new account is created in SugarCRM.

11 References

11.1 Finding SugarCRM Address

To find the address that can be used in the Address field of the Connection Tab, login into your SugarCRM instance and navigate to the Home Page.

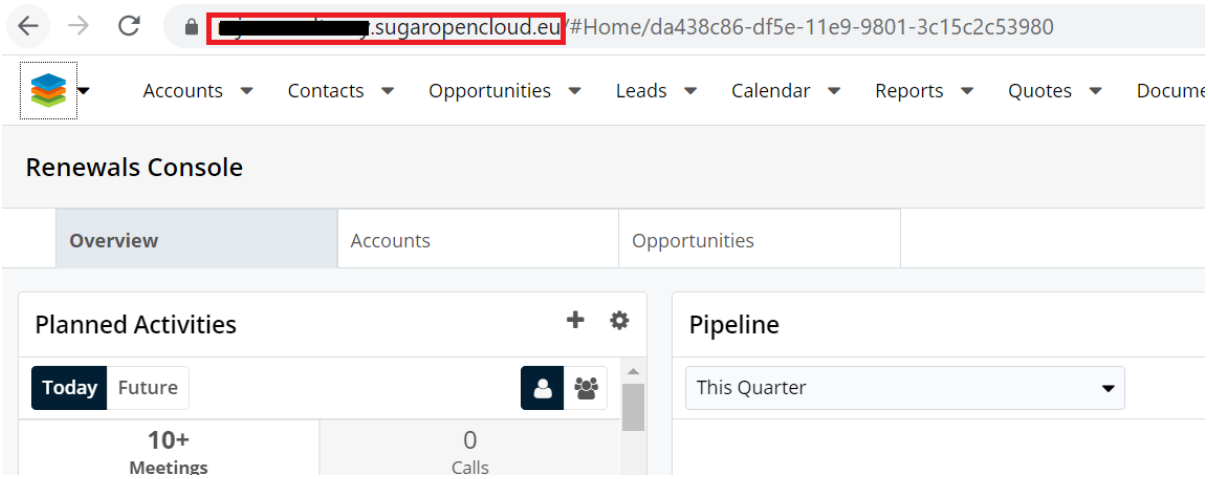

Figure 11.1 SugarCRM address

From the browser URL take the tenant name of your SugarCRM instance as highlighted in the Figure 11.1.

12 Support & Troubleshooting

In case of issues or errors, change the Log level of your integration flow to **Traces**. This can be done in Cloud Integration via the Monitor->Manage Integration Content page. An example of such a configuration can be seen in Figure 12.1.

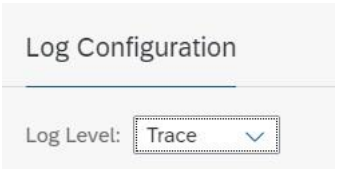

Figure 12.1: Activating Traces

The Trace Log level enables the collection of more traces that can be used to effectively understand the problem. These traces can also be used in a ticket.

The next section discusses few issues that you might encounter and possible solution for them.

12.1 SSLHandshakeException

Error Message:

> *javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException: PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target, cause: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target.*

Possible Solution: This error generally means that in the Keystore of SAP CPI, an entry for the root certificate of Sugar CRM is missing. At the time this document is published, Sugar CRM expects a DigiCert Global Root G2 certificate.

12.2 UnknownHostException

Error Message:

> *java.net.UnknownHostException: xxxxxxx.mysugartenant.com*

Possible Solution: This error generally indicates that the URL used in the Address field of the Connection tab in the adapter is not correct. Correct the value of this field. Refer to section 11.1 which explains how to find your SugarCRM Address.

12.3 No Artifact Descriptor Found

Error Message:

> *[CAMEL][IFLOW][CAUSE]: Cause: com.sap.it.nm.types.NodeManagerException: [CONTENT][CONTENT_DEPLOY][NoArtifactDecriptorFoundForArtifactName]:No artifact descriptor found for artifactName SugarCRM_ClientID*

Possible Solution: This error generally indicates that one of the security artifact used in the Connection tab in the adapter does not exists. Ensure that the artifact used in the connection tab exists as a Security Material.

12.4 Invalid JSON Input Structure

Error Message:

> *org.apache.camel.CamelException: Invalid JSON input structure. A JSONObject text must begin with '{' at 1 [character 2 line 1]*

Possible Solution: This error generally indicates that the adapter is expecting an JSON input, but is getting a non-valid JSON or even another format such as XML.

12.5 Invalid Input XML

Error Message:

> *org.apache.camel.CamelException: Invalid input XML.*
> *org.apache.camel.CamelException: Invalid input XML. The root node "request" is missing in the input.*

Possible Solution: This error generally indicates that the adapter is expecting an XML input but is getting a wrong XML or even another language such as JSON.

12.6 Cannot Produce Target Element

Error Message:

> *com.sap.xi.mapping.camel.XiMappingException:*
> *com.sap.aii.mappingtool.tf7.IllegalInstanceException: Cannot produce target element /request/entry/name. Queue has not enough values in context. Target xsd requires a value for this element, but target field mapping does not produce one. Probably the xml-instance is not valid to the source xsd, or the target field mapping does not fulfill the requirement of the target xsd., cause: com.sap.aii.mappingtool.tf7*

Possible Solution: This error generally indicates that the mapping requires a missing mandatory value in the message body you are sending when creating or updating an instance of an entity in Sugar CRM.

12.7 Non Existing Field

Error Message:

> *org.apache.camel.CamelException:org.apache.camel.CamelException: {"error":"invalid_parameter","error_message":"Non existing field: somefield in module: Accounts"}*

Possible Solution: This error usually indicates that a field does not match a known field found for this entity type. You may have used this field in **Filter** or **Order Response By**.

12.8 Could Not Find a Route with n-Elements

Error Message:

> *{"error": "no_method", "error_message": "Could not find a route with 1 element" }*

Possible Solution: This error indicates that the **Module Name** or the **Relationship Name** are incorrectly spelled.