# Microsoft Dynamics CRM Adapter Guide

For SAP Integration Suite and SAP Cloud Integration

Version 1.1.32 – November 2025

THE BEST RUN SAP

# TABLE OF CONTENTS

## 1. INTRODUCTION

This is the guide for the Microsoft Dynamics CRM Adapter for SAP Integration Suite and SAP Cloud Integration. This guide covers all relevant information for integration developers to start working with the Microsoft Dynamics CRM 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. The correctness and completeness of the Code given herein are not warranted.

### 1.2 Internet Hyperlinks

The documentation may contain hyperlinks to the Internet. These hyperlinks are intended to serve as hints on where to find related information. The availability and correctness of this related information or the ability of this information to serve a particular purpose are not guaranteed.

## 2. MICROSOFT DYNAMICS CRM INTEGRATION

### 2.1 Coding Samples

Dynamics 365 CRM or Microsoft Dynamics CRM is a customer relationship management software. It is a category of integrated, data-driven software solutions that improve how businesses interact and do business with their customers. Microsoft Dynamics CRM covers different customer management systems—such as sales, service, and marketing—and allows them to connect. Refer to the links below to read more about the Microsoft Dynamics CRM:

- https://dynamics.microsoft.com/en-us/crm/what-is-crm/
- https://dynamics.microsoft.com/en-us/crm/crm-software/

### 2.2 Microsoft Dynamics CRM Adapter

The Microsoft Dynamics CRM Receiver Adapter enables an SAP Cloud Integration tenant to accelerate the implementation time and reduce the complexity of connecting to Microsoft Dynamics CRM based on its API.

#### *2.2.1 Features*

From a high-level perspective, the following key features are provided by the Microsoft Dynamics CRM Adapter:

- Support for many versions: The Adapter Integrates with Microsoft Dynamics CRM using a preferred API version. At the moment of publishing this documentation, the Adapter supports up to Dynamics CRM API V9.0, V9.1, and V9.2.

- Support multiple operations, including:

    - Associate Objects

    - Disassociate Objects

    - Create Object

    - Delete Object

    - Get Object by ID

    - Query Objects
    - Update Object
- Support for OData Query
- Support for FetchXML Query
- Processing data in Single or Batch mode
- Ability to expand Navigation Properties
- Support Outbound Headers
- Support Pagination
- Secure authentication with OAuth 2.0: Every REST call between Dynamics CRM and SAP Cloud Integration is secured by the OAuth 2.0 industry-standard protocol for authorization.
- Dynamic configuration with headers and properties: It is possible to set up scenarios as dynamic as wanted. Reference to dynamic values from the header and properties stored in the exchange of the SAP Cloud Integration tenant is possible.
- Support for XML and JSON format: 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 Dynamics CRM.

### 2.3 Architecture Overview

From a technical perspective, the Microsoft Dynamics CRM Adapter can only be used on the receiver side.

This means that SAP Cloud Integration is the initiator of the calls. In case calls toward Microsoft Dynamics CRM need to be scheduled, use the Scheduler steps within the integration flow.

Figure 2.1 shows how the Microsoft Dynamics CRM Adapter can be used in a simple Request-Reply scenario. Various operations can be used in the Adapter – such as Query Data, Delete, Update, Associate, Disassociate or Create an Entity, etc. These operations are further explained in Section 3. Supported Operations and Versions.

To learn more about the different operations the Adapter supports and their configuration, refer to Section 8.2.

Figure 2.1 gives a high-level representation of how the Microsoft Dynamics CRM Adapter works.

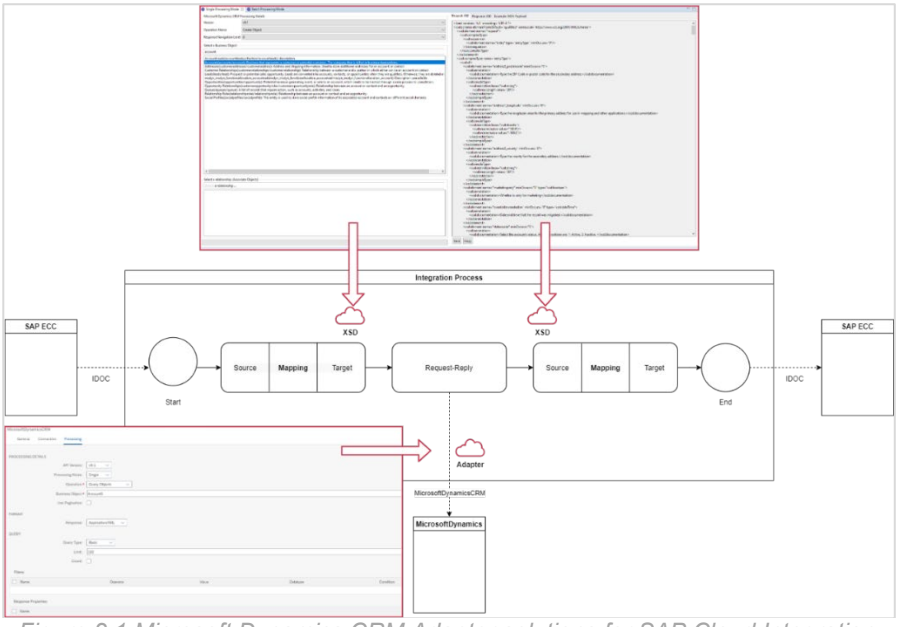



*Figure 2.1 Microsoft Dynamics CRM Adapter solutions for SAP Cloud Integration*

From Figure 2.1, it can be seen how the Request-Reply Step can use the Adapter to connect to and interact with Microsoft Dynamics CRM and invoke its different operations.

The Eclipse Plug-In delivered together with the Adapter helps in generating XSDs of the different entities or business objects in a Microsoft Dynamics CRM tenant. The XSDs generated by the Eclipse Plug-In can be imported in an Integration Flow and used in mappings. For instance, these XSDs can be used in the mapping in front of the Request-Reply Step. This way, the correct XML request message to create or update an entity in Microsoft Dynamics CRM (see the Mapping Step on the left side of the Request-Reply Step) can be created. Similarly, the XSDs can be used to create a mapping to handle the response message sent by Microsoft Dynamics CRM.

Note that the Eclipse Plug-In is not mandatory, it is just a facilitator that makes it easier to perform a mapping.

The Eclipse Plug-In is further explained in Section 6.

## 3. SUPPORTED OPERATIONS AND VERSIONS

Currently, the API versions 9.0,9.1, and 9.2 are supported by the Adapter. It supports two processing modes: Single and Batch.

The Single processing mode handles one record at a time. For example, with the Create Object operation, only one account is created in Microsoft Dynamics CRM each time the Adapter runs.

However, the Batch processing mode can process multiple records in a single request. For example, one hundred (100) account records in Microsoft Dynamics CRM can be created in one request. With a Batch request, several operations can also be performed at the same time.

The Single and Batch processing modes have operations that are associated with each.

Table 1 lists the different operations that are supported by each processing mode.

| Processing Mode | Operation | Description |
|---|---|---|
| Single | Associate objects | This creates an association and link between two entities. |
| | Disassociate objects | This removes an association and link between two entities. |
| | Create object | This creates an entity record(s) for different entities. |
| | Delete object | This deletes existing entity record(s). |
| | Get object by ID | This retrieves one record of an object by the ID of the record. |
| | Query objects | This retrieves data based on a Microsoft Dynamics CRM OData query. |
| | Update object | This updates existing entity record(s). |
| Batch | Associate objects | This creates an association and link between two entities. |
| | Disassociate objects | This removes an association and link between two entities. |
| | Create object | This creates an entity record(s) for different entities. |
| | Delete object | This deletes existing entity record(s). |
| | Query objects | This retrieves data based on a Microsoft Dynamics CRM OData query. |
| | Update object | This updates existing entity record(s). |

*Table 1 Description of the Supported Operations by Processing Mode*

For a detailed explanation of each of the above operations and how to configure each, refer to Section 8.2.

## 4. ADAPTER INSTALLATION

Refer to the Microsoft Dynamics CRM Adapter and Plug-In Installation Guide document included in the package for detailed steps on installing the Microsoft Dynamics CRM Adapter installation.

## 5. ECLIPSE PLUG-IN INSTALLATION

Refer to the Microsoft Dynamics CRM Adapter and Plug-In Installation Guide document included in the package for detailed steps on installing the Eclipse Plug-In installation.

## 6. USING THE ECLIPSE PLUG-IN

The Eclipse Plug-In or Workbench can be used to extract and generate XSDs of different Microsoft Dynamics CRM entities. These XSDs can then be used in mappings for the request or response messages in integration scenarios.

The next sections explain how to use the Eclipse Plug-In.

### 6.1 Configuration of Microsoft Dynamics Tenant Login Information

The Plug-In generates XSDs according to the metadata of the business objects or entities that are part of a Microsoft Dynamics CRM tenant.

To retrieve the metadata, it is required 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.

Navigate to the Microsoft Dynamics CRM Adapter Plug-In on the left side of the Preference dialogue box.

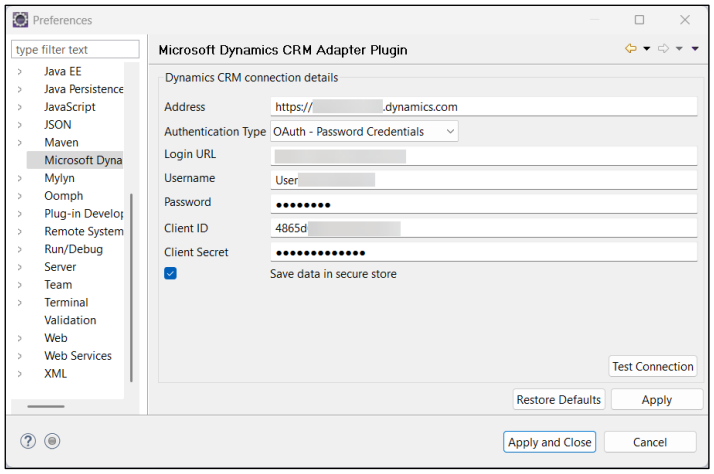



*Figure 6.1 Configure the Connection Details of the Plug-In*

The **Address** field refers to the address of the target Dynamics CRM tenant.

Example: https://abc123.api.crm4.dynamics.com.

Select an **Authentication Type** from the dropdown box. Possible values include OAuth - Password Credentials and OAuth - Client Credential Secret.

- **OAuth – Password Credentials**: This approach requires a **Login URL**, **Username**/**Password** pair (Credential Name), a **Client ID**, and an optional **Client Secret**.

- **OAuth – Client Credentials Secret**: This approach does not require a username/password pair (Credential Name). It only needs a **Login URL**, **Directory (tenant) ID**, **Client ID**, and an optional **Client Secret**. This Authentication type can also be used in case the account used in the Connection tab requires federated authentication.

If OAuth - Password Credentials is selected as the authentication type, a **Client ID** must be supplied.

Please follow the guide in Section 13.2 for extracting the **Client ID** configured in Azure.

Next, a **Username** and **Password** should be provided.

Note that these four fields correspond to the same information required in the Adapter's Connection tab (See Section 8.1).

As seen in Figure 6.1, there is a Save data in secure store checkbox that persist the authentication details on the secure storage of the local machine.

The Test Connection button validates whether the Microsoft Dynamics CRM tenant can be accessed using the provided login data.

It is important to have a successful test connection before processing to generate XSDs.

### 6.2 Generate XSD

As discussed in Section 3, the Microsoft Dynamics CRM Adapter Plug-In is divided into two processing modes: Single and Batch.

The Single Processing Mode allows the creation of schemas for single operations while Batch Processing Mode allows multiple operations per schema.

Each processing mode will be explained in the next sections.

#### *6.2.1 Single Processing Mode*

On the top left side of the Single Processing Mode view, select values for the fields: **Version**, **Operation Name**, and **Response Navigation Level**. Refer to Figure 6.2.

Note that the values specified for the version and operation name need to match the values specified in the Adapter.

Versions 9.0, 9.1, and 9.2 are currently supported along with the following operations:

- Query
- Get (by ID)
- Create
- Update
- Delete
- Associate
- Disassociate

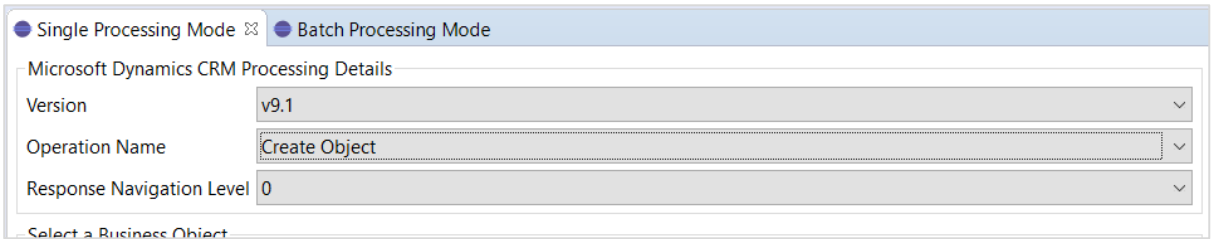



*Figure 6.2 Microsoft Dynamics CRM Processing Details*

The Response Navigation Level specifies how nested the response XSDs need to be. Possible options are 0 and 1.

For illustration purposes, it is assumed that the entity Account is linked to the Address entity.

In case the Response Navigation Level is specified as 0, only the fields or attributes of the concerned entity will be included in the XSD. In the example, it implies that only the property of the Account entity will be included. Selecting 1 will generate an XSD that includes properties of both the Account and the Address entities.

Choosing 1 is relevant if the OData expand feature is to be used.

Explore the functionality of the Plug-In by selecting a version and an operation that fits the needs of a scenario.

Figure 6.3 displays a search box for filtering the entities or business objects by name. Select the desired Business Object to access the XSDs.

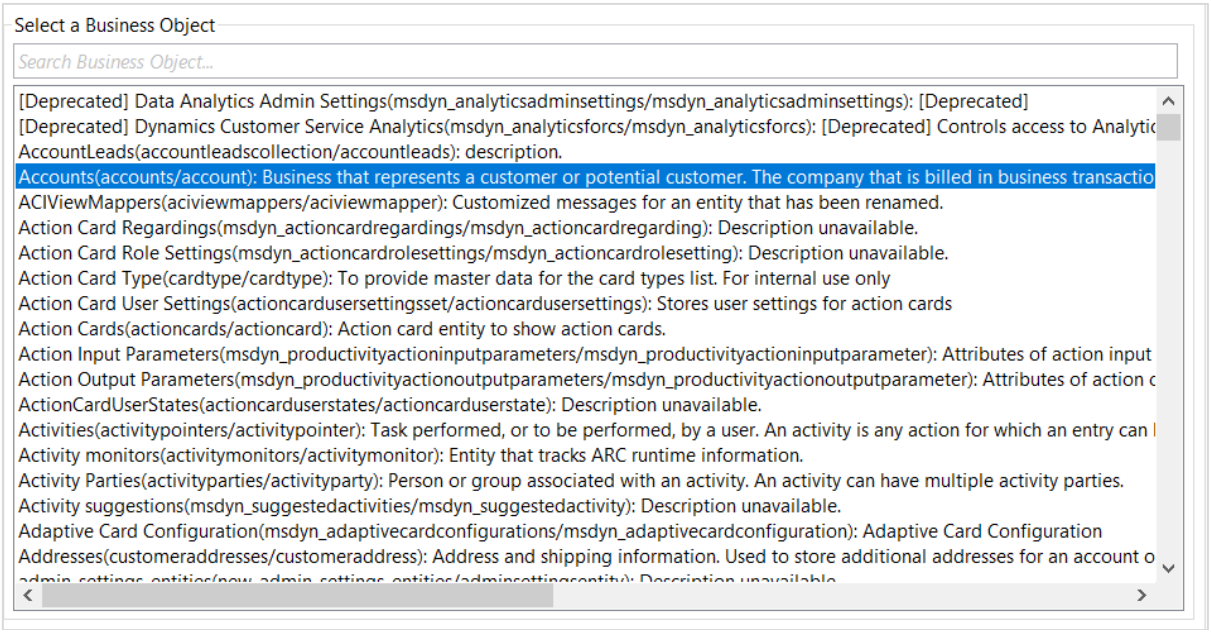



*Figure 6.3 Select a Business Object*

**Note** that when selecting an entity, it may take time to retrieve the relevant entities; related properties; and generate the XSDs.

On the bottom left side of the screen is the Select a Relationship search box (Figure 6.4).

**Note** that this feature is only available for Associate and Disassociate operations. By selecting an associated object, the XSD is enhanced with the necessary information to associate the entities that were previously selected.

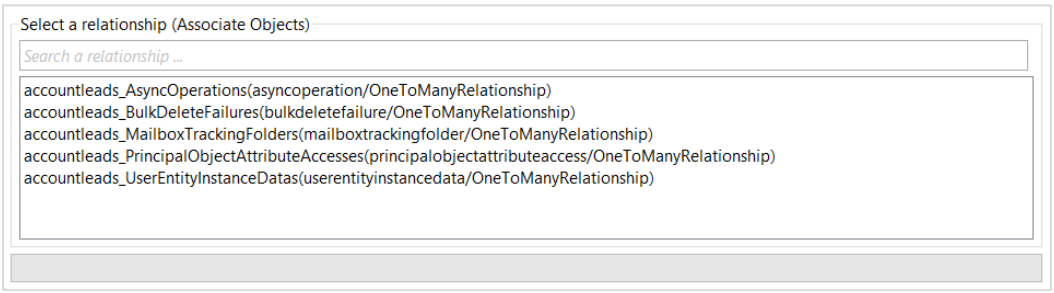



*Figure 6.4 Select a Relationship Search Box*

Three tabs are available on the right side of the screen: **Request XSD**, **Response XSD**, and **Example JSON Payload.**

The Request XSD tab (Figure 6.5) contains the XSD that represents the properties of the selected entity in Microsoft Dynamics CRM.

Note that this tab is only filled in for the Create, Update, Associate, and Disassociate operations.

This XSD can be copied or saved on the local machine and can later be used in a mapping of an integration flow.

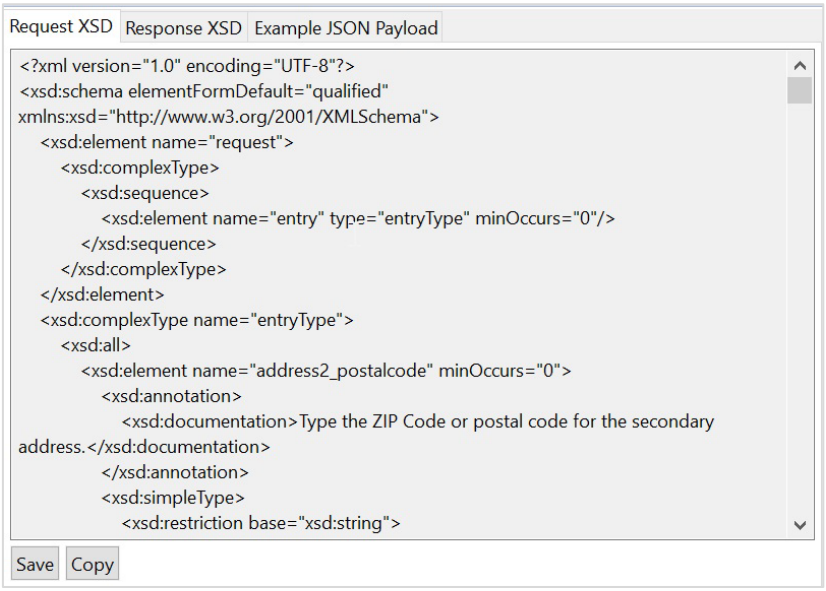



*Figure 6.5 Example Request XSD*

The Response XSD tab contains the XSD that represents the properties of the selected entity. This XSD only includes properties that are returned in the response to the specified operation. If the Create operation is used, the primary key property might not be present in the input (Request XSD) but it will be present in the Response XSD, as shown in Figure 6.6.

In contrast to the request XSD tab, the Response XSD tab is filled for all the operations.

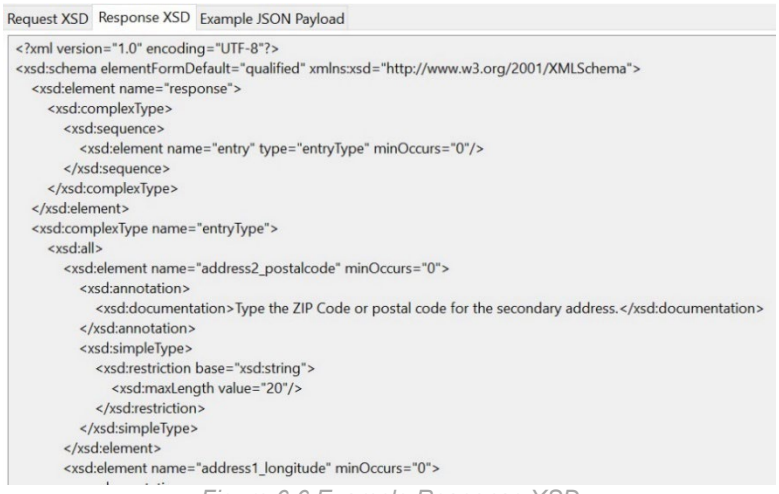



*Figure 6.6 Example Response XSD*

If using JSON in the Adapter instead of XSD is preferred, the Eclipse Plug-In can generate an example JSON Payload using the Example JSON Payload tab. Refer to Figure 6.7.

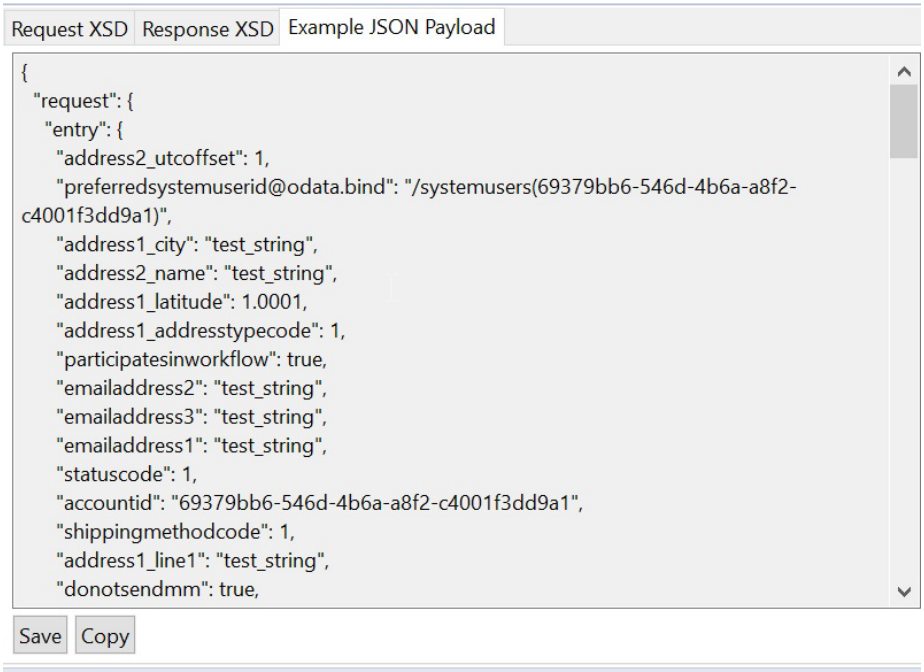



*Figure 6.7 Example JSON Payload*

The generated XSDs/ Example JSON Payload can be copied or saved to the local computer using the Save and Copy buttons. These buttons can be seen at the bottom of the panel.

### *6.2.2 Batch Processing Mode*

The **Batch Processing Mode** view (Figure 6.8) includes a table that allows the addition of more entities to the XSD. In addition to the entities, an operation for each entity must be selected. It should also be indicated whether the request should return a response or not.

To populate the table, click on the **Add** button. New lines can be added to the table by clicking on the **Add** button again. The object and operation can be selected from a dropdown box. Put a tick on the response box to indicate that a response is to be returned for the process.

**Note:** The Delete operation does not have any return properties in its response.

When required entities are added, tick the respective boxes and proceed to click on the Generate button. This creates the XSD schema.

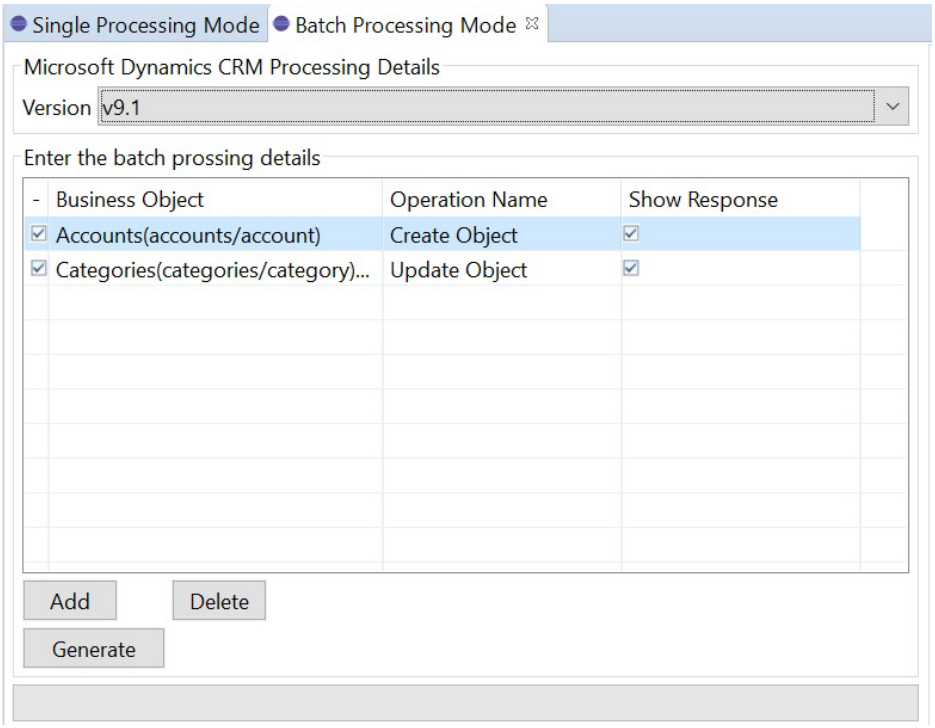



*Figure 6.8 Batch Processing Mode Options*

After clicking the Generate button, the generated XSDs can be copied from the right side of the Plug-In.

### 6.3 Query Formatter

The Microsoft Dynamics CRM Adapter supports three Query types: **Basic**, **Advanced**, and **FetchXML**.

The **Advanced** (or OData) and **FetchXML** query types require inputting a query string. Ensure that the rules in formatting a FetchXML query and the OData Query in the Adapter are properly observed.

The Query Formatter feature assists in ensuring that the format expected by the Adapter is followed.

Figure 6.9 shows an example of using the Query Formatter in the Eclipse Plug-In to format FecthXML Query. Also shown in Figure 6.9, a query placed on the left panel is displayed as the formatted version on the right panel.

Note that all spaces are removed.

Copy and add the formatted version to the Adapter.

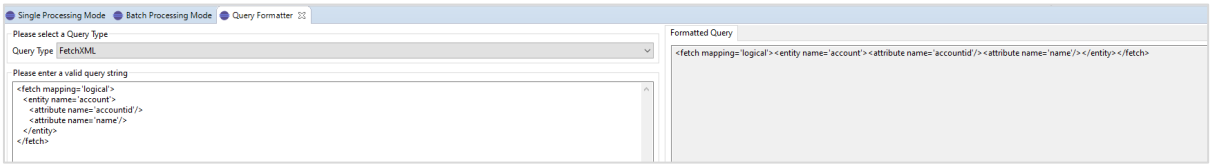



*Figure 6.9 Generate FetchXML with Plug-In*

Figure 6.10 shows an example of using the Query Formatter in the Eclipse Plug-In to format Advanced or OData Query. Input the query on the left panel and the formatted version is displayed on the right panel.

Note that all the ampersands (&) in the queries are shown as “_and_”.

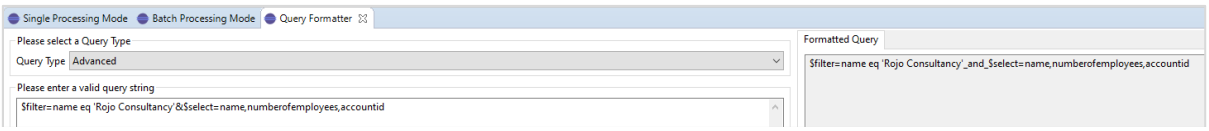



*Figure 6.10 Generate OData Query with Plug-In*

## 7. CREATING CREDENTIALS AND CLIENT ID AS SECURITY MATERIAL

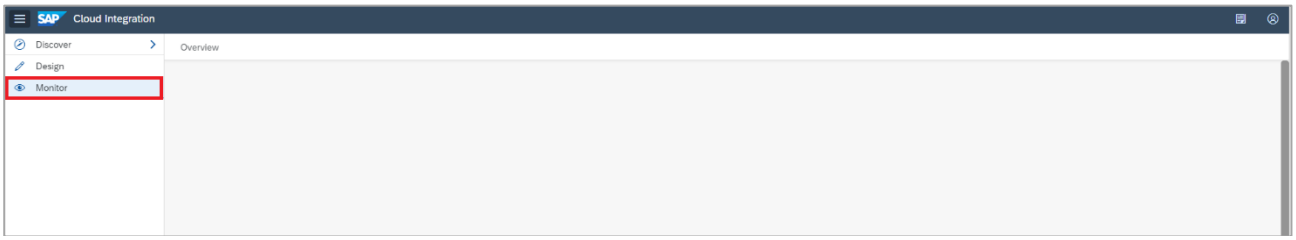
The Microsoft Dynamics CRM Adapter works with standard security artifacts such as Secure Parameter and User Credentials. These artifacts are stored in the SAP secure store. This way, username-password combinations, and secrets can be accessed safely via aliases configured in the Adapter.

The next sections give details on how to create User Credentials and Secure Parameter artifacts.

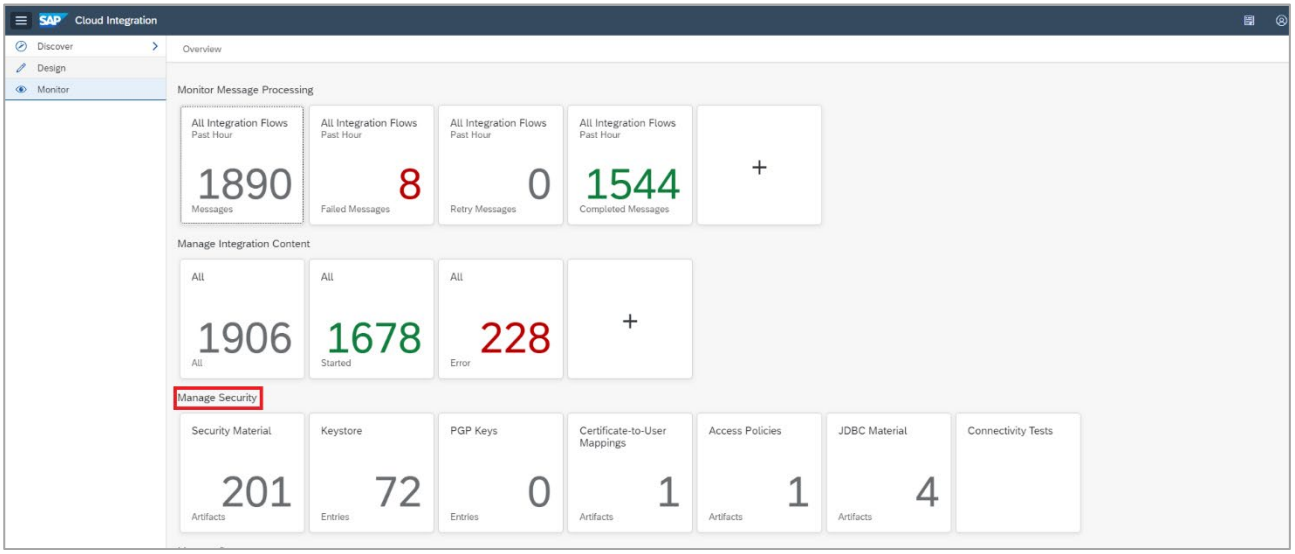
### 7.1 Create User Credential

To create a User Credential as Security Material for the Microsoft Dynamics CRM Adapter, proceed with the following steps:

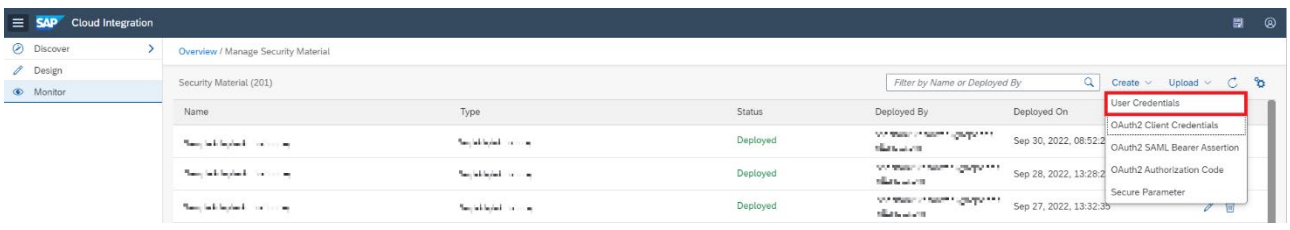
1. In SAP Cloud Integration, navigate to the **Monitor** View.



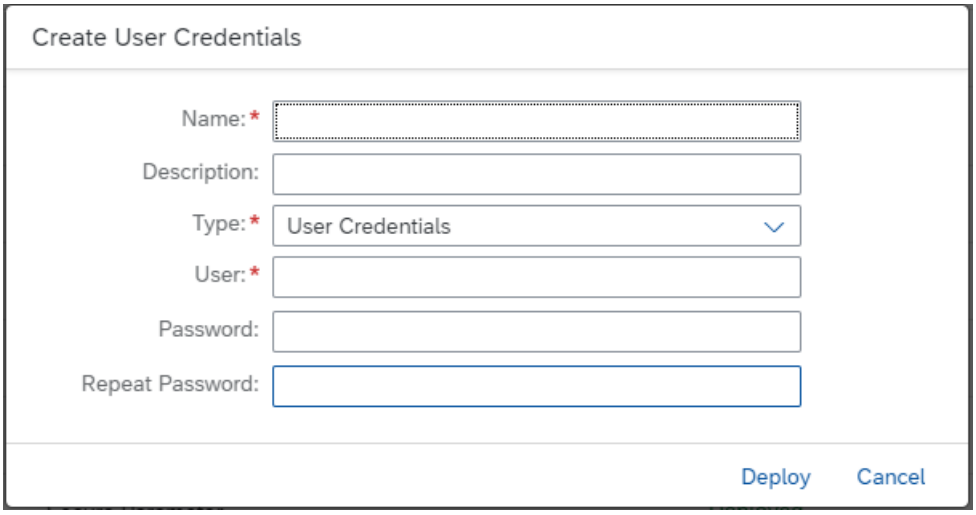

2. Select **Security Material** under the **Manage Security** section.




3. Click on the **Create** dropdown button at the top right and select **User Credential.**



4. Provide a **Name** to the Security Material and ensure that **User Credentials** is selected in the Type dropdown.



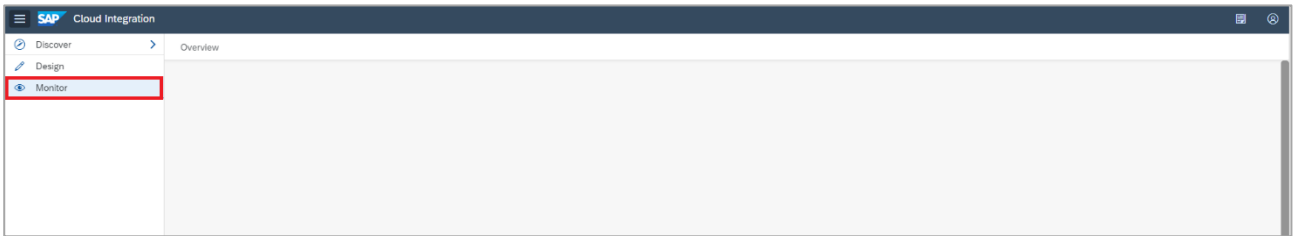

5. Supply the Microsoft Dynamics CRM **User** and **Password**.

6. Click on **Deploy**.

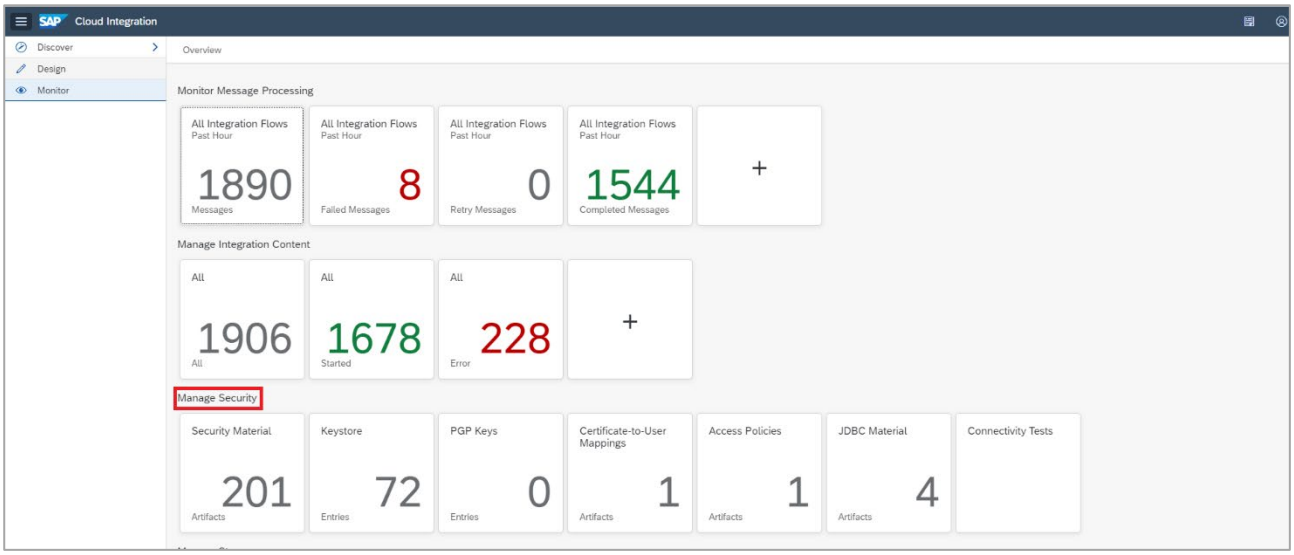
### 7.2 Create Client ID

To create a Client ID as Security Material for the Microsoft Dynamics CRM Adapter, proceed with the following steps:

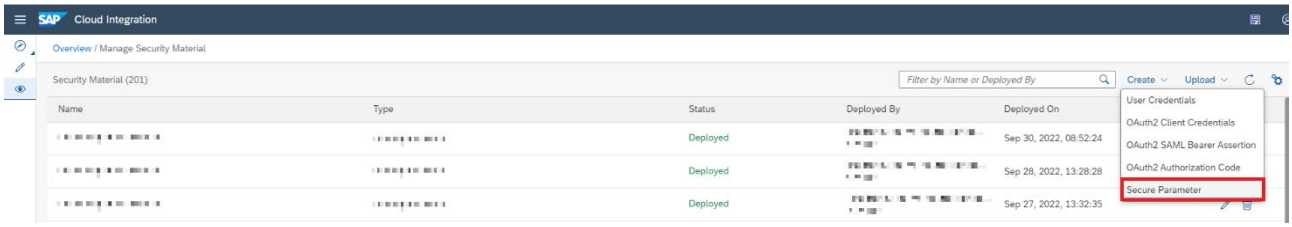
1. In SAP Cloud Integration, navigate to the **Monitor** View.



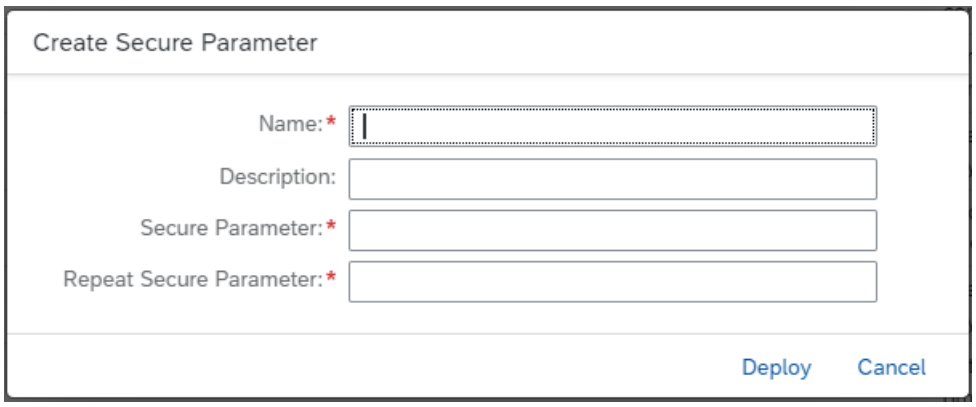

2. Select **Security Material** under the **Manage Security** section.



3. Click on the **Create** dropdown at the top right and select **Secure Parameter**.




4. Provide a **Name** to the Security Material and enter a **Description**.




5. Supply the Microsoft Dynamics CRM Client ID in the **Secure Parameter** and **Repeat Secure Parameter** fields.

   Refer to Section 13.2 on how to create a Microsoft Dynamics CRM Client ID.

6. Click on **Deploy**.

## 8. ADAPTER CONFIGURATION

This section describes the different parameters that can be configured for the Microsoft Dynamics CRM Adapter.

### 8.1 Connection Tab

The Connection Tab contains parameters that define how the Adapter connects to and authenticates against Microsoft Dynamics CRM.

Note that there are two different connectivity approaches:

- OAuth – Password Credentials
- OAuth – Client Credentials Secret

See examples of both approaches in Figure 8.1 for OAuth – Password Credentials and Figure 8.2 for OAuth – Client Credentials Secret.

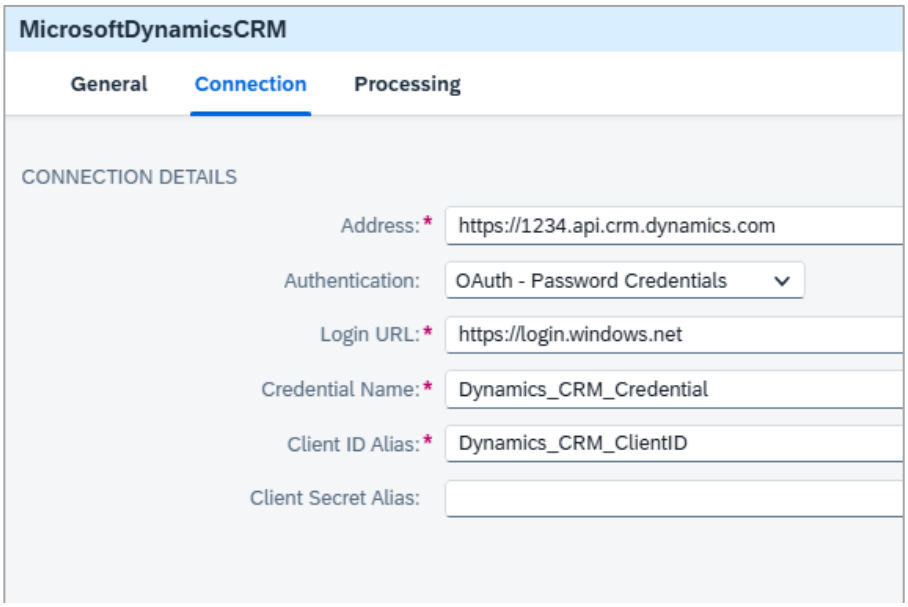



*Figure 8.1 Connection Tab, OAuth - Password Credentials*

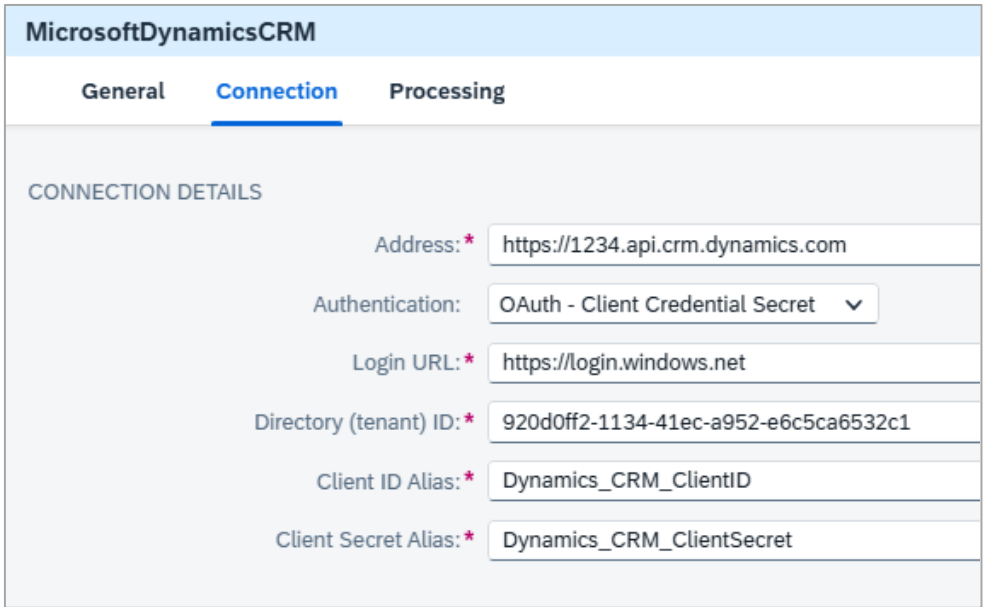



*Figure 8.2 Connection Tab, OAuth - Client Credential Secret*

The details of each field in the Connection Tab are explained in Table 2**.**

| Tab | Parameter Name | Description |
|---|---|---|
| Connection | Address | Specify the recipient's endpoint URL. This URL connects to the Microsoft Dynamics CRM tenant. The value of this field can also be read dynamically. To learn how to find this address, see 13.1 Finding Dynamics CRM Address.<br>Example: https://org1234.crm4.dynamics.com |
| | Authentication | Select the Authentication type to be used. Possible options include:<br>• OAuth – Password Credentials: This approach requires a username/password pair (Credential Name), a Client ID, and an optional Client Secret. |

| Tab | Parameter Name | Description |
|---|---|---|
| | | • OAuth – Client Credentials Secret: This approach does not require a username/password pair (Credential Name). It only needs a Directory (tenant) ID, Client ID, and an optional Client Secret. This Authentication type can also be used in case the account used in the Connection Tab requires Federated authentication. |
| | Login URL | Specify the login URL.<br>Example: https://login.windows.net |
| | Directory (tenant) ID | Specify the ID of the Directory or Tenant. |
| | Credential Name | Specify the credential name used to authenticate against the server. This represents the Microsoft Dynamics CRM credential name (username-password) pair stored as Security Material. Refer to Section 7.<br>Note that the user-provided material here needs to have the correct authorization in Microsoft Dynamics CRM. |
| | Client ID Alias | Specify the Client ID Alias created as a Security Material in Cloud Integration. Refer to Section 7.<br>Note that it is also possible to specify this value using property or header. |
| | Client Secret Alias | Specify the Client Secret Alias created as a Security Material in Cloud Integration. Refer to Section 7.<br>Note that it is also possible to specify this value using property or header. |

*Table 2 Connection Tab Description*

Note that the Adapter validates server certificates when connecting to Microsoft Dynamics CRM. If the root certificates of Microsoft Dynamics CRM are not present in the SAP Cloud Integration Keystore, an error will be returned.

### 8.2 Processing Tab

The type of operation and details of the request to be performed must be indicated in the Processing Tab.

All operations and requests require that the version of Microsoft Dynamics CRM is selected first. The version can be found in the Service Root URL section in the Settings and Customization of the Microsoft Dynamic CRM tenant.

Note: Currently, versions 9.0, 9.1, and 9.2 are supported by the Adapter.

In the Processing Mode, specify whether to run a batch or a single request.

The Single Processing mode, discussed in Section 6.2.1, supports the operations listed in Section 3.

Details on the Batch Processing mode are given in Section 6.2.2.

With a Batch request, operations can be combined in one request.

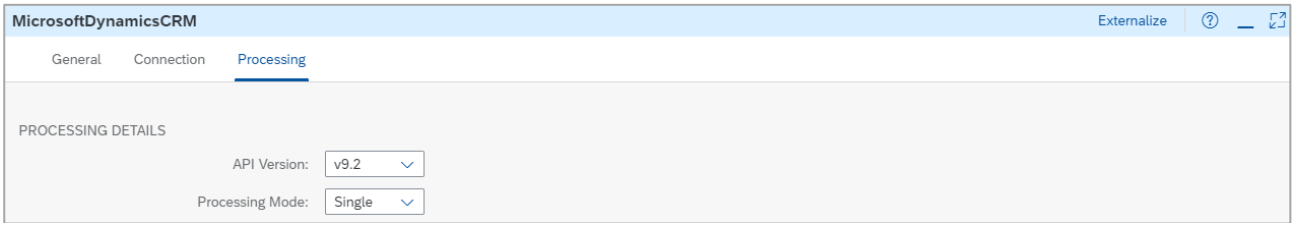



*Figure 8.3 Processing Tab, Microsoft Dynamics CRM Adapter*

### *8.2.1 Single Processing Mode*

Each one of the operations that are listed in Section 3 is discussed in the next sections.

#### *8.2.1.1 Query Types and Query Options*

Before configuring the different operations for a single request processing mode, it is important to understand the query types supported by the Adapter. Here follows the query types:

- Basic
- Advanced
- FetchXML

##### *8.2.1.1.1 Basic*

A **Basic** query is the simplest query type. The list of properties and feature options available for a basic query can be seen in Table 3.

**Note**: Not all query options are available for every operation. To visualize the fields that are listed in Table 3, refer to Figure 8.4.

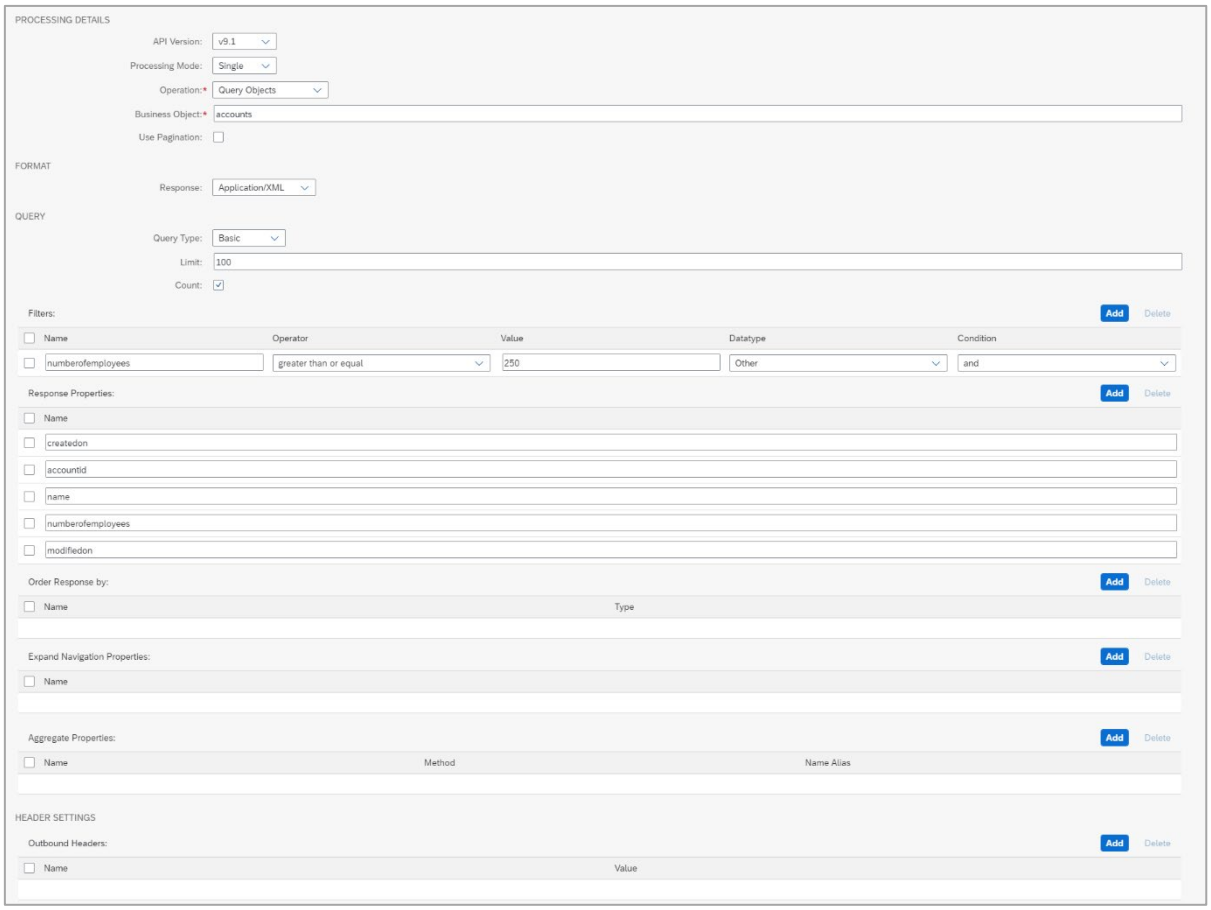



*Figure 8.4 Sample Query Type: Basic*

| Query Options | Field | Description | Value |
|---|---|---|---|
| Filters | Specify how to add filter conditions to determine which type of records should be returned by the server. | | |
| | Name | Name of the property of a Dynamics CRM entity (Business Object). | numberofemployees |
| | Operator | The operation to be used for the filter. It is possible to specify values such as equal, not equals, greater than, contains, etc. | greater than or equal |
| | Value | The value of the property is mentioned in the name field. | 250 |
| | Datatype | The data type can be String or Other depending on the type of data specified in the Name field. For example, if the data type is a date, then select Other. | Other |
| | Condition | Condition is a logical operator; it can be AND or OR. Note that AND is the default value. | AND |
| Response Properties | Specify the fields to be returned in the response. This is a way to limit which properties of the response should be returned. Each property can be added in a separate row of the table. Note that in case no properties are added to this property, all fields of the concerned entity will be returned. | | |
| | Name | Name of the property to be returned. | • createdon<br>• accountid<br>• name<br>• numberofemployees<br>• modifiedon |
| Order Response By | Specify in which order the records in the response should be returned. Note that this option is not always available if used with the Aggregate Properties option. | | |
| Order Response By | Name | Specify the name of the property to be used for ordering the records to be returned in the response. | createdon |
| | Type | Specify whether the response needs to be performed in ascending or descending order. | ascending |
| | Specify the properties to be included and expanded in the response. Note that aggregation is not supported while doing an expansion. | | |
| Expand Navigation Properties | Name | • Name of the property to be expanded. The expand feature enables the accessing of a nested entity.<br>Note that the name of the associated entity from the Eclipse Plug-In can be identified. When generating the XSD for an entity that needs to be extended, set the “Response Navigation level” to 1.<br>This way, the response XSD will include all nodes of the related entities.<br>• Alternatively, the “Associate Objects” operation can be selected, and see the list of names in the relationship section. | opportunity_customer_accounts |
| Aggregate Properties | Specify how to aggregate and group the response data. Note that aggregation is not supported while doing any other query options. | | |

| Query Options | Field | Description | Value |
|---|---|---|---|
| | Name | Name of the property to be used for aggregation. | versionnumber |
| | Method | Methods to choose from: max, min, sum, count (*), count distinct, and average. | max |
| Header settings: Outbound Headers | These are headers that will be sent to Microsoft Dynamics CRM while performing the request. More details about supported outbound headers can be found in Section 8.2.3. | | |
| | Name | Specify the name of the header. | prefer |
| | Value | Specify the value of the header. | Odata.maxpagesize=20 |

*Table 3 Description of Basic Query Options*

*8.2.1.1.2 Advanced*

Experienced users also can perform complex queries toward Microsoft Dynamics CRM using the **Advanced Query** feature.

With the Advanced Query, complex queries following the OData query options can be written.

Refer to Chapter 5 of the oasis-open org to learn more about the OData URL conventions.

Follow this link: http://docs.oasis-open.org/odata/odata/v4.0/errata03/os/complete/part2-url-conventions/odata-v4.0-errata03-os-part2-url-conventions-complete.html#_Toc453752360

**Note**: When writing queries, all occurrences of the ampersand (&) character should be replaced with "_and_". Alternatively, the query can be surrounded by the RAW().

See Figure 8.5 for an example of querying an entity using the Advanced Query type.

The Eclipse Plug-In includes a feature that assists in converting an OData Query to a format accepted by the Microsoft Dynamics Adapter.

Refer to Section 6.3 for further guidance.

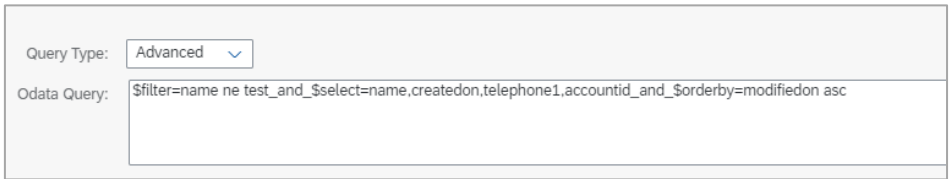



*Figure 8.5 Sample Query Type: Advanced*

In standard OData, a query is written as:

*$filter=name ne test&$select=name,createdon,telephone1,accountid&$orderby=modifiedon asc*

The equivalent query in the Adapter looks like the following:

*$filter=name ne test_and_$select=name,createdon,telephone1,accounted*
*_and_$orderby=modifiedon asc*

As the example demonstrates, the **&** character was expressed as **_and_**.

It is also possible to surround the query with RAW(), as seen below.

*RAW($filter=name ne test_and_$select=name,createdon,telephone1,accounted &$orderby=modifiedon asc)*

Table 4 explains the query options used in the example seen in Figure 8.5.

| Query Option | Description |
|---|---|
| $filter | Retrieve all accounts with a name not equal to “test”. |
| $select | The properties returned in the response. For example, name, createdon, telephone1 and accounted. |
| $orderby | Order the records in the response message according to the property modifiedon. Furthermore, the manner the records are ordered needs to be in ascending order. |

*Table 4 Description of Advanced Query Options*

Note that more query options are supported. For more details on supported queries in Microsoft Dynamics CRM, consult the Microsoft Dynamics Help.

*8.2.1.1.3 FetchXML*

The Adapter also supports the different FetchXML queries available in Dynamics CRM. Similar to the Advanced OData query, the FetchXML can be used to query data from Dynamics CRM.

An example of a FetchXML query is presented below.

```
<fetch mapping='logical'><entity name='opportunity'><attribute
name='estimatedclosedate'/><attribute name='estimatedvalue'/><attribute
name='budgetamount_base'/><attribute name='name'/><attribute name='opportunityid'/><filter
type='and'><condition attribute='name' operator='ne'
value='greencar'/></filter></entity></fetch>
```

*Listing 8.1 Example of FetchXML Query Type*

**Note**: The FetchXML query in the Adapter is formatted differently. The text of the FetchXML cannot contain spaces. The Eclipse Plug-In has a feature that converts a FetchXML query to a format accepted by the Microsoft Dynamics Adapter. Refer to Section 6.3.

The FetchXML query shown in Listing 8.1 filters opportunities that have a name not equal to “greencar”. In this query, the following properties are also requested to be returned in the response:

- estimatedclosedate
- estimatedvalue
- budgetamount_base
- name
- opportunityid

| Query Option | Description |
|---|---|
| <fetch mapping='logical'> | Represents the root of the FetchXML document. |
| <entity name='opportunity'> | Specifies an entity, in this case: opportunity. |
| <attribute name='estimatedclosedate'/> | Selecting a property: estimatedclosedate. |
| <attribute name='estimatedvalue'/> | Selecting a property: estimatedvalue. |
| <attribute name='budgetamount_base'/> | Selecting a property: budgetamount_base. |
| <attribute name='name'/> | Selecting a property: name. |
| <attribute name='opportunityid'/> | Selecting a property: opportunityid. |
| <filter type='and'> | Adding a filter. After selecting properties, filtering queries are also possible. Connect these query options by <filter type='and'>. |
| <condition attribute='name' operator='ne' value='opportunity100'/> | Adding a new condition to the filter. A filter for the name property. The name should not be equal to “greencar”. |

*Table 5 Description of FetchXML Query Options*

More information on FetchXML queries can be accessed here.

Each operation available in the Adapter is explained in the next sections.

### *8.2.1.2 Get Object by ID*

The operation Get Object by ID retrieves information and properties of a specific entity or a business object. The operation requires the unique ID of the concerned entity.

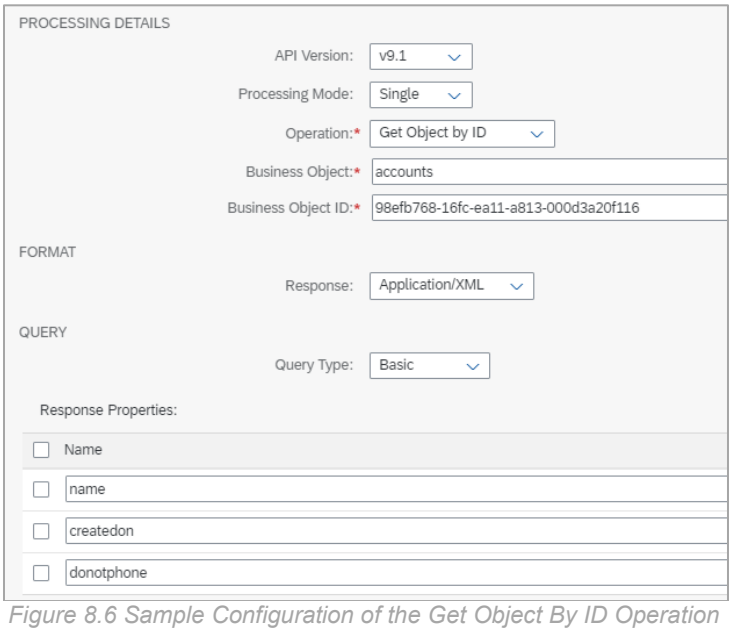



*Figure 8.6 Sample Configuration of the Get Object By ID Operation*

As depicted in Figure 8.6, the configurations for this operation include the following fields:

- **Business Object**: This needs to be filled in with the name of the entity to be retrieved. Examples: accounts, invoices, opportunities, etc. To see the full list of possible entities, refer to the Eclipse Plug-In discussed in Section 6.2.1.

- **Business Object ID**: The unique ID that represents the instance of the entity specified in the Business Object field.
- **Response**: This field specifies the format of the response message to be returned by Dynamics CRM. Possible values include Application/XML or Application/JSON.
  Note that the default value is Application/XML.
- **Query Type**:  Possible values include Basic, Advanced, and FetchXML.

Furthermore, the following query options are available: **Response Properties**, **Expand Navigation Properties**, and **Outbound Headers**.

These query options are discussed in Table 3.

For illustration purposes, the example in Figure 8.6 retrieves an account from Microsoft Dynamics CRM with the ID “98efb768-16fc-ea11-a813-000d3a20f116”. In this case, only certain properties in XML format are requested to be returned; name, createdon, and donotphone.

Figure 8.6 uses the Basic Query type. The same result can be achieved using the Advanced Query with the statement *$select=name,createdon,donotphone.*

Note that most of the fields listed above accept dynamic values using header and properties.

For example, ${header.header1} or ${property.property1}.

For more details on Advanced Query, refer to Section 8.2.1.1.2

*8.2.1.3 Query Objects*

As opposed to the Get Object by ID operation that retrieves a single record of an entity, the **Query Objects** operation retrieves multiple records of an entity.

For this operation, the following query options are available: **Use Pagination**, **Filters**, **Response Properties**, **Order Response by**, **Expand Navigation Properties**, **Aggregate Properties**, and **Outbound Headers**. Except for the Pagination, these options are explained in Table 3 of Section 8.2.1.1.

The use of the Pagination option is explained in Section 8.2.1.3.1.

For an example of the configuration of the Query Objects operation, refer to Figure 8.7.

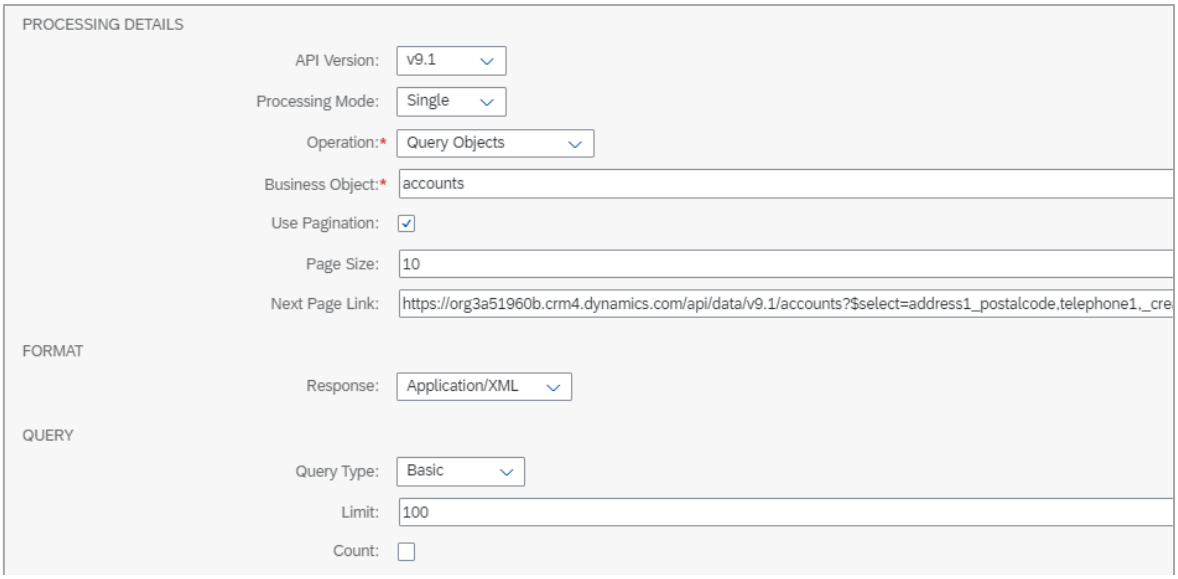



*Figure 8.7 Sample Configuration of the Query Objects Operation (Part 1)*

As depicted in Figure 8.7, the configurations for this operation include the following fields:

- **Business Object**: This needs to be filled in with the name of the entity to be retrieved.
  Examples: accounts, invoices, opportunities, etc.
  To see the full list of possible entities, refer to the Eclipse Plug-In in Section 6.2.1.
  Note that this field also accepts functions discussed in Section 8.2.1.4.
- **Use Pagination**: This field specifies how to control the number of returned records. This is ideal for use when dealing with multiple records. Once checked, the Page Size and the Next Page Link can be specified. This field is explained in detail in Section 8.2.1.3.1.
- **Response**: This field specifies the format of the response message to be returned by Dynamics CRM. Possible values include Application/XML and Application/JSON. Note that the default value is Application/XML.
- **Query Type**: Possible values include Basic, Advanced, and FetchXML. For details on each of these Query types, refer to the detailed explanation in Section 8.2.1.1.
- **Limit**: Specifies the maximum number of records to be returned as a response.
- **Count**: Select to include the total number of records that meet the query or filtering criteria in the response. The response includes a property called odata.count.

Figure 8.8 shows the different query options explained in Section 8.2.1.1.

Note that the Basic Query type is used in the example.

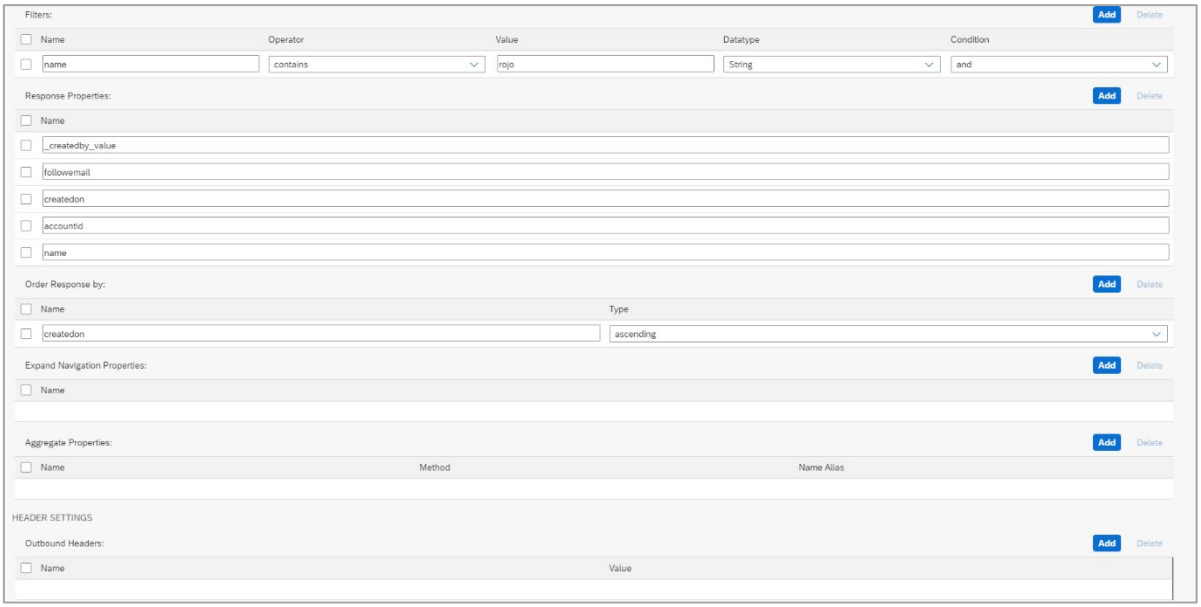



*Figure 8.8 Sample Configuration of the Query Objects Operation (Part 2)*

*8.2.1.3.1 Use Pagination*

The **Use Pagination** feature is useful if a large set of records to be returned by a query is expected. A large set of results can impact performance. With Pagination, the response is broken and sent in several small parts, and this improves performance.

When the **Use Pagination** feature is selected, the following fields become available:

- **Page Size:** Specifies the maximum number of records to be returned as a response. This field is set to 500 by default. Note that this field plays the same role as the header OData.maxpagesize. If both the Page Size and the OData.maxpagesize are available in the Adapter, Page Size takes precedence.
- **Next Page Link**: In case more records are returned in the response than the number specified in the Page Size, then a @odata.nextLink property will be returned with the results. The value @odata.nextLink property can be used to retrieve the next result page. The Next Page Link is meant to be filled with this value. In most cases, this field needs to be made dynamic by using a property. Example: ${property.nextLink}.

Note that this is an optional field.

For illustration purposes, assume that all the top one hundred (100) account records in Dynamics CRM are to be retrieved but only five (5) records are desired to be returned in each response. With these parameters set, twenty (20) pages containing five (5) records each are returned.

The Use Pagination functionality deals with use cases that have specific requirements set in the example. A sample configuration of the Adapter with the Use Pagination function is shown in Figure 8.9.

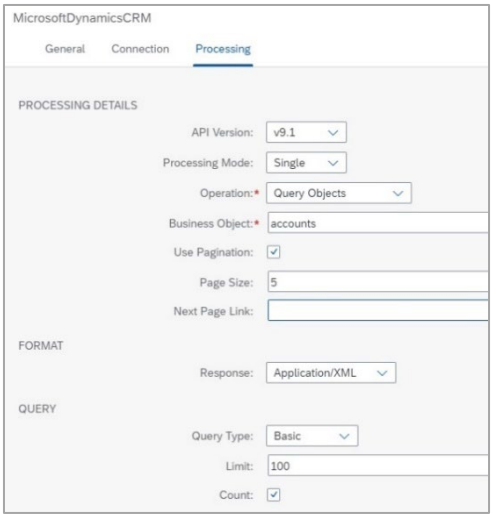



*Figure 8.9 Sample Configuration of the Use Pagination Function*

Figure 8.10 shows the response returned in the configuration seen in Figure 8.9.


*Figure 8.10 Example Response when using Pagination*

Note that the @odata.nextLink property returned in Figure 8.10 is used to retrieve the next page.

*8.2.1.4 Call Function*

The Adapter also supports calling Microsoft Dynamics CRM functions. For more Functions, refer to the Microsoft Dynamics 365 Documentation.

Note that the Function operation supports activities without observable side effects.

Figure 8.11 shows an example of using the Adapter's Function Operation with GetTimeZoneCodeByLocalizedName function. This retrieves the time zone code for the specified localized time zone name.

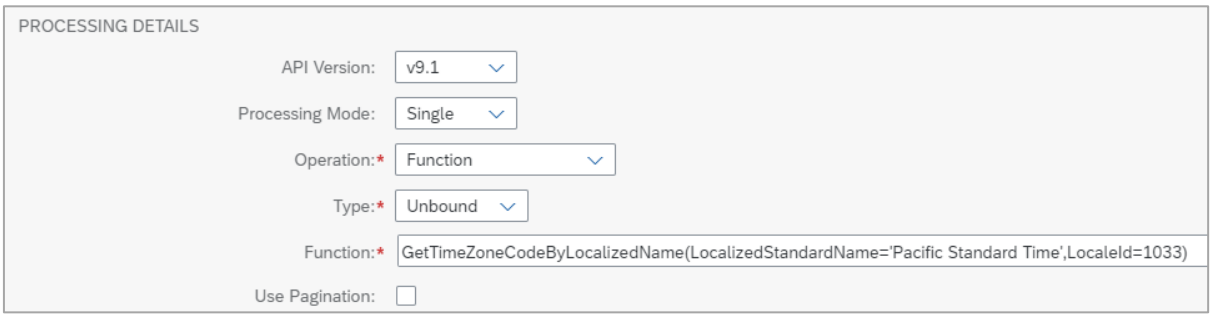



*Figure 8.11 Sample Configuration of the Function Operation, GetTimeZoneByLocalizedName()*

Figure 8.12 shows how to use the WhoAmI() function which returns the system user ID for the currently logged-on user or the user under whose context the code is running.

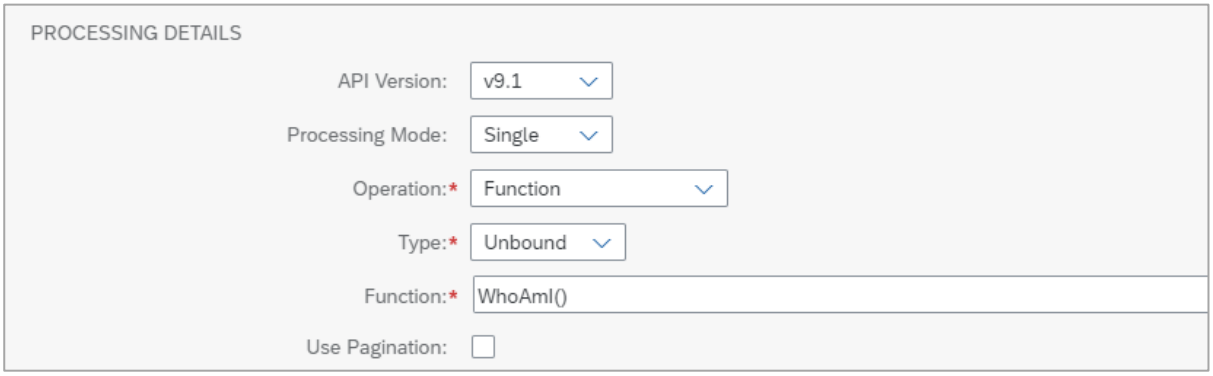



*Figure 8.12 Sample Configuration of the Function Operation, WhoAmI()*

The Function Operation can be further enhanced by selecting the type of action performed. The types available for selection are Bound and Unbound.

As seen in Figure 8.13, selecting the Bound type option requires supplying the Business Object and Business Object ID.


*Figure 8.13 Sample Configuration of the Function Operation, Bound Type*

*8.2.1.5 Update Object*

The Update Object Operation updates the properties of a specific record of an entity or a business object.

For this operation, the ID of the concerned entity is required.

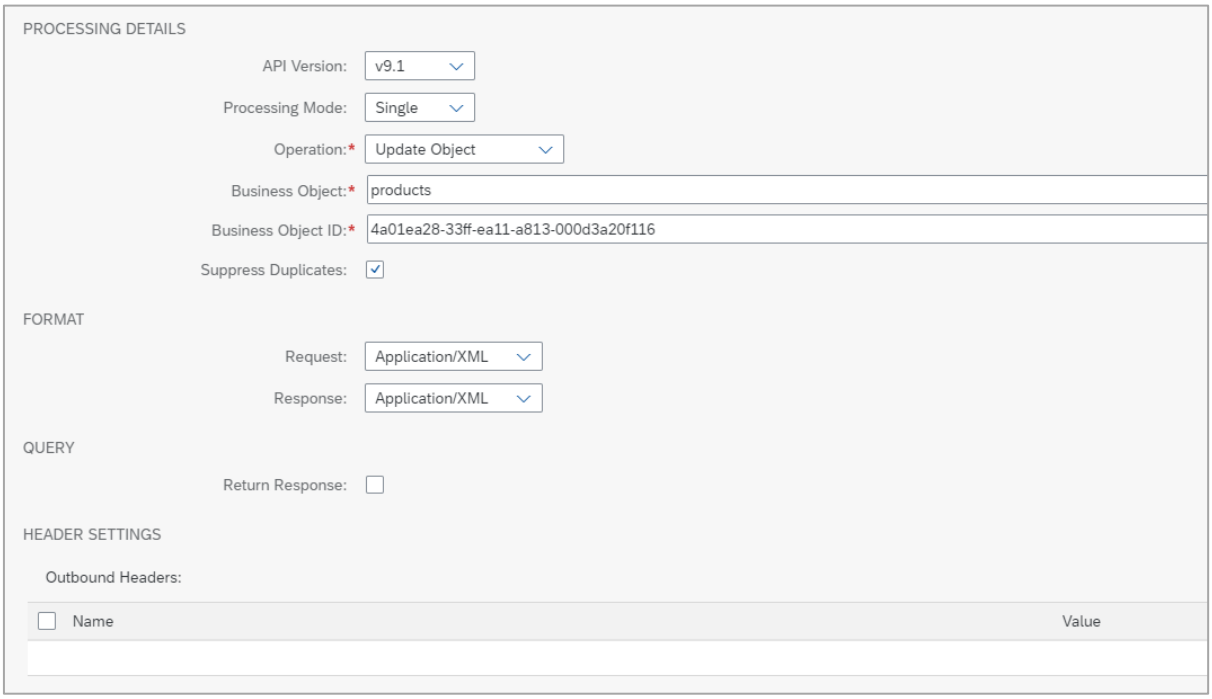



*Figure 8.14 Sample Configuration of the Update Object Operation (Part 1)*

To configure the Update Object operation, the following properties need to be specified:

- **Business Object**: This needs to be filled with the name of the entity to be updated.
  For example, accounts, invoices, opportunities, etc.
  To see the full list of possible entities, refer to the Eclipse Plug-In in Section 6.2.1.
- **Business Object ID:** The unique ID that represents the entity specified in the Business Object field.
- **Suppress Duplicates**: As a default behavior, the Update Object does not check for duplicates. This means that if an attempt to update records that already exist in Microsoft Dynamics CRM is made, these records are still updated. To prevent such, select this checkbox.
- **Request**: This field specifies the format of the request message to be sent to Dynamics CRM. 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 Dynamics CRM. Possible values include Application/XML and Application/JSON.
- Note that the default value is Application/XML.
- **Return Response**: As a default behavior, the Update Object operation does not return a response after it has been performed. If returning a response is desired, select this checkbox.

Furthermore, the following query options are available: **Response Properties**, **Expand Navigation Properties**, and **Outbound Headers**.

These query options are explained in Table 3 of Section 8.2.1.1.

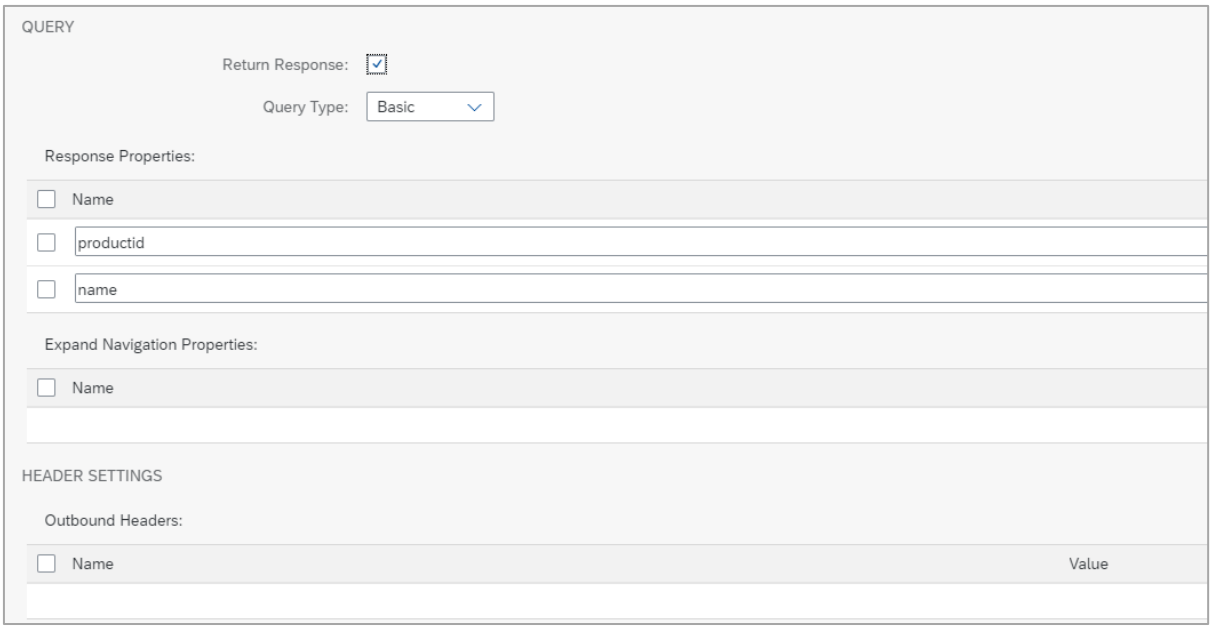



*Figure 8.15 Sample Configuration of the Update Object Operation (Part 2)*

As depicted in Figure 8.15, the Query type can be specified when performing the Update Object operation.

Possible values include **Basic**, **Advanced**, and **FetchXML**.

For details on each Query type, refer to the explanation in Section 8.2.1.1.

For illustration purposes, the example in Figure 8.14, updates a product in Microsoft Dynamics CRM with ID “c0693992-0cff-ea11-a813-000d3a20f116”.

Additionally, a Payload in a form of JSON or XML (using the Request format) is required to be configured in the Adapter. Use the Eclipse Plug-In to generate the XSDs that can be used to create the message mapping in front of the Adapter. Refer to Section 6.2.

#### *8.2.1.5.1 Sending Null Values in XML Request*

When performing an update in Microsoft Dynamics CRM, only the fields that have been specified in the request message are updated or inserted. In case an empty field is sent, Dynamics CRM will treat it as null and remove the field from the concerned entity.

Example: <name></name>.

**Note**: This does not work for all data types in Dynamics CRM. For instance, the DateTime data type. There are two alternative ways to set a field to null or clear it:

- Use the null as a value: Example <name>null</name>.
- Use the Delete operation: Using the Delete operation, it is possible to specify the field to be deleted in the **Affected Property** Adapter field. This is discussed in Section
- 8.2.1.6.

*8.2.1.6 Delete Object*

The Delete Object operation deletes a record of an entity or a business object.

For this operation, the ID of the concerned entity needs to be provided.

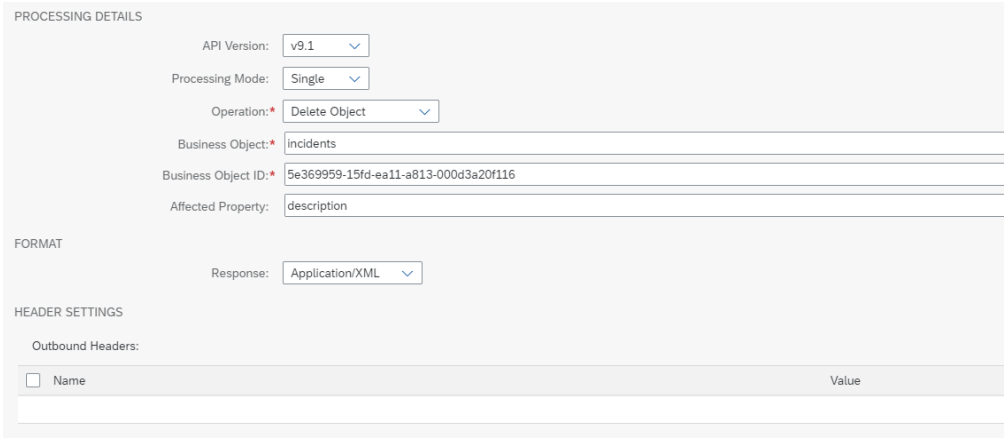



*Figure 8.16 Sample Configuration of the Delete Object Operation*

To configure the Delete Object operation, the following properties need to be specified:

- **Business Object**: This needs to be filled in with the name of the entity to be deleted.
  For example, accounts, invoices, opportunities, incidents, etc.
- **Business Object ID**: The unique ID that represents the entity specified in the Business Object field.
- **Affected Property**: It specifies the property of the entity that will be affected by the deletion. If a property in this field is specified, only the specified property will be deleted and the rest of the properties of the entity will be left intact. In case this field is left empty, the complete record of the entity or the business object will be deleted.
- **Response**: This field specifies the format of the response message to be returned by Dynamics CRM. Possible values include Application/XML and Application/JSON. Note that the default value is Application/XML.

Furthermore, the following query options are available: **Response Properties**, **Expand Navigation Properties**, and **Outbound Headers**.

These query options are explained in Table 3 of Section 8.2.1.1.

For illustration purposes, the example in Figure 8.16 deletes the description of an incident with the ID “5e369959-15fd-ea11-a813-000d3a20f116”.

*8.2.1.7 Create Object*

The operation Create Object creates a new record of an entity or a business object.

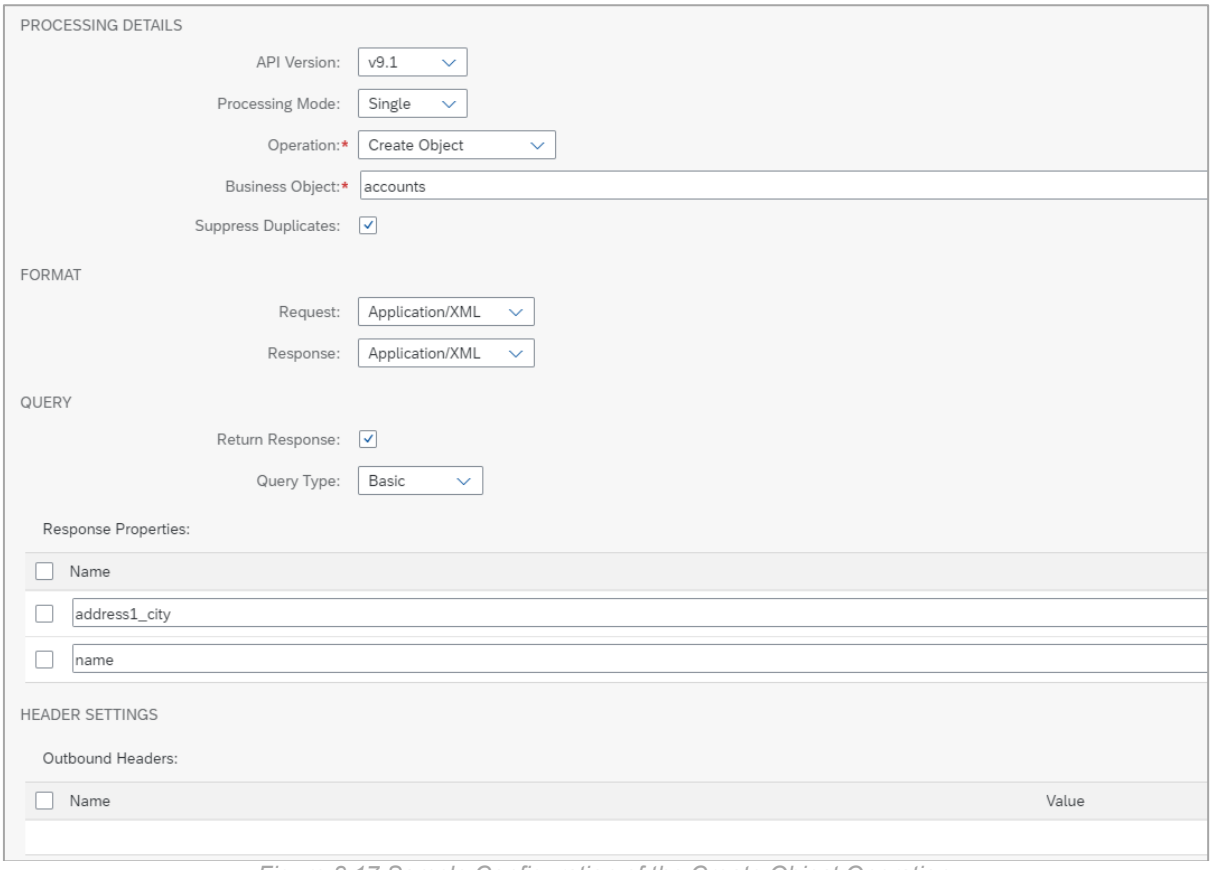



*Figure 8.17 Sample Configuration of the Create Object Operation*

To configure the Create Object operation, the following properties need to be specified:

- **Business Object**: This needs to be filled in with the name of the entity to be created.
  For example, accounts, invoices, opportunities, etc.
  To see the full list of possible entities, refer to the Eclipse Plug-In in Section 6.2.1.
- **Suppress Duplicates**: As a default behavior, the create object operation does not check for duplicates. This means that in case an attempt to create records that already exist in Microsoft Dynamics CRM is made, the records will still be created. To prevent such creation, select this checkbox.
- **Request**: This field specifies the format of the request message to be sent to Dynamics CRM. 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 Dynamics CRM. Possible values include Application/XML and Application/JSON. Note that the default value is Application/XML.
- **Return Response**: As a default behavior, the update object does not return a response after the operation has been performed. In case a response is to be returned, select this checkbox.

Furthermore, the following query options are available: **Response Properties** and **Outbound Headers**.

These query options are explained in Table 3 of Section 8.2.1.1.

The example in Figure 8.17 shows a Create account request in Microsoft Dynamics CRM which returns a response that contains only the indicated properties in XML format: name, createdon, and accountid.

Additionally, a Payload in a form of JSON or XML (using the Request format) is required to be configured in the Adapter. Use the Eclipse Plug-In to generate the XSDs that can be used to create the message mapping in front of the Adapter. Refer to Section 6.2.

Figure 8.18 shows an example request message.

```
<request>
      <entry>
            <name>Test001</name>
            <defaultuomid_lookup customertype="uoms">f5eb7e61-0cff-ea11-a813-000d3a20f116</defaultuomid_lookup>
            <defaultuomscheduleid_lookup customertype="uomschedules">fab3d3b0-0bff-ea11-a813-000d3a20f116</defaultuomscheduleid_lookup>
      </entry>
</request>
```

*Figure 8.18 Sample Request XML for Create Object Operation*

### *8.2.1.8 Associate Object*

The **Associate Object** operation creates a link or an association between two entities in Dynamics CRM. For example, a link between an Account and an Address record.

Figure 8.19 shows how to use the associate object operation in the Eclipse Plug-In to generate XSDs. Different entities that can be associated with the Accounts entity are available in the Plug-In.

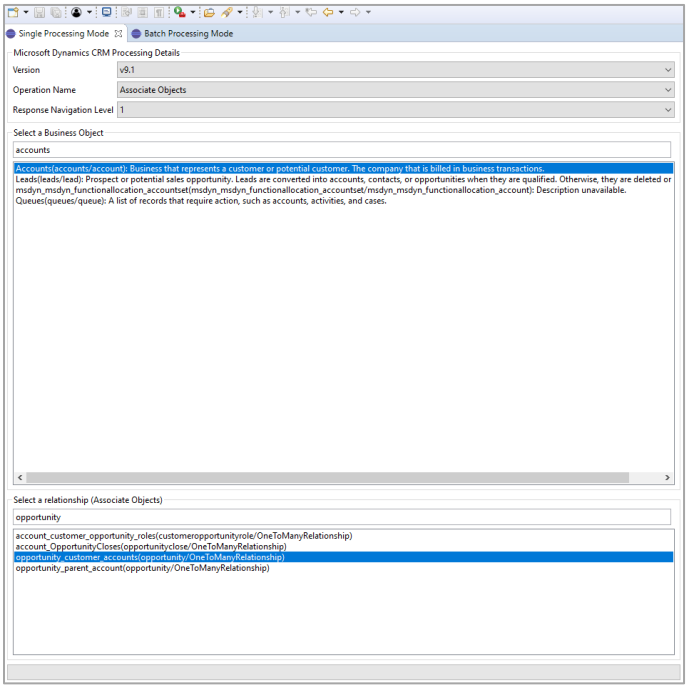



*Figure 8.19 Plug-In associations*

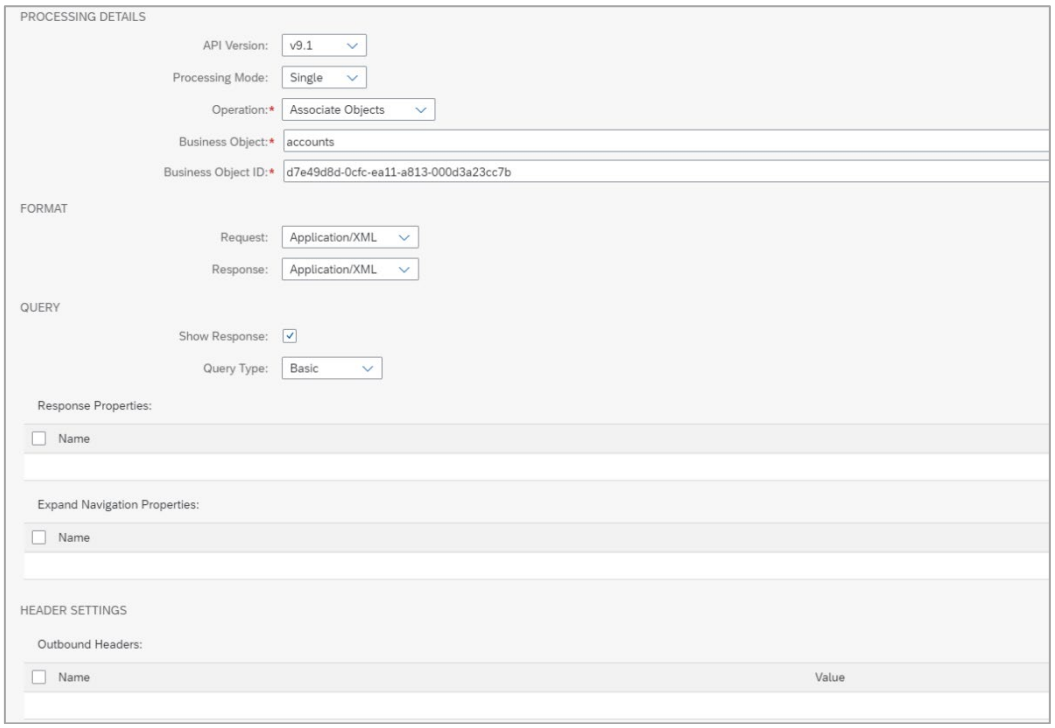



*Figure 8.20 Sample Configuration of Associate Object Operation*

Figure 8.20 shows a sample configuration when using the Associate Object operation. Note that the following properties are required:

- **Business Object**: Enter the name of the entity which the association is to be created with. For example, accounts, invoices, opportunities, etc. To see the full list of possible entities, refer to the Eclipse Plug-In in Section 6.2.1.
- **Business Object ID**: The unique ID that represents the entity specified in the Business Object field.
- **Request**: This field specifies the format of the request message to be sent to Dynamics CRM. 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 Dynamics CRM. Possible values include Application/XML and Application/JSON. Note that the default value is Application/XML.
- **Return Response**: As a default behavior, the update object does not return a response after the operation has been performed. If a response is to be returned, select this checkbox.

Furthermore, the following query options are available: **Response Properties**, **Expand Navigation Properties**, and **Outbound Headers**.

Figure 8.20 illustrates an example where an association between an account and an opportunity in Microsoft Dynamics CRM is created. This sample association is called opportunity_customer_accounts.

Refer to Figure 8.21 for a sample request XML that can be used in combination with the sample Adapter configuration in Figure 8.20.

Note that a Content Modifier can be used to send this payload. Alternatively, the XSD generated in the Eclipse Plug-In can be used to create a Message Mapping.

```
<request>
  <entry>
    <opportunity_customer_accounts>
      <opportunities>
        <id>b9668677-8552-44ea-a729-4d8f8d5b6559</id>
      </opportunities>
    </opportunity_customer_accounts>
  </entry>
</request>
```

*Figure 8.21 Sample Request XML for Associate Object Operation*

*8.2.1.9 Disassociate Object*

The Disassociate Object operation unlinks two entities in Dynamics CRM. For example, to remove the link/association between an Account and an Address record.

Figure 8.22 shows how to configure the Disassociate Object operation on the Account entity.

Like the Associate operation (Section 8.2.1.8), the Eclipse Plug-In can be used to generate XSDs for the Disassociate operation.

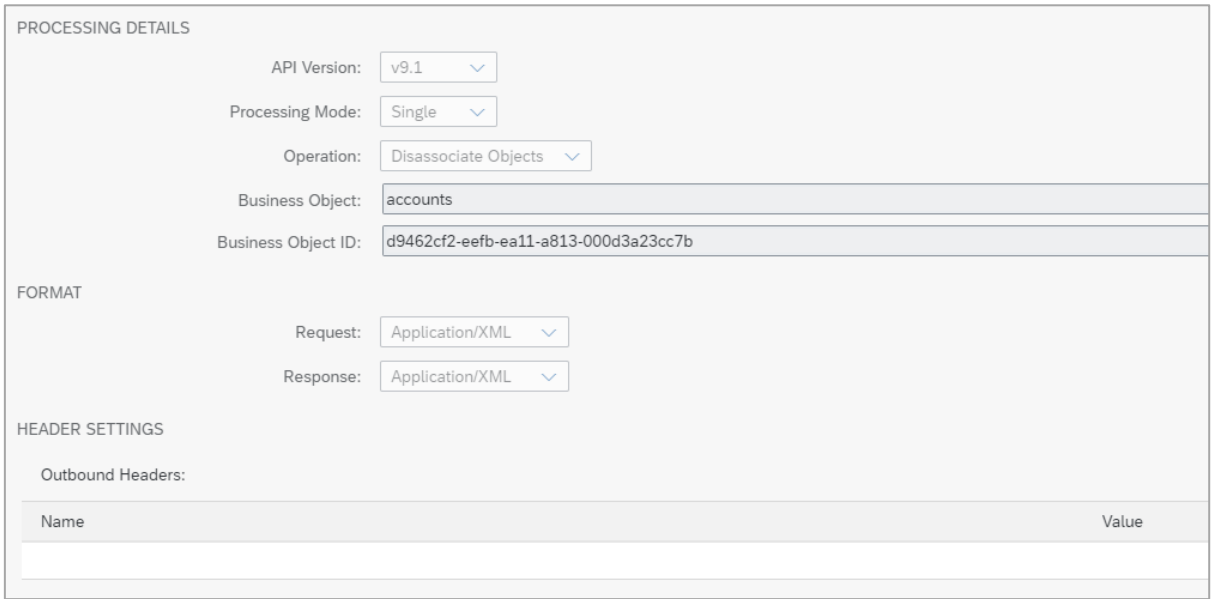



*Figure 8.22 Sample Configuration of Disassociate Object Operation*

As shown in Figure 8.22, configuring the Disassociate Object operation requires the following properties:

- **Business Object:** Supply the name of the entity that an existing association is to be removed from. For example, accounts, invoices, opportunities, etc.
  To see the full list of possible entities, refer to the Eclipse Plug-In in Section 6.2.1.
- **Business Object ID:** The unique ID that represents the entity specified in the Business Object field.
- Request: This field specifies the format of the request message to be sent to Dynamics CRM. 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 Dynamics CRM. Possible values include Application/XML and Application/JSON.
  Note that the default value is Application/XML.

Furthermore, Outbound Headers can be added.

Refer to Section 8.2.3

### *8.2.2 Batch Request Processing Mode*

The **Batch Request Processing Mode** allows the sending of multiple requests in a single call to Microsoft Dynamics CRM. This mode also supports the combination of different operations in a single request.

**Note:** The Create Object, Update, Delete, Associate, and Disassociate operations can be included in the batch.

Batch requests are best used when performing operations on entities that are not related to each other. It also implies that the operations and records included in the payload will be performed as single transactions. The Get operations (Get Object by ID and Query Objects) are supported since version 1.0.6 of the Adapter.

Note: Batch operations succeed or fail as a group.

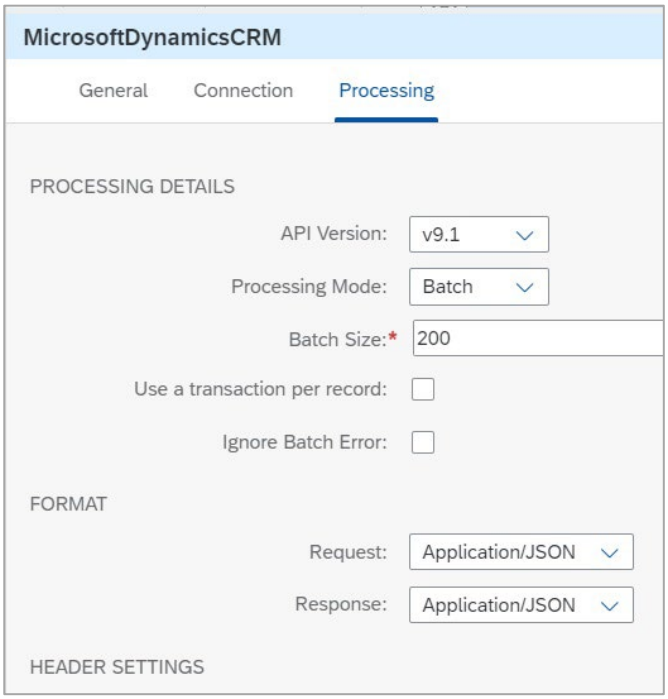



*Figure 8.23 Configurations for batch processing mode*

As depicted in Figure 8.23, the configuration for this operation includes the following fields:

- **Batch Size:** This represents the maximum number of records that will be sent to Dynamics at any given time as a single call. In case more records exist than this number, the Adapter will call Dynamics CRM multiple times.
  Note that the default batch size is 200 and that this field accepts dynamics values (header and properties).
  Example: ${property.MSD_Batchsize}.
- **Use a transaction per record:** When not selected, all records included in the input payload are sent to Microsoft Dynamics as a single transaction.
  This means that when one record fails, all records are rolled back. In case, this option is selected, each record in the input payload is treated as a separate transaction and executed individually.
  **Note**: Microsoft Dynamics CRM processes the records sent (as part of the Batch input payload) in sequential order. In case one of the input records has an error when being processed in Dynamics

CRM, the subsequent input records will not be processed. As a result, the response payload will only contain the processed records.

- **Ignore Batch Error:** By default, if any records in the input message fail, Microsoft Dynamics CRM returns a failed response with the description of what failed. Selecting the “Ignore Batch Error” option treats the failure as a successful response and includes details of the failed and success records.
- **Request:** This field specifies the format of the request message to be sent to Microsoft Dynamics CRM.
  Possible values include Application/XML or Application/JSON.
  Note that the default value is Application/XML.
- **Response:** This field specifies the format of the response message to be returned by Dynamics CRM.
  Possible values include Application/XML, Application/JSON, or Multipart/Mixed.
  Note that the default value is Application/XML.

Furthermore, the **Outbound Headers** query option is also available. Refer to Section 8.2.3 for more information.

A Message Mapping step or a Content Modifier step needs to be placed in front of the Microsoft Dynamics CRM Adapter integration flow which uses Batch.

In case a Message Mapping step is used, generate the related XSD schemas using the Eclipse Plug-In, as shown in Figure 8.24. The Batch processing mode supports both the JSON and XML formats.

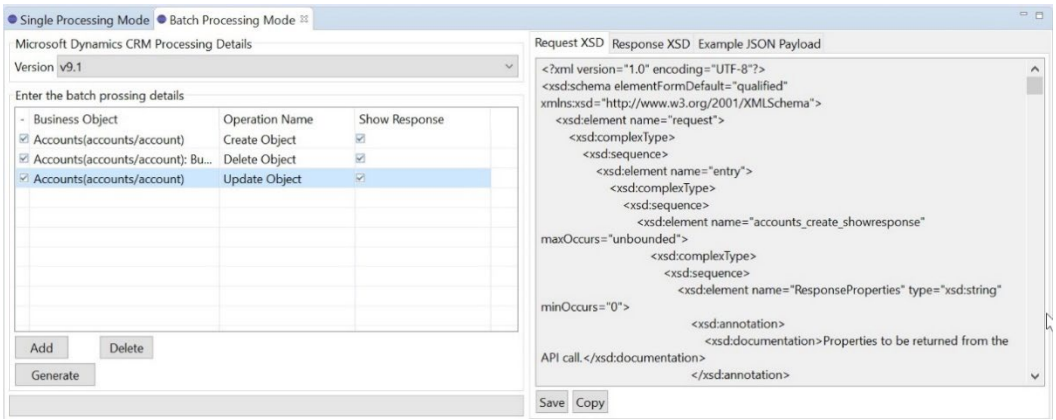



*Figure 8.24 Batch Processing Mode with Eclipse Plug-In*

As seen in Figure 8.24, it is possible to add multiple entities as part of the batch XSD. Figure 8.24 also shows an example where an XSD that supports the Creation, Deletion, and Update of the Account entity is generated.

It is possible to add as many entities and operations as needed. Before generating the XSD, select the checkbox next to entities and click on the **Generate** button. A JSON example of the message is also sent to Dynamics CRM for the batch.

For illustration purposes, the example payload in Figure 8.25 updates an account with ID “c27215d9-08c1-ea11-a812-000d3a20f116” while creating two accounts with the names “customer001” and “customer002”, and deleting an account with the ID “5930300c-41c8-ea11-a812-000d3a23cc7b” at the same time.

```
{
   "request":{
      "entry":{
         "accounts_update_noresponse":{
            "BusinessObjectId":"c27215d9-08c1-ea11-a812-000d3a20f116",
            "Payload":{
               "name":"testbatch300"
            }
         },
         "accounts_create_returnresponse":[
            {
               "ResponseProperties":"name",
               "Payload":{
                  "name":"testbatch100"
               }
            },
            {
               "ResponseProperties":"name",
               "Payload":{
                  "name":"testbatch200"
               }
            }
         ]
      }
   }
}
```


*Figure 8.25 Example of a Batch request input message*

The different parts of Figure 8.25 are explained below. Remember that this structure is automatically generated by the Eclipse Plug-In.

- **Request and entry:** Batch requests always start with:

  { "request": {s "entry":{

- **Entity, operation, and response:** The request starts with the name of the entity selected. This is then followed by the operation name (Create, Update, Associate, Dissociate, or Delete). Lastly, a keyword indicates whether a response should be returned or not. **Returnresponse** indicates if the values of certain properties back in the response are to be returned. If a response key is not needed, use **noresponse**.

- **ResponseProperties:** In ResponseProperties, certain properties can be added to the returned response for each entity included in the batch request. List the properties and separate each with a comma. There should be no spaces between the property names. Note that the Delete operation does not return any properties in its response.

- **Payload:** When creating an entity, specify what information the entity is to have in the payload.

- **BusinessObjectID:** For the Delete and Update operations, specify the ID of the entity.

*For the Delete Object operation, it is also possible to use the **affected property** option. This option is detailed in Section*

8.2.1.6. This only deletes the attribute instead of the entire entity.

See the example in Figure 8.26.


```
<accounts_delete_noresponse>
  <BusinessObjectId>5930300c-41c8-ea11-a812-000d3a23cc7b</BusinessObjectId>
  <AffectedProperty>telephone1</AffectedProperty>
</accounts_delete_noresponse>
```


*Figure 8.26 Example of Delete in the Batch Mode*

**Note:** Mandatory fields such as the name and the ID of an entity cannot be deleted.

*8.2.2.1 Use a Transaction Per Record and Ignore Batch Error Options*

As explained in the previous section, all records of a payload are treated as part of a single transaction by default.

Selecting both “Use a transaction per record” and “Ignore Batch Error” ensure that:

- Each record included in the input payload is treated as a separate transaction.
- The response returned by Microsoft Dynamics CRM specified failed as well as successful records.
- Each record in the response is automatically associated with the request records using the <content_id> element.

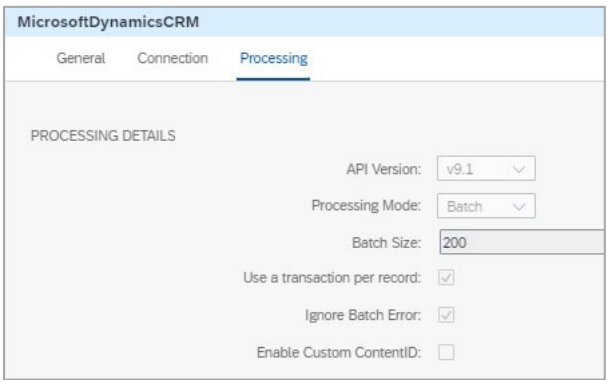



*Figure 8.27 Sample Configuration of Processing Mode: Batch, Transaction per Record Option Enabled*

Example:

```
<?xmlversion="1.0"encoding="UTF-8"?>
<response>
        <entry>
                <value>
                        <content_id>1</content_id>
                        <batch_response_payload>
                        <api_success>true</api_success>
                        </batch_response_payload>
                </value>
                <value>
                        <content_id>2</content_id>
                        <batch_response_payload>
                        <accountid>b0e2360e-93a6-eb11-b1ac-000d3ac255e3</accountid>
                        <odata.etag>W/&amp;quot;21279343&amp;quot;</odata.etag>
                        <name>testacc0006</name>
                        <odata.context>https://org3a51960b.crm4.dynamics.com/api/data/v9.1/$metadata#a
                        ccounts(name)/$entity</odata.context>
                        </batch_response_payload>
                </value>
        </entry>
</response>
```

*8.2.2.2 Conditional Operations*

Dynamics CRM uses the notion of entity tag (also known as ETag) to identify a version of a specific entity. Most operations of the adapter return the latest ETag of the entity. This is the case after performing an update or create operation. An example of a returned ETag is shown in Figure 8.28.

```
<accountid>0820f7ce-53d8-eb11-bacb-000d3abd055d</accountid>
<odata.etag>W/"26168245"</odata.etag>
<name>1</name>
```

*Figure 8.28 Example ETag after creating an Account*

In the Adapter, it is possible to use the If-Match and If-None-Match headers with ETag values to verify if the current version of an entity in Microsoft Dynamics CRM matches any previous version, or not.

In a scenario wherein an entity record is retrieved multiple times from Microsoft Dynamics CRM and only data regarding changes made to the record since the last time it was retrieved is desired, it is possible to pass the name **If-None-Match** and the value of the last ETag in the Outbound Headers section of the Adapter. Further details on the topic can be found in Section 8.2.2.4.

### *8.2.2.3 Specify Custom Content ID in the Batch*

When using the Batch functionality, the Adapter assigns a Content ID to uniquely identify each record of the payload by default.

It is possible to overwrite this behavior and assign a unique Content ID. To do so, choose the “Enable Custom ContentID” checkbox, as seen in Figure 8.29.

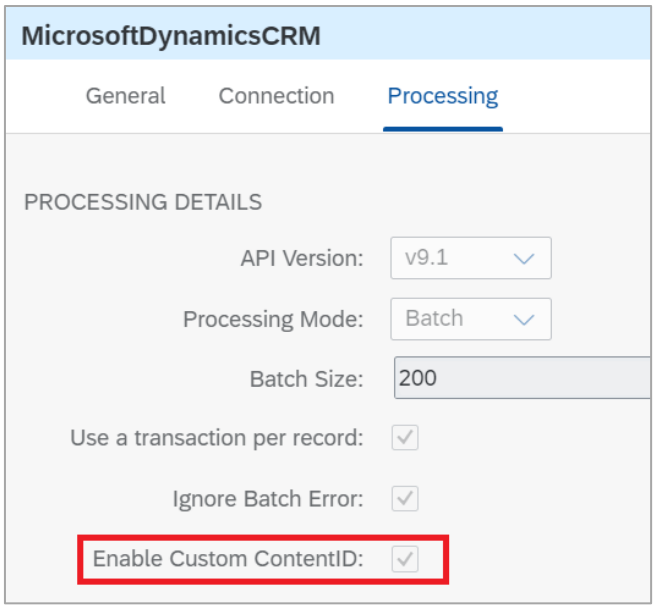



*Figure 8.29 Enable Custom Content ID Option, Microsoft Dynamics CRM Adapter, Processing Tab*

Additionally, a value needs to be assigned to the ContentId field from the payload that is sent to the Adapter. See an example input with a payload where custom ContentId is assigned:

```
<?xml version="1.0" encoding="utf-8"?>
<request>
        <entry>
                <new_salesarea_gbs_getobjectbyid>
                        <ContentId>1</ContentId>
                        <BusinessObjectId>5ea985d5-f1c9-eb11-    bacc</BusinessObjectId>
                        <ResponseProperties>new_name</ResponseProperties>
                </new_salesarea_gbs_getobjectbyid>
                <new_salesarea_gbs_create_noresponse>
                        <ResponseProperties>new_name</ResponseProperties>
```

```
                    <ContentId>2</ContentId>
                    <Payload>
                        <new_name>Test2000</new_name>
                    </Payload>
                </new_salesarea_gbs_create_noresponse>
                <accounts_query>
                    <ContentId>3</ContentId>
                    <Query querytype="odata">$filter=accountid eq 'cfbfdd2e-10fb-ea11-
                    a813'_and_$select=name</Query>
                </accounts_query>
                <new_salesarea_gbs_create_noresponse>
                    <ResponseProperties>new_name</ResponseProperties>
                    <ContentId>1000</ContentId>
                    <Payload>
                        <new_name>Test1000</new_name>
                    </Payload>
                </new_salesarea_gbs_create_noresponse>
        </entry>
</request>
```

Note that in case the “Enable Custom ContentID” is selected, the following rules apply:

- It is mandatory to provide the ContentID for every record.
- It is not allowed to have duplicate ContentID.

Furthermore, the content ID values specified in the input are also returned in the response. This assists in correlating and identifying matching request and response records.

*8.2.2.4 Reference URIs in a Batch Request*

The Adapter provides the functionality of using the $parameter (such as $1, $2, etc) in the body of a batch request to reference URIs returned for new entities created earlier in the same changeset in a batch request.

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<request>
  <entry>
    <accounts_create_noresponse>
      <ContentId>765</ContentId>
      <Payload>
        <name>test002020</name>
      </Payload>
    </accounts_create_noresponse>
    <new_salesarea_gbs_create_noresponse>
      <ResponseProperties>new_name</ResponseProperties>
      <ContentId>763</ContentId>
      <Payload>
            <new_company_codeid_lookup
customertype="accounts">$765</new_company_codeid_lookup>
        <new_name>Johnson</new_name>
      </Payload>
    </new_salesarea_gbs_create_noresponse>
  </entry>
</request>
```

Based on the above example, 2 records are being sent to Dynamics CRM (one Account and one new_salesarea_gbs). The execution of records to Dynamics happens sequentially. Notice that the field “**new_company_codeid_lookup**” in the second record refers to the content ID of the first record. This means

after the first record has been created, its unique identifier will be automatically assigned as a value for the field "**new_company_codeid_lookup**" in the second record. This is done by appending a $ character in front of the previous contentID. Example: $765. It is important to note that this is only possible if a custom Content ID is specified in the payload. This is discussed in Section 8.2.2.3.

Note that the Reference URIs are not supported in case the multiple transactions checkbox is enabled.

### *8.2.3 Outbound Headers*

The Adapter enables the inclusion of additional headers that can be sent to Microsoft Dynamics CRM. In this section, some of the supported outbound headers are listed.

Note that this is not an exhaustive list.

#### *8.2.3.1 Prefer*

**Name**: prefer

**Possible Values**:

- Microsoft.Dynamics.CRM.formattedvalue
- Odata.maxpagesize
- Return=representation

**Description**:

Use the Prefer header with the **Microsoft.Dynamics.CRM.formattedvalue** option to return formatted values in a response to a query.

Use the Prefer header with the **odata.maxpagesize** option to specify how many pages are to be returned.

To return data for the Create (POST) or Update (PATCH) operations for entities, including the **return=representation** preference.

When this preference is applied to a Create request, a successful response will have status code 201 (Created). For an update request, a successful response will have a status code 200 (OK).

If this preference is not applied, both operations will return status code 204 (No Content). In this case, a single no data is returned in the body of the response by default.

**Example**: prefer odata.maxpagesize=10

#### *8.2.3.2 If-match*

**Name:** If-Match

**Value**: ETag

In Dynamics CRM, an ETag defines an entity tag. It can be used to filter and retrieve a specific version of an entity.

**Description**: If-Match is mostly used with state-changing methods (e.g., create, update, delete) to prevent accidental overwrites when multiple calls are made in parallel on the same resource.

**Example**: If-Match W/"9966158"

*8.2.3.3 If-none-match*

**Name**: If-None-Match

**Value**: ETag

In Dynamics CRM, an ETag defines an entity tag. It can be used to filter and retrieve a specific version of an entity.

**Description**: To control whether an upsert operation should create or update an entity, the If-Match and If-None-Match headers can also be used. The header field "If-None-Match" makes the request method conditional on a recipient cache or origin server.

**Example**: If-Match W/"9966158"

## 9. A QUERY EMBEDDED IN A REQUEST PAYLOAD

There are cases where it is required to query or retrieve values from Microsoft Dynamics CRM while performing a create or update operation. Without the possibility of embedding a query in a request, the developer will need to first perform a separate call using the Get by ID or Query operation in the Integration Flow. With this feature, the Adapter will automatically perform the embedded query and replace its value in the payload before calling an operation.

This feature is supported for the following operations:

- In the Single processing mode: Create, Update, Associate, and Disassociate.
- In the Batch processing mode: Create, Delete, Update, Associate, and Disassociate.

As an illustration, assume that an account with the statecode field needs to be retrieved from a query. The Adapter accepts a FetchXML or OData query instead of a value in the payload.

An example of how to achieve this is shown below.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<request>
  <entry>
    <accounts_create_returnresponse>
      <ResponseProperties>name,statecode</ResponseProperties>
      <Payload>
        <name>Test Batch With Query</name>
        <statecode querytype="fetchXML" queryEntityCollectionName="accounts"
queryTargetFieldName="statecode">
          <fetch mapping="logical">
            <entity name="account">
              <attribute name="statecode"/>
              <filter type="and">
                <condition attribute="accountid" operator="eq" value="437f099b-bfad-eb11-8236-
002248811925"/>
              </filter>
            </entity>
          </fetch>
        </statecode>
      </Payload>
    </accounts_create_returnresponse>
    <accounts_create_returnresponse>
```

```
    <ResponseProperties>name,statecode</ResponseProperties>
    <Payload>
      <name>Test Batch Without Query</name>
      <statecode>0</statecode>
    </Payload>
   </accounts_create_returnresponse>
  </entry>
</request>
```

As can be seen in the example above, the field statecode contains a FetchXML query instead of a normal string value. To specify the query, add the “queryType”, “queryEntityCollectionName”, and “queryTargetFieldName” attributes in the concerned field. Each one of these attributes is explained below:

- queryType: Represents the type of query to be used. Possible values include “FetchXML” or “OData”.
- queryEntityCollectionName: Represents the Business Object that needs to be queried.
- queryTargetFieldName: Represents the target property in the query response payload. This value will be used in the field of the XML payload.

Note that this feature is only supported for the XML Request format. In case multiple records are returned by the query, only the first record is used. Furthermore, when using an update operation, it is also possible to specify an attribute named “createonFail”. This attribute can be used to ensure that in case a unique identifier of a record is not found, a record will be created. It, therefore, behaves like an upsert operation.

Example using the createonFail attribute:

```
<?xml version="1.0"?>
<accounts_update_returnresponse>
  <ResponseProperties>name,emailaddress1</ResponseProperties>
  <BusinessObjectId querytype="fetchXML" queryEntityCollectionName="accounts"
queryTargetFieldName="accountid" createonFail="true">
    <fetch mapping="logical">
      <entity name="account">
        <attribute name="accountid"/>
        <filter type="and">
          <condition attribute="accountid" operator="eq" value="c27215d9-08c1-ea11-a812-000d3a20f117"/>
        </filter>
      </entity>
    </fetch>
  </BusinessObjectId>
  <Payload>
    <name>Test Batch With Query</name>
  </Payload>
</accounts_update_returnresponse>
```

It is also possible to use an OData query. The example below showcases an OData query example.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<request>
  <entry>
    <accounts_create_returnresponse>
      <ResponseProperties>name,statecode</ResponseProperties>
      <Payload>
        <name>Test Batch With OData Query</name>
        <statecode querytype="OData" queryEntityCollectionName="accounts"
queryTargetFieldName="statecode">
$filter=accountid eq '437f099b-bfad-eb11-8236-002248811925'&amp;amp;$select=statecode
</statecode>
      </Payload>
```

```
    </accounts_create_returnresponse>
    <accounts_create_returnresponse>
      <ResponseProperties>name,donotbulkemail</ResponseProperties>
      <Payload>
        <name>Test Batch Without OData Query</name>
        <statecode>0</statecode>
      </Payload>
    </accounts_create_returnresponse>
  </entry>
</request>
```

Note that the different attributes related to the lookup can be added to the XSD in the Eclipse Workbench using the “Enable Query Attribute” checkbox in the Batch Processing Mode Tab – as shown in Figure 9.1.

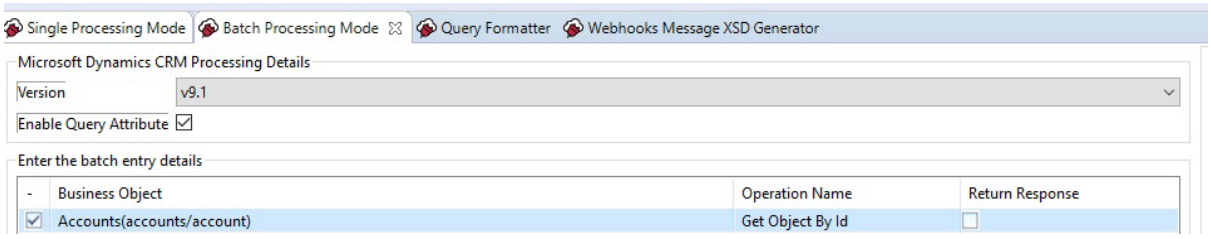



*Figure 9.1 Adding Query Attribute to the XSD*

## 10. USING WEBHOOKS TO SEND CHANGED DATA TO SAP CLOUD INTEGRATION

Some integration scenarios require sending events from Microsoft Dynamics CRM to SAP Cloud Integration as soon as an operation like creating, updating, deleting, etc. is performed. To achieve that, following the steps specified here to create a webhook pushes events to an external Handler. It is necessary to create an integration flow in SAP Cloud Integration that uses the HTTP sender adapter to accept the events.

The endpoint of this integration flow can be used in the **Endpoint URL** when creating a webhook. The Eclipse Plug-In that comes along with the Adapter can be used to generate the XSD to be used in SAP Cloud Integration from the events. Figure 10.1 shows where the XSD can be found in the Eclipse Plug-In.

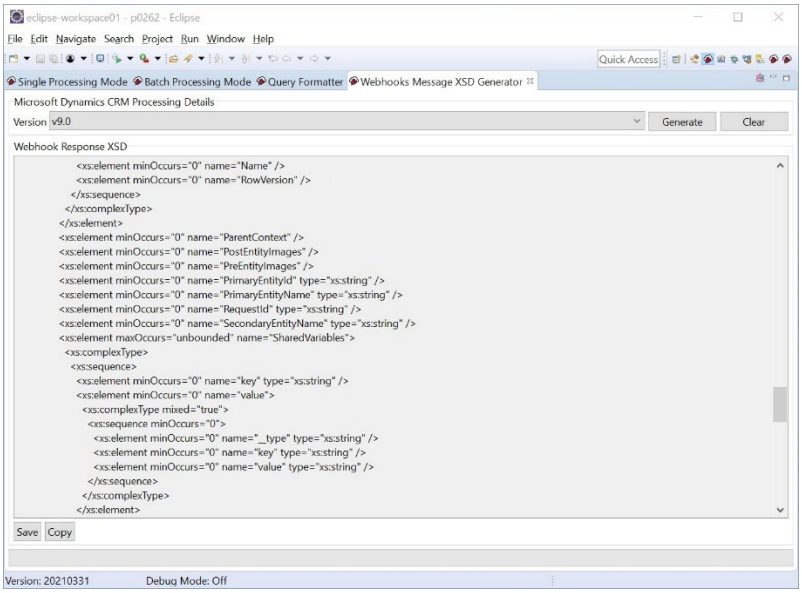



*Figure 10.1 XSD for Events Coming from Microsoft Dynamics CRM*

## 11. ASSIGNING VALUES TO CREATEDON FIELD

When creating records in Dynamics CRM, a few system fields are automatically assigned. This is the case for fields like createdOn and CreatedBy. There are integration scenarios that require these fields to be assigned specific values. The next section will discuss how to assign values to the CreatedOn and CreatedBy fields.

### 11.1 CreatedOn

By default, when a record is created, the field CreateOn is automatically assigned the current DateTime. To give the CreateOn a DateTime in the past, assign a value to the field "**overriddencreatedon**".

As an illustration, here is an example payload that performs a create on the Account entity:

```
<request>
  <entry>
    <name>ABC Inc</name>
    <overriddencreatedon>2010-10-10T12:12:12</overriddencreatedon>
  </entry>
</request>
```

The above example will create a record on the Account entity with the **createdon** field assigned to "2010-10-10T12:12:12". Note that Microsoft Dynamics CRM will automatically pick the value of **overriddencreatedon** and assign it to the **createdon** field.

### 11.2 CreatedBy

By default, when a record is created in Microsoft Dynamics CRM, the userid of the credential in the Connection Tab of the Adapter is automatically used and assigned to the CreatedBy field. It is possible to assign another user to the createdBy field. To achieve that, add an Outbound Header named "CallerObjectId" and assign to it the ObjectID of the user that needs to be used for the createdBy. See Figure 11.1.


*Figure 11.1 Outbound Header, CallerObjectID*

In the above example, a property is used to dynamically assign the value. Note that the value to be used for the CallerObjectId can be retrieved from Azure Active Directory. As depicted in Figure 11.2, the value of the Object ID should be used.

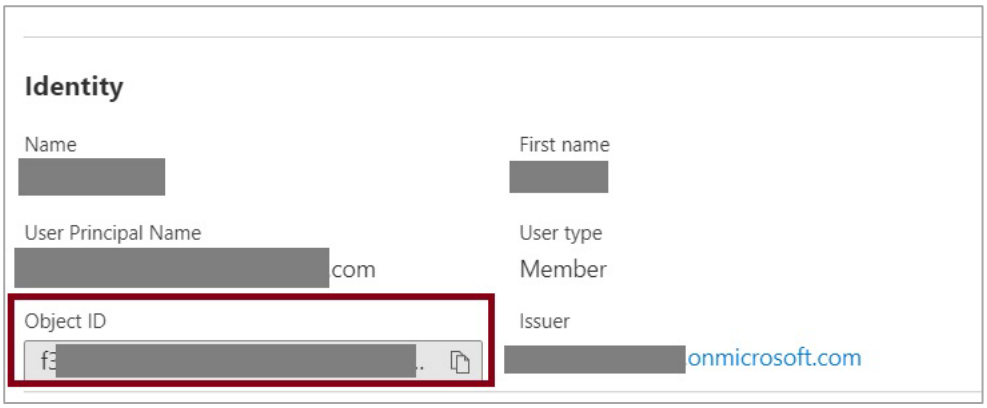



*Figure 11.2 Object ID of a User in Azure Active Directory*

Before a user is used on behalf of another user, the impersonate security privilege needs to be assigned as a pre-requisite. To achieve that, add the security privilege, in Dynamics CRM, to enable Impersonate for the concerned user as shown in Figure 11.3.

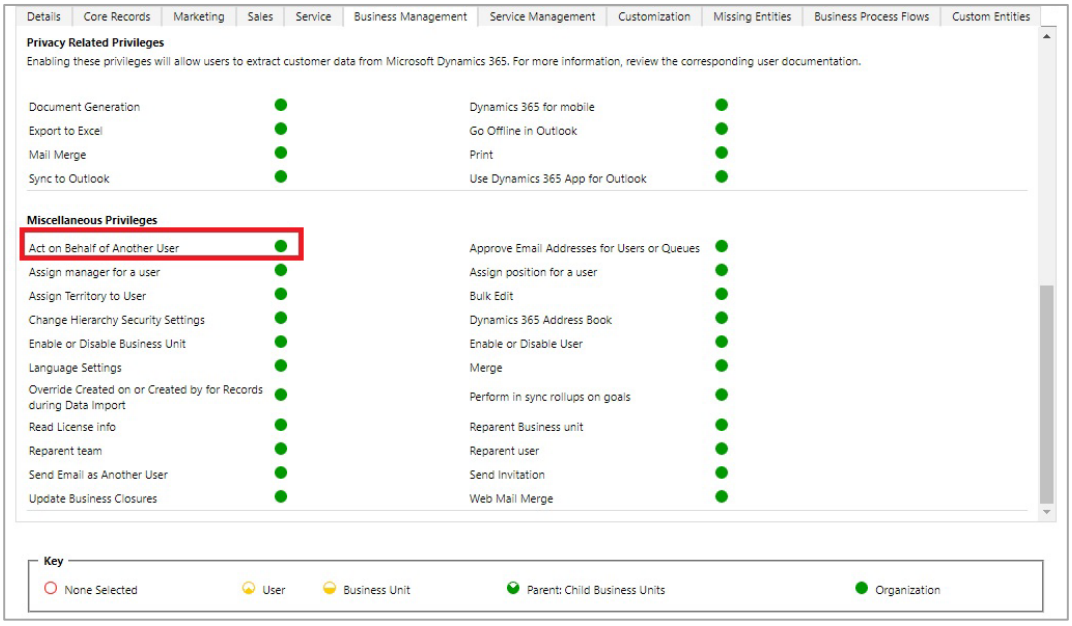



*Figure 11.3 Add the Security Privilege to Enable Impersonate*

## 12. SAMPLE SCENARIO EXPLAINED

In this section, a sample business scenario will be explained to show the use of the Adapter and the Eclipse Plug-In in a real-life scenario.

Figure 12.1 shows an integration flow in SAP Cloud Integration that is replicating Business Partners (customers) from SAP S/4HANA On-premise to Accounts in Microsoft Dynamics CRM.

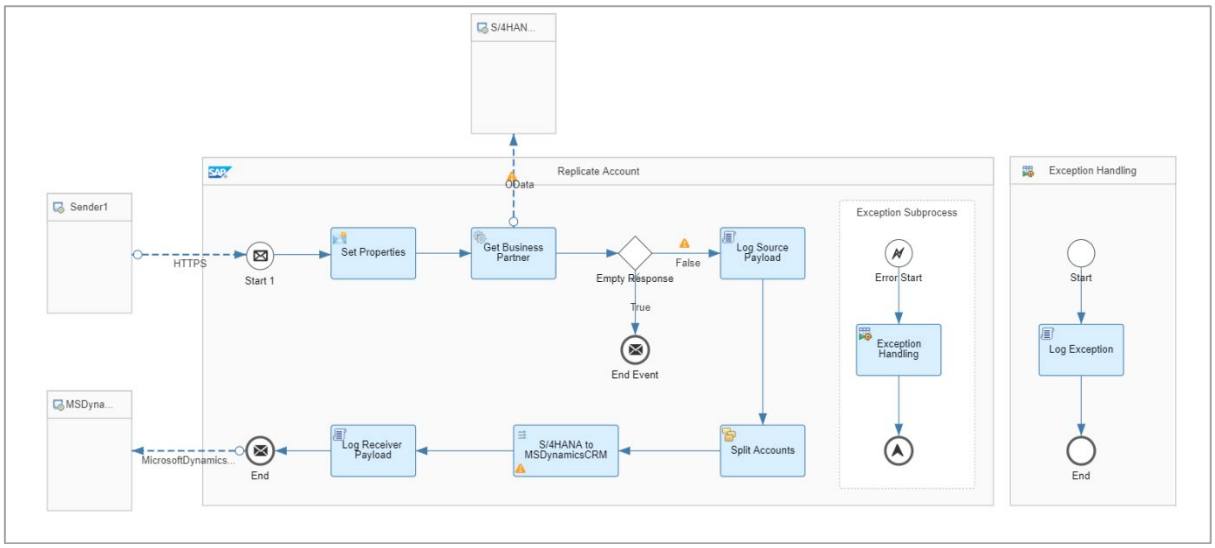



*Figure 12.1 Sample Scenario, Replicate Account (SAP S4/HANA to Microsoft Dynamics CRM)*

For simplicity reasons, the integration flow exposes an HTTPS endpoint. This means that the integration flow needs to be triggered externally by an HTTPS call.

Using an OData call, the iFlow queries SAP S/4HANA for all Business Partners created and modified since the last time the integration flow was deployed. In case there are no new Business Partners, the process ends. To retrieve these details from SAP S/4HANA, use the OData service named “API_BUSINESS_PARTNER”.

If new records are found, the source payload is logged. The returned Business Partners collection is split one at a time and mapped from the SAP S/4HANA Business Partner structure to the Microsoft Dynamics CRM Account entity.

To transform the structure coming from SAP S/4HANA to the Microsoft Dynamics CRM structure, XSDs for the Business Partner (SAP S/4HANA) and Accounts (Microsoft Dynamics CRM) are needed. The XSD of the data returned by S/4HANA is generated by the OData adapter. After selecting the fields to be queried, a new file is created as ‘A_BusinessPartnerEntityGET0’.

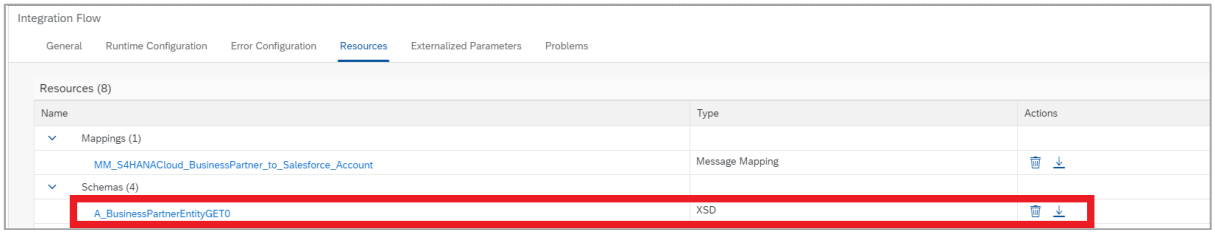



*Figure 12.2 Resources Tab in the Replicate Account (SAP S/4HANA to Microsoft Dynamics CRM) Integration Flow*

To generate the XSD needed on the Microsoft Dynamic CRM side to create an account, use the Eclipse Plug-In delivered along with the Adapter.

See Figure 12.3 for an example setup of the Plug-In.

Note that Create Object operation in Single Processing Mode is used in the scenario.

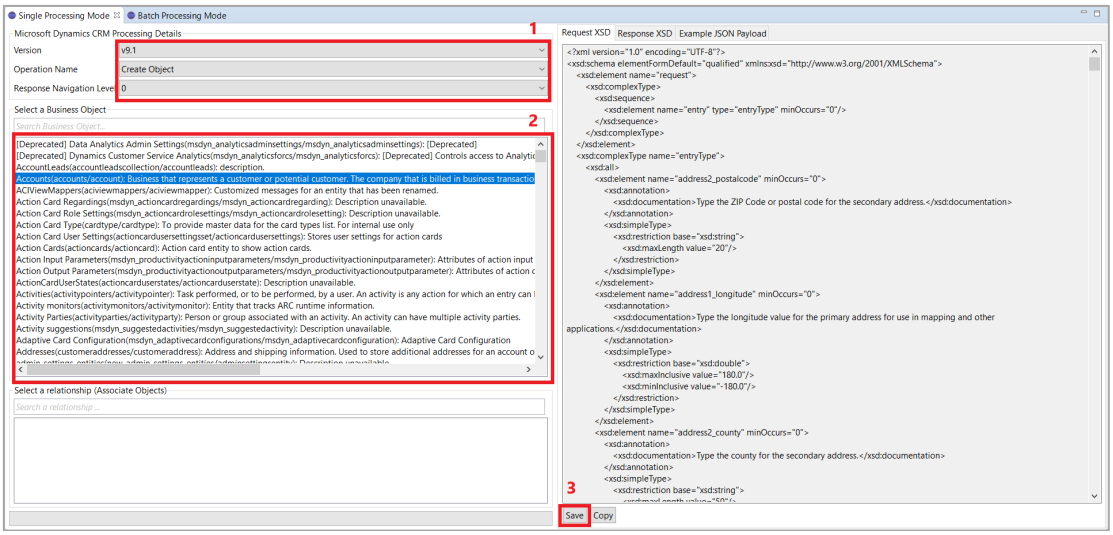



*Figure 12.3 Microsoft Dynamics CRM Plug-In*

First, as shown in step 1 in Figure 12.3, select the Version and the Operation Name (Create). Then, select the concerned Business Object (Accounts), see step 2 in Figure 12.3. After selecting the Account Business Object, the Request and response XSDs will be populated on the right side of the screen. Save these XSDs on the local file system as shown in step 3 in Figure 12.3.

Use both schemas for the mapping in Cloud Integration. This concerns the mapping step **S/4HANA to MSDynamicsCRM** in the iFlow shown inFigure 12.1.

After mapping, configure the Microsoft Dynamics CRM Adapter. Figure 12.4 shows an example of the configuration of the Adapter's Connection Tab.

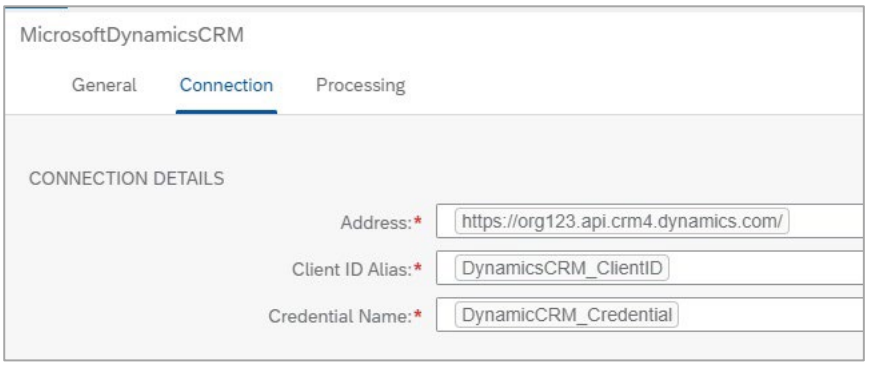



*Figure 12.4 Connection Tab, Microsoft Dynamics CRM Adapter*

In the Connection Tab shown in Figure 12.4, provide the Microsoft Dynamics CRM details. Credentials previously deployed for Client ID 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 in the Connection tab is explained in Section 8.1.

After the Connection Tab is configured, proceed to the Process Tab. Figure 12.5 shows the necessary configuration details to achieve the integration scenario.

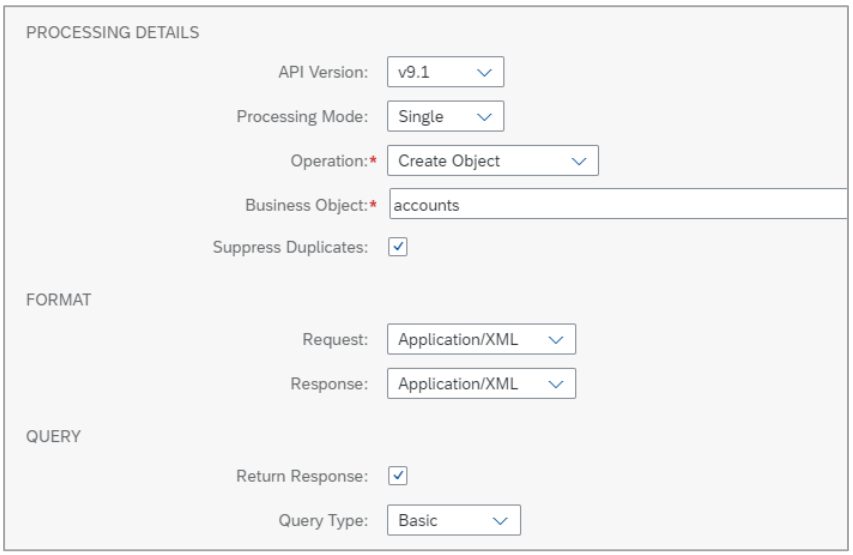



*Figure 12.5 Processing Tab, Microsoft Dynamics CRM Adapter*

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). For simplicity, select Single as the Processing Mode.

Note that Batch Processing Mode can also be chosen. But, in that case, the Business Partners retrieved from SAP S/4HANA should not be split, they would all be replicated in the batch process.

Select Create Object as Operation and specify the Business Object as Accounts. The example uses XML as the format for both request and response. Select the Return Response checkbox and Basic as the Query Type.

With all the necessary configurations completed, the integration flow can be deployed and then called.

Figure 12.6 shows an example of the response returned by SAP S/4HANA using the OData adapter.

```
<A_BusinessPartner>
    <A_BusinessPartnerType>
        <CreationDate>2020-10-27T00:00:00.000</CreationDate>
        <CreatedByUser>MICHAELM</CreatedByUser>
        <Customer>1000151</Customer>
        <BusinessPartner>1000151</BusinessPartner>
        <BusinessPartnerGrouping>BP02</BusinessPartnerGrouping>
        <LastChangedByUser>MICHAELM</LastChangedByUser>
        <BusinessPartnerUUID>c81f66e8-4df0-1edb-868d-7355f7654fa4</BusinessPartnerUUID>
        <CreationTime>1970-01-01T15:47:20.000</CreationTime>
        <to_BusinessPartnerAddress>
            <A_BusinessPartnerAddressType>
                <StreetName></StreetName>
                <BusinessPartner>1000151</BusinessPartner>
                <PostalCode>08024</PostalCode>
                <AddressUUID>c81f66e8-4df0-1edb-868d-74672503efa5</AddressUUID>
                <CityName>Barcelona</CityName>
                <Country>ES</Country>
                <Region>08</Region>
                <ValidityStartDate>2020-10-27T00:00:00.000</ValidityStartDate>
            </A_BusinessPartnerAddressType>
        </to_BusinessPartnerAddress>
        <BusinessPartnerCategory>2</BusinessPartnerCategory>
        <LastChangeTime>1970-01-01T15:49:29.000</LastChangeTime>
        <GroupBusinessPartnerName1></GroupBusinessPartnerName1>
        <GroupBusinessPartnerName2></GroupBusinessPartnerName2>
        <LastChangeDate>2020-10-27T00:00:00.000</LastChangeDate>
    </A_BusinessPartnerType>
</A_BusinessPartner>
```

*Figure 12.6 Input XML from SAP S/4HANA*

Below Figure 12.7 shows an example payload for the account entity used and sent as a request to the Adapter to create the object in Microsoft Dynamics CRM Adapter.

```
<request>
    <entry>
        <address1_country>2020-10-27T00:00:00.000</address1_country>
        <address1_postofficebox>10</address1_postofficebox>
        <telephone1>600000000</telephone1>
        <address1_stateorprovince>08</address1_stateorprovince>
        <address1_city>Barna</address1_city>
        <address1_line1>calle</address1_line1>
        <name>Group S4HPremise3</name>
        <address1_postalcode>08024</address1_postalcode>
        <to_BusinessPartnerAddress>
    </entry>
</request>
```

*Figure 12.7 Output XML to Microsoft Dynamics CRM*

As a result, a new account is created in Microsoft Dynamics CRM with the information extracted from SAP S/4HANA.

## 13. REFERENCES

This section includes reference guides needed for the configuration requirements of the Adapter and for Microsoft Dynamics CRM to connect to SAP Cloud Integration.

### 13.1 Finding Dynamics CRM Address

The address required in the Address field of the Connection Tab is retrieved in the Microsoft Dynamics CRM tenant. Note that the following sample screens may look slightly different, depending on the version of the tenant.

Proceed as follows:

1. Log in to the Microsoft Dynamics CRM tenant and navigate to **Settings** and **Customization**.

   See Figure 13.1 and Figure 13.2.

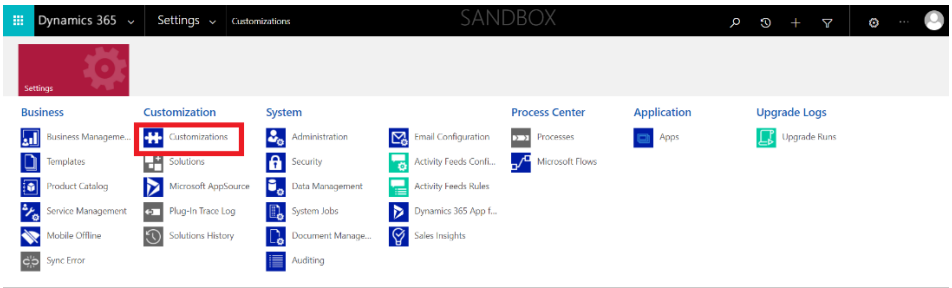



*Figure 13.1 Landing Page of Microsoft Dynamics CRM, Settings Option Highlighted*

2. Click on the **Developer Resources** link as shown in Figure . This opens a new screen.

*Figure 13.2 Customizations Settings View, Microsoft Dynamics CRM*

3. Retrieve the Address from the **Service Root URL** field framed in red below.

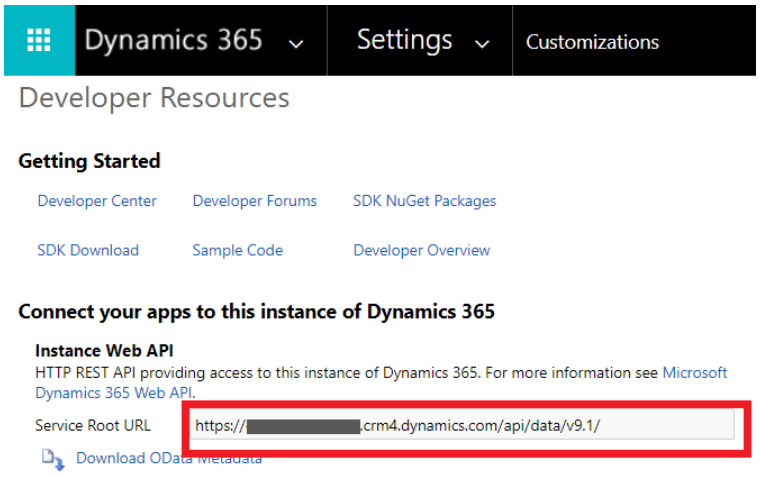



*Figure 13.3 Developer Resources Link*

## 13.2 Microsoft Dynamics CRM Configuration

To connect SAP Cloud Integration to Microsoft Dynamics CRM, it is required to register an app in Microsoft Azure for SAP Cloud Integration.

Follow the steps below to register an application in Microsoft Azure:

1. Access the Microsoft Azure page using an account with an administrative role. This can be accessed using the link: http://portal.azure.com

2. Select **Azure Active Directory**.

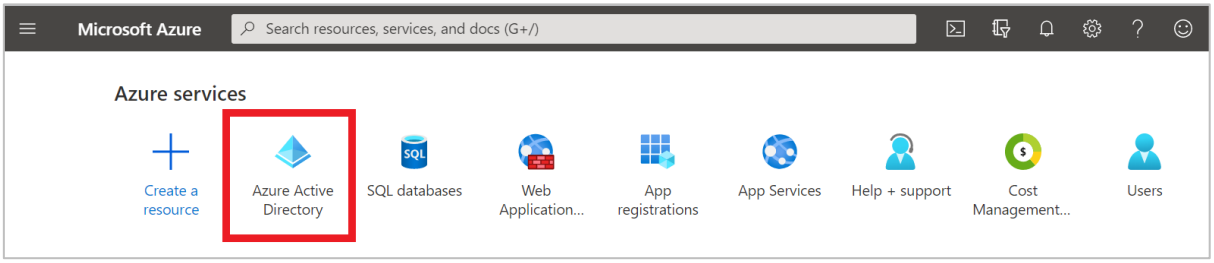



*Figure 13.4 Azure Services, Azure Active Directory Selected*

3. Click on **App registrations**.

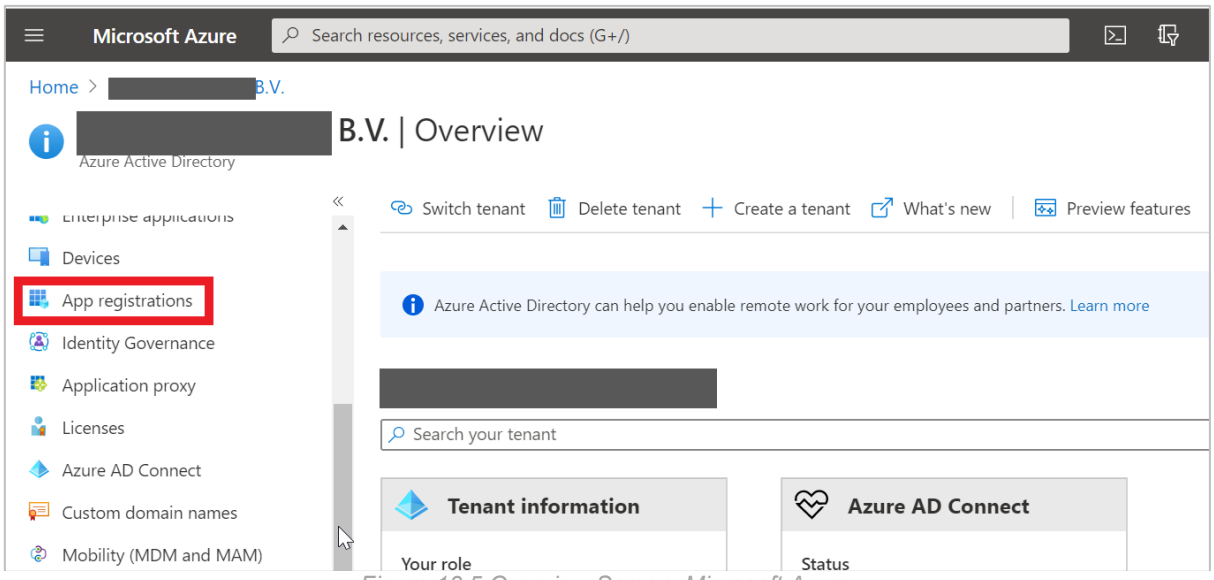



*Figure 13.5 Overview Screen, Microsoft Azure*

4. Select **New registration**.

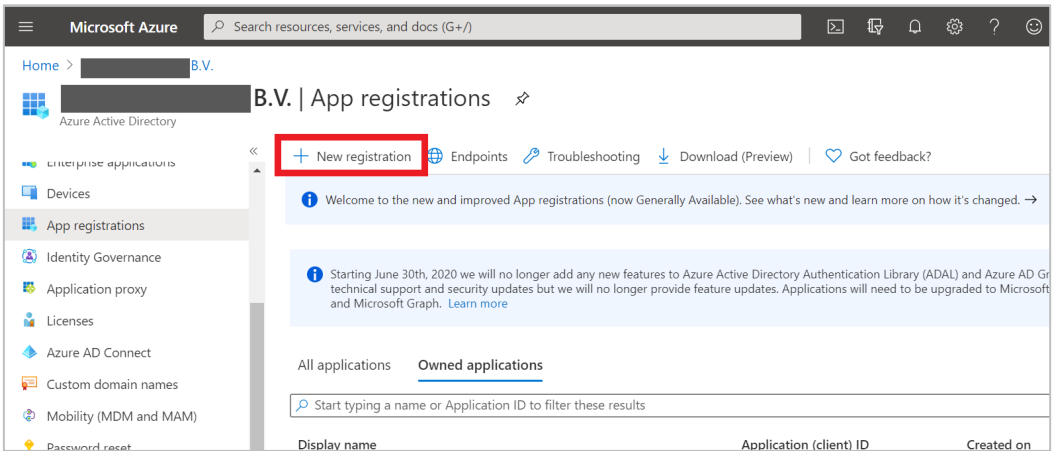



*Figure 13.6 App Registration Screen, Microsoft Azure*

5. Fill in the Name and select an ***account***. Fill in the Web Type and **Register**.

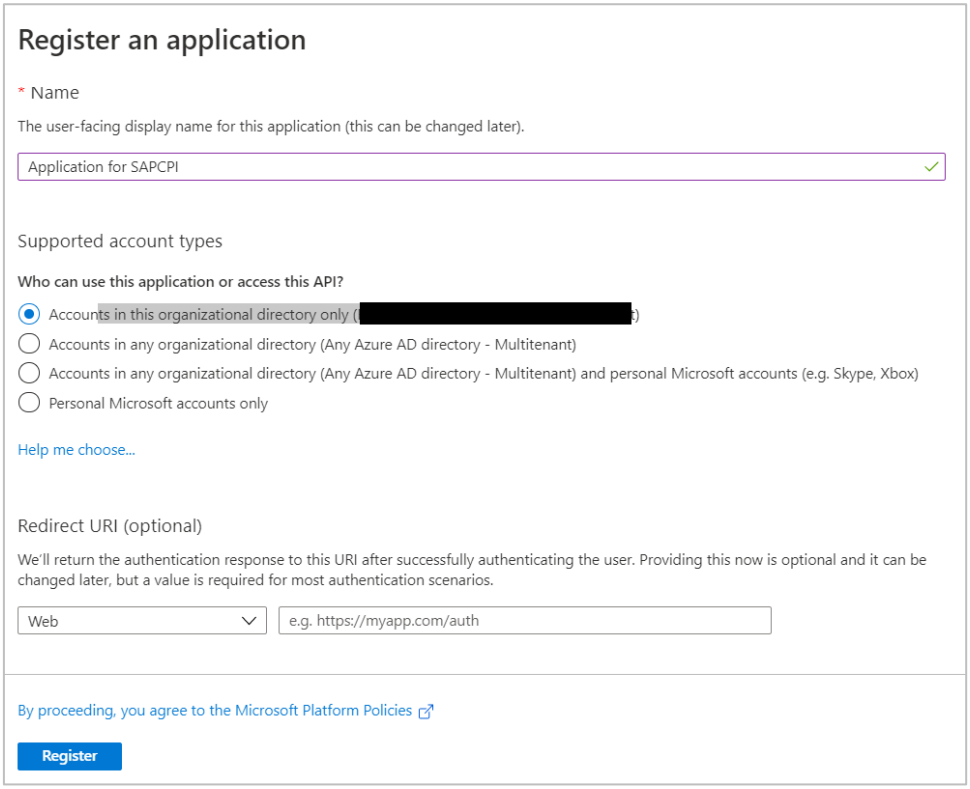



*Figure 13.7 Register an Application Screen, Microsoft Azure*

6. Select **API Permissions** on the left menu to add permissions.

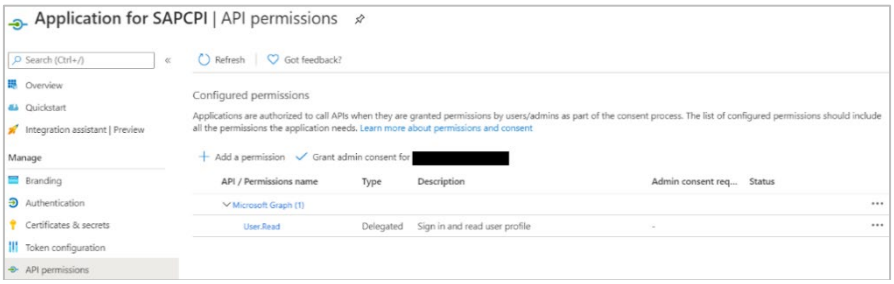



*Figure 13.8 API Permissions Screen, Microsoft Azure*

7. Select **Dynamics CRM**.

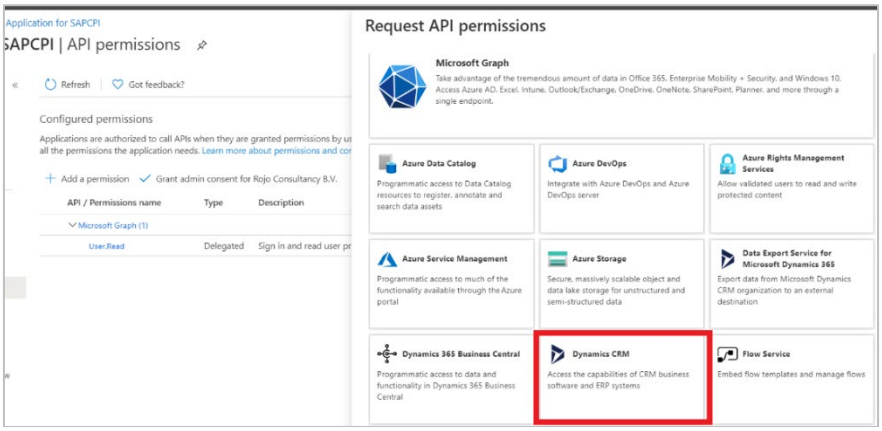



*Figure 13.9 Request API permissions Screen, Microsoft Azure*

8. Assign and **Add permissions**. In the example below the user_impersonation permission has been added.

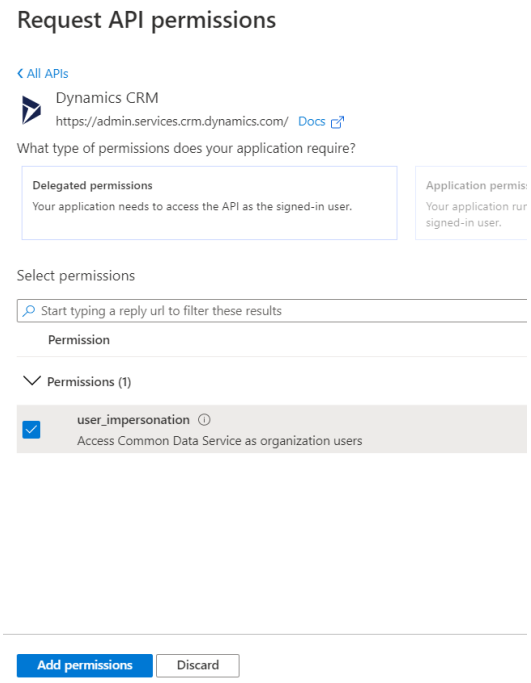



*Figure 13.10 Select Permissions Screen, Microsoft Azure*

9. Grant consent.

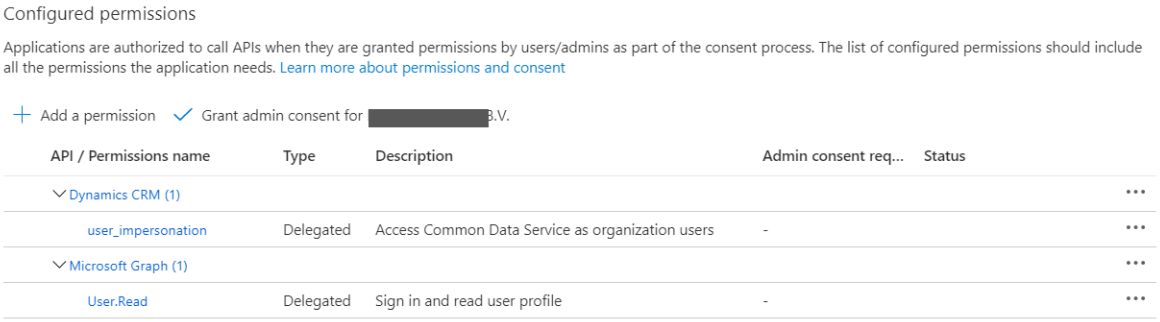



*Figure 13.11 Configured Permissions Screen, Microsoft Azure*

10. Set the “Enable public client flows” to Yes.

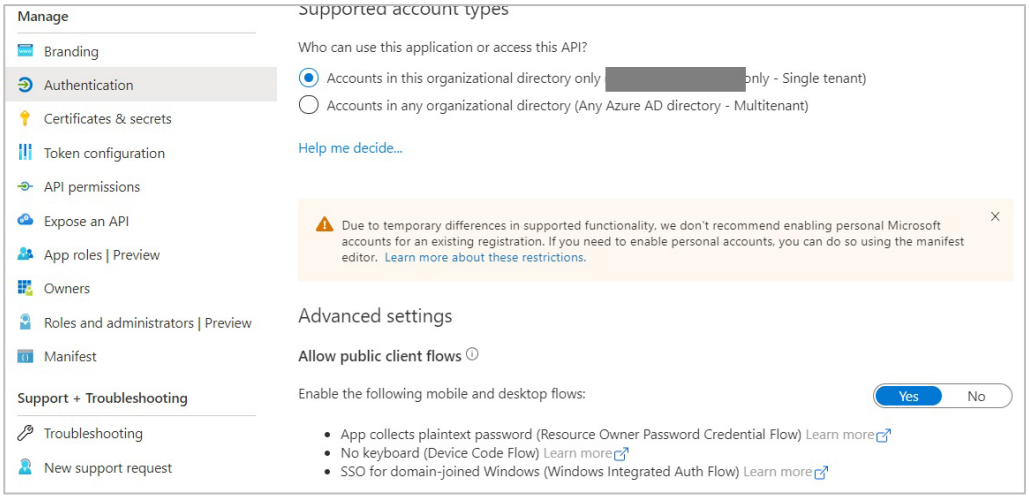



*Figure 13.12 Allow Public Client Flows Screen, Microsoft Azure*

Note: If the “Enable public client flows” is set to Yes, there is no need to set the Client Secret in the Adapter. In case the “Enable public client flows” is set to No, then it is important to set the Client Secret in the Adapter.

11. Copy the value of the Application (client) ID. This value will be later needed in the Adapter and be used as the Client ID. See Figure 13.13.

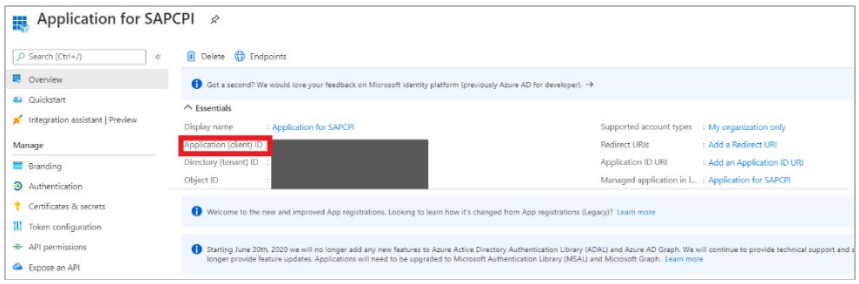
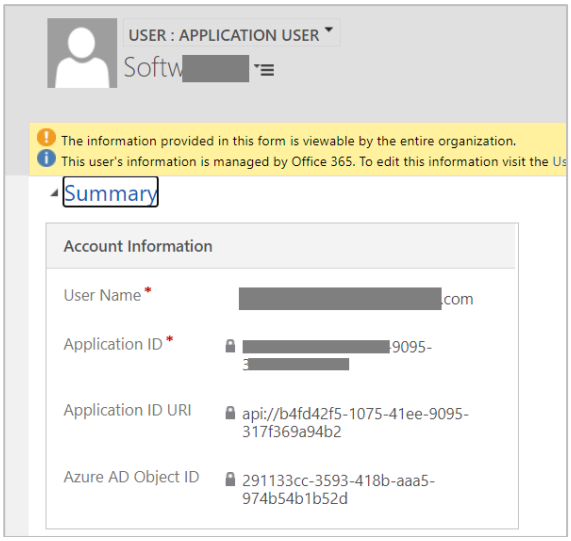



*Figure 13.13 Client ID Screen, Microsoft Azure*

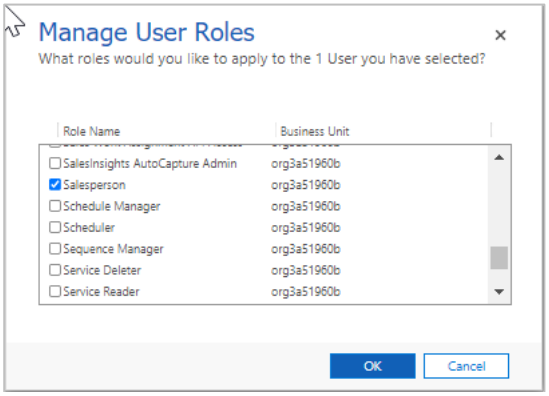
In case “OAuth – Client Credentials Secret” is used, the Directory (tenant) ID shown in Figure 13.13 below the Application (client) ID should also be copied. Furthermore, it will also be necessary to create a new Client Secret under the “Certificates & Secrets” section. See Figure 13.14.


*Figure 13.14 Creating a Client Secret*

**Note**: In Microsoft Dynamics, a user needs to be created which should be linked to the Application (client) ID. It is necessary to add the “Application (client) ID” in the Application ID of the user. See example set up below.




This user in Dynamics needs to have all the relevant roles needed to perform the required operations in the adapter. As an example, to be able to access accounts data, the user will need the “Sales” role.




### 13.3 Using Alternatives Keys with the Adapter

Alternate keys are useful features in Dynamics CRM, especially when integrating with other third-party systems. This enables the use of keys from external systems instead of using keys generated in the Microsoft Dynamics CRM system to identify records. For some integration use cases, it is easier to use alternate keys to efficiently retrieve and update records.

For this to work, ensure that the alternative key is first available and maintained in the entity. As shown in Figure 13.15, an alternative key named “new_alternative_key” was created for the entity Account.

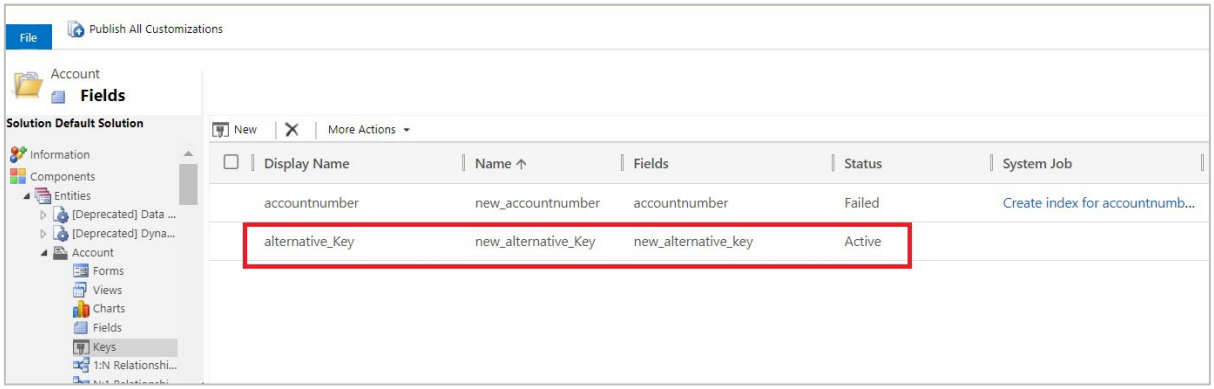



*Figure 13.15 Alternative Keys*

Ensure that the alternative key has been indexed in Microsoft Dynamics CRM – in which case the Status field will have the status Active.

For illustration purposes, assume that an alternative key is added to the Account entity and that another entity references the Account entity (as a lookup). It becomes possible to use the “new_alternative_key” instead of the Account primary key to reference it. Below an example of JSON and XML payloads using an alternative key to reference the Account entity are shown.

**JSON Example:**

```
{
  "request": {
    "entry": {
      "new_multipleline": "test_string",
```

```
    "new_testcustomer_account@odata.bind": "/accounts(new_alternative_key='123457')"
  }
 }
}
```

**XML Example:**

```
<request>
        <entry>
                <new_testcustomer_customer>new_alternative_key=&amp;apos;123457&amp;apos;
                </new_testcustomer_customer>
                <new_sales_groupid_lookupcustomertype="accounts">b794ad5f-ec0d-eb11-a813-
                000d3a23cc7b</new_sales_groupid_lookup>

        </entry>
</request>
```

Note from the above example that the string “='” used in the field “new_alternative_key” represents a single quote. Furthermore, the Eclipse Workbench can be used to generate the required XSD.

Alternatively, the customertype attribute can be added to specify that the alternative keys used are for the entity Account.

```
<new_testcustomer_customer
customertype="accounts">new_alternative_key=&amp;apos;123457&amp;apos;</new_testcustomer_customer>
```

In case the same property needs to reference the Contact instead of the Account entity, the following line should be used instead:

```
<new_testcustomer_customer
customertype="contacts">new_alternative_key=&amp;apos;123&amp;apos;</new_testcustomer_customer>
```

Note that if the customertype attribute is not specified, the Account entity is used as default.

## 14. SUPPORT AND TROUBLESHOOTING

In case of issues or errors, change the Log level of the 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 14.1.

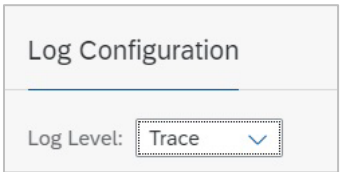



*Figure 14.1 Activating Traces, SAP Cloud Integration Suite*

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 a few issues that might be encountered and possible solutions for such.

### 14.1 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 Cloud Integration, an entry for the root certificate of Microsoft Dynamics CRM is missing. At the time this document is published, Microsoft Dynamics CRM expects a DigiCert Baltimore Root certificate.

### 14.2 UnknownHostException

Error Message:

> *java.net.UnknownHostException: xxxxxxx.crm4.dynamics.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 13.1 for instructions on how to find the Microsoft Dynamics CRM Address.

### 14.3 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 DynamicsCRM_ClientID*

Possible Solution:

This error generally indicates that one of the security artifacts used in the Connection Tab in the Adapter does not exist. Ensure that the artifact used in the Connection Tab exists as a Security Material. Refer to Section 7.

### 14.4 Could Not Find a Property Named

Error Message:

> *org.apache.camel.CamelException:org.apache.camel.CamelException: {"error":{"code":"0x0","message":"Could not find a property named 'Name' on type 'Microsoft.Dynamics.CRM.incident'."}}*

Possible Solution:

This error generally indicates that one of the entries in the “Response Properties” field of the Adapter does not exist as a property of the indicated Business Object.

### 14.5 Illegal Character in Opaque Part at Index-xxx

Error Message:

> *[CAMEL][IFLOW][CAUSE]: Cause: java.net.URISyntaxException: Illegal character in opaque part at index 212.*

Possible Solution:

This error generally indicates that the FetchXML query is wrongly formatted. Refer to Section 8.2.1.1.3 for the right format type. The Plug-In can help format a FetchXML, refer to Section 6.3. Possible solutions include removing all the line returns and spaces between the different queries within the FetchXML query.

### 14.6 There Are 2 Parameters That Couldn’t Be Set on the Endpoint

Error Message:

> *[CAMEL][IFLOW][CAUSE]: Cause: There are 2 parameters that couldn't be set on the endpoint. Check the uri if the parameters are spelt correctly and that they are properties of the endpoint. Unknown parameters=[{$filter=name eq 'test', $orderby=name}]*

Possible Solution:

This error generally indicates that the Advanced Query format is wrong. Refer to Section 8.2.1.1.2 for the right format for this specific query. The Plug-In can help format the Query, refer to Section 6.3. Possible solutions include replacing the ‘&’ sign to connect queries with ‘_and_’, or putting the query inside of the following text: RAW({advanced query}).

### 14.7 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 indicates that the Adapter is expecting a JSON input but is getting a non-valid JSON or even another format such as XML. The Plug-In can help format the Query, refer to Section 6.3.

### 14.8 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 indicates that the Adapter is expecting an XML input but is getting the wrong XML or even another language such as JSON. The Plug-In can help format the Query, refer to Section 6.3.

### 14.9 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 indicates that a mandatory field has not been mapped or is missing in the payload sent to Microsoft Dynamics CRM for the creation or updating of an entity.

### 14.10 Could Not Find a Property Name

Error Message:

*org.apache.camel.CamelException: org.apache.camel.CamelException:*
*{"error":{"code":"0x0","message":"Could not find a property named 'housingsituation' on type 'Microsoft.Dynamics.CRM.account'."}}*

Possible Solution:

This error usually indicates that a property specified in the Adapter setting does not match any known property for the relevant entity type. This property may have been used in **Filter**, **Response Properties**, **Order Response By**, **Expand Navigation Properties,** and **Aggregate** Properties in the Adapter.

### 14.11 "Bad Request - Error in query syntax" When Using Alternative Keys

Error Message:

*An internal server error occured: org.apache.camel.CamelException:*
*{"error":{"code":"0x0","message":"Bad Request - Error in query syntax."}}*

Possible Solution:

When using alternative keys, this error usually indicates that a property specified in the payload as an alternative key might not be an alternative key for the concerned entity. It might also mean that the property is a key but might not be unique or the property is not indexed.

### 14.12 Error: "invalid_client", AADSTS7000218: The request body must contain the following parameter: 'client_assertion' or 'client_secret'

Error Message:

*org.apache.camel.CamelException: {"error":"invalid_client","error_description":"AADSTS7000218: The request body must contain the following parameter: 'client_assertion' or 'client_secret'*

Possible Solution: This problem is generally caused when “OAuth – Password Credentials” authentication without setting a value for the Client Secret Alias is used.

In case “OAuth – Password Credentials” as authentication without a Client Secret is preferred, ensure that the following settings are applied in the Azure App:

1. From the Azure Portal (https://portal.azure.com), click on "App Registrations".
2. Click on the app that was previously registered and select the "Authentication" tab.

3. At the bottom of the screen (in the Advanced settings), please ensure that the "Allow public client flows" is set to yes.
4. Then click on the Save button.

The result should look as below.


In case the "Allow public client flows" is set to No, note that it is important to assign a value to the Client Secret Alias property in the Adapter.

### 14.13 Invalid_grant - AADSTS50126: Error validating credentials due to invalid username or password

Error Message:

> *org.apache.camel.CamelException: {"error":"invalid_grant","error_description":"AADSTS50126: Error validating credentials due to invalid username or password.*

Possible Solution:

This error usually indicates that the account used requires **Federated authentication**. Federated authentication does not necessarily mean that ADFS (Active Directory Federation Service) is used. Any third-party IDPs such as Auth0, OneLogin, etc. can be selected.

Consider one of the following approaches to solve this error:

1. Create a new account with UPN like *username@your_tenant.onmicrosoft.com* to be certain that federated authentication is not used. This account can then be used in the Adapter with the authentication type “OAuth – Password Credentials”.

2. Use the authentication type “OAuth – Client Credentials Secret”.

   This authentication type does not require a username/password, but instead a client ID and Client Secret. Note that this approach also requires the creation of an application user in Microsoft Dynamics CRM based on opening the Client ID. This authentication approach was discussed in Section 8.1.

**www.sap.com/contactsap**



THE BEST RUN SAP