Salesforce Adapter Guide

For SAP Integration Suite and SAP Cloud Integration

Version 1.2.0 – April 2026

THE BEST RUN

Table of Contents

1 INTRODUCTION

This is a guide for the Salesforce Adapter for SAP Cloud Integration. This guide covers all relevant information for integration developers to start working with the Salesforce Adapter. Read this guide carefully before using the Salesforce 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 assured.

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 the correctness of these links to serve a particular purpose are not warranted.

2 SALESFORCE INTEGRATION

2.1 Introduction

Salesforce is a SaaS Cloud solution that covers four main services; Marketing, Sales, Orders, and Support. Salesforce Cloud supports sales, marketing, and customer support in both B2B and B2C contexts. It helps track customer information and interactions in one place, automates complex business processes, keeps all information up to date, nurtures leads, and tracks the effectiveness of marketing campaigns.

For most organizations, it is required to exchange different types of data between Salesforce and any other systems while streamlining processes. This Adapter allows the exchange of data between Salesforce CRM (Customer Relationship Management) and these other systems. For illustration purposes, some business scenarios that the Adapter can be used for including:

- Synchronizing transactional and master data.
- Consolidating data for CRM migration.
- Retrieving data from reports and dashboards.
- Managing order flow to Salesforce from SAP and non-SAP backend applications.
- Processing batches for large data sets.
- Order fulfillment and campaign tracking in Marketing & Sales.
- Aggregating data for compliance or auditing purposes.
- Invoice creation in SAP when opportunities are successful.

In the next sections, the available options and the Adapter features are discussed.

2.2 Salesforce Adapter

From a high-level perspective, the Salesforce Adapter has the following key features:

- *Support for multiple API versions*: Salesforce releases a few versions of its API every year. The Adapter, as a standard, supports up to Salesforce API version 52.0.
- *Secure authentication with OAuth 2.0:* Every REST call between Salesforce and SAP Cloud Integration is secured by OAuth 2.0. Authentication mechanisms include OAuth 2.0 Client Credentials, OAuth 2.0 Username-Password and OAuth 2.0 JWT Bearer.
- *Support for multiple Salesforce operations*.
- *Querying and searching:* In addition to all the operations that the Salesforce Adapter offers, the Adapter allows the querying of data from a Salesforce system using SOQL and SOSL queries.

- *Dynamic configuration with headers and properties:* The Salesforce Adapter provides the freedom to assign dynamic values to its different properties. It allows the referring to dynamic parameters using SAP Cloud Integration exchange headers and properties.
- *Processing large sets of data:* With support for the Bulk operation, the Salesforce Adapter is optimized for loading or deleting large sets of data. Using Batches, numerous Records can be processed asynchronously.
- *Support for XML & JSON:* XML and JSON are fully incorporated in the Adapter, while the REST Bulk API additionally supports CSV and ZIP files. Furthermore, the Salesforce Adapter also processes binary content from a Salesforce system. For example, the REST – Get Blob operation supports Documents and Attachment objects.
- *Pretty print:* It is possible to “pretty print” or format XML and JSON payloads returned by Salesforce to improve legibility.
- *Query and XSD Eclipse plug-in:* The Eclipse Query plug-in enables the creation of SOQL and SOSL queries. The Eclipse XSD plug-in helps in generating all necessary up-to-date messages XSDs for the specific Salesforce version and Custom Objects.
- *Full integration support for Custom Objects and Fields:* In addition to supporting all the standard Salesforce Objects, the Adapter integrates with all the Custom Objects in Salesforce.
- *Support for creation/upsert of Aggregated Structures:* The Salesforce Adapter supports a composite Salesforce call where the request body can contain a combination of different operations (such as Upsert and Read) on multiple sObjects at the same time.
- *Support nested structures in the CRUD operations:* This feature supports the references that link nested sObjects to a main sObject in the XSD Generation plug-in. This allows the mapping of referenced sObjects in the structure. For example, the Account XSD contains the referenced subobjects Account, ReportsTo, and Owner.
- Option to persist replay ID together with subscription on single/multiple run time nodes.

2.3 Architecture Overview

From a technical perspective, the Salesforce Adapter can be used both as a Sender and as a Receiver Adapter. The Sender Adapter subscribes to Salesforce to retrieve events. As a Receiver Adapter, SAP Cloud Integration acts as the initiator of the calls. To schedule calls toward Salesforce, the User can follow the Scheduler step within the integration flow.

Figure 2.1 shows how the Salesforce Adapter can be used in a simple Request-Reply scenario and gives a high-level representation of how the Adapter works. Various operations can be used in the Adapter – such as Query Data, Delete, Update, Upsert, or Create a sObject, etc. The different adapter operations and their configuration are explained in Section 5 for the Sender Adapter and Section 6 for the Receiver Adapter.

The Request-Reply Step can use the Salesforce Adapter as shown in Figure 2.1. This Adapter is responsible for connecting and interacting with Salesforce and invoking its different operations.

The Eclipse Plug-in, which is delivered together with the Adapter, can also be used to help the User generate the XSDs of the different entities or business objects that are part of the Salesforce tenant.

The XSDs generated by the Eclipse Plug-in can be imported in an Integration Flow and used in mappings. For instance, it can be utilized in the mapping in front of the Request-Reply Step. This way, the correct XML request message to create or update a sObject in Salesforce (see the Mapping Step on the left side of the Request-Reply Step) can be designed. Similarly, the XSDs can be used to create a mapping to handle the response message sent by Salesforce.

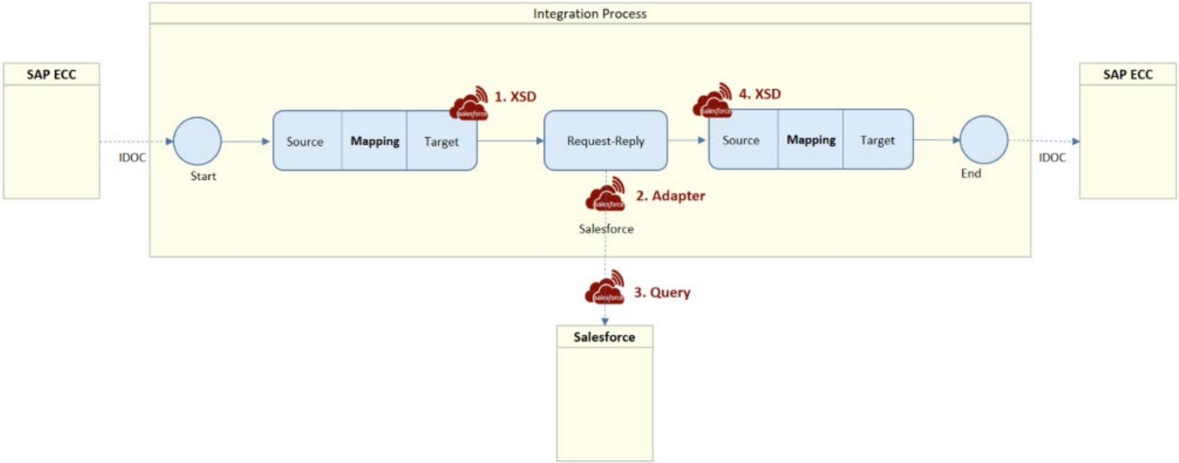
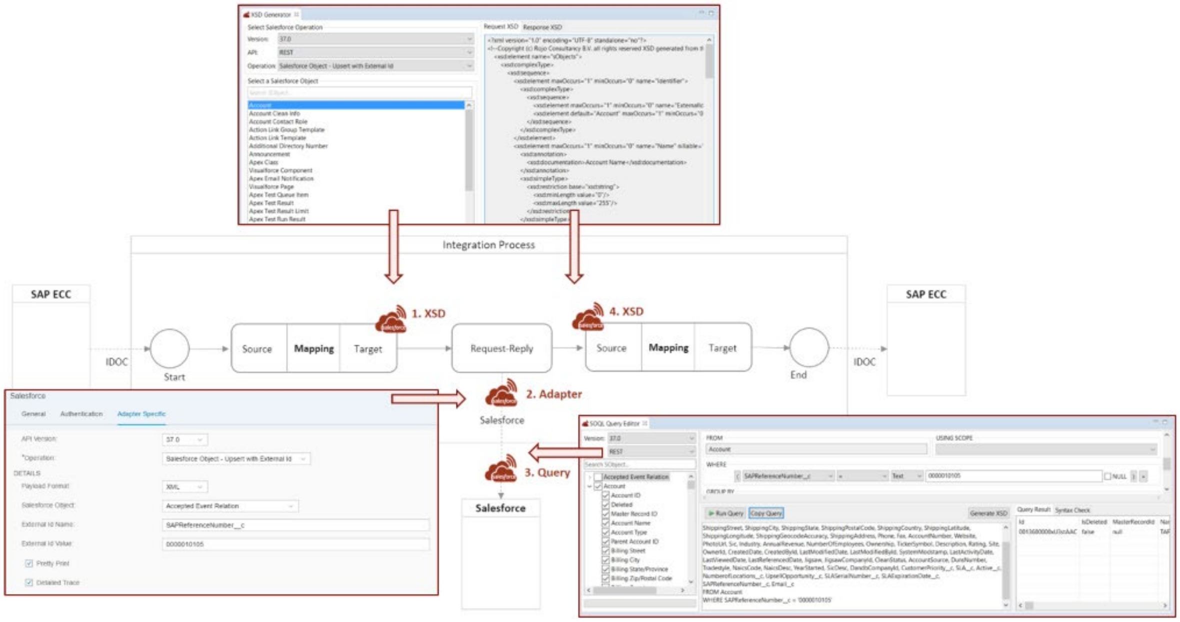


Figure 2.1 Salesforce Adapter solutions for SAP Cloud Integration

Furthermore, the Eclipse Plug-in has the following features:

- The XSD Generator helps in the generation of up-to-date XSDs for Salesforce sObjects for different operations (Create, Delete, Update, etc.). This makes mappings to other systems' message types a straightforward step. For the Salesforce plug-in and XSD Generator configuration please refer to the Salesforce Adapter and Plug-in Installation Guide.
- The Query Editor assists in creating valid SOQL or SOSL queries. With these queries, the best out of the REST, and REST Bulk API. To know more about the Salesforce Eclipse plug-in, refer to the document Salesforce Adapter and Plug-in Installation Guide.

Figure 2.2 shows at which stage of the SAP Cloud integration flow, the Salesforce Adapter can be used and where the XSD Generator and the SOSL or SOQL Query Editor are utilized.



Version: 37.0
API: REST
Operation: Salesforce Object - Upsert with External Id

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!--Copyright (c) Rojo Consultancy B.V. all rights reserved XSD generated from tl
  <xsd:element name="sObjects">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element maxOccurs="1" minOccurs="0" name="Identifier">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element maxOccurs="1" minOccurs="0" name="Externalic
              <xsd:element default="Account" maxOccurs="1" minOccurs="0
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element maxOccurs="1" minOccurs="0" name="Name" nillable="
          <xsd:annotation>
            <xsd:documentation>Account Name</xsd:documentation>
          </xsd:annotation>
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:minLength value="0"/>
              <xsd:maxLength value="255"/>
            </xsd:restriction>
          </xsd:simpleType>
```

API Version: 37.0
*Operation: Salesforce Object - Upsert with External Id
DETAILS
Payload Format: XML
Salesforce Object: Accepted Event Relation
External Id Name: SAPReferenceNumber__c
External Id Value: 0000010105
☑ Pretty Print
☑ Detailed Trace

Version: 37.0
REST
FROM Account
WHERE (SAPReferenceNumber__c = Text 0000010105 □ NULL)

```
ShippingStreet, ShippingCity, ShippingState, ShippingPostalCode, ShippingCountry, ShippingLatitude,
ShippingLongitude, ShippingGeocodeAccuracy, ShippingAddress, Phone, Fax, AccountNumber, Website,
PhotoUrl, Sic, Industry, AnnualRevenue, NumberOfEmployees, Ownership, TickerSymbol, Description, Rating, Site,
OwnerId, CreatedDate, CreatedById, LastModifiedDate, LastModifiedById, SystemModstamp, LastActivityDate,
LastViewedDate, LastReferencedDate, Jigsaw, JigsawCompanyId, CleanStatus, AccountSource, DunsNumber,
Tradestyle, NaicsCode, NaicsDesc, YearStarted, SicDesc, DandbCompanyId, CustomerPriority__c, SLA__c, Active__c,
NumberofLocations__c, UpsellOpportunity__c, SLASerialNumber__c, SLAExpirationDate__c,
SAPReferenceNumber__c, Email__c
FROM Account
WHERE SAPReferenceNumber__c = '0000010105'
```

Id	IsDeleted	MasterRecordId	Na
0013600000xU3oAAC	false	null	TAF

Figure 2.2 The Salesforce Adapter, Eclipse Plug-in

3 SUPPORTED OPERATIONS AND POST-PROCESSING ACTIVITIES

The Adapter supports the following main API categories:

- REST API
- REST Place Order API
- REST Bulk API
- REST Bulk 2.0 API
- Apex Call
- Event Processing

Brief descriptions of each one of these APIs are presented in the next sections.

3.1 REST API

The REST API is selected by default and allows integration with Salesforce using either XML or JSON formats. The integration is done securely by utilizing the OAuth protocol for authenticating REST calls. The REST API enables accessing the full data model of Salesforce and execution of CRUD operations and more. Note that these operations generally handle single sObject records as input.

However, if there are numerous records to process, the Bulk API (explained in detail in Section 3.2) is optimized for large sets of data. The following operations are currently supported in the Adapter's REST API Category:

- Metadata – All
- Metadata – Basic
- Metadata – Comprehensive
- Other – Get REST API Versions
- Other – Get REST Resource Endpoints
- Salesforce Object – Create
- Salesforce Object – Delete
- Salesforce Object – Delete with External ID
- Salesforce Object – Get
- Salesforce Object – Get Blob
- Salesforce Object - Get with External ID
- Salesforce Object – Update
- Salesforce Object – Upsert with External ID
- Salesforce Objects – Composite
- Salesforce Objects – sObject Collections
- SOQL – Execute Query
- SOQL – Execute Query for More Results
- SOSL – Search Resource
- Salesforce Objects – Custom Request
- Salesforce – List Organization Limits

To get a description of the purpose of each of the above operations and how to configure them, refer to Section 6.2.1.

3.2 REST Bulk API

The Bulk API is specifically designed to load or update large sets of data into a Salesforce organization. With this API, it is possible to query, insert, update, upsert, or delete many records asynchronously by submitting batches for Salesforce to process in the background. The following operations are currently supported in the Adapter:

- Batch – Create
- Batch – Get
- Batch – Get Request
- Batch – Get Results
- Batch Query – Create
- Batch Query – Get Result
- Batch Query – Get Result IDs
- Job – Abort
- Job – Close
- Job – Create
- Job – Get
- Job – Get All Batches

To get a description of the purpose of each of the above operations and how to configure them, refer to Section 6.2.3.

3.3 REST Bulk 2.0 API

The Bulk 2.0 API is specifically designed to load or update large sets of data into a Salesforce organization. The operation makes it easier to upload a large set of data asynchronously by automatically breaking data into batches. Note that this operation only accepts the upload of CSV files. Currently, the following operations are supported in the Adapter:

- Get – All Jobs
- Job – Abort
- Job – Close
- Job – Create
- Job – Get
- Job - Delete
- Job – Get All Batches
- Query – Abort Query Job
- Query – Create Query Job
- Query – Delete Query Job
- Query – Get All Query Jobs
- Query – Get Query Information
- Query – Get Query Results
- Upload – Job Data

To get a description of the purpose of each of the above operations and how to configure them, refer to Section 6.2.4.

3.4 REST Place Order API

The Place Order API allows the accessing of an organization’s Order and Contract data programmatically. This is a composite API that supports the creation of Contract, Order, Order Product, and Custom Object records in a single call. The following operations are currently supported in the Adapter:

- Account – Add Contracts and Orders
- Account – Add Orders
- Contract – Add Orders
- Contract – Filter Details
- Contract – Get
- Order – Add Products
- Order – Filter Details

- Order – Get

Each of the above operations is discussed in Section 6.2.2.

3.5 Event Processing

The Event processing feature enables Cloud Integration to subscribe (for the Sender Adapter) and poll (for the Receiver Adapter) Events from Salesforce. The following Event types are currently supported in the Adapter:

- PushTopic Events
- Platform Events
- Generic Events
- Change Data Capture

The Events supported by the Adapter are further discussed in Sections 5.2 and 6.2.5.

3.6 Apex Call

Using an Apex class or method, a custom class and its methods can be exposed to external applications. The Adapter can call APEX Classes and methods using this operation. The following Event types are currently supported in the Adapter:

- GET
- DELETE
- PATCH
- POST
- PUT

Apex Calls are further discussed in Section 6.2.6.

4 AUTHENTICATION

The Salesforce Adapter supports multiple OAuth 2.0 authentication methods. The Adapter makes use of common security artifacts in SAP Cloud Integration. It is required to use Secure Parameter and User Credentials to safely store OAuth tokens, user-password combinations, and secrets. These security artifacts can then be accessed in the Adapter using aliases.

The Salesforce Adapter uses the OAuth 2.0 Salesforce Autonomous Client type. The Adapter uses a Basic Authentication flow which requires a Salesforce username and password to connect to the Salesforce tenant data. In addition to the username and password, Salesforce app credentials are also required.

These app credentials consist of a Consumer Key (client_id) and Consumer Secret (client_secret). The diagram in Figure 4.1 explains the background process of connecting to Salesforce.

With the access token that is returned from the authorization server, the Salesforce Adapter can make different requests to Salesforce resources.

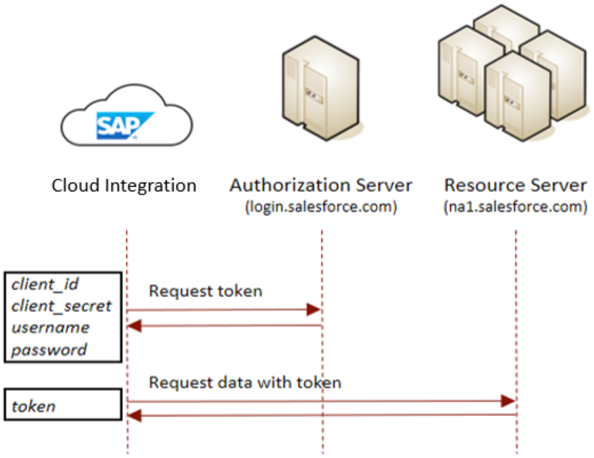


Figure 4.1 Salesforce Adapter authenticating using OAuth 2.0

Sections 5 and 6 discuss the Adapter’s configurations.

5 SENDER ADAPTER CONFIGURATION

This section describes the different parameters that can be configured for the Sender Adapter. The Sender Adapter enables SAP Cloud Integration to subscribe to and receive events that are generated in Salesforce. Events can be generated in Salesforce each time there is a change in the state of an Object. These events can be the trigger needed to perform an integration scenario. Such events can be used to synchronize data between multiple systems. Currently, the Adapter supports four types of events: PushTopic Events, Platform Events, Generic Events, and Change Data Capture.

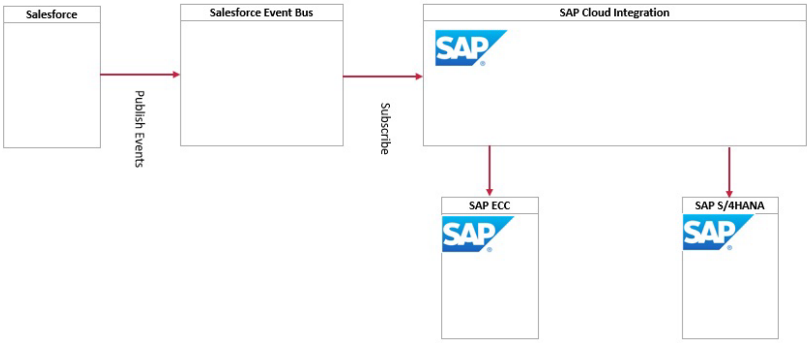


Figure 5.1 SAP Cloud Integration Subscription to Salesforce Events

The Sender Adapter has Connection and Processing tabs that require configuration. Each one of these is explained in the next sections.

5.1 Connection Tab

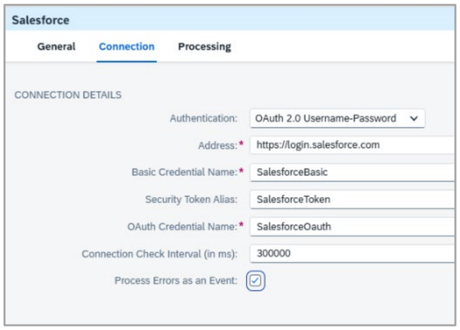
The Connection tab contains parameters that define how to connect to and how to authenticate against Salesforce. There are three authentication types:

- OAuth 2.0 Username-Password
- OAuth 2.0 JWT Bearer
- OAuth 2.0 Client Credentials

These authentication types are discussed in the next sections.

5.1.1 OAuth 2.0 Username-Password

For OAuth 2.0 Username-Password, a sample configuration is shown in Figure .



CONNECTION DETAILS

Authentication: OAuth 2.0 Username-Password

Address:* https://login.salesforce.com

Basic Credential Name:* SalesforceBasic

Security Token Alias: SalesforceToken

OAuth Credential Name:* SalesforceOauth

Connection Check Interval (in ms): 300000

Process Errors as an Event: ☑

Figure 5.2 Connection Tab of Salesforce Adapter, OAuth 2.0 Username Password

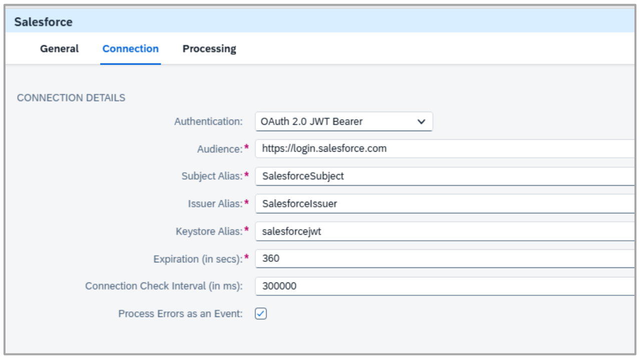
As seen in Figure 5.2, the Connection Tab contains the following fields:

- **Address***:* Specifies the recipient's endpoint URL. By default, the URL https://login.salesforce.com is used. This can be changed according to a scenario.
  - For Salesforce production environments: https://login.salesforce.com.
  - For Sandbox environments: https://test.salesforce.com. In case MyDomain is enabled in the organization, https://<MyDomain>.my.salesforce.com can be used.
- **Basic Credential Name***:* Specifies the name of the User Credentials artifact that contains the credentials for basic authentication. This refers to the username-password pair used to log in to Salesforce. This pair must be created as a Security Artifact of User Credential type and subsequently referred to in the Adapter.
- **Security Token Alias***:* Specifies the name of the Secure Parameter artifact that contains the security token needed to connect to Salesforce. This property enables the system to fetch the security token from Keystore for authentication. This field can be omitted if the IP has been whitelisted in Salesforce. To whitelist an IP address, add the IP range in the Network Access section in the Security Control menu in Salesforce.
- **OAuth Credential Name***:* Specifies the name of the User Credentials artifact that contains the Salesforce's OAuth Consumer Key-Consumer Secret pair. Configure the Consumer Key as **User** and the Consumer Secret as the **Password** in the Security Artifact of type **User Credential**. Then refer to it in the Adapter's Connection Tab.
- **Connection *Check Interval (in ms):*** Specifies the interval in milliseconds to verify the connection validity to Salesforce.
- **Process Errors as an Event:** Process Errors as an Event will create message exceptions in SAP Cloud Integration for each connection error during Salesforce Streaming. Errors that prevent SAP Cloud Integration from picking up Events via Streaming will be raised as a message exception in SAP Cloud Integration. An example exception message during streaming is "Organization concurrent user limit exceeded".

For more details on how to retrieve details of the Security Token and Consumer Key-Consumer Secret pair, refer to Section 12.1.

5.1.2 OAuth 2.0 JWT Bearer

For OAuth 2.0 JWT Bearer, a sample configuration is shown in Figure 5.3.



Salesforce

CONNECTION DETAILS

Authentication: OAuth 2.0 JWT Bearer

Audience:* https://login.salesforce.com

Subject Alias:* SalesforceSubject

Issuer Alias:* SalesforceIssuer

Keystore Alias:* salesforcejwt

Expiration (in secs):* 360

Connection Check Interval (in ms): 300000

Process Errors as an Event: ☑

Figure 5.3 Connection Tab of Salesforce Adapter with OAuth JWT Bearer

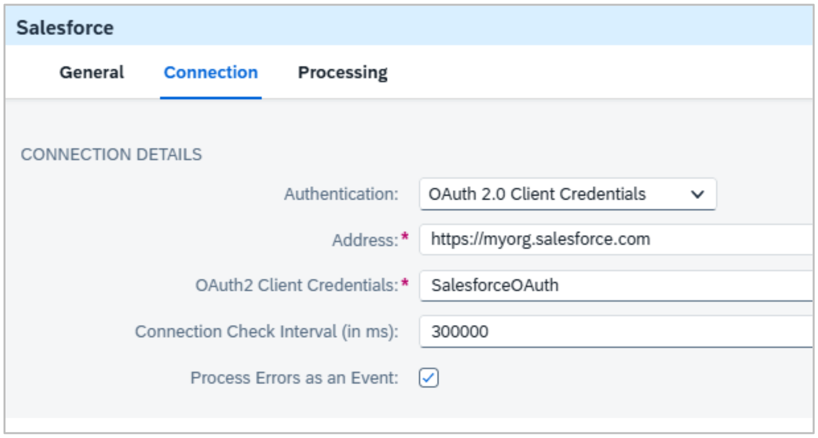
As seen in Figure 5.3, the Connection Tab contains the following fields:

- **Audience***:* Specifies the recipient's endpoint URL. By default, the URL https://login.salesforce.com is used. This can be changed based on a scenario.
  - For Salesforce production environments: https://login.salesforce.com.
  - For Sandbox environments, https://test.salesforce.com. In case MyDomain is enabled in the organization, https://<MyDomain>.my.salesforce.com may be used.
- **Subject Alias***:* The alias name of the deployed Secure Parameter artifact. It specifies the Salesforce username.
- **Issuer Alias***:* The alias name of the deployed Secure Parameter artifact. It specifies the OAuth Consumer Key of the connected app for which the certificate was registered.
- **Keystore Alias***:* The alias name of the added JKS file in Keystore as a Key Pair. It consists of a key and certificate to sign the JWT.
- **Expiration *(in secs)****:* Specifies the validity of the assertion in seconds.
- **Connection Check Interval (in ms):** Specifies the interval in milliseconds to verify the connection validity to Salesforce.
- **Process Errors as an Event:** Process Errors as an Event will create message exceptions in SAP Cloud Integration for each connection error during Salesforce Streaming. Errors that prevent SAP Cloud Integration from picking up events via Streaming will be raised as a message exception in SAP Cloud Integration. An example exception message during streaming is "Organization concurrent user limit exceeded".

For more details on how to retrieve the above details, refer to Section 12.2.

5.1.3 OAuth 2.0 Client Credentials

For OAuth 2.0 Client Credentials, a sample configuration is shown .



CONNECTION DETAILS

Authentication: OAuth 2.0 Client Credentials

Address:* https://myorg.salesforce.com

OAuth2 Client Credentials:* SalesforceOAuth

Connection Check Interval (in ms): 300000

Process Errors as an Event: ☑

Figure 5.5 Connection Tab of Salesforce Adapter with OAuth 2.0 Client Credentials

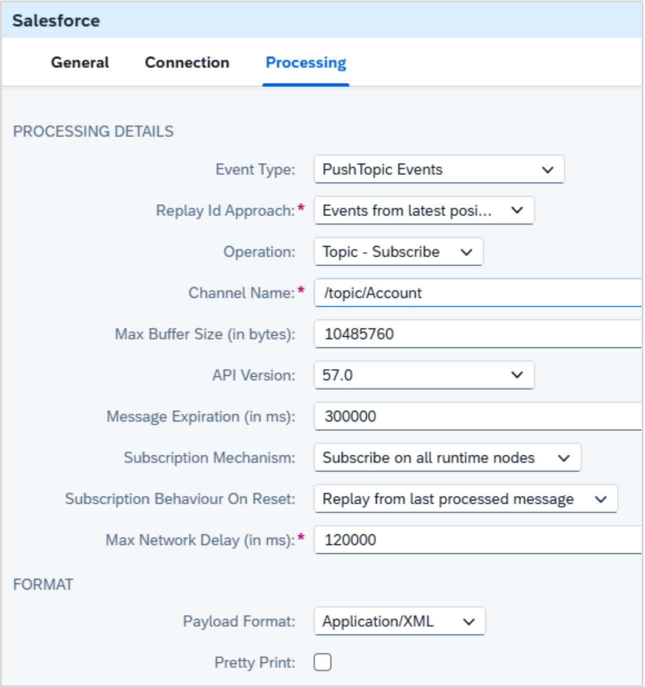
As seen in the Figure 5.5 , the Connection Tab contains the following fields:

- **Address***:* Specifies the recipient's endpoint URL. By default, the URL https://login.salesforce.com is used. This can be changed based on a scenario.
    - For Salesforce production environments: https://login.salesforce.com.
    - For Sandbox environments, https://test.salesforce.com. In case MyDomain is enabled in the organization, https://<MyDomain>.my.salesforce.com may be used.
- **OAuth2 Client Credentials***:* The alias name of the deployed OAuth2 Client Credentials artifact.
- **Connection Check Interval (in ms):** Specifies the interval in milliseconds to verify the connection validity to Salesforce.
- **Process Errors as an Event:** Process Errors as an Event will create message exceptions in SAP Cloud Integration for each connection error during Salesforce Streaming. Errors that prevent SAP Cloud Integration from picking up events via Streaming will be raised as a message exception in SAP Cloud Integration. An example exception message during streaming is "Organization concurrent user limit exceeded".

For more details on how to create an OAuth2 Client Credential, refer to Section 12.3.

5.2 Processing Tab

The Processing Tab contains all the operation configurations for the selected Salesforce API. An example configuration for the operation Salesforce Object – Create is shown in Figure 5.4.



Salesforce

PROCESSING DETAILS

Event Type:	PushTopic Events
Replay Id Approach:*	Events from latest posi...
Operation:	Topic - Subscribe
Channel Name:*	/topic/Account
Max Buffer Size (in bytes):	10485760
API Version:	57.0
Message Expiration (in ms):	300000
Subscription Mechanism:	Subscribe on all runtime nodes
Subscription Behaviour On Reset:	Replay from last processed message
Max Network Delay (in ms):*	120000

FORMAT

Payload Format:	Application/XML
Pretty Print:	□

Figure 5.4 Processing Tab of Salesforce Adapter, PushTopic Events

The different properties of the Adapter shown in Figure 5.4 are described below:

- **Event Type***:* Specifies the category of the event to be used for interacting with Salesforce. Possible values include PushTopic Events, Platform Events, Generic Events, and Change Data Capture Events. These different types of events are further explained in Section 5.2.1.
- **Replay Id Approach***:* Refers to the position of the Event in the event stream. The Replay Id is further explained in Section 5.2.2.
- **API Version***:* The API Version specifies the version of the Salesforce API. Since the 1st of June 2022 Salesforce API Versions 7.0 through 20.0 have been retired by Salesforce. These are therefore no longer part of the standard Salesforce API Version selection in the Adapter. Note that the Salesforce API version can be overwritten by manually entering the Salesforce API version.
- **Operation***:* Specifies the operation to perform towards Salesforce by choosing one of the provided operation options. Currently, only the Subscribe option is available. More options will be provided in the future.
- **Channel name**: Made of a Topic name preceded with a prefix. For instance, a PushTopic can be prefixed with “/topic/”. Example: /topic/MyPushTopic. Note that the names are case-sensitive.
- **Max Buffer Size (in bytes):** Specify the maximum permissible capacity (in bytes) for Buffer.
- **Replay Id:** The Replay Id is populated by Salesforce and refers to the position of the Event in the event stream. Note that the Replay Id values are not guaranteed to be continuous for consecutive Events. By specifying the value of the Replay Id, the Integration Flow will retrieve events that are within the retention window and that have followed the specified Replay Id.
  Example: In case Replay Id 6 is specified, the Integration Flow will receive all messages with a Replay Id greater than 6.
- **Payload Format:** This field specifies the format of the request message to be sent to and the response to be returned from Salesforce. Possible values include Application/XML and Application/JSON. Note that the default value is Application/XML.
- **Pretty Print:** Select the Pretty Print checkbox to nicely format the XML payload and improve its readability.
- **Message Expiration (in ms):** Specify the expiry duration for a message. The minimum acceptable value is 300000 ms.

- **Subscription Mechanism:** Select the subscription mode:
  - **Subscribe on all runtime nodes** creates a subscription for every runtime node in the tenant.
  - **Subscribe on one runtime node** creates a single subscription on one worker node, regardless of the number of runtime nodes.
- **Subscription Behaviour on Reset:** Select the subscription behavior on reset. Resubscription may occur due to many reason such errors in the subscription listener, restart of worker node, redeployment of iFlow etc.
  - **Replay from last processed message** restarts the subscription from the last replay ID successfully consumed by the iFlow.
  - **Replay from UI Configured Value** restarts from the replay ID defined in the Replay ID Approach setting.

5.2.1 Type of Events

The following Events are supported:

- *PushTopic Events:* These are Events that provide notification messages for record changes to Salesforce data. These Events include creation, updating, deletion and are specified in Salesforce using a SOQL query. Events matching the query are sent as notifications. Note that Salesforce sends PushTopic events for a pre-defined set of SObjects and all Custom Objects. To read more about supported Objects, refer Salesforce documentation.
- *Platform Events:* Used to connect Salesforce business processes to other systems by sending real-time Event data.
- *Generic Events:* Used by Salesforce to send custom notifications that are not linked to a data change.
- *Change Data Capture*: Used by Salesforce to send and publish near-real-time change Events. These changes represent any changes to Salesforce records. Including the creation, updates, and deletion of a record. To read more about the topic names to be used for Change Data Capture, refer to the Salesforce Salesforce documentation.

5.2.2 Replay Id Approach

Every Event or message retrieved from Streaming is assigned a Replay ID value. This field refers to the position of the event in the event stream. There are three possible settings:

- ***Events from latest position (-1)****:* Enables SAP Cloud Integration to receive any new Event. It represents Replay ID value -1. Note that the iFlow needs to be deployed and running for any new Event to be received. Any Event is generated while the iFlow is not available will be missed by the iFlow.
- ***Events from earliest position (-2)****:* Enables SAP Cloud Integration to receive all Events that have not yet expired. It represents Replay ID value -2.
- ***Events from a specific position****:* This option gives the possibility to specify any Replay ID. In this case, SAP Cloud Integration will receive all Events that were created after the specified Replay ID. In case the Replay ID is 5, a message containing Replay Id 6, 7, 8, etc. will be retrieved.

5.2.3 Subscription Mechanisms

Salesforce Sender adapter provides the option to select a subscription mechanism that utilises cluster-level locks to ensure event consumption is done by only one worker or node instance at a time. **Subscription Mechanism** and **Subscription Behavior on Reset** can be set to various configurations to optimize event consumption.

Note: Configuring your subscription must be done carefully, you must keep in mind the event consumption requirements specific to your production scenario. To help you understand the impact of each combination and make an informed choice, see the **Notes** tab in the table below.

Subscription Mechanism	Subscription Behaviour on Reset	Notes
Subscribe on all runtime nodes	**Replay from UI Configured Value**	This is the default configuration where all runtime nodes participate in subscription.
Subscribe on all runtime nodes	**Replay from last processed message**	This is a recommended configuration where all nodes participate in the subscription and the adapter retains the last successfully received message. In scenarios such as iFlow redeployment or tenant restarts, the adapter resumes the subscription using the stored replay ID rather than the value configured in the UI, thereby ensuring no data loss. **Note:**The “Replay ID approach” configuration is applied only during initial deployment. For all subsequent subscriptions, the adapter uses the replay ID stored in the persistence layer to establish new subscriptions.
Subscribe on one runtime node	**Replay from last processed message**	This configuration is suitable for scenarios where multi-node subscription is not required. In a scenario where the Salesforce event consumption count is higher, the “Subscribe on one runtime node” option can be selected. Please note that since the subscription is active on only one node, a node failure may cause a temporary delay in event consumption until another node takes over the subscription. This setup ensures that event consumption resumes from the last stored replay ID, once the node becomes available again after an unexpected downtime. **Note:** The “Replay ID approach” configuration is applied only during the

		initial deployment. For all subsequent subscriptions, the adapter uses the replay ID stored in the persistence layer to establish new subscriptions.
Subscribe on one runtime node	**Replay from UI Configured Value**	This is the least preferred configuration and should only be used when you need to process events from a specific replay ID, rather than using the value stored in the persistence layer. As the subscription is active on a single node, a node failure may lead to potential event loss, since the replay ID is not persisted.

If you have any queries regarding the above subscription options, you can raise an OSS ticket for further clarifications.

5.2.3 Channel Name Prefix

Depending on the type of Event selected, the Channel name needs to be prefixed according to the table below.

Event Type	**Channel name Prefix**	**Example**
PushTopic	/topic/	/topic/Account
Platform Events	/event/	/event/TestEvent__e
Generic Events	/u/	/u/notifications/ExampleUserChannel
Change Data Capture	/data/	/data/ChangeEvents

6 RECEIVER ADAPTER CONFIGURATION

This section describes the different parameters that can be configured for the Receiver Adapter. This Adapter has the Connection and Processing Tabs. Each one of these will be explained in the next sections.

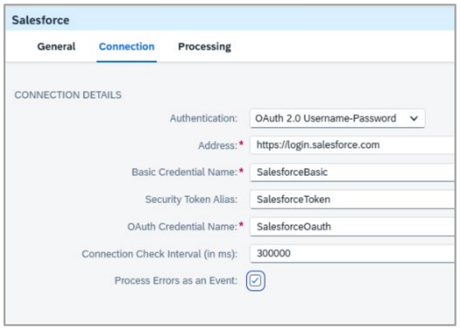
6.1 Connection Tab

The Connection Tab contains parameters that define how to connect to and how to authenticate against Salesforce. There are three authentication types:

- OAuth 2.0 Username-Password
- OAuth 2.0 JWT Bearer
- OAuth 2.0 Client Credentials

6.1.1 OAuth 2.0 Username-Password

For OAuth 2.0 Username-Password, a sample configuration is shown in Figure .



CONNECTION DETAILS

Authentication: OAuth 2.0 Username-Password

Address:* https://login.salesforce.com

Basic Credential Name:* SalesforceBasic

Security Token Alias: SalesforceToken

OAuth Credential Name:* SalesforceOauth

Connection Check Interval (in ms): 300000

Process Errors as an Event: ☑

Figure 5.2 Connection Tab of Salesforce Adapter, OAuth 2.0 Username Password

As seen in Figure 5.2, the Connection Tab contains the following fields:

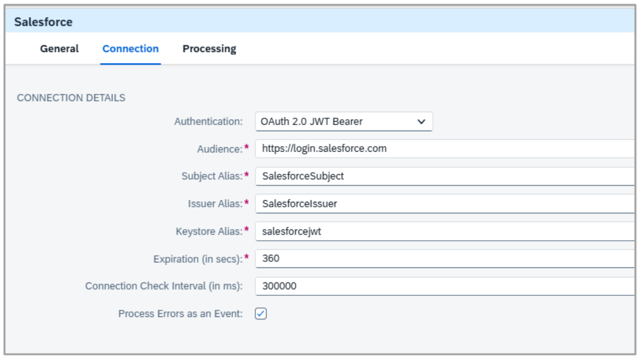
- **Address***:* Specifies the recipient's endpoint URL. By default, the URL https://login.salesforce.com is used. This can be changed according to a scenario.
  - For Salesforce production environments: https://login.salesforce.com.
  - For Sandbox environments: https://test.salesforce.com. In case MyDomain is enabled in the organization, https://<MyDomain>.my.salesforce.com can be used.
- **Basic Credential Name***:* Specifies the name of the User Credentials artifact that contains the credentials for basic authentication. This refers to the username-password pair used to log in to Salesforce. This pair must be created as a Security Artifact of User Credential type and subsequently referred to in the Adapter.
- **Security Token Alias***:* Specifies the name of the Secure Parameter artifact that contains the security token needed to connect to Salesforce. This property enables the system to fetch the security token from Keystore for authentication. This field can be omitted if the IP has been whitelisted in Salesforce. To whitelist an IP address, add the IP range in the Network Access section in the Security Control menu in Salesforce.
- **OAuth Credential Name***:* Specifies the name of the User Credentials artifact that contains the Salesforce's OAuth Consumer Key-Consumer Secret pair. Configure the Consumer Key as **User** and the Consumer Secret as the **Password** in the Security Artifact of type **User Credential**. Then refer to it in the Adapter's Connection Tab.
- **Connection *Check Interval (in ms):*** Specifies the interval in milliseconds to verify the connection validity to Salesforce.
- **Process Errors as an Event:** Process Errors as an Event will create message exceptions in SAP Cloud Integration for each connection error during Salesforce Streaming. Errors that prevent SAP Cloud Integration from picking up Events via Streaming will be raised as a message exception in

SAP Cloud Integration. An example exception message during streaming is "Organization concurrent user limit exceeded".

For more details on how to retrieve details of the Security token and Consumer Key-Consumer Secret pair, refer to Section 12.1.

6.1.2 OAuth 2.0 JWT Bearer

For OAuth 2.0 JWT Bearer, a sample configuration is shown in Figure 5.3.



CONNECTION DETAILS

Authentication: OAuth 2.0 JWT Bearer

Audience:* https://login.salesforce.com

Subject Alias:* SalesforceSubject

Issuer Alias:* SalesforceIssuer

Keystore Alias:* salesforcejwt

Expiration (in secs):* 360

Connection Check Interval (in ms): 300000

Process Errors as an Event: ☑

Figure 5.3 Connection Tab of Salesforce Adapter with OAuth JWT Bearer

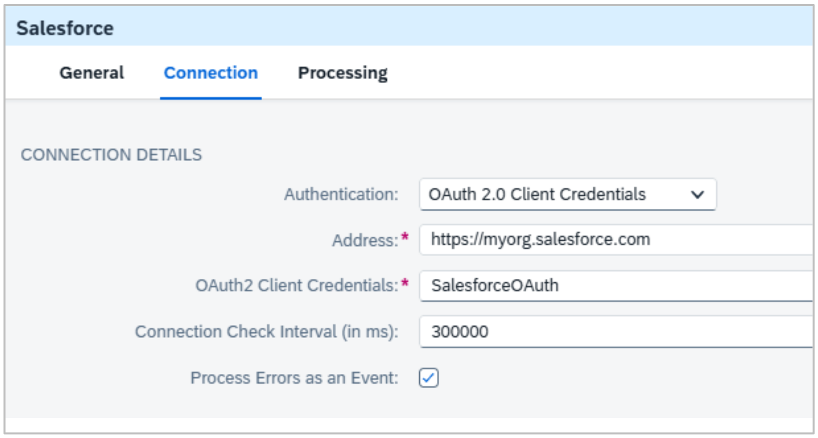
As seen in Figure 5.3, the Connection Tab contains the following fields:

- **Audience***:* Specifies the recipient's endpoint URL. By default, the URL https://login.salesforce.com is used. This can be changed based on a scenario.
  - For Salesforce production environments: https://login.salesforce.com.
  - For Sandbox environments, https://test.salesforce.com. In case MyDomain is enabled in the organization, https://<MyDomain>.my.salesforce.com may be used.
- **Subject Alias***:* The alias name of the deployed Secure Parameter artifact. It specifies the Salesforce username.
- **Issuer Alias***:* The alias name of the deployed Secure Parameter artifact. It specifies the OAuth Consumer Key of the connected app for which the certificate was registered.
- **Keystore Alias***:* The alias name of the added JKS file in Keystore as a Key Pair. It consists of a key and certificate to sign the JWT.
- **Expiration *(in secs)****:* Specifies the validity of the assertion in seconds.
- **Connection Check Interval (in ms):** Specifies the interval in milliseconds to verify the connection validity to Salesforce.
- **Process Errors as an Event:** Process Errors as an Event will create message exceptions in SAP Cloud Integration for each connection error during Salesforce Streaming. Errors that prevent SAP Cloud Integration from picking up events via Streaming will be raised as a message exception in SAP Cloud Integration. An example exception message during streaming is "Organization concurrent user limit exceeded".

For more details on how to retrieve the above details, refer to Section 12.2.

6.1.3 OAuth 2.0 Client Credentials

For OAuth 2.0 Client Credentials, a sample configuration is shown .



Salesforce

General | Connection | Processing

CONNECTION DETAILS

Authentication: OAuth 2.0 Client Credentials

Address:* https://myorg.salesforce.com

OAuth2 Client Credentials:* SalesforceOAuth

Connection Check Interval (in ms): 300000

Process Errors as an Event: ☑

Figure 5.5 Connection Tab of Salesforce Adapter with OAuth 2.0 Client Credentials

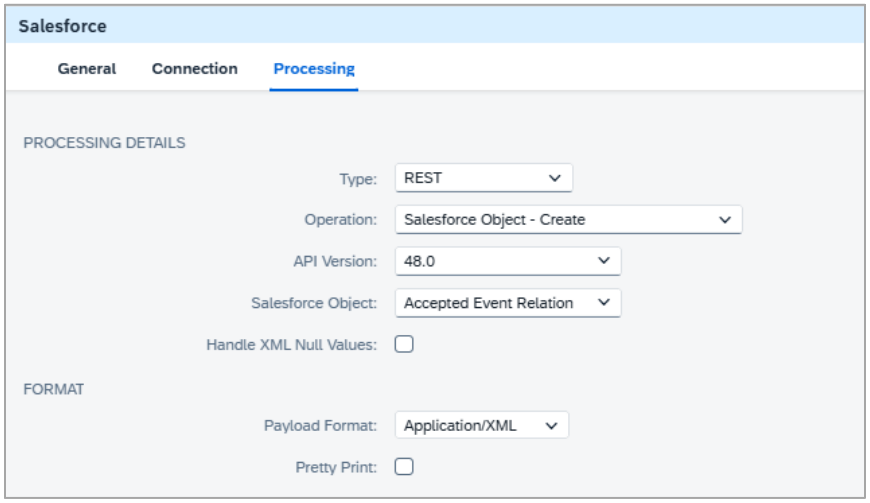
As seen in the Figure 5.5 , the Connection Tab contains the following fields:

- **Address***:* Specifies the recipient's endpoint URL. By default, the URL https://login.salesforce.com is used. This can be changed based on a scenario.
  - For Salesforce production environments: https://login.salesforce.com.
  - For Sandbox environments, https://test.salesforce.com. In case MyDomain is enabled in the organization, https://<MyDomain>.my.salesforce.com may be used.
- **OAuth2 Client Credentials***:* The alias name of the deployed OAuth2 Client Credentials artifact that uses Client ID and Client Secret.
- **Connection Check Interval (in ms):** Specifies the interval in milliseconds to verify the connection validity to Salesforce.
- **Process Errors as an Event:** Process Errors as an Event will create message exceptions in SAP Cloud Integration for each connection error during Salesforce Streaming. Errors that prevent SAP Cloud Integration from picking up events via Streaming will be raised as a message exception in SAP Cloud Integration. An example exception message during streaming is "Organization concurrent user limit exceeded".

For more details on how to create an OAuth2 Client Credential, refer to Section 12.3.

6.2 Processing Tab

The Processing tab contains all the operation configurations for the selected Salesforce API. An example configuration for the operation Salesforce Object – Create is shown in Figure 6.3.



PROCESSING DETAILS

Type: REST

Operation: Salesforce Object - Create

API Version: 48.0

Salesforce Object: Accepted Event Relation

Handle XML Null Values: □

FORMAT

Payload Format: Application/XML

Pretty Print: □

Figure 6.3 Processing Tab of Salesforce Adapter, REST

The different properties of the Adapter shown in Figure 6.3.are described below:

- **Type:** Specifies the category of the API to be used to interact with Salesforce. Possible values include REST, REST Bulk, and REST Place Order. These different API types are explained in Section.
- **API Version**: The API Version specifies the version of the Salesforce API. Since the 1st of June 2022 Salesforce API Versions 7.0 through 20.0 have been retired by Salesforce and are no longer part of the standard Salesforce API Version selection in the Salesforce Adapter. Note that the Salesforce API version can be overwritten by manually entering the Salesforce API version.
- **Operation:** One of the provided operation options can be selected to specify the type of action desired to be executed in Salesforce. Section 0 dives into all the different operations. Section 6.2.1 assists in deciding which operation to use for an integration scenario.
- **Payload Format:** This field specifies the format of the request message to be sent to and the response to be returned from Salesforce. Possible values include Application/XML and Application/JSON. Note that the default value is Application/XML.
- **Salesforce Object:** Select the name of the Object in Salesforce to operate on.
  Example: Accounts, Lead, Order, etc. Custom can be selected in case any custom sObject is involved or if the Salesforce Object is not available in the list of Salesforce Objects provided.
- **Pretty Print:** Select the Pretty Print checkbox to nicely format the XML payload and improve its readability.
- **Handle XML Null Values**: Enable to use null values in the payload to nullify properties in Salesforce
- **Treat all XML Payload Values as String**: Select to treat all the values in the XML payload as string before sending it to Salesforce

6.2.1 REST API

The REST API supports fifteen operations to retrieve or send data to Salesforce. These operations are categorized into five groups known as Salesforce Object, Metadata, SOQL, SOSL, and Others.

There are a set of common fields that are often used across different operations in the salesforce Adapter. These common fields are described below:

- *Operation:* To access and exchange data with Salesforce, one of the operation options listed in the dropdown can be selected.
- *Salesforce Object:* Represents a business object or entity in Salesforce. For example, Salesforce Object (or sObject) includes Account, Contact, Order, Product, etc. Each Salesforce console tab represents one Salesforce Object. If a Custom Object name is preferred, select Custom Object

from the list with Salesforce objects and type the name of the Custom Object. As the dropdown list is editable, it is possible to specify or write the name of the Custom Object.

- *Salesforce Object ID:* Specifies a unique record of a sObject.
- *External ID Name:* Refers to the name of a custom field of the sObject which is configured as an external ID. This is an alternative way to identify a Salesforce Object record based on a unique identifier provided by an external system (other than Salesforce).
- *External ID Value:* Refers to the value of the External ID Name field. For example, in a scenario where an SAP ERP (Enterprise Resource Planning) system needs to synchronize master data with Salesforce, SAPReferenceNumber in Salesforce (as External ID Name) and 10005210032 as (External ID Value) can be used to identify that record in Salesforce based on the SAP identifier.
- *SOQL Query:* Specifies the SOQL Query to be executed in Salesforce. SOQL stands for Salesforce Object Query Language and is an adaptation of the well-known SQL query language. A simple SELECT ID, Name FROM Account provides all the ID and Name for all Account records in Salesforce. For more advanced queries, the SOQL Query Editor highlighted in detail in Section 8.1 may be used.
- *SOSL Search:* Specifies the SOSL Search to be executed in Salesforce. SOSL stands for Salesforce Object Search Language and is used to construct text-based search queries against the search index. Text, email, and phone fields for multiple objects, including Custom Objects may be searched for. An example SOSL is FIND {Bakery*} RETURNING account (industry, ID, billingStreet, name). Section 8.2 explains how to use the SOSL Search Editor to get the most out of the SOSL Salesforce searches.
- *Filter Fields:* Specifies the fields of the sObject that need to be returned as part of the result of a query toward Salesforce. Without any field specified in this property, all fields of the sObject will be returned. For example, instead of getting back all the fields for a specific Salesforce Object, the operation Salesforce Object – Get can be narrowed down by applying a filter for only ID, name, and industry.

The following sections explain each operation of the REST API and what they can be used for.

6.2.1.1 Other – Get REST API Versions

This operation retrieves a list of information about each REST API version currently available.

6.2.1.2 Other – Get REST Resource Endpoints

This operation retrieves a list of resources available for the specified API version.

6.2.1.3 Metadata – All

This operation retrieves a list of available objects for the Salesforce tenant. This operation also returns the Salesforce org encoding and the maximum batch size permitted in queries.

6.2.1.4 Metadata – Basic

This operation retrieves metadata for a specified Salesforce object. The below table shows an example configuration of this operation.

Example Configuration	
API Version	50.0
Operation	Metadata - Basic
Salesforce Object	Account
Payload Format	Application/JSON or Application/XML
Pretty Print	Enable

6.2.1.5 Metadata – Comprehensive

This operation retrieves all the metadata for a specified Salesforce object.

6.2.1.6 Salesforce Object – Get

This operation retrieves a record in Salesforce and its properties. The table below shows an example configuration of this operation. Also, note in this example the configuration of specifying the names (ID, name) in the fields to be returned (in the property named List of Fields).

Example Configuration	
API Version	50.0
Operation	Salesforce Object - Get
Payload Format	Application/JSON or Application/XML
Salesforce Object	Account
Salesforce Object ID	0013600000xU52T
Filter Fields	Enable
List of Fields	ID, name
Pretty Print	Enable

6.2.1.7 Salesforce Object – Get Blob

This operation retrieves the binary content of a Salesforce Attachment or Document object. The response content type is not in JSON or XML formats as the document body content is returned in binary form. The table below shows an example configuration of this operation.

Example Configuration	
API Version	50.0
Operation	Salesforce Object - Get Blob
Salesforce Object	Attachment
Salesforce Object ID	0013600000xU52T

6.2.1.8 Salesforce Object – Create

This operation creates a new record of a specified Salesforce Object. The request payload provides the property and field values in the request data. The response body contains the unique identifier (ID) of the created record if the call is successful.

6.2.1.9 Salesforce Object – Update

This operation updates a Salesforce Object record. The request payload contains the property values that are needed to update a record in Salesforce based on a specific record ID.

Note: HTTP Protocol supports HTTP_1_1 which is a plaintext framing that includes persistent connections.

The below table shows an example configuration of this operation.

Example Configuration	
API Version	50.0
Operation	Salesforce Object - Update
Payload Format	Application/JSON or Application/XML
Salesforce Object	Case
Salesforce Object ID	5003600000Ag1YK
HTTP Protocol	Default
Pretty Print	Enable

6.2.1.10 Salesforce Object – Delete

This operation deletes a Salesforce Object record. It requires the unique identifier of the Salesforce record to be deleted.

6.2.1.11 Salesforce Object – Get with External ID

This operation retrieves details of a Salesforce Object record using the value of a specified external ID. This operation differs from Salesforce Object – Get as the External ID is specified instead of the Salesforce record ID. This is shown in the example configuration below.

Example Configuration	
API Version	50.0
Operation	Salesforce Object - Get with External ID
Payload Format	Application/JSON or Application/XML
Salesforce Object	Account
External ID Name	SAPReferenceNumber__c
External ID Value	1000521110
Filter Fields	Enable
Fields	ID, name, industry
Pretty Print	Enable

Note that this operation is meant to be used with 'External ID'. As seen in the table, the field "SAPReferenceNumber__c " is defined as a reference ID from another external system. Example: SAP.

6.2.1.12 Salesforce Object - Upsert with External ID

This operation creates or updates Records in Salesforce depending on whether it already exists or not. It uses the value of a specified External ID to determine whether to create or update a Record. If this value does not exist, a new Record is created. If a record does exist with the same value, the field values specified in the request body are updated.

Example Configuration	
API Version	50.0
Operation	Salesforce Object - Upsert with External ID
Payload Format	Application/JSON or Application/XML
Salesforce Object	Account
External ID Name	SAPReferenceNumber__c
External ID Value	1000521110
HTTP Protocol	Default
Filter Fields	Enable
Fields	ID, name, industry
Pretty Print	Enable

Note that the field needs to be defined as an 'External ID' in Salesforce.

6.2.1.13 Salesforce Object - Delete with External ID

This operation deletes a record using the value of a specified External ID. Therefore, the External ID should be specified instead of the Salesforce ID of the record.

Note that the field needs to be defined as an 'External ID' in Salesforce.

6.2.1.14 SOQL - Execute Query

This operation allows the execution of powerful SOQL queries to retrieve Salesforce data. When using this operation, it is possible to filter the records to be returned. For example, the query can return the IDs and the names of all the Accounts based in France with more than one billion yearly revenues. By default, the SOQL query returns up to 2000 records on a single response page. In case there are more than 2000 records, a URL link (nextRecordsUrl) is returned by Salesforce which enables the retrieval of additional records using the SOQL Execute Query for More Results operation.

For more details on how to achieve this, refer to the operation SOQL Execute Query for More Results explained in Section 6.2.1.15. Furthermore, the SOQL Query Editor (which is provided in the Eclipse Plug-in) can be used to build specific and complex queries. This is specified in Section 8.1.

The following table shows an example configuration of the SOQL – Execute Query operation.

Example Configuration	
API Version	50.0
Operation	Execute SOQL query
Payload Format	Application/JSON or Application/XML
SOQL Query	Select ID, name, industry FROM Account
Auto Pagination	Enable
Include deleted records in query result	Enable to include the records that have been deleted or merged as part of the result.
Pretty Print	Enable

It is possible to automatically retrieve all records within a single call if there are more than 2000 records. This can be done by selecting the "Auto Pagination" option.

Note that by default, the Query operation only returns active data records. Deleted objects are excluded from the result set. To include deleted or archived records in the result set, the "Include deleted records in query result" checkbox must be ticked.

Note that the Outbound Headers table under the HEADER SETTINGS section can be used to specify the request headers.

Example:
Name as 'Sforce-Query-Options' and Value as 'batchSize=200'.

Note that dynamic values like headers or properties can be used in queries.
Example:
Select name,SAP_Account_number__c FROM Account WHERE SAP_Account_number__c= '${property.myID}'

6.2.1.15 SOQL - Execute Query for More Results

This operation is an extension of the previous operation. It retrieves the query results that were not returned in the original SOQL – Execute Query response using a Next Records URL. This operation exists for instances where the result of a query can generate an amount of data that cannot all be returned in one response payload. In such a case, a URL is returned in the SOQL - Execute Query. This URL can then be used in the Execute Query for More Results operation. The table below shows an example configuration.

Example Configuration	
API Version	50.0
Operation	SOQL - Execute Query for More Results

Payload Format	Application/JSON or Application/XML
Next Records URL	/services/data/v44.0/query/01g3600000XpvoNAAR-1000
Pretty Print	Enable

6.2.1.16 SOSL – Search Resource

This operation supports a SOSL query to search for useful information in Salesforce. The Salesforce Object Search language lets users search using strings throughout the entire Salesforce tenant and acts as an alternative for the SOQL query. Users can utilize the SOSL Search Editor to build complex queries. For details on how to use the SOSL Search editor in the Eclipse Plug-in, see Section 8.2. The table below shows an example configuration.

Data Insert	
API Version	50.0
Operation	SOSL - Search Resource
Payload Format	Application/JSON or Application/XML
SOSL Search	FIND {Bakery*} RETURNING account (industry, ID, billingStreet, name)
Pretty Print	Enable

6.2.1.17 Salesforce Objects – Composite

This operation allows the creation of a composite call with a request body that contains various individual calls such as Upsert and Read actions on multiple sObjects. The Composite operation supports the usage of multiple sObject records and multiple sObject types in one operation.

This reduces the number of roundtrips between SAP Cloud Integration and Salesforce. For instance, an integration scenario that has one Account and two Contacts can be retrieved in a single call to Salesforce. This eliminates the need for multiple calls to retrieve data associated with a single Record. It is also possible to use the output of one request as the input of a subsequent request with this operation.

This operation supports control over rollback actions for the operations on sObjects specified in the composite collection. The allorNone can be used to specify what to do when an error occurs while processing a subrequest.

Additionally, the request payload has a field named “collateSubrequests” which controls whether the API collates unrelated subrequests to bulkify them (true) or not (false).

When “collateSubrequests” is set to true, subrequests are collated and the processing speed is faster. It is important to note however that order of execution is not guaranteed. When “collateSubrequests” is set to false, the subrequests are executed sequentially in the order in which they are received.

A single composite call can consist of multiple subrequests that require different methods and URLs based on the request type. The table below provides an overview of the type of request, operation, method, and corresponding URL that the operation can handle.

Composite Subrequest Type	**Operation**	**Method**	**URL**
sObject	Create	POST	Structure: /services/data/<VERSION>/ sobjects/<SOBJECT NAME> Sample: /services/data/v50.0/sobjects/Account
	Upsert	PATCH	Structure: /services/data/<VERSION>/sobjects/<SOBJECT NAME>/<EXTERNALID FIELD NAME>/<EXTERNALID VALUE> Sample: /services/data/v50.0/sobjects/Account/ExternalAcctId__c/EXTID12345
	Update	PATCH	Structure: /services/data/<VERSION>/sobjects/<SOBJECT NAME>/<INTERNALID VALUE> Sample: /services/data/v50.0/sobjects/Account/INTID12345

Composite Subrequest Type	Operation	Method	URL
	Retrieve	GET	Structure: /services/data/<VERSION>/sobjects/<SOBJECT NAME>/<INTERNALID VALUE> Sample: /services/data/v50.0/sobjects/Account/INTID12345
	Delete	DELETE	Structure: /services/data/<VERSION>/sobjects/<SOBJECT NAME>/<INTERNALID VALUE> Sample: /services/data/v50.0/sobjects/Account/INTID12345
sObject Collection	Create	POST	Structure: /services/data/<VERSION>/composite/sobjects Sample: /services/data/v50.0/composite/sobjects
	Upsert	PATCH	Structure: /services/data/<VERSION>/composite/sobjects/<SOBJECT NAME>/<EXTERNALID FIELD NAME> Sample: /services/data/v50.0/composite/sobjects/Account/ExternalAcctId__c
	Update	PATCH	Structure: /services/data/<VERSION>/composite/sobjects Sample: /services/data/v50.0/composite/sobjects
	Retrieve	GET	Structure: /services/data/<VERSION>/composite/sobjects/<SOBJECT NAME>?ids=<INTERNALID VALUE>&fields=<FIELD NAME> Sample: /services/data/v50.0/composite/sobjects/Account?ids=INTID12345&fields=id
	Delete	DELETE	Structure: /services/data/<VERSION>/composite/sobjects?ids=<INTERNALID VALUE1> Sample: /services/data/v50.0/composite/sobjects?ids=INTID12345

A more detailed example scenario can be found in Section 11.5. The table below shows an example configuration.

Example Configuration	
API Version	50.0
Operation	Salesforce Objects - Composite
Payload Format	Application/JSON or Application/XML
Pretty Print	Enable

Note that when using a Composite operation, it is possible to include Records on the sObject in the request. However, it is also possible to make a request that includes another sObject Collection as a subrequest.

The image below demonstrates three scenarios of a composite request.

- Scenario 1: Indicates multiple subrequests i.e., Account and Contact (Example 1).
- Scenario 2: Indicates only sObject collections as subrequests (Example 2).
- Scenario 3: Indicates a mixture of composite and sObject Collection records as in a single request (Example 2).

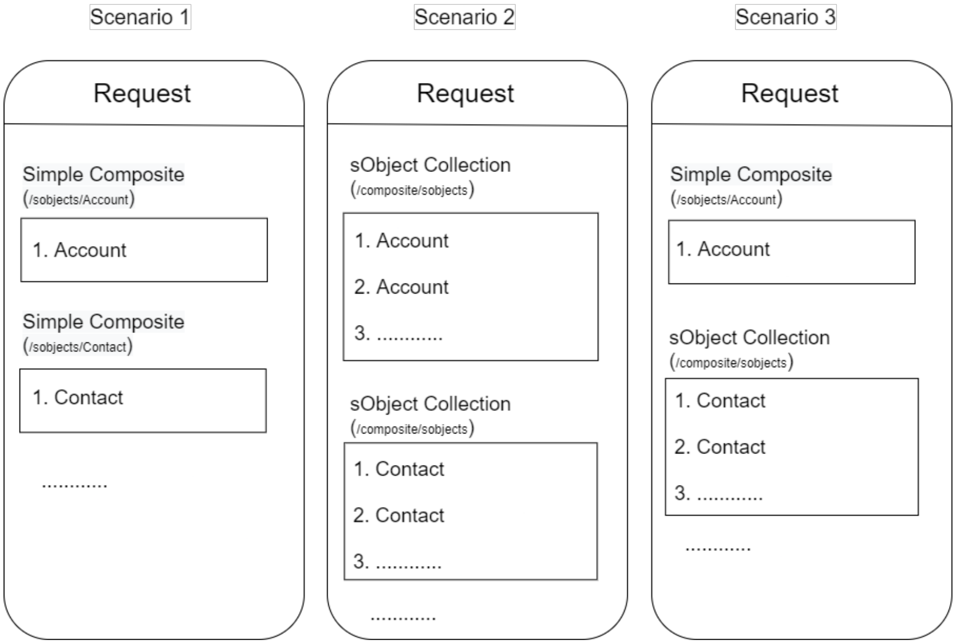


Figure 6.4 Illustration of the Difference of the Three Composite Scenarios

Example 1 illustrates the payload of a simple Composite.

Example 1: XML Request message (Composite) - Scenario 1

```
<sObjectsComposite>
   <allOrNone>false</allOrNone>
   <collateSubrequests>false</collateSubrequests>
   <compositeRequest_Account>
      <method>POST</method>
      <url>/services/data/v48.0/sobjects/Account</url>
      <referenceId>refAccount</referenceId>
      <body>
         <Name>John Doe</Name>
      </body>
   </compositeRequest_Account>
   <compositeRequest_Contact>
      <method>POST</method>
      <url>/services/data/v48.0/sobjects/contact</url>
      <referenceId>refContact</referenceId>
      <body>
         <LastName>Smith</LastName>
         <Email>smith@salesforce.com</Email>
      </body>
   </compositeRequest_Contact>
</sObjectsComposite>
```

A payload of a Composite with only sObject Collection as Sub Request looks is shown below in Example 2.

Example 2: XML Request message (Composite with only sObject Collection as SubRequest) - Scenario 2

```
<sObjectsComposite>
  <allOrNone>false</allOrNone>
  <collateSubrequests>false</collateSubrequests>
  <compositeRequest_SObjectCollection_Account>
    <method>POST</method>
    <url>/services/data/v48.0/composite/sobjects</url>
    <referenceId>refAccounts</referenceId>
    <body>
      <allOrNone>false</allOrNone>
      <records>
        <attributes>
          <type>Account</type>
        </attributes>
        <Name>John Doe</Name>
      </records>
      <records>
        <attributes>
          <type>Account</type>
        </attributes>
        <Name>Jane Doe</Name>
      </records>
    </body>
  </compositeRequest_SObjectCollection_Account>
  <compositeRequest_SObjectCollection_Contact>
    <method>POST</method>
    <url>/services/data/v48.0/composite/sobjects</url>
    <referenceId>refContacts</referenceId>
    <body>
      <allOrNone>false</allOrNone>
      <records>
        <attributes>
          <type>Contact</type>
        </attributes>
        <LastName>Reter</LastName>
        <Email>peter@salesforce.com</Email>
      </records>
      <records>
        <attributes>
          <type>Contact</type>
        </attributes>
        <LastName>Ronald</LastName>
        <Email>ronald@salesforce.com</Email>
      </records>
    </body>
  </compositeRequest_SObjectCollection_Contact>
</sObjectsComposite>
```

Note that this sObject Collection as Sub Request has a tag with the name **compositeRequest_SObjectCollection_Contact** instead of the tag **compositeRequest_Contact** which is used for the normal Composite.

Example 3 below shows what the payload of a Composite with sObject Collection as Sub Request looks like.

Example 3: XML Request message (Composite along with sObject Collection as SubRequest) - Scenario 3

```
<sObjectsComposite>
  <allOrNone>false</allOrNone>
  <collateSubrequests>false</collateSubrequests>
  <compositeRequest_Account>
    <method>POST</method>
    <url>/services/data/v48.0/sobjects/Account</url>
    <referenceId>refAccount</referenceId>
    <body>
      <Name>John Doe</Name>
    </body>
  </compositeRequest_Account>
  <compositeRequest_SObjectCollection_Contact>
    <method>POST</method>
    <url>/services/data/v48.0/composite/sobjects</url>
    <referenceId>refContact1</referenceId>
    <body>
      <allOrNone>false</allOrNone>
      <records>
        <attributes>
          <type>Contact</type>
        </attributes>
        <LastName>Smith</LastName>
        <Email>smith@salesforce.com</Email>
      </records>
      <records>
        <attributes>
          <type>Contact</type>
        </attributes>
        <LastName>Desmond</LastName>
        <Email>desmond@salesforce.com</Email>
      </records>
    </body>
  </compositeRequest_SObjectCollection_Contact>
  <compositeRequest_SObjectCollection_Contact>
    <method>POST</method>
    <url>/services/data/v48.0/composite/sobjects</url>
    <referenceId>refContact2</referenceId>
    <body>
      <allOrNone>false</allOrNone>
      <records>
        <attributes>
          <type>Contact</type>
        </attributes>
        <LastName>Reter</LastName>
        <Email>peter@salesforce.com</Email>
      </records>
      <records>
        <attributes>
          <type>Contact</type>
        </attributes>
        <LastName>Ronald</LastName>
        <Email>ronald@salesforce.com</Email>
      </records>
    </body>
  </compositeRequest_SObjectCollection_Contact>
</sObjectsComposite>
```

Note that this request has tags with the name **compositeRequest_Account** for normal composite and **compositeRequest_SObjectCollection_Contact** for sObject Collection.

Also notice the difference in URL tags i.e., /**services/data/v48.0/sobjects/Account** for normal composite and **/services/data/v48.0/composite/sobjects** for sObjectCollection.

Example 4 below indicates a sample payload with Upsert operation for Composite Subrequests. This Composite request includes two different subrequests. The first subrequest represents an Upsert operation on a single object Account. The second subrequest contains an Upsert operation on an SObject Collection of two accounts. Note the differences in structure, URL specification, and the way the external ID value is specified in the XML.

Example 4: XML Request message for Upsert operation (Composite along with sObject Collection as SubRequest)

```
<sObjectsComposite>
  <allOrNone>false</allOrNone>replay
  <collateSubrequests>false</collateSubrequests>
  <compositeRequest_Account>
    <method>PATCH</method>
    <url>/services/data/v48.0/sobjects/Account/External_Id_Field/99999</url>
    <referenceId>refAccountComposite</referenceId>
    <body>
      <Name>John Doe</Name>
    </body>
  </compositeRequest_Account>
  <compositeRequest_SObjectCollection_Account>
    <method>PATCH</method>
    <url>/services/data/v48.0/composite/sobjects/Account/External_Id_Field</url>
    <referenceId>refAccountCompositeColl</referenceId>
    <body>
      <allOrNone>false</allOrNone>
      <records>
        <attributes>
          <type>Account</type>
        </attributes>
        <Name>Will Smith</Name>
        <External_Id_Field>99999_1</External_Id_Field>
      </records>
      <records>
        <attributes>
          <type>Account</type>
```

```
            </attributes>
            <Name>Jane Doe</Name>
            <External_Id_Field>99999_2</External_Id_Field>
         </records>
      </body>
   </compositeRequest_SObjectCollection_Account>
</sObjectsComposite>
```

Example 5: XML Request message (Composite with reference within records)

```
<sObjectsComposite>
   <allOrNone>true</allOrNone>
   <collateSubrequests>false</collateSubrequests>
   <compositeRequest_Account>
      <method>PATCH</method>
      <url>/services/data/v48.0/sobjects/Account/SAPReferenceNumber__c/99999</url>
      <referenceId>refAccount</referenceId>
      <body>
         <Name>John Doe</Name>
      </body>
   </compositeRequest_Account>
   <compositeRequest_Contact>
      <method>POST</method>
      <url>/services/data/v48.0/sobjects/contact</url>
      <referenceId>contact</referenceId>
      <body>
         <AccountId>@{refAccount.id}</AccountId>
         <LastName>Smith</LastName>
         <Email>smith@salesforce.com</Email>
      </body>
   </compositeRequest_Contact>
</sObjectsComposite>
```

In the above example, the result of the first record (of type Account) is used for the subsequent request of type Contact using the value **@{refAccount.id}** in the AccountId field.

Example 6: JSON Request message Upsert operation (Composite along with sObject Collection as SubRequest)

```
{
"allOrNone": false,
"collateSubrequests": false,
"compositeRequest": [
{
          "method": "PATCH",
          "url": "/services/data/v48.0/sobjects/account/External_Id_Field/99999",
          "referenceId": "refAccountComposite",
          "body": {
                    "Name": "John Doe"
                    }
},
{
          "method": "PATCH",
          "url": "/services/data/v48.0/composite/sobjects/account/External_Id_Field",
          "referenceId": "refAccountCompositeColl",
          "body": {
          "records": [
          {
          "attributes": {
                    "type": "account"
                    },
          "Name": "Will Smith",
          "External_Id_Field": "99999_1"
                    },
          {
          "attributes": {
                    "type": "account"
                    },
          "Name": "Jane Doe",
          "External_Id_Field": "99999_2"
                    }
          ]
          }
}
]
}
```

The above example indicates a sample payload with Upsert operation for Composite Subrequests in JSON format. This Composite request includes two different subrequests. The first subrequest represents an Upsert operation on a single object Account. The second subrequest contains an SObject Collection Upsert operation on two accounts. Note the differences in structure, URL specification, and the way the external ID value is specified.

Up to twenty-five subrequests can be made on a single call. Up to five of these subrequests can be sObject Collections or query operations, including Query and QueryAll requests. If a Composite request uses sObject Collections, there are two or more allOrNone parameters that can interact, one on the Composite request and one on each sObject Collections subrequest.

If the Composite request has allOrNone set to true, then the all-or-none behavior also applies to each of the sObject Collections subrequests. This overrides the value of allOrNone in the subrequests.

6.2.1.18 Salesforce Objects – sObject Collections

The sObject Collections operation allows the selection of one of the following methods; Create, Retrieve, Update, or Upsert, and applies it to one or more records of the same sObject type. In an integration scenario where the creation of multiple accounts within the same call to Salesforce is desired, the sObject Collection can be used.

This operation supports control over rollback actions for the operations on sObjects specified in the request. The allOrNone can be used to specify what to do when an error occurs while processing a subrequest.

Note that this service is available in API version 42.0 or later. A maximum of 200 records can be used with the methods Create, Update, or Upsert and up to 2000 for Retrieve.

The table below shows an example configuration.

Example Configuration	
API Version	50.0
Operation	Salesforce Objects – sObject Collections
Method	Select any of the following options: Create, Delete, Retrieve, Update, or Upsert
Payload Format	Application/JSON or Application/XML
HTTP Protocol	Default (Only applicable when Operation is Update or Upsert)
Pretty Print	Enable

Create: Use this method with sObject Collections to create up to 200 records, returning a list of SaveResult objects.

Delete: Use this method with sObject Collection to delete up to 200 records. The IDs of the records are required.

Retrieve: Use this method with sObject Collections to retrieve one or more records of the same object type. A list of sObjects that represents the individual records of the specified type is returned. The number of sObjects returned matches the number of IDs passed in the request.

Update: Use this method with sObject Collections to update up to 200 records. It returns a list of SaveResult objects.

Upsert: Use this method with sObject Collections to either create or update (upsert) up to 200 records based on an external ID field.

Note that the Delete method is not supported in this version.

Example XML Request message:

```
<SObjectsCollection>
   <allOrNone>false</allOrNone>
   <records>
      <attributes>
         <type>Account</type>
      </attributes>
      <Name>John Doe</Name>
   </records>
   <records>
      <attributes>
         <type>Account</type>
      </attributes>
      <Name>Jane Doe</Name>
   </records>
</SObjectsCollection>
```

Note that sObject collection is mostly suited for scenarios where the same type of Salesforce entity is used. The Composite operation is more suited when different types of Salesforce objects are involved.

The Adapter provides the capability to automatically call Salesforce multiple times (orchestration) in cases where the payload contains more than 200 records. For that, the “Enable Multiple Batch Processing” checkbox needs to be selected. Note that this feature is only supported when allOrNone is set to false. The Adapter also supports SOQL lookup queries which enable the User to embed a SOQL query within an existing Payload. For that, the “Enable Lookup Query” checkbox needs to be selected. Note that the lookup query only works for XML.

The Adapter also supports SOQL lookup queries which enable the User to embed a SOQL query within an existing Payload. The Adapter resolves the SOQL lookup query in the payload by replacing the SOQL lookup query in the payload with the returned value of the SOQL query itself. This way, the User can enrich the payload with the value returned from the specified SOQL lookup query.

Its behavior can be further illustrated by the examples in the table below.

The Adapter also supports SOQL lookup queries which enable the User to embed a SOQL query within an existing Payload.

Query	Behavior
<records> <Name>Demo123</Name> <attributes> <type>Account</type> </attributes> <Id queryType="SOQL">SELECT id from Account where CustomExternalId __c='53153011024'</Id> </records>	The query is performed in Salesforce and the value of the returned ID is assigned to the <Id> tag before sending data to Salesforce.
<records> <Name>Demo456</Name> <attributes> <type>Account</type> </attributes> <Id queryType="SOQL">&lt;![CDATA[SELECT id from Account where CustomExternalId __c='53153011024']]&gt;</Id> </records>	The result is the same as the previous query. Note that a query can be explicitly specified, or it can start with "<![CDATA[“and ends with “]]>”
<records> <Name>Demo123</Name> <attributes> <type>Account</type> </attributes> <Id queryType="SOQL">SELECT id from Account limit 3</Id> </records>	The query is performed in Salesforce. Since the query expects to return three records, only the first ID is used. The remaining records are ignored.
<records> <Name>Demo123</Name> <attributes> <type>Account</type> </attributes> <Id queryType="SOQL">&lt;![CDATA[SELECT id,name from Account where CustomExternalId __c='53153011024']]&gt;</Id> </records>	The query will result in two columns. The Adapter will pick the first column (id) and will ignore the second column.
<records> <Name>Demo123</Name> <attributes> <type>Account</type> </attributes> <Id queryType="SOQL" defaultValue="0011Q00002PzXyTQAV">SELECT id,name from Account where CustomExternalId __c='53153011024'</Id> </records>	The defaultValue attribute indicates that in case the query returns no results, a default value needs to be assigned. Without the defaultValue attribute, an empty string value will be assigned.
<records> <Name>Demo123</Name> <attributes> <type>Account</type> </attributes> <Id queryType="SOQL" defaultValue="null">SELECT id from Account where CustomExternalId__c='53153011024'</Id> </records>	The defaultValue=”null” indicates that in case the query returns no results, a null value will be assigned. This means that the field will be nullified.

6.2.1.19 Salesforce Objects – Custom Request

The Custom Request operation enables the sending of any Salesforce REST-based API to Salesforce. This option can be used for any operation that is currently not covered by the set of operations provided in the Adapter. Add the needed URL and select the corresponding method.

Example: If the VersionData of the SObject names “ContentVersion” is to be retrieved, the Adapter can be configured as shown below:

Example Configuration	
Operation	Salesforce Object – Custom Request
Input Payload Format	Application/JSON or Application/XML or Application/octetstream
Response Payload Format	Application/JSON or Application/XML or Application/octetstream
Method Name	GET
Resource Relative URL	/services/data/v48.0/sobjects/ContentVersion/06836000001k0O9AAI/VersionData

6.2.1.20 Salesforce Objects – List Organization Limits

The List Organization Limits operation enables one to list the organization limits in terms of API call limits and current usage for the Salesforce tenant. It will return both the max limit for the Salesforce org and the remaining number of calls or events left.

For example, part of a response message for Salesforce – List Organization Limits in application/JSON:

```
"DailyApiRequests": {
   "Max": 5000,
   "Remaining": 4937
}
```

Example Configuration	
API Version	52.0
Operation	Salesforce – List Organization Limits
Payload Format	Application/JSON or Application/XML

6.2.2 Place Order API

The Place Order API supports eight operations that retrieve or push data to Salesforce. These operations are categorized into three groups; Account, Contract, and Order.

The configuration for each operation can differ in terms of dynamic setting and whether the API expects a request body or returns a response body.

There are a set of common fields that are often used across different operations in the Place Order API. These common fields are described below:

- *Order ID:* The unique identifier (ID) of the Order that is being processed. Similar to the Salesforce Object ID, this is a unique identifier that can be found in the last part of the URL in the Order record view in the Salesforce Console.
- *Contract ID:* The unique identifier (ID) of the Contract that is being processed. Within the context of the place order API, Contracts act as the parent Salesforce Object and Orders as the child Salesforce Object.
- *Filter Name:* Contains the name of the child object field to be used for filtering purposes.
- *Filter Value:* Contains the value of the child object field to be used for filtering purposes. Together, the Filter Name and Filter Value can help to retrieve specific fields of an Order or Contract. An example is provided in Section 6.2.2.4.

The eight operations are explained in further detail in the next sections.

6.2.2.1 Account – Add Contracts and Orders

This operation creates contract-based Orders in Salesforce. The table below shows an example configuration.

Example Configuration	
API Version	50.0
Operation	Account - Add Contracts and Orders

Payload Format	Application/JSON or Application/XML
Pretty Print	Enable

6.2.2.2 Account – Add Orders

This operation can create multiple Orders with their order items (also known as Order Products) for an existing Account.

6.2.2.3 Contract - Add Orders

This operation creates an Order with Order Products for an existing Contract. The table below shows an example configuration.

Example Configuration	
API Version	50.0
Operation	Contract - Add Orders
Payload Format	Application/JSON or Application/XML
Contract ID	80036000000dq8E
HTTP Protocol	Default
Pretty Print	Enable

6.2.2.4 Contract - Filter Details

This operation queries a Contract based on a filter criterion specified in the Contract ID, Filter Name, and Filter Value field. The table below shows an example configuration.

In this example, Orders that have the StatusCode equal to “Activated” with the contract with ID “80036000000dq8E” are filtered.

Example Configuration	
API Version	50.0
Operation	Contract - Filter Details
Payload Format	Application/JSON or Application/XML
Contract ID	80036000000dq8E
Filter Name	contract.order.StatusCode
Filter Value	‘Activated’
Pretty Print	Enable

6.2.2.5 Contract – Get

This operation retrieves the details of a Contract record in Salesforce based on its ID. Note that it also returns its child Orders, Order Products, and Custom Objects.

6.2.2.6 Order – Add Products

This operation adds Order Products to an existing Order. It is required to specify the order ID.

Example Configuration	
API Version	57.0
Operation	Order - Add Products
Payload Format	Application/JSON or Application/XML
Order ID	80036000000dq8E
HTTP Protocol	Default
Pretty Print	Enable

6.2.2.7 Order – Filter Details

This operation queries Orders based on filter criteria. The table below shows an example configuration.

Example Configuration	
API Version	50.0
Operation	Order - Filter Details
Payload Format	Application/JSON or Application/XML
Contract ID	80036000000dq8E
Filter Name	order.orderItems.effectivedate
Filter Value	2013-08-05
Pretty Print	Enable

6.2.2.8 Order – Get

This operation retrieves the details of an Order Record in Salesforce based on its ID. Note that it also returns its Order Products and child Custom Objects.

6.2.3 Bulk API

The Bulk API supports twelve operations to retrieve or push large sets of data. It uses the concepts of Jobs and Batches. A Job works as a container that holds one or more Batches. Each Batch can contain ten thousand records. Note that the inserted Batches need to conform to the settings of the Job.

Figure 6.5**Figure** depicts the relationship between a Job, Batches, and the processing of data. After the Batches are added to a specific Job, the included Records are processed in Salesforce. Besides being able to create, abort, close, and retrieve jobs, it is also possible to retrieve the status and the results with specific operations that this Adapter provides.

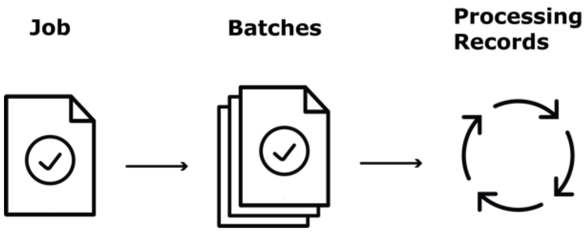


Figure 6.5 Bulk API – schematic overview of jobs and batches

There are a set of common fields that are often used across different operations in the Bulk API. These common fields are described below:

- *Content Type:* Specifies the file content of the batches. Available options are CSV, JSON, XML, ZIP CSV, ZIP JSON and ZIP XML. This implies that a batch can be created in CSV, JSON, XML, and its ZIP equivalent.
- *Job ID:* The unique identifier of the Job being processed.
- *Batch ID:* The unique identifier of the Batch being processed.
- *Result ID:* represents the unique identifier for the Result in the response to the batch result list request. For Bulk API Query operations, the Result ID may be used to retrieve the results of a Bulk Query.

Each of the Bulk API operations is explained in the next sections.

6.2.3.1 Job – Create

This operation can be used to create a Job in Salesforce. In the request message, the properties of the Job can be specified. This acts as a specification for Batches to be attached later. The table below shows an example configuration.

Example Configuration	
API Version	50.0
Operation	Job – Create
Payload Format	Application/JSON or Application/XML
Pretty Print	Enable

6.2.3.2 Job – Get

This operation retrieves all details for an existing Job. The table below shows an example configuration.

Example Configuration	
API Version	50.0
Operation	Job – Get
Job ID	75036000006MDYb
Payload Format	Application/JSON or Application/XML
Pretty Print	Enable

6.2.3.3 Job – Abort

This operation aborts an existing Job. When a Job is identified and aborted, no more Records are processed. Changes to data that have already been committed are not rolled back.

6.2.3.4 Job – Close

This operation closes the Job after it has finished processing. When a Job is identified and closed, no more Batches can be added to it.

6.2.3.5 Job – Get all Batches

Information about all batches associated with a specific Job can be retrieved by executing this operation.

6.2.3.6 Batch – Create

This operation adds a new Batch to an existing Job. The request body contains a list of Records to be processed. The table below shows an example configuration.

Data Insert	
API Version	50.0
Operation	Batch - Create
Content Type	XML/ZIP CSV/ZIP JSON/ZIP XML
Job ID	75036000006MDYb
Pretty Print	Enable

6.2.3.7 Batch – Get

This operation retrieves information associated with an existing Batch.

Data Insert	
API Version	50.0
Operation	Batch - Get
Payload Format	Application/JSON or Application/XML
Job ID	75036000006MDYb

Batch ID	75136000005kbnC
Pretty Print	Enable

6.2.3.8 Batch – Get Request

This operation retrieves a Batch Request. It is also possible to retrieve the original request which was used to initiate the Batch.

6.2.3.9 Batch – Get Result

This operation retrieves the results for a specific Batch. As Batches are processed asynchronously, this operation is needed to retrieve results after Batch processing has been completed. Note that the content type of the result matches one of the input request records. This means that in case the Batch is a CSV file, the response will also be in CSV format. Alternatively, a Bulk API batch can be monitored in the Salesforce console.

6.2.3.10 Batch Query – Create

This operation adds a batch query to a bulk query job.

Example Configuration	
API Version	50.0
Operation	Batch Query - Create
Payload Format	Application/JSON or Application/XML
Job ID	75036000006MDYb
SOQL Query	SELECT ID, Name FROM Account
Pretty Print	Enable

6.2.3.11 Batch Query – Get Result

When a batch query has completed processing records, results can be displayed with this operation using the Result ID returned by the operation Batch Query – Get Result IDs. Alternatively, a Bulk API batch can be monitored in Salesforce.

Example Configuration	
API Version	50.0
Operation	Batch Query - Get Result
Payload Format	Application/JSON or Application/XML
Job ID	75036000006MDYb
Batch ID	75136000005kbnC
Result ID	752360000033TwU
Pretty Print	Enable

6.2.3.12 Batch Query – Get Result IDs

This operation retrieves a list of Result IDs. These Result IDs can then be used in the Batch Query – Get Result operation to retrieve the actual results. Alternatively, Result IDs can be viewed in the Salesforce console.

6.2.4 Bulk API 2.0

The Bulk API 2.0 supports twelve operations to retrieve or push large sets of data. In a comparable way to Bulk 1.0 discussed in Section 6.2.3, it uses the concepts of jobs and batches. This component works with Salesforce API version 47.0 and later. It is important to be aware that Salesforce places a set of limitations on Bulk API 2.0.

6.2.4.1 Job – Create

This operation can be used to create a Job in Salesforce. In the request message, the properties of the Job can be specified. This acts as a specification for Batches to be attached later. The table below shows an example configuration.

Example Configuration	
API Version	50.0
Operation	Job – Create
Advance Job Properties	X
Object	Contact
Operation Name	Insert
Column Delimeter for Job	Comma character (,)
Content Type	CSV
Line Ending	LF-LineFeed character
Payload Format	Application/JSON or Application/XML
Pretty Print	Enable

This operation does not support the property ‘concurrencyMode’ in the request body.

6.2.4.2 Job – Get

This operation retrieves all details for an existing Job. The table below shows an example configuration.

Example Configuration	
API Version	50.0
Operation	Job – Get
Job ID	75036000006MDYb
Job Result Type	None
Payload Format	Application/JSON or Application/XML
Pretty Print	Enable

6.2.4.3 Job – Abort

This operation aborts an existing Job. When a Job is identified and aborted, no more records are processed. If changes to data have already been committed, they are not rolled back. The table below shows an example configuration.

Example Configuration	
API Version	50.0
Operation	Job – Abort
Job ID	75036000006MDYb
Payload Format	Application/JSON or Application/XML
HTTP Protocol	Default
Pretty Print	Enable

6.2.4.4 Job – Close

This operation closes the Job after it has finished processing. When a Job is identified and closed, no more Batches can be added to it. The table below shows an example configuration.

Example Configuration	
API Version	50.0
Operation	Job – Close
Job ID	75036000006MDYb
Payload Format	Application/JSON or Application/XML
Pretty Print	Enable

6.2.4.5 Job – Get all Jobs

This retrieves information about all Jobs that are available in Salesforce. The table below shows an example configuration.

Example Configuration	
API Version	50.0
Operation	Get – All Jobs
Enable PkChunking	X
Job Type	Classic
Payload Format	Application/JSON or Application/XML
Pretty Print	Enable

6.2.4.6 Job – Delete

This operation deletes and removes a specific Job. The table below shows an example configuration.

Example Configuration	
API Version	50.0
Operation	Job – Delete
Job ID	75036000006MDYb
Payload Format	Application/JSON or Application/XML
Pretty Print	Enable

6.2.4.7 Job Query – Create

This operation creates a Query Job. The table below shows an example configuration.

Example Configuration	
API Version	50.0
Operation	Query – Create Query Job
Payload Format	Application/JSON or Application/XML
Pretty Print	Enable

Using the above configuration, all required attributes can be specified in the XSD which can be generated using the **Error! Reference source not found.**.

Note that it is also possible to specify Job properties using the Adapter interfaces instead of using details from the Payload. This is achieved by selecting the “Advanced Job Properties” checkbox. This allows for the configuration of the Operation, Delimiter, Content Type, Line Ending, and SOQL Query for the Query Job.

Example Configuration	
Operation Name	QueryAll
Column Delimiter for Job	Comma Character (,)
Content-Type	CSV
Line Ending	LF—linefeed character
SOQL Query	SELECT Id FROM Account

6.2.4.8 Job Query – Get All Query Jobs

This operation retrieves all information about Query Jobs in Salesforce. The table below shows an example configuration.

Example Configuration	
API Version	50.0
Operation	Query – Get All Query Jobs
Payload Format	Application/JSON or Application/XML
Pretty Print	Enable

6.2.4.9 Job Query – Get Information

This operation retrieves all information about a specific Query Job in Salesforce. The table below shows an example configuration.

Example Configuration	
API Version	50.0
Operation	Query – Get Query Information
Query Job ID	75036000006MDYb
Payload Format	Application/JSON or Application/XML
Pretty Print	Enable

6.2.4.10 Job Query – Abort

This operation aborts and stops a specific Query Job. The table below shows an example configuration.

Example Configuration	
API Version	50.0
Operation	Query – Abort Query Job
Query Job ID	75036000006MDYb
Payload Format	Application/JSON or Application/XML
HTTP Protocol	Default
Pretty Print	Enable

6.2.4.11 Job Query – Get Results

This operation retrieves the result for a specific Query Job. The table below shows an example configuration.

Example Configuration	
API Version	52.0
Operation	Query – Get Query Results
Query Job ID	${property.JobID}
Max Records	${property.Max}
Auto Pagination	Enable
Pretty Print	Enable

By default, a maximum of 2000 Records is returned for each result page. It is possible to return the entire result set (all pages) using the “Auto Pagination” checkbox. The Max Records field allows the User to specify the maximum number of records to retrieve per set of results for the query.

6.2.4.12 Upload – Job Data

This operation facilitates the upload of CSV data to an existing Job. It requires a CSV payload. The table below shows an example configuration.

Example Configuration	
API Version	50.0
Operation	Upload – Job Data
Job ID	${property.JobID}
Payload Format	Application/JSON or Application/XML

6.2.4.13 Query Job – Delete

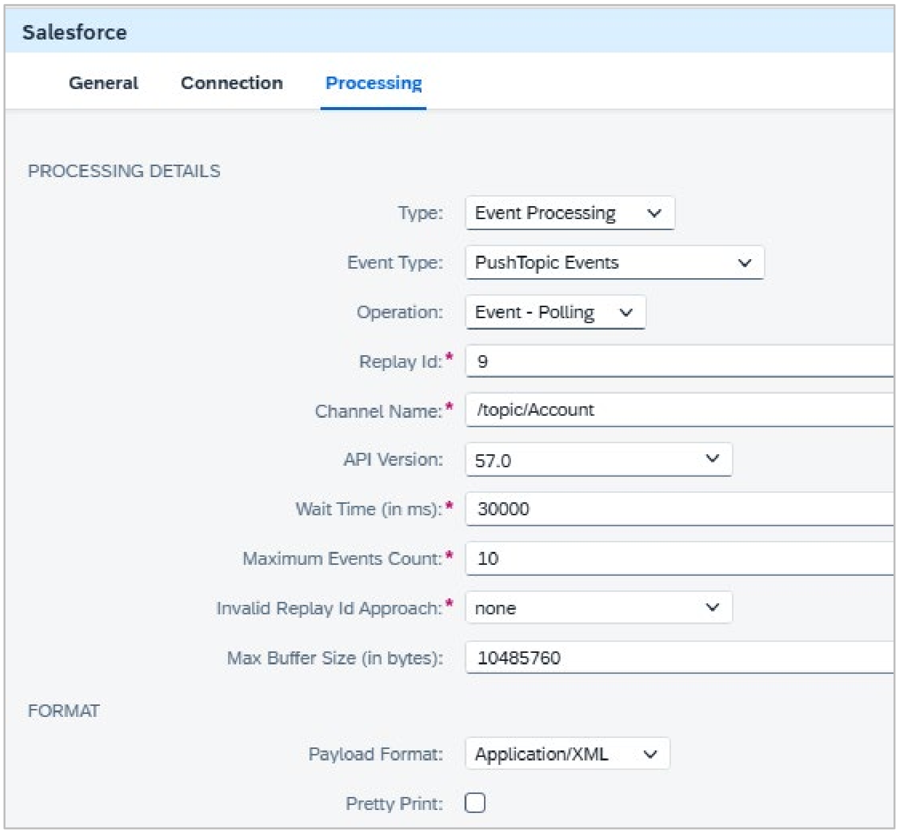
This operation deletes a Query Job. The table below shows an example configuration.

Example Configuration	
API Version	50.0
Operation	Query Job - Delete

Payload Format	Application/JSON or Application/XML

6.2.5 Event Processing

The Event processing feature enables Cloud Integration to poll Events from Salesforce. Using this operation, it is possible to retrieve Events from Salesforce based on the latest ReplayID and the number of Events to be processed. An example configuration for this operation is shown in Figure 6.6.



PROCESSING DETAILS

Type: Event Processing

Event Type: PushTopic Events

Operation: Event - Polling

Replay Id:* 9

Channel Name:* /topic/Account

API Version: 57.0

Wait Time (in ms):* 30000

Maximum Events Count:* 10

Invalid Replay Id Approach:* none

Max Buffer Size (in bytes): 10485760

FORMAT

Payload Format: Application/XML

Pretty Print: □

Figure 6.6 Processing events with the Receiver Adapter

The different properties of the Adapter shown in Figure 6.6 are described below:

- *Event Type:* Specifies the category of the Event to be retrieved from Salesforce. Possible values include PushTopic Events, Platform Events, Change Data Capture Events, and Generic events. These distinct types of events are explained in Section 5.2.1.
- *Operation:* Specifies the operation to perform. Currently, only the Event Polling option is available.
- *Replay Id:* Specifies the Replay Id which is populated by Salesforce and refers to the position of the Event in the event stream. Note that the Replay Id values are not guaranteed to be contiguous for consecutive events. By specifying the value of the Replay Id, the integration flow retrieves events that are within the retention window and that have followed the specified Replay Id.
  Example: In case a Replay Id 200 is specified, the integration flow receives all messages with a Replay Id greater than 200.
  This field supports dynamic properties and a header.
  Example: ${property.replayID}.
- *Channel Name*: Made of a Topic name preceded with a prefix. For instance, a PushTopic can be prefixed with “/topic/”. Example: /topic/MyPushTopic. Note that the names are case-sensitive.
- *API Version:* The API Version specifies the version of the Salesforce API.

- *Wait Time (in ms):* Specifies how long the Receiver Adapter connects to Salesforce to retrieve Events. The Adapter automatically disconnects after this period.
- *Maximum Events Count:* Specifies the maximum number of Events that will be collected in a single run.
- *Invalid Replay Id Approach:* Select the approach to be applied when an invalid replayId has been found.
- *Max Buffer Size (in bytes):* Specify the maximum permissible capacity (in bytes) for Buffer.
- *Payload Format:* This field specifies the format of the request message to be sent to and the response to be returned from Salesforce. Possible values include Application/XML and Application/JSON. The default value is Application/XML.
- *Pretty Print:* Select the Pretty Print checkbox to nicely format the XML payload and improve its readability.

In case an iflow uses a timer step in combination with the Event processing operation, ensure that the scheduling period is higher than the value specified in the Wait Time property.

6.2.6 REST Apex Call

The Apex Call feature enables Cloud Integration to call custom classes and methods exposed to external applications from Salesforce. The Adapter can call Apex Classes and methods using this operation. This operation supports both JSON and XML. The table below shows an example configuration.

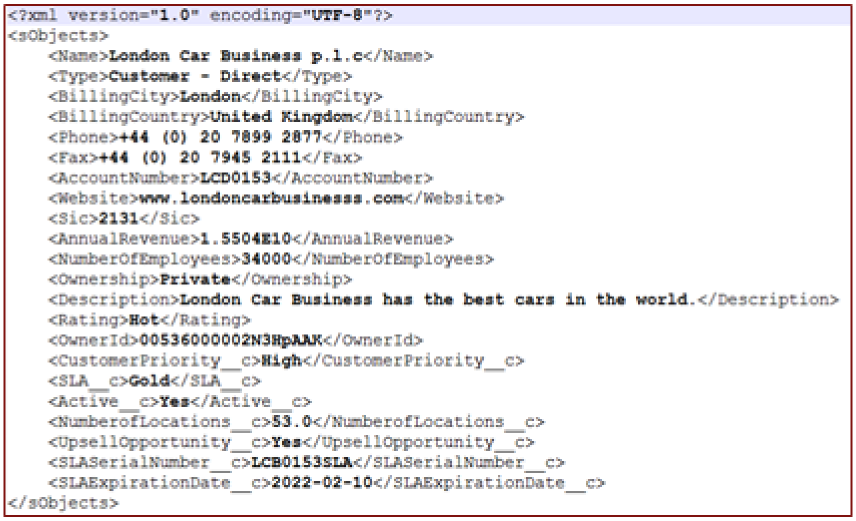
Example Configuration	
Operation	GET
Resource Relative URL	services/apexrest/AccountApex/0011Q00002NlppdQAB
Payload Format	Application/JSON or Application/XML
HTTP Protocol	Default
Pretty Print	Enable

7 PAYLOAD AND DYNAMIC CONFIGURATION

The Salesforce Adapter supports different message formats. It also supports dynamic configuration of its properties using both the header and property features of the SAP Cloud Integration Exchange. Both header and property features of the exchange can be used in most Salesforce Adapter properties.

7.1 Message Payloads

Many operations require a Message as input to be sent to Salesforce. For example, a message body that specifies the fields and values to be inserted is required to create an Object in Salesforce using the Salesforce Object – Create operation. After specifying the Salesforce Object Type in the Adapter, a payload must be included. An example payload to create an Account Salesforce Object can be seen in Figure 7.1.



```
<?xml version="1.0" encoding="UTF-8"?>
<sObjects>
    <Name>London Car Business p.l.c</Name>
    <Type>Customer - Direct</Type>
    <BillingCity>London</BillingCity>
    <BillingCountry>United Kingdom</BillingCountry>
    <Phone>+44 (0) 20 7899 2877</Phone>
    <Fax>+44 (0) 20 7945 2111</Fax>
    <AccountNumber>LCD0153</AccountNumber>
    <Website>www.londoncarbusinesss.com</Website>
    <Sic>2131</Sic>
    <AnnualRevenue>1.5504E10</AnnualRevenue>
    <NumberOfEmployees>34000</NumberOfEmployees>
    <Ownership>Private</Ownership>
    <Description>London Car Business has the best cars in the world.</Description>
    <Rating>Hot</Rating>
    <OwnerId>00536000002N3HpAAK</OwnerId>
    <CustomerPriority__c>High</CustomerPriority__c>
    <SLA__c>Gold</SLA__c>
    <Active__c>Yes</Active__c>
    <NumberofLocations__c>53.0</NumberofLocations__c>
    <UpsellOpportunity__c>Yes</UpsellOpportunity__c>
    <SLASerialNumber__c>LCB0153SLA</SLASerialNumber__c>
    <SLAExpirationDate__c>2022-02-10</SLAExpirationDate__c>
</sObjects>
```

Figure 7.1 Create an Account Salesforce Object payload

The XSD Generator (configuration details in Section 8.3) helps in two ways. First, it generates the XSDs that may be used in the mappings of an integration scenario. Following the example, the XSD includes elements/properties of the sObjects.

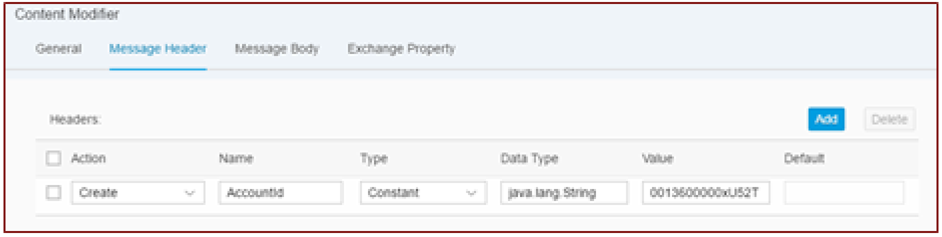
If a Create operation in Salesforce is performed, use the correct request XSD as the target of the mapping before the Request-Reply. This assists in creating a valid input message (payload) for Salesforce.

The XSDs obtained from the Eclipse Plug-in can help to provide a good understanding of the request message that various Salesforce operations expect as well as the response messages expected from Salesforce.

7.2 Basic Dynamic Configuration Example

This section illustrates the use of dynamic values in the Adapter property. It depicts a simple Salesforce Object – Get operation configuration using a dynamic header reference.

1. Use a Content Modifier before the Request-Reply step in the integration flow. The Content Modifier should create a Message Header. The name of the Message Header is the name required to be referred to in the Salesforce Adapter. Any name is allowed here. Note that it is possible to use properties instead of a header to achieve the same results. As seen in the example, Constant is selected as the Type and Salesforce Object ID is entered in the Value field.

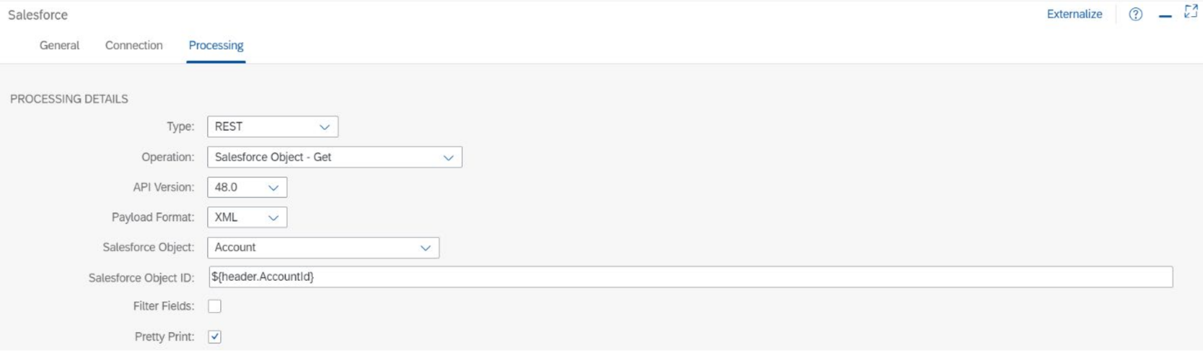


Headers:

	Action	Name	Type	Data Type	Value	Default
□	Create	AccountId	Constant	java.lang.String	0013600000xU52T	

Figure 7.2 Message Header in Content Modifier

2. The Salesforce Receiver Adapter (which is linked to the Request-Reply step) needs to refer to this header by using **${header.name}**. To refer to a property, a syntax similar to **${property.name}** should be used. As seen in Figure 7.3, the AccountID header is referred to with the syntax $**{header.AccountId}**. The Salesforce Adapter automatically replaces this reference with the ID 0013600000xU52T.



Salesforce

General Connection Processing

Externalize

PROCESSING DETAILS

Type: REST

Operation: Salesforce Object - Get

API Version: 48.0

Payload Format: XML

Salesforce Object: Account

Salesforce Object ID: ${header.AccountId}

Filter Fields: ☐

Pretty Print: ☑

Figure 7.3 Adapter Specific Tab for Salesforce Object - Get Operation

7.3 Handling Null Values

When performing Write operations like Create, Update and Upsert in Salesforce, the integration scenario may require that some properties are set with a null value. This can be achieved in diverse ways depending on whether XML or JSON is selected in the Adapter.

<u>In XML:</u>
Use null value as in the payload.

Example:

```
<sObjects>

            <Name>Company1</Name>

               <BillingCity>null</BillingCity>

               <BillingState>Amsterdam</BillingState>

</sObjects>
```

Additionally, it is required to set select the “Handle XML Null Values” checkbox in the Processing Tab of the Adapter.

<u>In JSON:</u>
In JSON, it is sufficient to set use null in the JSON property.

Example:

```
{

  "Name" : "Company2",

  "BillingCity" : null,

  "BillingState" : "Catalunya"

}
```

7.4 Treat all XML Payload Values as String

When performing operations like sObject Collections, Composite, and Place Order, the integration scenarios may require that all the values in the XML payload are treated as a string before sending it to Salesforce.

For example, XML <Name>123E268</Name> is translated to "Name" : "123E268".

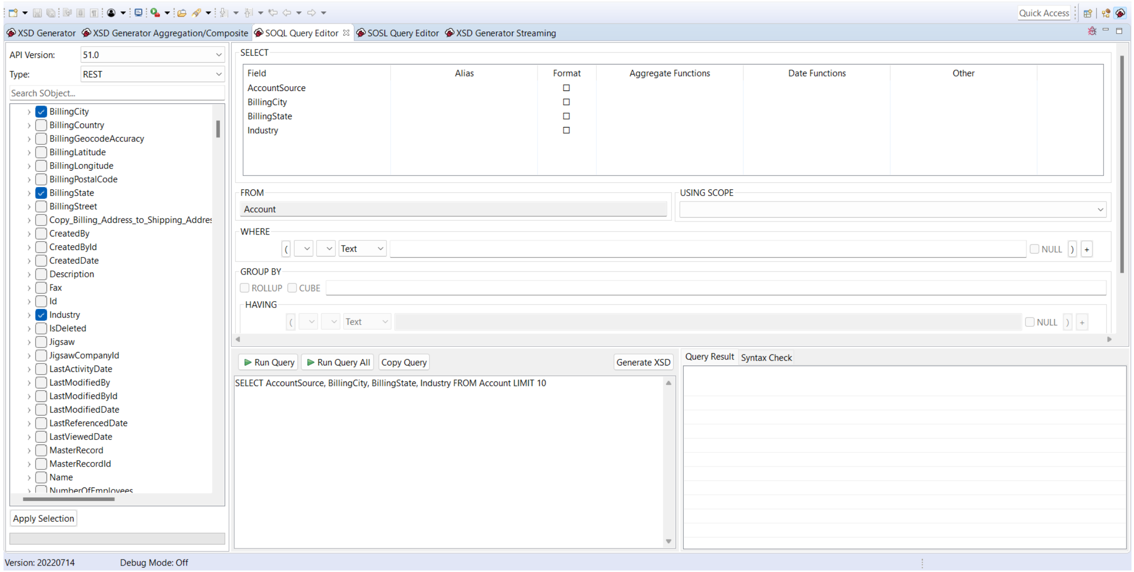
It is required to select the “Treat all XML Payload Values as String” checkbox in the Processing Tab of the Adapter.

8 ECLIPSE PLUG-IN CONFIGURATION

8.1 SOQL Query Editor

The SOQL Query Editor helps build and test well-formatted queries in the Salesforce Object Query Language. It supports the building of queries for the Rest and Bulk API. Depending on the selected API, different functionality is supported according to the Salesforce documentation. The offset functionality, for example, can be used in the Rest API, but not in the Bulk API. The Query Editor is limited to queries within one object since relationships and joins have not been implemented. The SOQL Query Editor is the first tab in the Salesforce Cloud Integration Adapter Tool (see Figure 8.1).

The following steps show how to create a query with the SOQL Query Editor. Descriptions of the sections and elements of the Editor are also included.



API Version: 51.0

Type: REST

- ☑ BillingCity
- □ BillingCountry
- □ BillingGeocodeAccuracy
- □ BillingLatitude
- □ BillingLongitude
- □ BillingPostalCode
- ☑ BillingState
- □ BillingStreet
- □ Copy_Billing_Address_to_Shipping_Addre:
- □ CreatedBy
- □ CreatedById
- □ CreatedDate
- □ Description
- □ Fax
- □ Id
- ☑ Industry
- □ IsDeleted
- □ Jigsaw
- □ JigsawCompanyId
- □ LastActivityDate
- □ LastModifiedBy
- □ LastModifiedById
- □ LastModifiedDate
- □ LastReferencedDate
- □ LastViewedDate
- □ MasterRecord
- □ MasterRecordId
- □ Name
- □ NumberOfEmployees

Field	Alias	Format	Aggregate Functions	Date Functions	Other
AccountSource		□			
BillingCity		□			
BillingState		□			
Industry		□			

FROM: Account

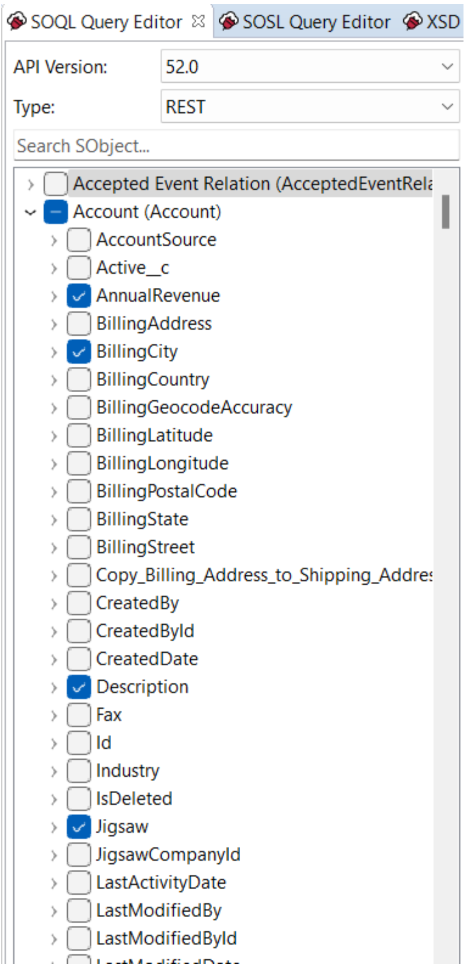
```
SELECT AccountSource, BillingCity, BillingState, Industry FROM Account LIMIT 10
```

Version: 20220714 Debug Mode: Off

Figure 8.1 SOQL Query Editor

8.1.1 Objects and Fields

Objects and their subsequent fields can be selected in the left component of the SOQL Query Editor, as shown in Figure 8.2. At the bottom of the component is the progress bar that runs when data is retrieved from Salesforce and prepared for display.



API Version: 52.0

Type: REST

Search SObject...

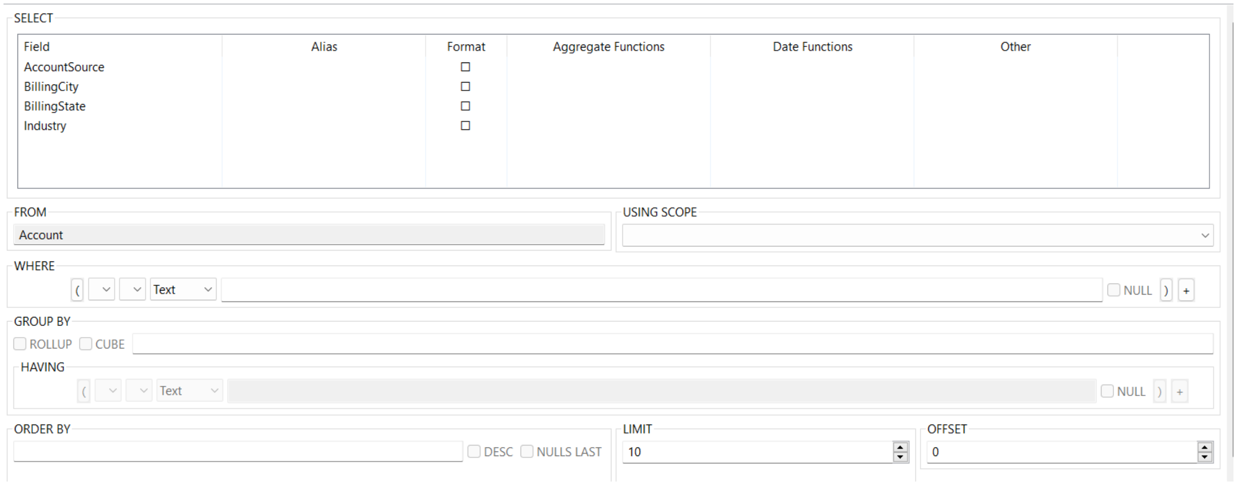
- ☐ Accepted Event Relation (AcceptedEventRela
- ▣ Account (Account)
  - ☐ AccountSource
  - ☐ Active__c
  - ☑ AnnualRevenue
  - ☐ BillingAddress
  - ☑ BillingCity
  - ☐ BillingCountry
  - ☐ BillingGeocodeAccuracy
  - ☐ BillingLatitude
  - ☐ BillingLongitude
  - ☐ BillingPostalCode
  - ☐ BillingState
  - ☐ BillingStreet
  - ☐ Copy_Billing_Address_to_Shipping_Addres
  - ☐ CreatedBy
  - ☐ CreatedById
  - ☐ CreatedDate
  - ☑ Description
  - ☐ Fax
  - ☐ Id
  - ☐ Industry
  - ☐ IsDeleted
  - ☑ Jigsaw
  - ☐ JigsawCompanyId
  - ☐ LastActivityDate
  - ☐ LastModifiedBy
  - ☐ LastModifiedById

Figure 8.2 Objects and Field Overview

1. Select the API version in the Version dropdown. The version is synchronized between the different tabs. If a version is selected in another tab, this will be automatically selected here.
2. Select the API type with which the Query will be created for. Available options are REST BULK.
3. Optional: Search for the object the Query to be created will query with the "Search sObject…" field. For example, type "acc" and the list below will only show objects with that letter combination such as "Account" or "Access".
4. Click on the arrow in front of an object to load all the fields of the object. Alternatively, the box in front of the object can be selected. This will load and select all fields of that object.
5. Select the objects or specific fields that the query to be created is desired to search for. The selected fields will be visible in the Select section.

8.1.2 Query Options

The component on the top right-hand side of the SOQL Query Editor enables the setting of different Query options. The different syntax elements have their sections labeled accordingly as seen in Figure 8.3.



Field	Alias	Format	Aggregate Functions	Date Functions	Other
AccountSource		□			
BillingCity		□			
BillingState		□			
Industry		□			

FROM: Account

LIMIT: 10

OFFSET: 0

Figure 8.3 Query options component

- The Select section displays the following fields.
  - Formatting fields to local settings can be done by selecting the box in the Format column. If a field is formatted, it is possible to give it an alias.
  - In the Aggregate Functions column, the desired function can be selected. Click the arrow to show a dropdown with available functions (e.g., AGV, COUNT, SUM). Available options depend on the data type of the field, for example, AVG and SUM are available for numbers, but not for text fields.
  - Date Functions are available for Date/DateTime fields. This column enables formatting the field, for example, as a calendar year or month. Click on the arrow to show the dropdown with available options and select accordingly.
  - The Other column holds other functions such as ”ConvertCurrency”. This dropdown is available when there is a function available.
- The From section shows which Salesforce Object is selected. This cannot be changed in this section. The fields change dynamically by selecting fields from a different Object in the Objects and Fields component.
- In the Using Scope section, the scope of the query can be set. If no scope is entered in the field, the default scope of the Query includes “everything”.
- In the Where section, the field and operator can be selected for the statement. Depending on the selected field, the input changes accordingly.
  - Text fields facilitate free text input.
  - For picklists fields, the input is a dropdown with all available options.
  - Numeric fields allow numbers only.
  - For DateTime fields, the input is split into two parts. The first part is a calendar to select the date from. The second is an input field for a time.

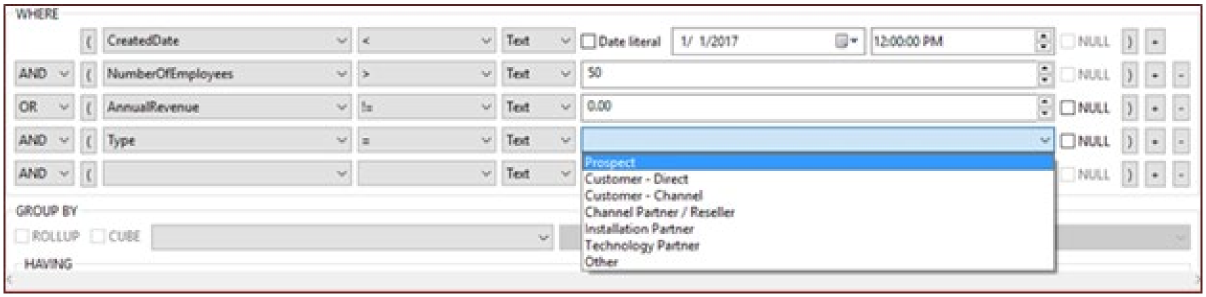
  Select the Null box if records without values are to be queried.

  Additional Where statements can be added by clicking on the plus (+) sign on the right-hand side of the section. This adds a new row to create an extra statement. New rows are preceded by a dropdown to select And/Or.

  To remove a row, click the minus (-) sign on the right-hand side of the row.

  See Figure 8.4 for an overview of the Where section.
- The Group By section enables the grouping of query results. Grouping options are Group by Rollup or Group by Cube (to include subtotals in the query).

  Having is a sub-section of Group By where the query results with having statements are grouped. This section functions in the same way as the Where section described above.



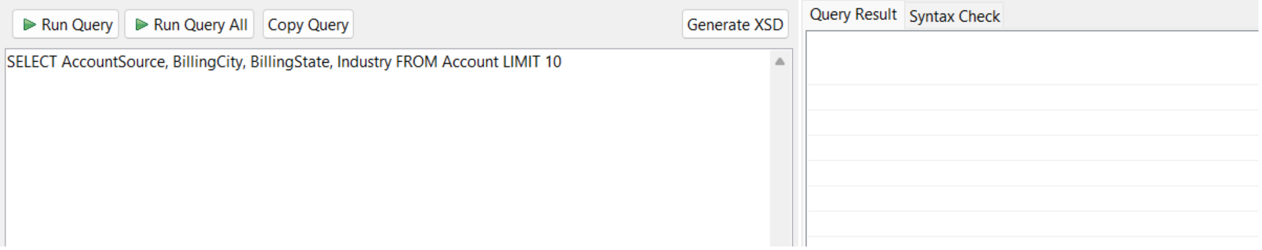
	(	CreatedDate	<	Text	□ Date literal 1/ 1/2017 12:00:00 PM	□ NULL	)
AND	(	NumberOfEmployees	>	Text	50	□ NULL	)
OR	(	AnnualRevenue	!=	Text	0.00	□ NULL	)
AND	(	Type	=	Text		□ NULL	)
AND	(			Text		□ NULL	)

Figure 8.4 Overview of the WHERE section

- In the Order By section, the field to sort on can be selected. The default order that the results are displayed is ascending. Selecting the DESC box sorts the query results in descending order. Additionally, records without an entry for the selected fields are placed at the bottom of the results by selecting the box Nulls Last.
- The Limit section allows setting the limit of results returned from a query. This is only available for REST and Bulk API.
- In the Offset section, it is possible to set the starting row of the returned result set. This is only available for REST API.

8.1.3 Results

Created Queries can be tested using the component on the bottom left-hand side of the Editor. Figure 8.5 shows an overview of this component.



```
SELECT AccountSource, BillingCity, BillingState, Industry FROM Account LIMIT 10
```

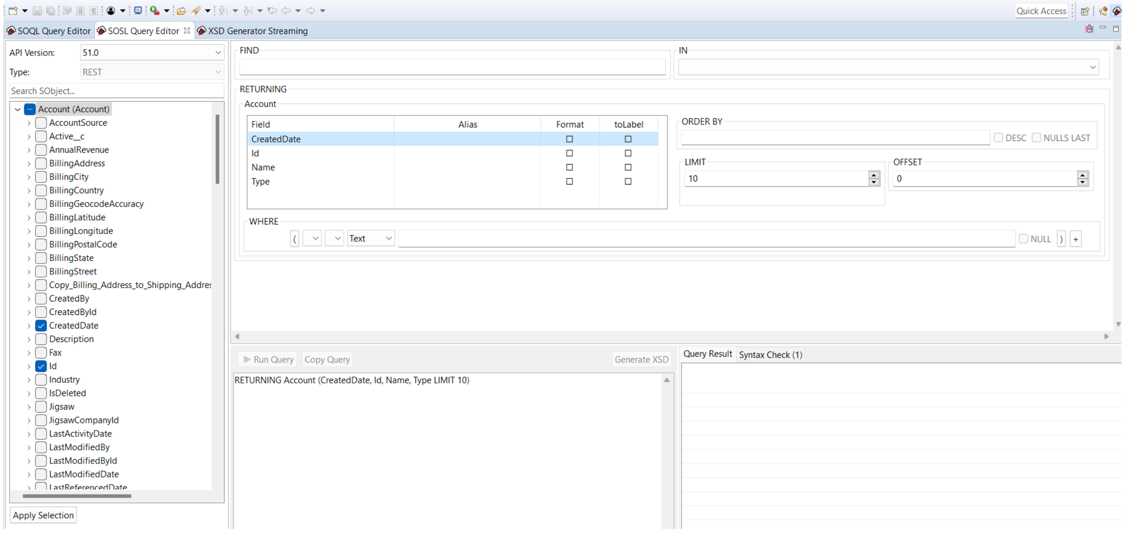
Figure 8.5 Overview of result component

- The text field on the left shows the created query. This query cannot be edited manually. It can only be changed by the settings in the other components.
- Click on Run Query to test the query and verify if the results are as expected. Note that takes some time to process this request. The button indicates when the query is running and returns to its original state when the query is finished.
- To test the Query and include previously Salesforce Records that match the query parameters, click on Press Run Query All.
- The Query Result tab on the right displays the results in a table.
- If the query is not valid, the Syntax Check tab on the right can be consulted. This displays additional information such as the Message and Error Code of the Response.
- If the results are satisfactory, the query can be copied for use in the Salesforce Cloud Integration Adapter. To do this, the “Copy Query” button can be clicked on to copy the query to the clipboard. Selecting the query and copying it manually is also an option.
- An XSD for the query result can be generated by pressing the “Generate XSD” button. The generated XSD pops up on a new screen. Options to either cancel or save the XSD are available in this pop-up. The generated XSD can be used to map the query results in an integration scenario.

8.2 SOSL Query Editor

The SOSL Query Editor helps build and test well-formatted searches in the Salesforce Object Search Language. It supports the building of searches for the Rest API. In a search query, data on multiple Objects can be looked up, for example, in both Account and Contact.

The SOSL Query Editor is the second tab in the Salesforce Cloud Integration Adapter Tool perspective, as shown in Figure 8.6. The following steps show how to create a query with the SOSL Query Editor. Additionally, descriptions of the sections and elements of the Editor are available.



API Version: 51.0

Type: REST

Search SObject...

- Account (Account)
  - ☐ AccountSource
  - ☐ Active__c
  - ☐ AnnualRevenue
  - ☐ BillingAddress
  - ☐ BillingCity
  - ☐ BillingCountry
  - ☐ BillingGeocodeAccuracy
  - ☐ BillingLatitude
  - ☐ BillingLongitude
  - ☐ BillingPostalCode
  - ☐ BillingState
  - ☐ BillingStreet
  - ☐ Copy_Billing_Address_to_Shipping_Addres
  - ☐ CreatedBy
  - ☐ CreatedById
  - ☑ CreatedDate
  - ☐ Description
  - ☐ Fax
  - ☑ Id
  - ☐ Industry
  - ☐ IsDeleted
  - ☐ Jigsaw
  - ☐ JigsawCompanyId
  - ☐ LastActivityDate
  - ☐ LastModifiedBy
  - ☐ LastModifiedById
  - ☐ LastModifiedDate
  - ☐ LastReferencedDate

Account

Field	Alias	Format	toLabel
CreatedDate		☐	☐
Id		☐	☐
Name		☐	☐
Type		☐	☐

LIMIT: 10

OFFSET: 0

```
RETURNING Account (CreatedDate, Id, Name, Type LIMIT 10)
```

Figure 8.6 SOSL Query Editor

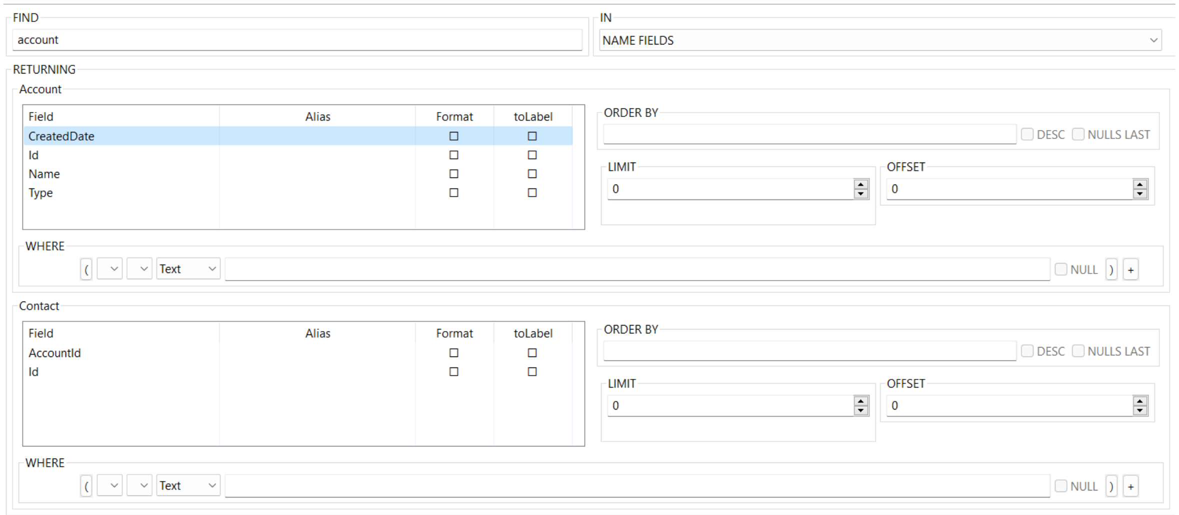
8.2.1 Objects and Fields

Objects and Fields of Objects can be selected in the components found on the left-hand side of the SOSL Query Editor. At the bottom of the component is a progress bar that runs when data is retrieved from Salesforce and prepared for display. This component is identical to the one in the SOQL Query Editor described above.

1. Select the API version in the “Version” dropdown. The version is synchronized between the different tabs. If a version is selected in another tab, this will be automatically selected here.
2. The selection of an API is disabled. SOSL can only be used for the Rest API.
   Optional: Search for the Object desired to be queried using the “Search sObject…” field.
3. For example, typing “acc” on the field will only show objects with that letter combination such as “Account” or “Access” in the name.
4. Click on the arrow in front of an object to load all the fields of the object. Alternatively, the box in front of the object can be selected. This loads and selects all fields of that object.
5. Select the Objects or specific fields desired to be included in the search.

8.2.2 Search Options

The component on the top right of the SOSL Query Editor enables the setting of different search options. The different syntax elements have their sections labeled accordingly. See Figure 8.7.



FIND: account

IN: NAME FIELDS

RETURNING

Account

Field	Alias	Format	toLabel
CreatedDate		□	□
Id		□	□
Name		□	□
Type		□	□

ORDER BY: □ DESC □ NULLS LAST

LIMIT: 0

OFFSET: 0

WHERE: (Text □ NULL) +

Contact

Field	Alias	Format	toLabel
AccountId		□	□
Id		□	□

ORDER BY: □ DESC □ NULLS LAST

LIMIT: 0

OFFSET: 0

WHERE: (Text □ NULL) +

Figure 8.7 Search Options Component

- The Find section serves as a search field. Name, number, or partial strings can be used as search parameters. An entry is required to use this function.
- The In section shows which fields are used for the search. The default setting is “All Fields”. The dropdown allows for the selection of other options such as “Name” and “Email Fields.”
- The Returning section is the largest of the components of the SOSL Query Editor. It has a subsection for each selected Salesforce Object, for instance, Account or Contact. Each object has multiple query options. The data to be returned can be set per Object, the following five points describe all options.
- The Fields Table displays the selected fields of an object.
  - In the table, fields can be formatted according to local settings. To do this, tick the corresponding box. A formatted field can be given an alias.
- To translate the search result into a preferred language, tick the toLabel box.
- In the Order By section, the field to sort on can be selected. The default order that the results are displayed is ascending. Selecting the DESC box sorts the query results in descending order. Additionally, records without an entry for the selected fields are placed at the bottom of the results by selecting the box Nulls Last.
- The Limit section allows setting the limit of results returned from a query. This is only available for REST and Bulk API.
- In the Offset section, it is possible to set the starting row of the returned result set. This is only available for REST API.
- In the Offset section, it is possible to set the starting row of the returned result set.
- In the Where section, the field and operator can be selected for the statement. Depending on the selected field, the input changes accordingly.
  - Text fields facilitate free text input.
  - For picklists fields, the input is a dropdown with all available options.
  - Numeric fields allow numbers only.
  - For DateTime fields, the input is split into two parts. The first part is a calendar to select the date from. The second is an input field for a time.

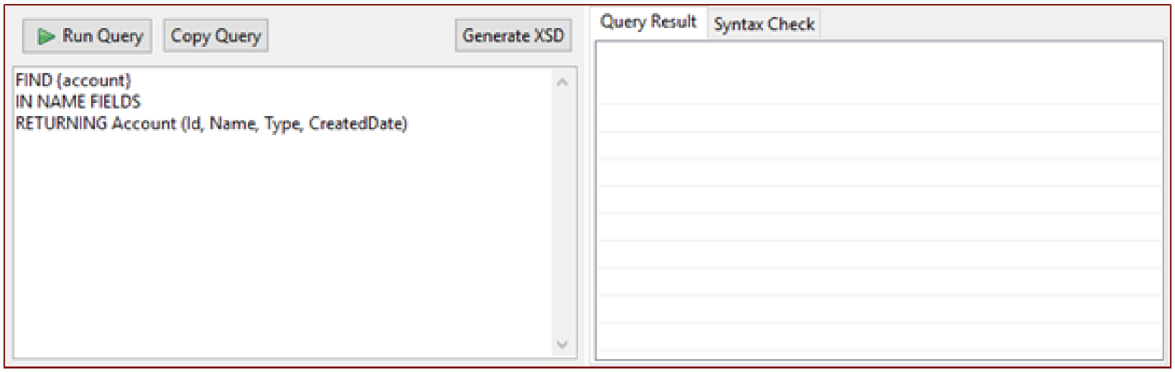
  Select the Null box if records without values are to be queried.

  Additional Where statements can be added by clicking on the plus (+) sign on the right-hand side of the section. This adds a new row to create an extra statement. New rows are preceded by a dropdown to select And/Or.

  To remove a row, click the minus (-) sign on the right-hand side of the row.

8.2.3 Results

Created Queries can be tested using the component on the bottom left-hand side of the Editor. See Figure 8.8 an overview of this component.



```
FIND {account}
IN NAME FIELDS
RETURNING Account (Id, Name, Type, CreatedDate)
```

Figure 8.8 SOSL Result Component

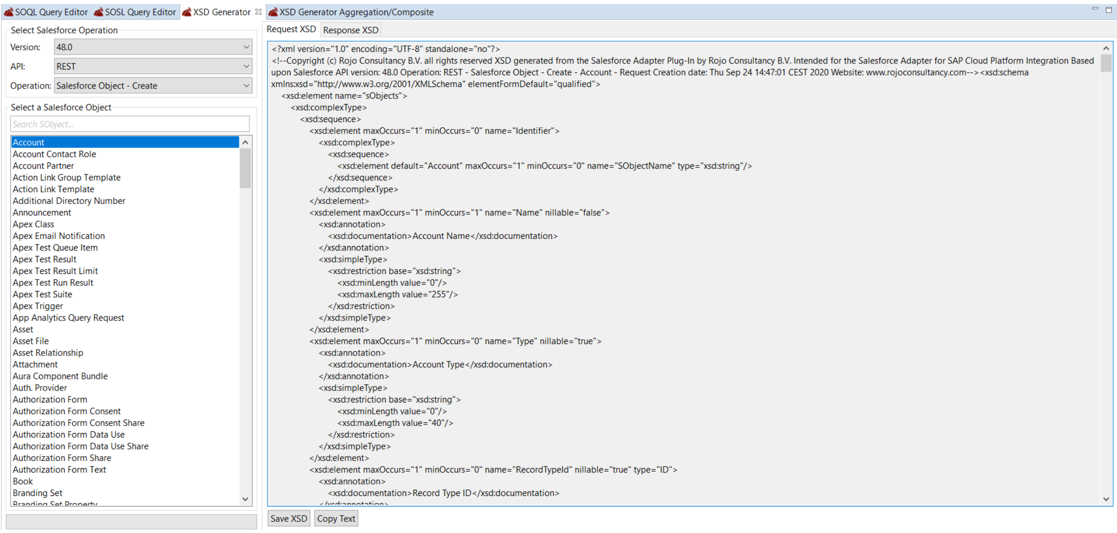
1. The text field on the left shows the created Query. Queries in this field cannot be edited manually. Changes can only be applied through the other components.

- Click on the Run Query button to test the Query. It may take some time to process this request. The button displays if the Query is running. It returns to its original state when the query finishes.
- The Query Result tab on the right displays the results in a table.
- If the Query is invalid, the Syntax Check tab on the right can be consulted. This displays additional information such as the Message and Error Code of the Response.
- If the results are satisfactory, the Query can be copied for use in the Salesforce Cloud Integration Adapter. To do this, the “Copy Query” button can be clicked on to copy the Query to the clipboard. Selecting the Query and copying it manually is also an option.

An XSD for the query result is generated by pressing the “Generate XSD” button. The generated XSD will pop up on a new screen. The option to either cancel or save the XSD is available. The generated XSD can be used to map the query results in an integration scenario.

8.3 XSD Generator

The XSD Generator was developed to support mapping operations in integration scenarios. It generates Request and Response XSDs for multiple APIs that are specific to the Salesforce instance. All Custom Objects are available in the list of Salesforce objects and custom fields are included in all Objects. The REST, Bulk, and Place Order APIs are implemented. Available versions depend on the API. Apart from the Query and Search operations that have been implemented in query builders, all operations that require a payload are implemented. The left side of the view enables the selection of the version, API, operation, and Salesforce object (if required). The right side of the view contains Request and Response Tabs that displays the resulting XSD (Figure 8.9). In case not all required fields have been selected, it will indicate which selection is needed. If an operation has no XSD for a Request or Response, it will indicate this. Figure 8.9 gives an overview of the different XSDs that can be generated.



Select Salesforce Operation

Version: 48.0

API: REST

Operation: Salesforce Object - Create

Select a Salesforce Object

Search SObject...

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!--Copyright (c) Rojo Consultancy B.V. all rights reserved XSD generated from the Salesforce Adapter Plug-In by Rojo Consultancy B.V. Intended for the Salesforce Adapter for SAP Cloud Platform Integration Based
upon Salesforce API version: 48.0 Operation: REST - Salesforce Object - Create - Account - Request Creation date: Thu Sep 24 14:47:01 CEST 2020 Website: www.rojoconsultancy.com--><xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xsd:element name="sObjects">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element maxOccurs="1" minOccurs="0" name="Identifier">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element default="Account" maxOccurs="1" minOccurs="0" name="SObjectName" type="xsd:string"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element maxOccurs="1" minOccurs="1" name="Name" nillable="false">
          <xsd:annotation>
            <xsd:documentation>Account Name</xsd:documentation>
          </xsd:annotation>
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:minLength value="0"/>
              <xsd:maxLength value="255"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element maxOccurs="1" minOccurs="0" name="Type" nillable="true">
          <xsd:annotation>
            <xsd:documentation>Account Type</xsd:documentation>
          </xsd:annotation>
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:minLength value="0"/>
              <xsd:maxLength value="40"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element maxOccurs="1" minOccurs="0" name="RecordTypeId" nillable="true" type="ID">
          <xsd:annotation>
            <xsd:documentation>Record Type ID</xsd:documentation>
          </xsd:annotation>
```

Figure 8.9 XSD Generator overview with the request XSD as output

1. Select the API version in the Version dropdown.
2. Select the API in the API dropdown.
3. Select the operation in the “Operation” dropdown.
   Note that for Bulk API, Batch – Create, the operations applicable may be selected in the Job Operation dropdown.
5. Select a Salesforce Object if applicable. The Search sObject box can be used to filter the list of sObjects.
6. Click Save XSD to save the displayed XSD. Switch tabs to save the corresponding Response/Request.

REST API			
Operation	**Object-specific**	**Request**	**Response**
Metadata - All			X
Metadata - Basic	X		X
Metadata - Comprehensive	X		X
Other - Get REST API Versions			X
Other - Get REST Resource Endpoints			X
Salesforce Object - Create	X	X	X
Salesforce Object - Delete			X
Salesforce Object - Delete with External ID			X
Salesforce Object - Get	X		X
Salesforce Object - Get with External ID	X		X
Salesforce Object - Update	X	X	X
Salesforce Object - Upsert with External ID	X	X	X
Bulk API			
Operation	**Object-specific**	**Request**	**Response**
Batch - Create - delete		X	X
Batch - Create - hard delete		X	X
Batch - Create - insert	X	X	X
Batch - Create - query		X	X
Batch - Create - update	X		X

Batch - Create - upsert	X	X	X
Batch - Get			X
Batch - Get Request	X		X
Batch - Get Results			X
Batch Query - Get Result	X		X
Batch Query - Get Result IDs			X
Job - Abort		X	X
Job - Close		X	X
Job - Create		X	X
Job - Get			X
Job - Get All Batches			X
Place Order API			
Operation	**Object-specific**	**Request**	**Response**
Account - Add Contracts and Orders		X	X
Account - Add Orders		X	X
Contract - Add Orders		X	X
Contract - Filter Details			X
Contract - Get			X
Order - Add Products		X	X
Order - Filter Details			X
Order - Get			X

Table 1 Overview of REST, BULK API, and Place Order API Operations

8.4 XSD Generator Aggregation/Composite

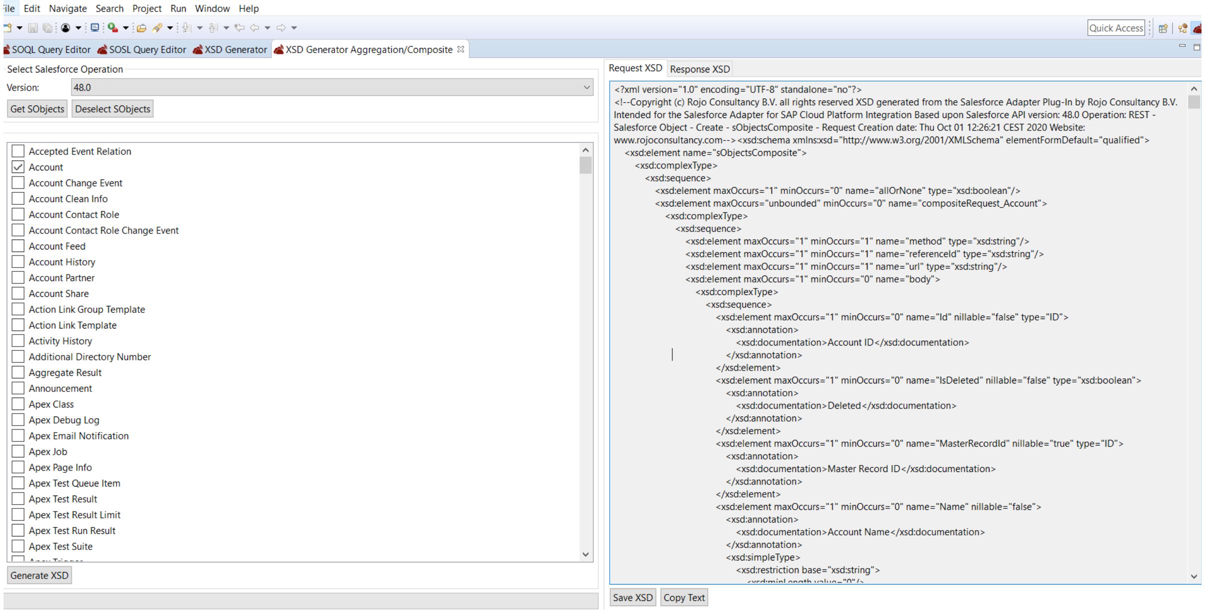
The XSD Generator Aggregation/Composite is developed to execute a series of REST API Requests in a single call. The output of one Request can be used as the input to a subsequent Request. The response bodies and HTTP statuses of the requests are returned in a single Response body. The entire series of Requests count as a single call toward API limits.

The Requests in a composite call are called subrequests. All subrequests are executed in the context of the same User. In a subrequest's body, specify a reference ID that maps to the subrequest's response. Refer to the ID in the URL or body fields of later subrequests by using a JavaScript-like reference notation.

It is possible to specify that an entire composite request rolls back should there be an error in a subrequest or just the subrequests that depend on it. Headers can also be assigned to each subrequest.

1. Select the API version in the Version dropdown.
2. Click on Get sObjects to retrieve a list of sObjects for the specific Salesforce tenant.
3. Select the sObjects to be included in the Aggregation/Composite operation, for example, Account and Contact.
4. Click Generate XSD to create the respective Request XSD.
5. Click Save XSD to save the displayed XSD. Switch tabs to save the corresponding Response/Request.

This component works with Salesforce API version 43.0 and later. A detailed explanation of the Aggregation/Composite operation is available in Section 11.5.



Select Salesforce Operation

Version: 48.0

- □ Accepted Event Relation
- ☑ Account
- □ Account Change Event
- □ Account Clean Info
- □ Account Contact Role
- □ Account Contact Role Change Event
- □ Account Feed
- □ Account History
- □ Account Partner
- □ Account Share
- □ Action Link Group Template
- □ Action Link Template
- □ Activity History
- □ Additional Directory Number
- □ Aggregate Result
- □ Announcement
- □ Apex Class
- □ Apex Debug Log
- □ Apex Email Notification
- □ Apex Job
- □ Apex Page Info
- □ Apex Test Queue Item
- □ Apex Test Result
- □ Apex Test Result Limit
- □ Apex Test Run Result
- □ Apex Test Suite

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!--Copyright (c) Rojo Consultancy B.V. all rights reserved XSD generated from the Salesforce Adapter Plug-In by Rojo Consultancy B.V.
Intended for the Salesforce Adapter for SAP Cloud Platform Integration Based upon Salesforce API version: 48.0 Operation: REST -
Salesforce Object - Create - sObjectsComposite - Request Creation date: Thu Oct 01 12:26:21 CEST 2020 Website:
www.rojoconsultancy.com--><xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xsd:element name="sObjectsComposite">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element maxOccurs="1" minOccurs="0" name="allOrNone" type="xsd:boolean"/>
        <xsd:element maxOccurs="unbounded" minOccurs="0" name="compositeRequest_Account">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element maxOccurs="1" minOccurs="1" name="method" type="xsd:string"/>
              <xsd:element maxOccurs="1" minOccurs="1" name="referenceId" type="xsd:string"/>
              <xsd:element maxOccurs="1" minOccurs="1" name="url" type="xsd:string"/>
              <xsd:element maxOccurs="1" minOccurs="0" name="body">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="0" name="Id" nillable="false" type="ID">
                      <xsd:annotation>
                        <xsd:documentation>Account ID</xsd:documentation>
                      </xsd:annotation>
                    </xsd:element>
                    <xsd:element maxOccurs="1" minOccurs="0" name="IsDeleted" nillable="false" type="xsd:boolean">
                      <xsd:annotation>
                        <xsd:documentation>Deleted</xsd:documentation>
                      </xsd:annotation>
                    </xsd:element>
                    <xsd:element maxOccurs="1" minOccurs="0" name="MasterRecordId" nillable="true" type="ID">
                      <xsd:annotation>
                        <xsd:documentation>Master Record ID</xsd:documentation>
                      </xsd:annotation>
                    </xsd:element>
                    <xsd:element maxOccurs="1" minOccurs="0" name="Name" nillable="false">
                      <xsd:annotation>
                        <xsd:documentation>Account Name</xsd:documentation>
                      </xsd:annotation>
                      <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                          <xsd:minLength value="0"/>
```

Figure 8.10 XSD Generator Aggregation/Composite overview with part of the selection on the left and resulting request XSD on the right side

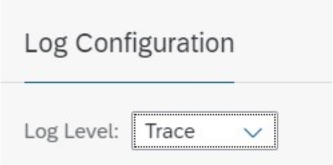
9 ERROR CODES

The table below lists some common error codes.

Error Code	Probable reason
300	There is more than one record for the provided external ID.
400	The request message cannot be executed because the XML or JSON contains an error.
401	The OAuth token is invalid or has expired.
403	The request sent was not accepted. This can be due to permission limits or because the Salesforce organization limit was exceeded.
404	The concerned resource or entity is not found. Check if the provided entity name is correct.

10 SUPPORT

In case of issues or errors, change the Log level of the integration flow to **Traces**. This can be done in SAP Cloud Integration via the Monitor > Manage Integration Content page. An example of such a configuration can be seen in Figure 10.1.


Log Configuration

Log Level: Trace

Figure 10.1 Activating Traces

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

The next section discusses a few issues that may be encountered when using the Adapter and practical solutions for such.

10.1 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 means that an entry for the root certificate of Salesforce is missing in the Keystore of SAP Cloud Integration. At the time this document is published, Salesforce expects a DigiCert Global Root certificate.

10.2 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 SalesforceOauth.*

Possible Solution: This error 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 Security Material.

10.3 There Are [x] Parameters That Could Not Be Set on the Endpoint

Error Message:

> *[CAMEL][IFLOW][CAUSE]: Cause: There are [x] parameters that couldn't be set on the endpoint. Check the URI if the parameters are spelled correctly and if they are properties of the endpoint. Unknown parameters=…*

Possible Solution: This error indicates that the SOQL or SOSL Query format is wrong. The Plug-in can help format the Query. Viable solutions include replacing the ‘&’ sign to connect queries with **_and_ or** putting the query inside the following text: **RAW ({advanced query})**.

10.4 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 the mapping requires a missing mandatory value in the message body sent when creating or updating an instance of an entity in Salesforce.

10.5 Login Error Code: Authentication Failure

Error Message:

```
"errorCode":"Login error code:[invalid_grant] description:[authentication
failure]","message":"authentication failure"
```

Possible Solution: This error indicates that the authentication failed. Verify if the credential names specified in the Salesforce Adapter are the correct reference names associated with the deployed Security Artifacts. Additionally, ensure the correct credentials are deployed. In case no Security Token is specified in the Adapter, check that the IP address ranges of the SAP Cloud Integration tenant are whitelisted in Salesforce (as specified in Section 6.1).

10.6 Login Error Code: 400: Bad Request

Error Message:

```
{
        "error": "invalid_grant",
        "error_description": "authentication failure"
}
```

Possible Solution: This error occurs due to invalid credential(s). Please verify the credentials. Example: Username, Password, Security Token, Client Key, and Consumer Secret. Another solution could be resetting them and trying again.

10.7 Creation Error Code: DUPLICATES_DETECTED

Error Message:

```
"errorCode":"com.sap.it.rt.adapter.http.api.exception.HttpResponseException: An internal server error occurred: <?xml version="1.0" encoding="UTF-8"?><Errors><Error><errorCode>DUPLICATES_DETECTED</errorCode><message>Use one of these records?</message></Error></Errors>"
```

Possible Solution: This error is returned by Salesforce and indicates that it has detected that the Records sent already exist.

10.8 Creation Error Code: INVALID_CROSS_REFERENCE_KEY

Error Message:

```
org.apache.camel.CamelException: <?xml version="1.0" encoding="UTF-8"?><Errors><Error><errorCode>INVALID_CROSS_REFERENCE_KEY</errorCode><message>invalid cross reference id</message></Error></Errors>
```

Possible Solution: This error is returned by Salesforce. This indicates that an ID provided in the Request does not exist. Ensure that the provided Object ID exists and matches with the expected sObject type.

10.9 Creation Error Code: 403: Handshake Denied

Error Message:

```
Caused by: org.apache.camel.CamelException: Cannot connect. "error": "403::Handshake
denied".
```

Possible Solution: This error occurs with the Sender Adapter when connecting and subscribing to Events. It occurs when the number of concurrent users has exceeded the limit of the organization. It means that there are multiple subscribers reading Events and the Salesforce instance does not allow such. For a full list of potential errors when subscribing to events, refer to:

https://developer.salesforce.com/docs/atlas.en-us.api_streaming.meta/api_streaming/streaming_error_codes.htm

10.10 Error Subscribing 403: denied_by_security_policy

Error Message:

> *org.apache.camel.CamelException: Error subscribing to /event/TestField__c: 403:denied_by_security_policy:create_denied*

Possible Solution: This error can occur with the Sender Adapter. It may indicate that the topic used in the subscription does not exist. Taking the example of Platform events, ensure that the name used for the topic comes from the field “API Name” in Platform Event Definition in Salesforce.

10.11 Error: spec must not be null

Error Message:

> *org.apache.camel.CamelException: spec must not be null*

Possible Solution: This error can occur when using a splitter with Parallel Processing and the Adapter with “Lazy Authentication” set to true. When using a splitter with parallel processing the lazy authentication must be disabled.

11 SAMPLE SCENARIOS EXPLAINED

In this section, three commonly used scenarios are used to display the use of the Adapter and the Eclipse Plug-in.

Requirements:

- SAP Cloud Integration tenant
- Salesforce tenant
- Salesforce Adapter

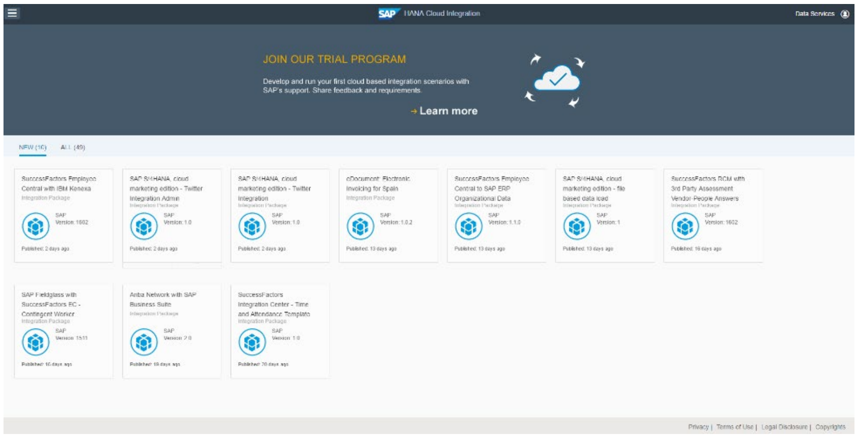
11.1 Initial Steps

All these scenarios begin with the creation of an Integration Package and adding a new Integration Flow.

Find the initial steps below.

11.1.1 Create an Integration Package in SAP Cloud Integration

1. Launch the SAP Cloud Integration Web application by accessing the URL provided by SAP.
2. If there is no existing account, choose Signup.
3. If User credentials are available, choose Login.

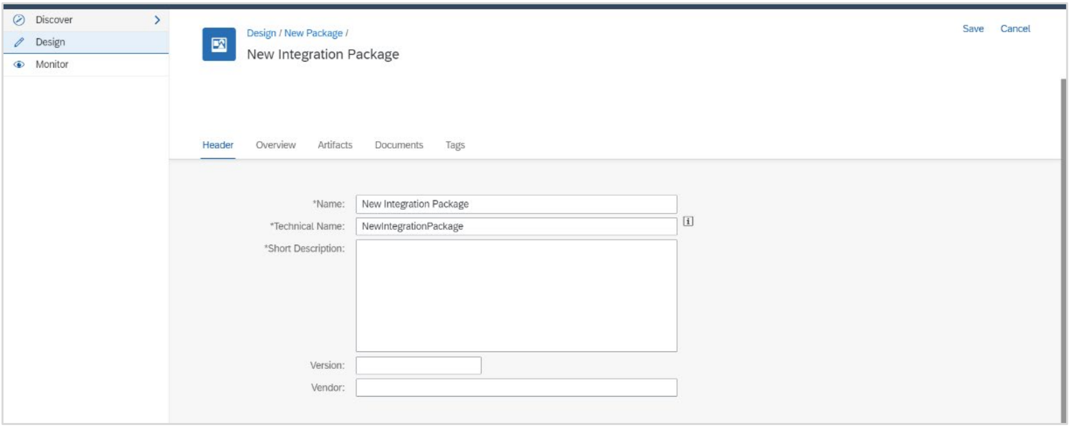

SAP HANA Cloud Integration
Data Services
JOIN OUR TRIAL PROGRAM
→ Learn more
NEW (10)
ALL (49)
SuccessFactors Employee Central with IBM Kenexa
Integration Package
SAP
Version: 1602
Published: 2 days ago
SAP S/4HANA, cloud marketing edition - Twitter Integration Admin
Integration Package
SAP
Version: 1.0
Published: 2 days ago
SAP S/4HANA, cloud marketing edition - Twitter Integration
Integration Package
SAP
Version: 1.0
Published: 2 days ago
eDocument: Electronic Invoicing for Spain
Integration Package
SAP
Version: 1.0.2
Published: 13 days ago
SuccessFactors Employee Central to SAP ERP Organizational Data
Integration Package
SAP
Version: 1.1.0
Published: 13 days ago
SAP S/4HANA, cloud marketing edition - file based data load
Integration Package
SAP
Version: 1
Published: 13 days ago
SuccessFactors RCM with 3rd Party Assessment Vendor-People Answers
Integration Package
SAP
Version: 1602
Published: 16 days ago
SAP Fieldglass with SuccessFactors EC - Contingent Worker
Integration Package
SAP
Version: 1511
Published: 16 days ago
Ariba Network with SAP Business Suite
Integration Package
SAP
Version: 2.0
Published: 19 days ago
SuccessFactors Integration Center - Time and Attendance Template
Integration Package
SAP
Version: 1.0
Published: 20 days ago
Privacy | Terms of Use | Legal Disclosure | Copyrights

Develop and run your first cloud based integration scenarios with SAP's support. Share feedback and requirements.

Figure 11.1 SAP Cloud Integration – Home

4. Select ☰ and choose the Design tab.
5. Choose Create and insert the required data to create a new Integration Package.

Data Insert	
Name	New Integration Package
Version	<Version>
Owned by	<Owned By>
Description	<Description>



*Name: New Integration Package

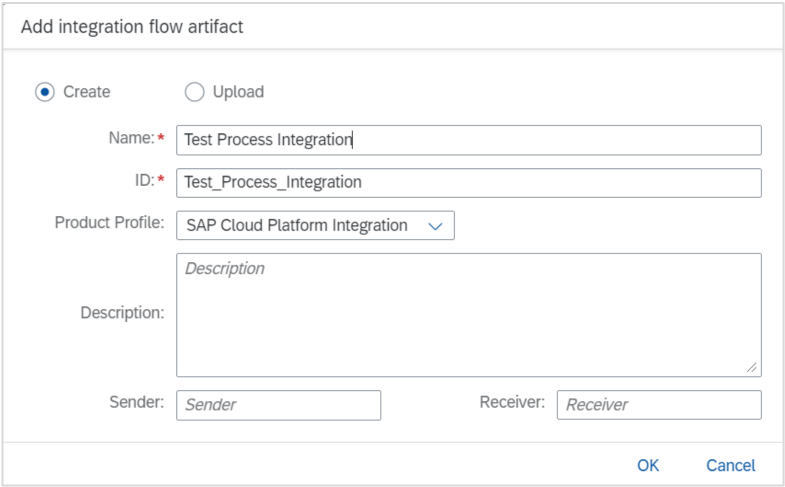
*Technical Name: NewIntegrationPackage

Figure 11.2 SAP Cloud Integration - New Integration Package

11.1.2 Create and Add a New Integration Flow in SAP Cloud Integration

1. Choose Artifacts, select Add and select Integration Flow to add a new integration flow (iFlow) as an artifact.
2. Select Create and enter the required data.

Data Insert	
Create	X
Name	Test Process Integration
Description	<Description>
Sender	Sender
Receiver	Receiver



Add integration flow artifact

● Create ○ Upload

Name:* Test Process Integration

ID:* Test_Process_Integration

Product Profile: SAP Cloud Platform Integration

Description: Description

Sender: Sender Receiver: Receiver

Figure 11.3 SAP Cloud Integration - Add Process Integration Artifact

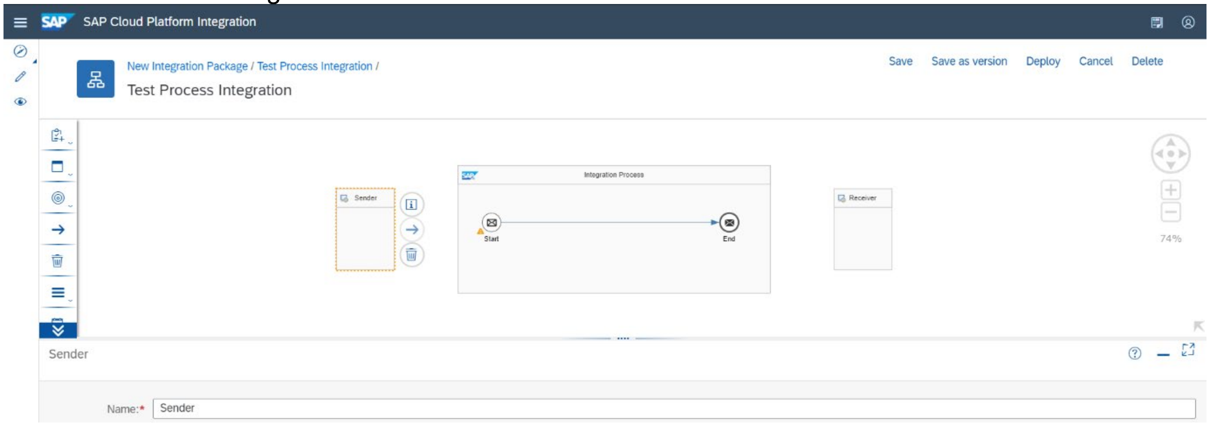
3. Choose **OK** to add the process integration artifact.
4. Choose **Save** to keep the integration package.

11.2 Synchronize Data

This scenario explains the synchronization of data via the Upsert capability in Salesforce. It replicates a message coming from a sender system via a Timer component and a Content Modifier component. It can Create or Update respective Records in the receiver system of Salesforce.

11.2.1 Assign Timer Instead of Sender and Start Message

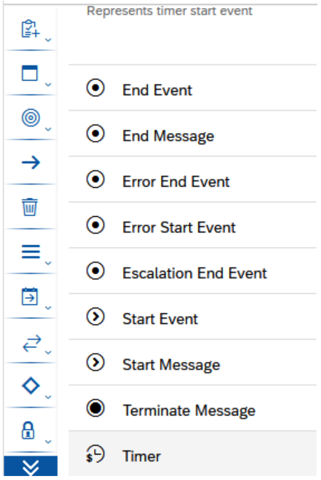
1. Click on the integration flow artifact name and choose Edit.



Name:* Sender

Figure 11.4 SAP Cloud Integration- Configuration Integration Flow

2. Select Sender and click Delete.
3. Select Start Message and click Delete.



Represents timer start event

Figure 11.5 SAP Cloud Integration – Select Events

4. Click on Events and select the Timer. Connect the Timer to End Message.

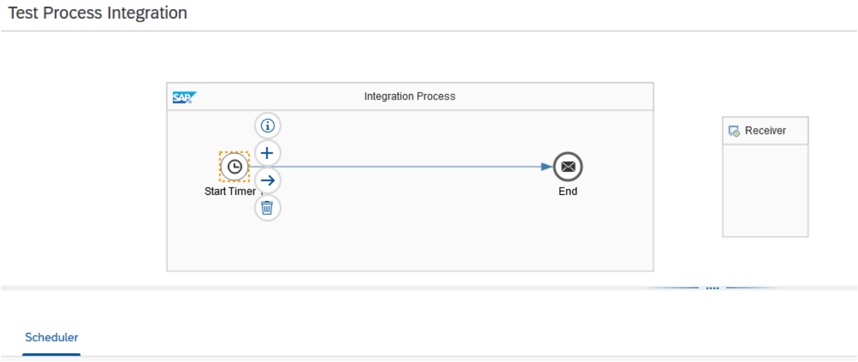
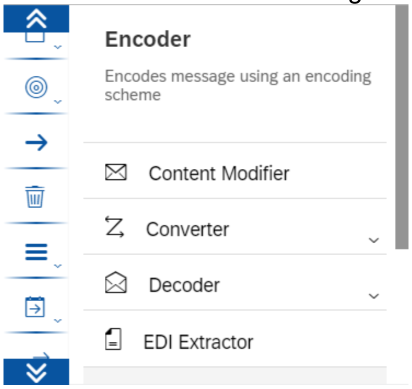


Figure 11.6 SAP Cloud Integration – Start Timer

5. Add a Content Modifier and connect it to the Start Message.



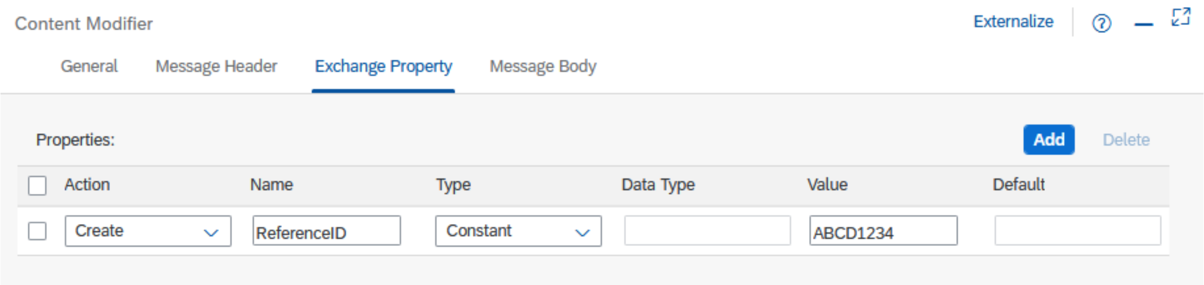
Encoder

Encodes message using an encoding scheme

Figure 11.7 SAP Cloud Integration - Add Content Modifier

6. Add a message containing Account data to the Content Modifier to replicate the message coming from the Sender. In this case, the message has an XML format.

7. Add an Exchange Property with the Reference ID value (ID used for correlation of the Record between the two systems).
   Example: ReferenceID.



□	Action	Name	Type	Data Type	Value	Default
□	Create	ReferenceID	Constant		ABCD1234	

Figure 11.8 Content Modifier – Exchange Property

11.2.2 Message Mapping Request

1. Subsequently, add a Message Mapping element to map the incoming message to the expected target message structure for Salesforce.

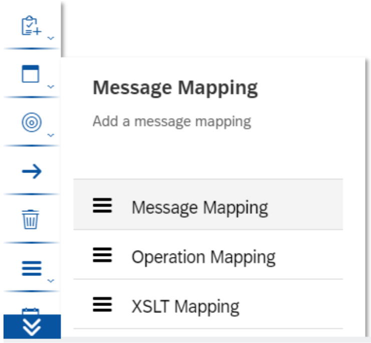
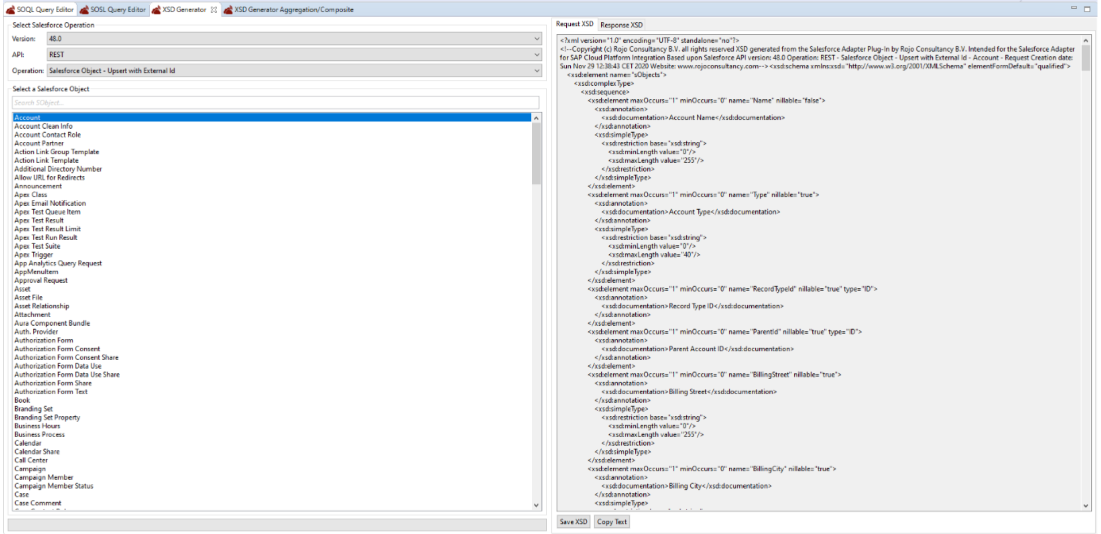


Figure 11.9 SAP Cloud Integration - Add Message Mapping

2. In Eclipse, select the XSD Generator and choose the REST API. Select the Salesforce Object – Upsert with External ID operation. Select Account under the Salesforce Objects and save the corresponding Request XSD file.



Version: 48.0

API: REST

Operation: Salesforce Object - Upsert with External Id

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!--Copyright (c) Rojo Consultancy B.V. all rights reserved XSD generated from the Salesforce Adapter Plug-In by Rojo Consultancy B.V. Intended for the Salesforce Adapter
for SAP Cloud Platform Integration Based upon Salesforce API version: 48.0 Operation: REST - Salesforce Object - Upsert with External Id - Account - Request Creation date:
Sun Nov 29 12:38:43 CET 2020 Website: www.rojoconsultancy.com--> <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xsd:element name="sObjects">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element maxOccurs="1" minOccurs="0" name="Name" nillable="false">
          <xsd:annotation>
            <xsd:documentation>Account Name</xsd:documentation>
          </xsd:annotation>
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:minLength value="0"/>
              <xsd:maxLength value="255"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element maxOccurs="1" minOccurs="0" name="Type" nillable="true">
          <xsd:annotation>
            <xsd:documentation>Account Type</xsd:documentation>
          </xsd:annotation>
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:minLength value="0"/>
              <xsd:maxLength value="40"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element maxOccurs="1" minOccurs="0" name="RecordTypeId" nillable="true" type="ID">
          <xsd:annotation>
            <xsd:documentation>Record Type ID</xsd:documentation>
          </xsd:annotation>
        </xsd:element>
        <xsd:element maxOccurs="1" minOccurs="0" name="ParentId" nillable="true" type="ID">
          <xsd:annotation>
            <xsd:documentation>Parent Account ID</xsd:documentation>
          </xsd:annotation>
        </xsd:element>
        <xsd:element maxOccurs="1" minOccurs="0" name="BillingStreet" nillable="true">
          <xsd:annotation>
            <xsd:documentation>Billing Street</xsd:documentation>
          </xsd:annotation>
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:minLength value="0"/>
              <xsd:maxLength value="255"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element maxOccurs="1" minOccurs="0" name="BillingCity" nillable="true">
          <xsd:annotation>
            <xsd:documentation>Billing City</xsd:documentation>
          </xsd:annotation>
          <xsd:simpleType>
```

Figure 11.10 XSD Generator

3. Select the Message Mapping in the iFlow, and then select Create and create a name.

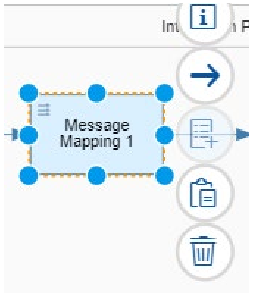
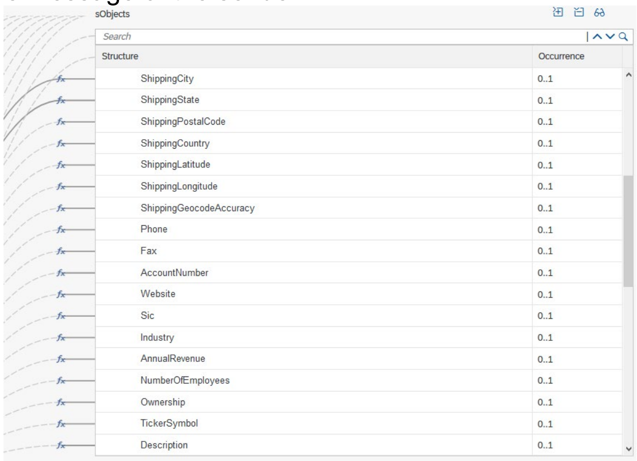


Figure 11.11 Select Create

4. Upload the XSD Request file as the Target Message. The Source Message should be configured with the XSD of the message of the sender.


sObjects

Search

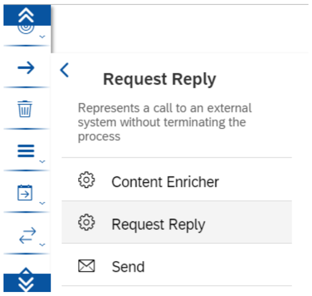
Structure	Occurrence
ShippingCity	0..1
ShippingState	0..1
ShippingPostalCode	0..1
ShippingCountry	0..1
ShippingLatitude	0..1
ShippingLongitude	0..1
ShippingGeocodeAccuracy	0..1
Phone	0..1
Fax	0..1
AccountNumber	0..1
Website	0..1
Sic	0..1
Industry	0..1
AnnualRevenue	0..1
NumberOfEmployees	0..1
Ownership	0..1
TickerSymbol	0..1
Description	0..1

Figure 11.12 Salesforce Object – Upsert with External ID – Account as Target XSD in message mapping

5. Map the two message structures from Source to Target using the Message Mapping element.

11.2.3 Configure Receiver Salesforce Adapter in SAP Cloud Integration

1. Add a Request Reply step to the integration flow after the Message Mapping.



Represents a call to an external system without terminating the process

Figure 11.13 Add Request Reply

2. Select Request Reply and choose the arrow to drag the pointer to Receiver. Then, select the Salesforce Adapter.

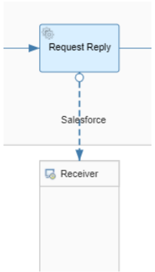
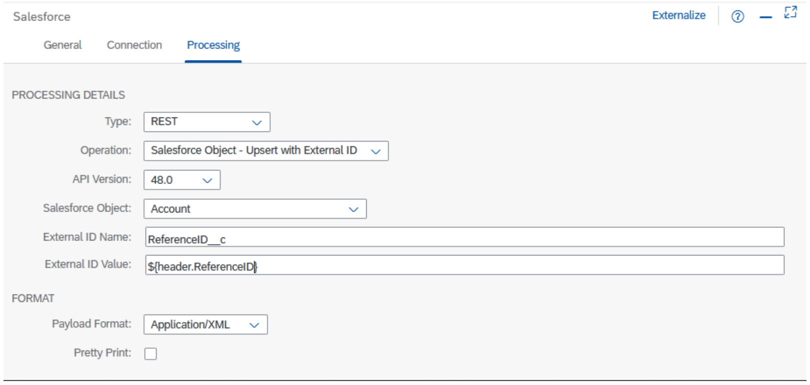


Figure 11.14 Add the Salesforce Adapter

3. Select REST as the Type in the Processing Tab of the Salesforce Adapter. Select Salesforce Object – Upsert with External ID as the Operation, and Account as the Salesforce Object. Specify *${property.ReferenceID}* in the External ID Value field, and *ReferenceID__c* in the External ID Name field.



PROCESSING DETAILS

Type: REST

Operation: Salesforce Object - Upsert with External ID

API Version: 48.0

Salesforce Object: Account

External ID Name: ReferenceID__c

External ID Value: ${header.ReferenceID}

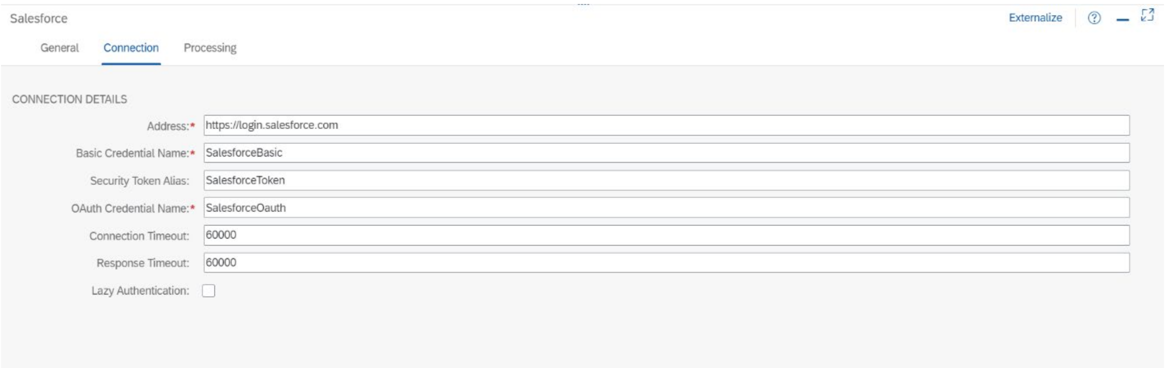
FORMAT

Payload Format: Application/XML

Pretty Print: □

Figure 11.15 Salesforce Adapter Processing Tab

4. Navigate to the Connection Tab in the Salesforce Adapter and enter the required data into the fields Basic Credential Name and OAuth Credential Name, respectively. After this step, supply the Security Token by referring to a secure parameter.



CONNECTION DETAILS

Address:* https://login.salesforce.com

Basic Credential Name:* SalesforceBasic

Security Token Alias: SalesforceToken

OAuth Credential Name:* SalesforceOauth

Connection Timeout: 60000

Response Timeout: 60000

Lazy Authentication: □

Figure 11.16 Salesforce Adapter Connection Tab

11.2.4 Message Mapping Response

1. Add another Message Mapping after the Salesforce Receiver Adapter and connect them. In Eclipse, save the corresponding Response XSD (as generated before together with the Request XSD) and add it as the Target Message and click OK.
2. Select Message Mapping and then select the arrow to drag it to the End Message.

11.2.5 Result of the iFlow

1. Save the iFlow as a version and click on Deploy.
2. Depending on whether the ReferenceID exists or not, the Account will be updated or created in Salesforce with the data present in the XML request message.

11.3 Synchronize Data Using Event Polling on the Receiver

The Salesforce Adapter provides a feature to poll Events using the Request-Reply step as discussed in Section 6.2.5.

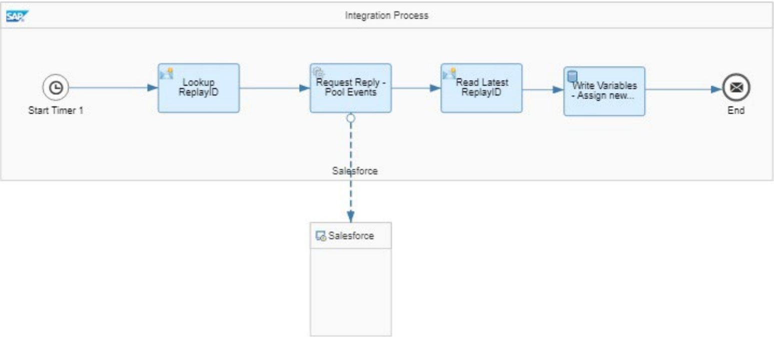


Figure 11.17 Sample Integration Flow to retrieve events using a Request-Reply step

As a reminder, Events in Salesforce are assigned with a unique identifier known as Replay ID. Each retrieved event message contains a ReplayID that can be used as a correlation ID in the iFlow.

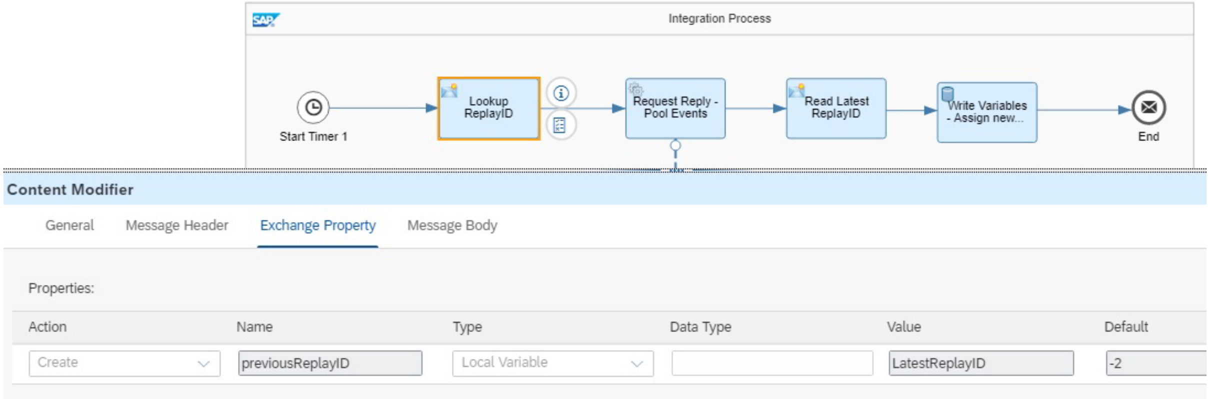
Each of the steps of the iFlow are discussed below:

- Start Timer

As shown in the above figure, the iFlow can be scheduled using a Timer step. The Timer can be scheduled to the desired frequency to poll events from Salesforce.

- Content Modifier

Use to create or to look up a local variable that contains the ReplayID. As Figure 11.17 shows, a value is assigned to a variable named “LatestReplayID.” Also, note that its value defaults to -2. This means that the first time the iFlow runs, the value of the ReplayID will be initialized to -2 (retrieve all events still available in Salesforce’s Event Bus).



Content Modifier

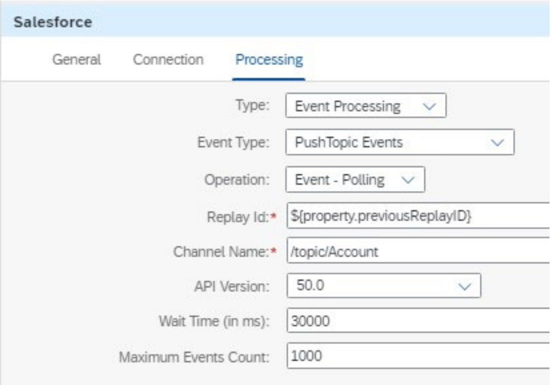
Properties:

Action	Name	Type	Data Type	Value	Default
Create	previousReplayID	Local Variable		LatestReplayID	-2

Figure 11.18 Create a local variable to store the Replay ID

- Request-Reply

Use the Salesforce Adapter and configure it as shown below. Note that the Replay Id field is filled with a dynamic property: **${property.previousReplayID}**. Furthermore, the Wait Time and Maximum Events Count have also been set. This means that a maximum of 1000 events will be collected within the specified Wait Time. In case the maximum number of events is reached before the Wait Time, the subscription will close. If the maximum number of events is not yet reached, the subscription will only close after the Wait Time elapses.



Type: Event Processing

Event Type: PushTopic Events

Operation: Event - Polling

Replay Id:* ${property.previousReplayID}

Channel Name:* /topic/Account

API Version: 50.0

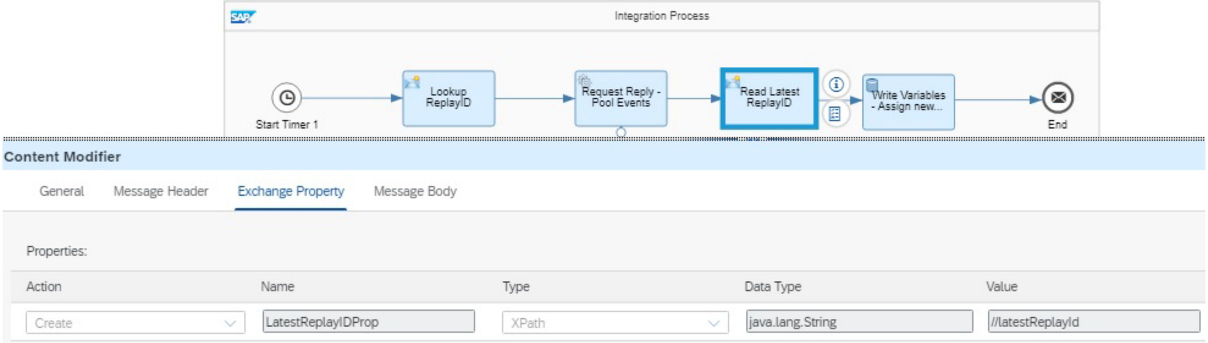
Wait Time (in ms): 30000

Maximum Events Count: 1000

Figure 11.19 Configuration of the Receiver Adapter

- Read Latest Replay ID

Use the Content Modifier to extract the value of the Latest Replay ID returned by the Request-Reply step.



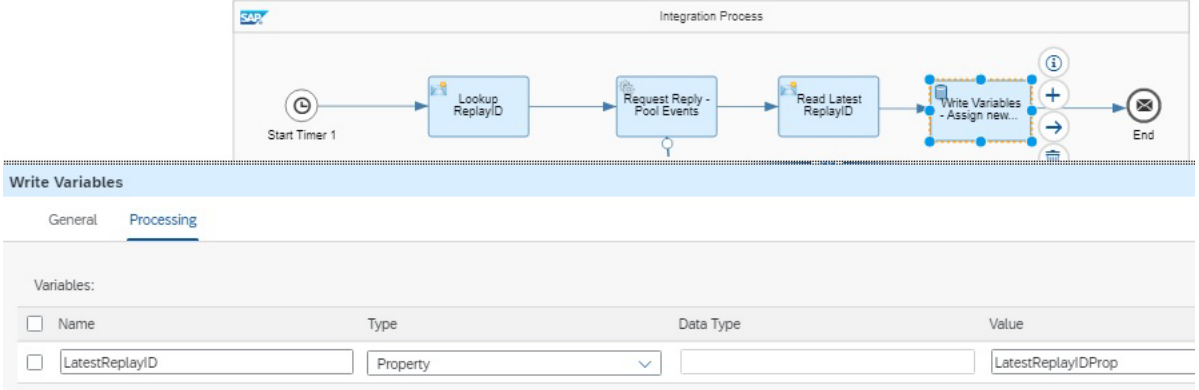
Properties:

Action	Name	Type	Data Type	Value
Create	LatestReplayIDProp	XPath	java.lang.String	//latestReplayId

Figure 11.20 Retrieve a value from the response of Salesforce

- Write Variable - Assign New Replay ID

Use the Write Variable step to assign the value of the Latest Replay ID to a variable. A sample configuration is available below.



Variables:

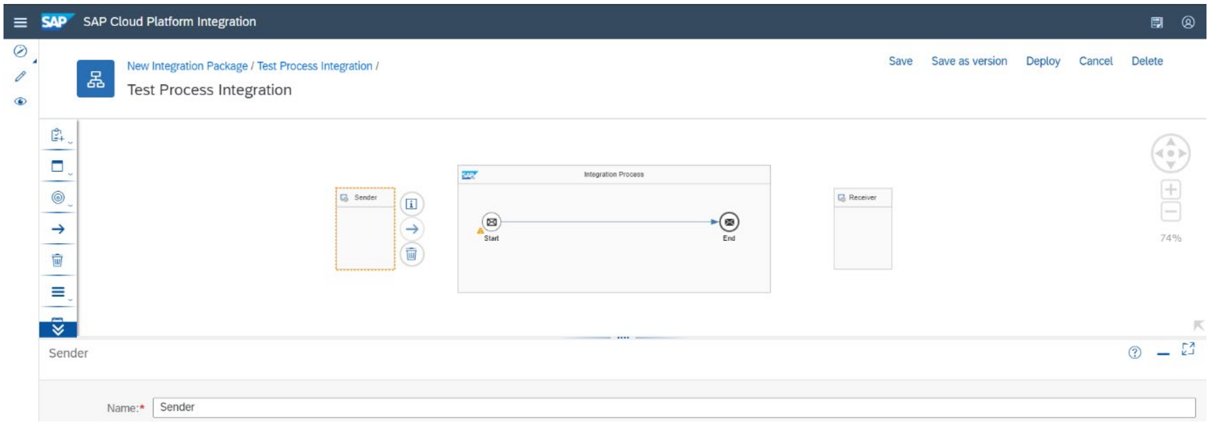
☐	Name	Type	Data Type	Value
☐	LatestReplayID	Property		LatestReplayIDProp

Figure 11.21 Assigning a value to a variable

11.4 Query Data

11.4.1 Assign Timer Instead of Sender and Start Message

1. Click on the iFlow artifact name and choose Edit.



Name:* Sender

Figure 11.22 SAP Cloud Integration - Configuration Integration Flow

2. Select Sender and click Delete.
3. Select Start Message and click Delete.

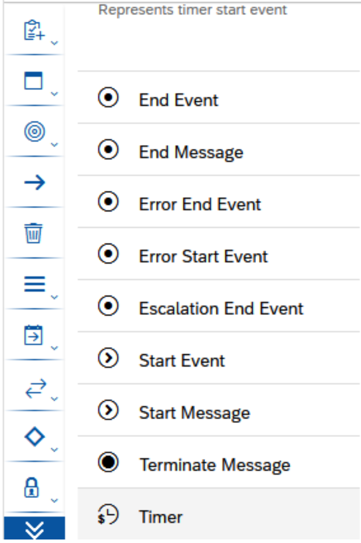


Figure 11.23 SAP Cloud Integration – Select Events

4. Click on Events and select the Timer. Connect Timer to End Message.

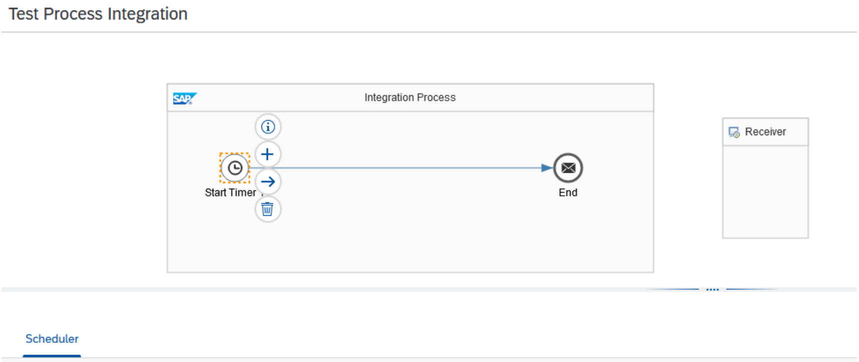
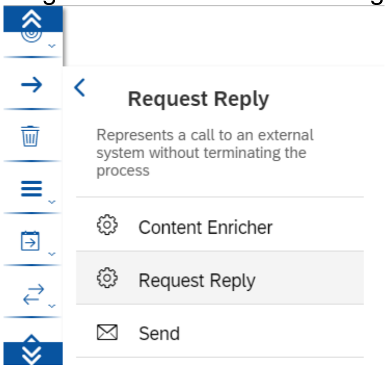


Figure 11.24 SAP Cloud Integration – Start Timer

11.4.2 Configure Salesforce Receiver Adapter

1. Add a Request Reply to the integration flow after the Message Mapping.



Represents a call to an external system without terminating the process

Figure 11.25 Add Request Reply

2. Select Request Reply and choose the arrow to drag the pointer to Receiver. Then, select the Salesforce Adapter.

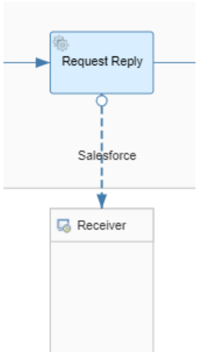
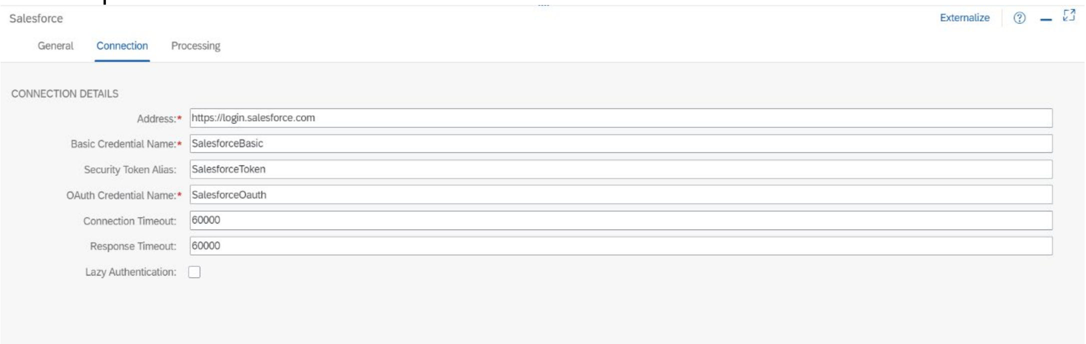


Figure 11.26 Add the Salesforce Adapter

3. Select the Connection Tab in the Salesforce Adapter and enter the required data into the fields Basic Credential Name and OAuth Credential Name. Supply the Security Token by referring to a secure parameter.

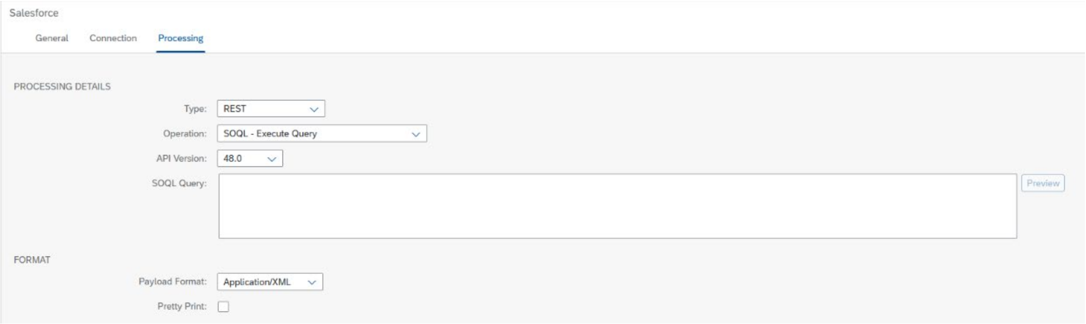


CONNECTION DETAILS

Address:* https://login.salesforce.com
Basic Credential Name:* SalesforceBasic
Security Token Alias: SalesforceToken
OAuth Credential Name:* SalesforceOauth
Connection Timeout: 60000
Response Timeout: 60000
Lazy Authentication: □

Figure 11.27 Salesforce Adapter Connection Tab

4. In the Processing Tab select the operation SOQL – Execute Query and set Payload Format to Application/XML.



Salesforce

PROCESSING DETAILS

Type: REST

Operation: SOQL - Execute Query

API Version: 48.0

SOQL Query:

FORMAT

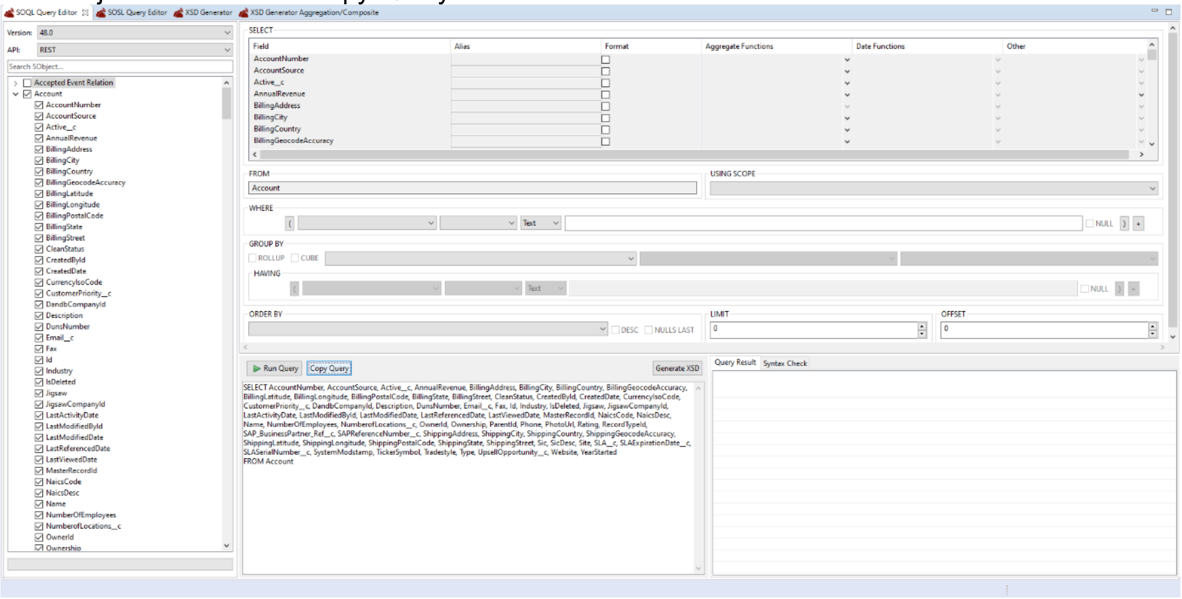
Payload Format: Application/XML

Pretty Print: □

Figure 11.28 Salesforce Adapter Processing Tab – SOQL – Execute Query

11.4.3 Create SOQL Query

1. In Eclipse, select the SOQL Query Editor. Choose REST API. Select Account under the Salesforce Objects and click on Copy Query.



Version: 48.0

API: REST

FROM: Account

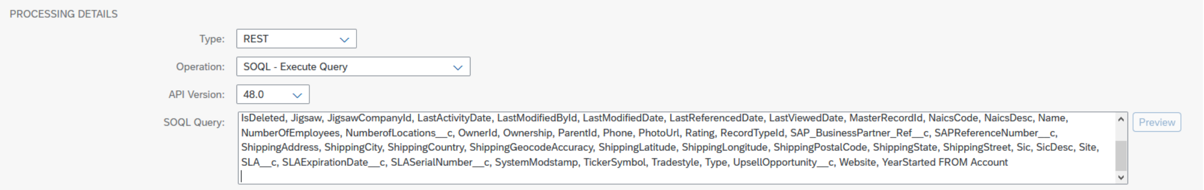
LIMIT: 0

OFFSET: 0

```
SELECT AccountNumber, AccountSource, Active__c, AnnualRevenue, BillingAddress, BillingCity, BillingCountry, BillingGeocodeAccuracy,
BillingLatitude, BillingLongitude, BillingPostalCode, BillingState, BillingStreet, CleanStatus, CreatedById, CreatedDate, CurrencyIsoCode,
CustomerPriority__c, DandbCompanyId, Description, DunsNumber, Email__c, Fax, Id, Industry, IsDeleted, Jigsaw, JigsawCompanyId,
LastActivityDate, LastModifiedById, LastModifiedDate, LastReferencedDate, LastViewedDate, MasterRecordId, NaicsCode, NaicsDesc,
Name, NumberOfEmployees, NumberofLocations__c, OwnerId, Ownership, ParentId, Phone, PhotoUrl, Rating, RecordTypeId,
SAP_BusinessPartner_Ref__c, SAPReferenceNumber__c, ShippingAddress, ShippingCity, ShippingCountry, ShippingGeocodeAccuracy,
ShippingLatitude, ShippingLongitude, ShippingPostalCode, ShippingState, ShippingStreet, Sic, SicDesc, Site, SLA__c, SLAExpirationDate__c,
SLASerialNumber__c, SystemModstamp, TickerSymbol, Tradestyle, Type, UpsellOpportunity__c, Website, YearStarted
FROM Account
```

Figure 11.29 SOQL Query Editor

2. Copy the query into the SOQL Query field in the Salesforce Adapter.



PROCESSING DETAILS

Type: REST

Operation: SOQL - Execute Query

API Version: 48.0

SOQL Query:

```
IsDeleted, Jigsaw, JigsawCompanyId, LastActivityDate, LastModifiedById, LastModifiedDate, LastReferencedDate, LastViewedDate, MasterRecordId, NaicsCode, NaicsDesc, Name,
NumberOfEmployees, NumberofLocations__c, OwnerId, Ownership, ParentId, Phone, PhotoUrl, Rating, RecordTypeId, SAP_BusinessPartner_Ref__c, SAPReferenceNumber__c,
ShippingAddress, ShippingCity, ShippingCountry, ShippingGeocodeAccuracy, ShippingLatitude, ShippingLongitude, ShippingPostalCode, ShippingState, ShippingStreet, Sic, SicDesc, Site,
SLA__c, SLAExpirationDate__c, SLASerialNumber__c, SystemModstamp, TickerSymbol, Tradestyle, Type, UpsellOpportunity__c, Website, YearStarted FROM Account
```

Figure 11.30 SOQL Query

11.4.4 Message Mapping Response

1. In Eclipse, click on Generate XSD and Save XSD to create and save the XSD for the response mapping for the create SOQL query.
2. Add a Message Mapping.

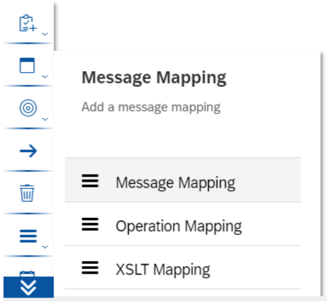
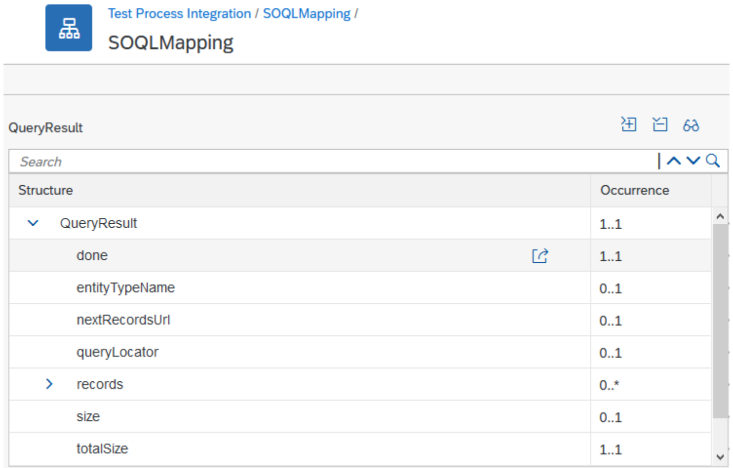


Figure 11.31 SAP Cloud Integration- Add Message Mapping

3. Use the generated XSD as Source Message (and for test purposes also as Target Message). Click on OK.


Test Process Integration / SOQLMapping /

SOQLMapping

QueryResult

Search

Structure	Occurrence
QueryResult	1..1
done	1..1
entityTypeName	0..1
nextRecordsUrl	0..1
queryLocator	0..1
records	0..*
size	0..1
totalSize	1..1

Figure 11.32 SAP Cloud Integration – Add XSD as Source Structure

4. Select Message Mapping and then select the arrow to drag it to the End Message.

11.4.5 Result of the iFlow

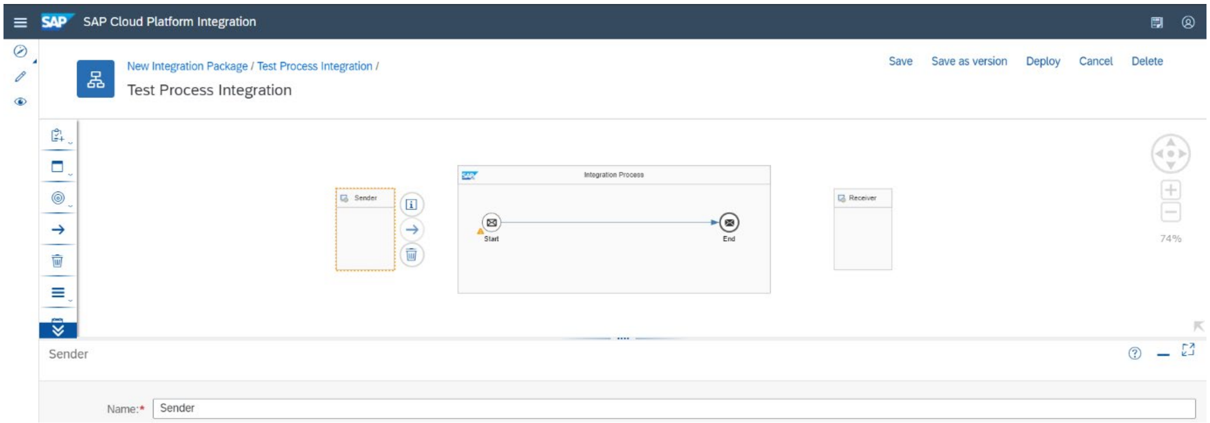
1. Save the iFlow as a version and click on Deploy.
2. The iFlow will return the data output matching the result of the query in Salesforce in the exchange body. This body can be used as input in the next step in the iFlow to complete the scenario.

11.5 Composite Call

The Composite operation supports the use of multiple sObject records and multiple sObject types in one operation. It is applicable in an integration scenario where it is desired to create one Account and two Contacts in the same message to Salesforce.

11.5.1 Assign Timer Instead of Sender and Start Message

1. Click on the integration flow artifact name and choose Edit.



Name:* Sender

Figure 11.33 SAP Cloud Integration - Configuration Integration Flow

2. Select Sender and click Delete.
3. Select Start Message and click Delete.

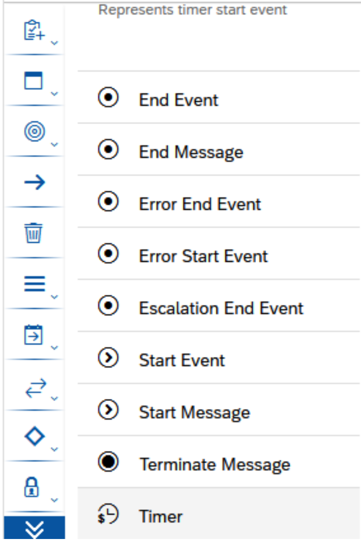


Figure 11.34 SAP Cloud Integration – Select Events

4. Click on Events and select the Timer. Connect Timer to End Message.

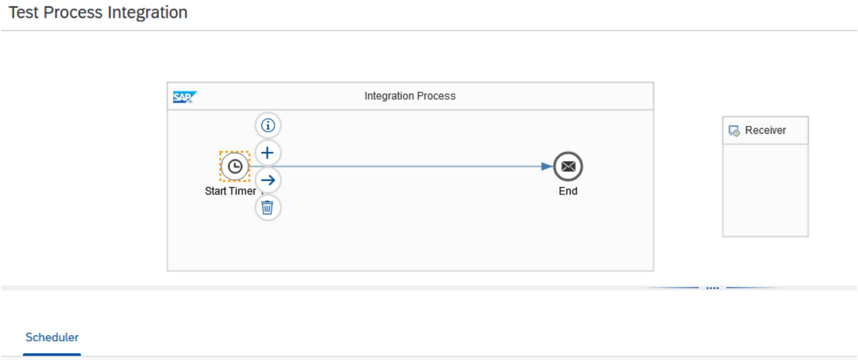


Figure 11.35 SAP Cloud Integration – Start Timer

5. Add a Content Modifier and connect it to the Start Message.

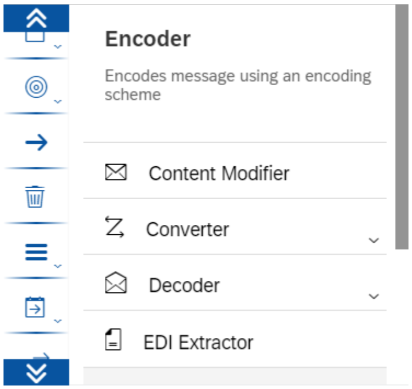
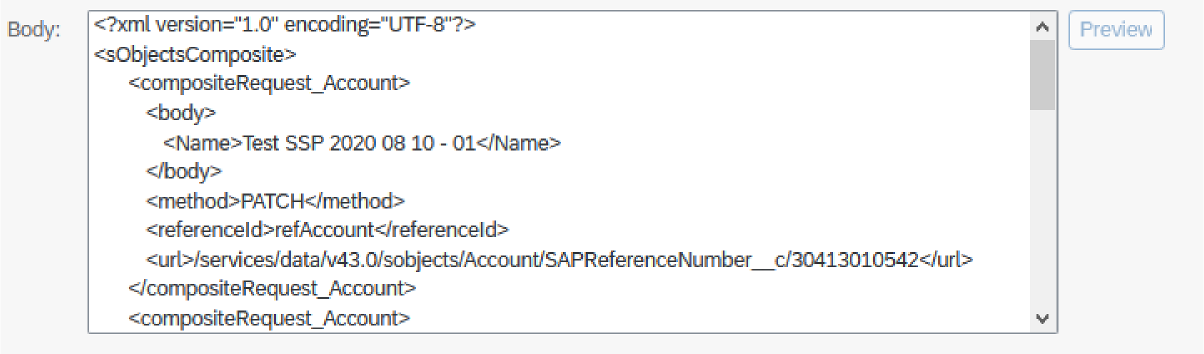


Figure 11.36 SAP Cloud Integration - Add Content Modifier

6. Add a Message containing Composite data to the Content Modifier to replicate the message coming from the Sender. In this example, the message is in XML format. Find an example of a start of a Composite Message below.



```
<?xml version="1.0" encoding="UTF-8"?>
<sObjectsComposite>
    <compositeRequest_Account>
      <body>
        <Name>Test SSP 2020 08 10 - 01</Name>
      </body>
      <method>PATCH</method>
      <referenceId>refAccount</referenceId>
      <url>/services/data/v43.0/sobjects/Account/SAPReferenceNumber__c/30413010542</url>
    </compositeRequest_Account>
    <compositeRequest_Account>
```

Figure 11.37 Content Modifier – Composite body

7. For an overview of how the message structure of a Composite message is organized, use the XSD Generator Aggregation/Composite tool or refer to the Composite Instructions Salesforce.

11.5.2 Message Mapping Request

1. Add a Message Mapping.

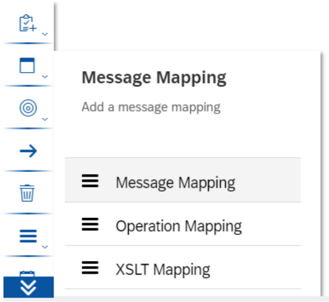
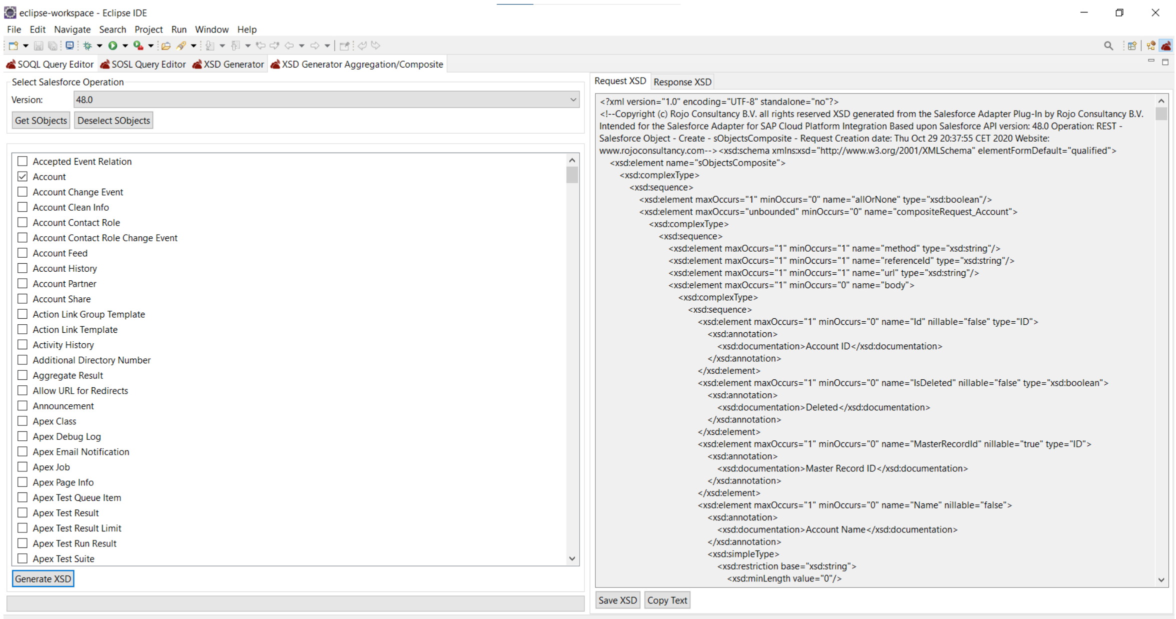


Figure 11.38 Add a Message Mapping

2. In Eclipse, select the XSD Generator Aggregation/Composite. Click on Account and Contacts under the Salesforce Objects. Save the corresponding Request XSD file.



Select Salesforce Operation

Version: 48.0

- ☐ Accepted Event Relation
- ☑ Account
- ☐ Account Change Event
- ☐ Account Clean Info
- ☐ Account Contact Role
- ☐ Account Contact Role Change Event
- ☐ Account Feed
- ☐ Account History
- ☐ Account Partner
- ☐ Account Share
- ☐ Action Link Group Template
- ☐ Action Link Template
- ☐ Activity History
- ☐ Additional Directory Number
- ☐ Aggregate Result
- ☐ Allow URL for Redirects
- ☐ Announcement
- ☐ Apex Class
- ☐ Apex Debug Log
- ☐ Apex Email Notification
- ☐ Apex Job
- ☐ Apex Page Info
- ☐ Apex Test Queue Item
- ☐ Apex Test Result
- ☐ Apex Test Result Limit
- ☐ Apex Test Run Result
- ☐ Apex Test Suite

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!--Copyright (c) Rojo Consultancy B.V. all rights reserved XSD generated from the Salesforce Adapter Plug-In by Rojo Consultancy B.V.
Intended for the Salesforce Adapter for SAP Cloud Platform Integration Based upon Salesforce API version: 48.0 Operation: REST -
Salesforce Object - Create - sObjectsComposite - Request Creation date: Thu Oct 29 20:37:55 CET 2020 Website:
www.rojoconsultancy.com--><xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xsd:element name="sObjectsComposite">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element maxOccurs="1" minOccurs="0" name="allOrNone" type="xsd:boolean"/>
        <xsd:element maxOccurs="unbounded" minOccurs="0" name="compositeRequest_Account">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element maxOccurs="1" minOccurs="1" name="method" type="xsd:string"/>
              <xsd:element maxOccurs="1" minOccurs="1" name="referenceId" type="xsd:string"/>
              <xsd:element maxOccurs="1" minOccurs="1" name="url" type="xsd:string"/>
              <xsd:element maxOccurs="1" minOccurs="0" name="body">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="0" name="Id" nillable="false" type="ID">
                      <xsd:annotation>
                        <xsd:documentation>Account ID</xsd:documentation>
                      </xsd:annotation>
                    </xsd:element>
                    <xsd:element maxOccurs="1" minOccurs="0" name="IsDeleted" nillable="false" type="xsd:boolean">
                      <xsd:annotation>
                        <xsd:documentation>Deleted</xsd:documentation>
                      </xsd:annotation>
                    </xsd:element>
                    <xsd:element maxOccurs="1" minOccurs="0" name="MasterRecordId" nillable="true" type="ID">
                      <xsd:annotation>
                        <xsd:documentation>Master Record ID</xsd:documentation>
                      </xsd:annotation>
                    </xsd:element>
                    <xsd:element maxOccurs="1" minOccurs="0" name="Name" nillable="false">
                      <xsd:annotation>
                        <xsd:documentation>Account Name</xsd:documentation>
                      </xsd:annotation>
                      <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                          <xsd:minLength value="0"/>
```

Figure 11.39 XSD Generator

3. Select the Message Mapping in the iFlow. Select Create and supply a name.

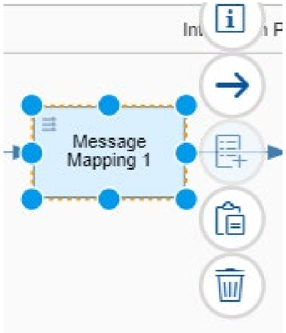
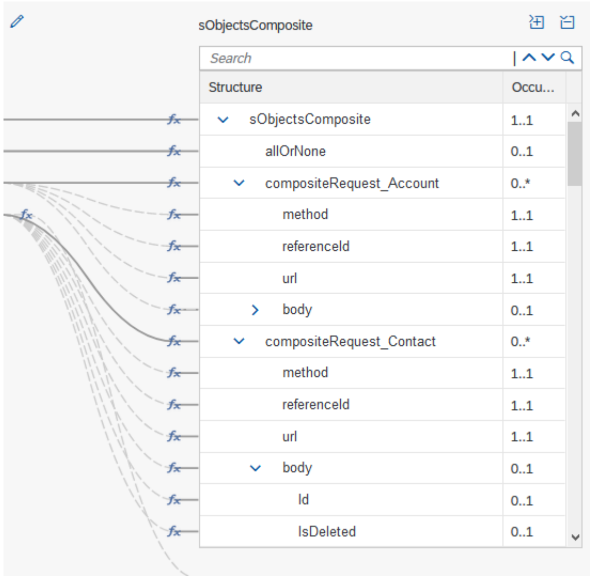


Figure 11.40 Select Create

4. Upload the XSD Request file as the Target Message (for testing purposes, the same file can also be used as the Source Message).
5. Map the two Message structures from Source to Target using the Message Mapping element.



sObjectsComposite

Structure	Occu...
sObjectsComposite	1..1
allOrNone	0..1
compositeRequest_Account	0..*
method	1..1
referenceId	1..1
url	1..1
body	0..1
compositeRequest_Contact	0..*
method	1..1
referenceId	1..1
url	1..1
body	0..1
Id	0..1
IsDeleted	0..1

Figure 11.41 Sample Mapping

11.5.3 Configure Receiver Salesforce Adapter in SAP Cloud Integration

1. Add a Request Reply to the iFlow after the Message Mapping.

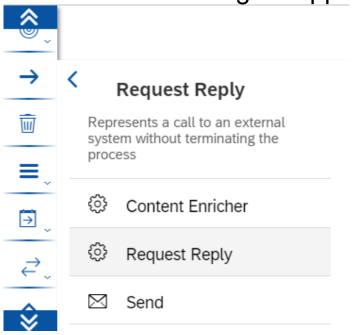

Request Reply
Represents a call to an external system without terminating the process
Content Enricher
Request Reply
Send

Figure 11.42 Add Request Reply

2. Select Request Reply and choose the arrow to drag the pointer to Receiver. Select the Salesforce Adapter.

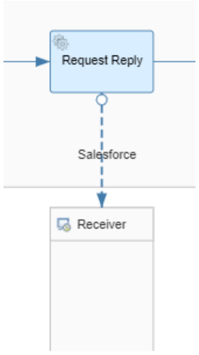
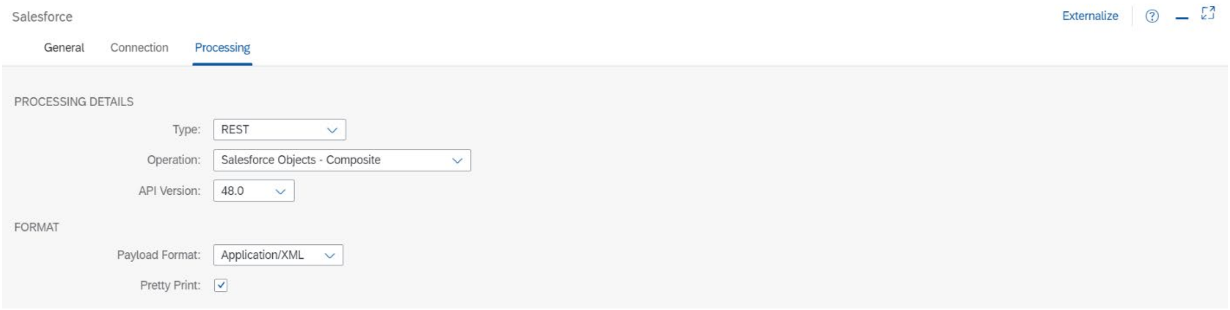


Figure 11.43 Add the Salesforce Adapter

3. Select REST as the Type in the Processing Tab of the Salesforce Adapter. Select Salesforce Object – Composite as the Operation.



Salesforce

General Connection Processing

PROCESSING DETAILS

Type: REST

Operation: Salesforce Objects - Composite

API Version: 48.0

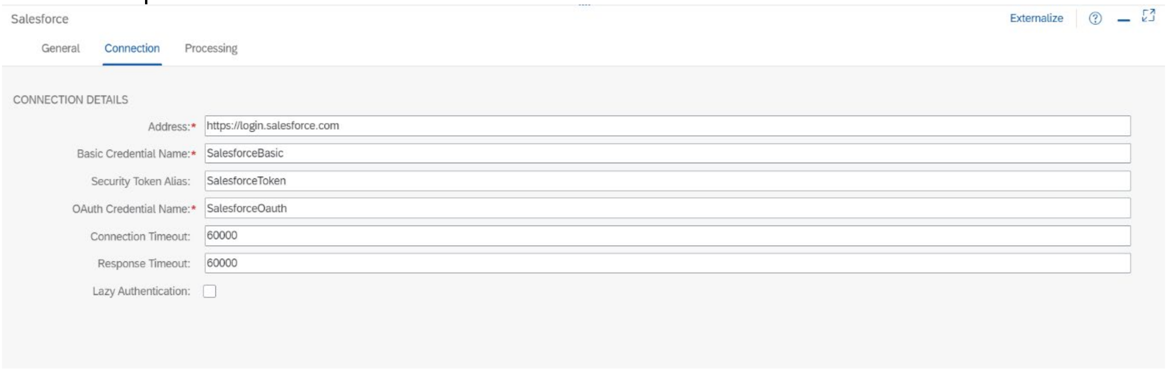
FORMAT

Payload Format: Application/XML

Pretty Print: ☑

Figure 11.44 Salesforce Adapter Processing Tab

4. Navigate to the Connection Tab in the Salesforce Adapter and enter the required data into the fields Basic Credential Name and OAuth Credential Name. Provide the Security Token by referring to a secure parameter.



Salesforce

General Connection Processing

CONNECTION DETAILS

Address:* https://login.salesforce.com

Basic Credential Name:* SalesforceBasic

Security Token Alias: SalesforceToken

OAuth Credential Name:* SalesforceOauth

Connection Timeout: 60000

Response Timeout: 60000

Lazy Authentication: □

Figure 11.45 Salesforce Adapter Connection Tab

11.5.4 Result of the iFlow

1. Save the iFlow as a version and click on Deploy.
2. Depending on whether the external IDs mentioned in the individual calls exist or not, the Account and/or Contact sObjects will be updated or created in Salesforce with the data present in the Request Message.

11.6 Content Version – Version Data

In this section, how to retrieve and insert Version Data is described.

11.6.1 Retrieving Content Version - Version Data

In addition to retrieving the binary data for the Document or Attachment object, described in Section 6.2.1.17 Salesforce Object – Get Blob, it is possible to retrieve a representation of a specific version of a document in Salesforce CRM Content or Salesforce Files.

1. This is an optional step to retrieve the Version Data URL. SOQL Query can be configured with the following query (returning VersionData) as an example:
   Select VersionData from ContentVersion where Id = '0685E000002nXoqQAE'
   In the response XML the Version Data URL will be returned:

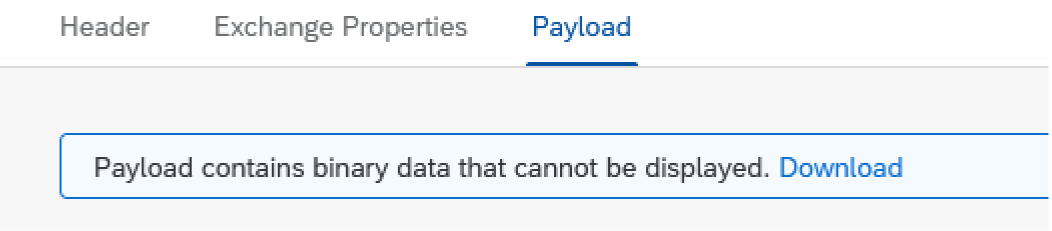
```
<?xml version="1.0" encoding="UTF-8"?>
<QueryResult>
  <done>true</done>
  <records type="ContentVersion"
url="/services/data/v51.0/sobjects/ContentVersion/0685E000002nXoqQAE">
```

```
<VersionData>/services/data/v51.0/sobjects/ContentVersion/0685E000002nXoqQAE/VersionData</Versi
onData>
  </records>
  <totalSize>1</totalSize>
</QueryResult>
```

2. Configure the Salesforce receiver adapter to retrieve the content in octetstream content-type with the Content Version id, using the Section 6.2.1.18 Salesforce Object - Custom Request. Note that Input Payload Format configuration does not have any impact as this GET operation has no Request payload.

Example Configuration	
Operation	Salesforce Objects – Custom Request
Input Payload Format	Application/octetstream
Response Payload Format	Application/octetstream
Method Name	GET
Resource Relative URL	/services/data/v51.0/sobjects/ContentVersion/0685E000002nXoqQAE/VersionData

3. As a response, the binary data will be returned.

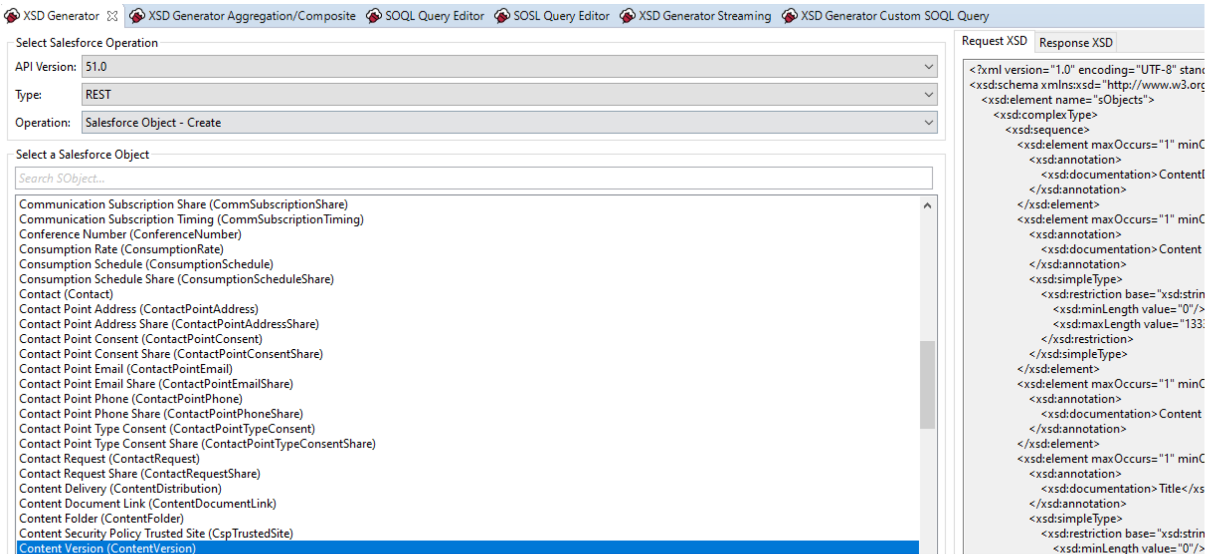


Payload contains binary data that cannot be displayed. Download

Figure 11.46 Binary Data as the response on Custom Request, VersionData

11.6.2 Inserting Content Version - Version Data

1. Insert Version Data is supported by the Salesforce Object – Create operation. To know the structure that is expected for XML, navigate to that operation. Select the Salesforce Object Content Version.
2. Save the Request XSD.



Select Salesforce Operation

API Version: 51.0

Type: REST

Operation: Salesforce Object - Create

Select a Salesforce Object

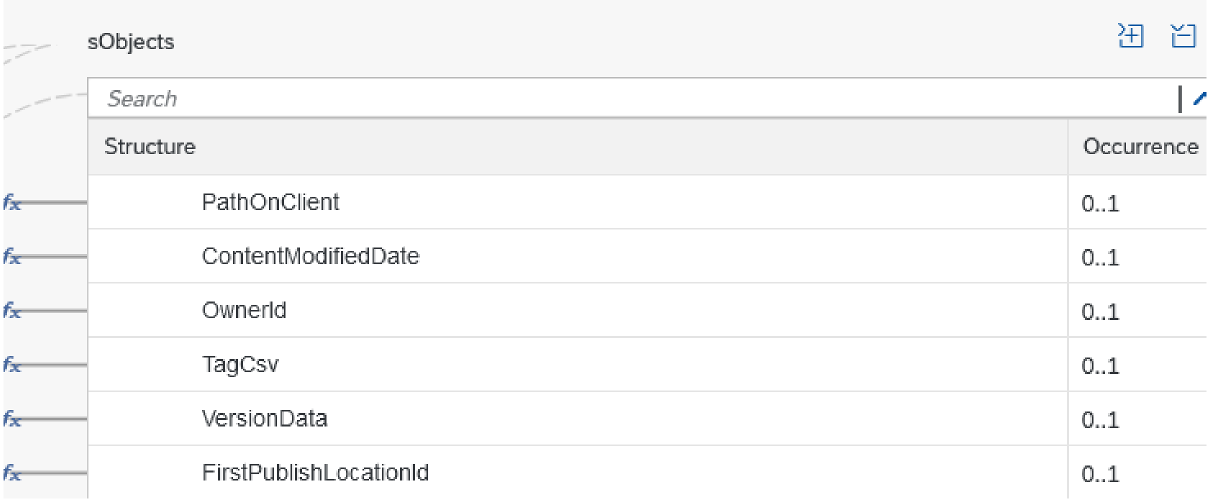
Search SObject...

```
Communication Subscription Share (CommSubscriptionShare)
Communication Subscription Timing (CommSubscriptionTiming)
Conference Number (ConferenceNumber)
Consumption Rate (ConsumptionRate)
Consumption Schedule (ConsumptionSchedule)
Consumption Schedule Share (ConsumptionScheduleShare)
Contact (Contact)
Contact Point Address (ContactPointAddress)
Contact Point Address Share (ContactPointAddressShare)
Contact Point Consent (ContactPointConsent)
Contact Point Consent Share (ContactPointConsentShare)
Contact Point Email (ContactPointEmail)
Contact Point Email Share (ContactPointEmailShare)
Contact Point Phone (ContactPointPhone)
Contact Point Phone Share (ContactPointPhoneShare)
Contact Point Type Consent (ContactPointTypeConsent)
Contact Point Type Consent Share (ContactPointTypeConsentShare)
Contact Request (ContactRequest)
Contact Request Share (ContactRequestShare)
Content Delivery (ContentDistribution)
Content Document Link (ContentDocumentLink)
Content Folder (ContentFolder)
Content Security Policy Trusted Site (CspTrustedSite)
Content Version (ContentVersion)
```

```
<?xml version="1.0" encoding="UTF-8" stanc
<xsd:schema xmlns:xsd="http://www.w3.org
  <xsd:element name="sObjects">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element maxOccurs="1" minC
          <xsd:annotation>
            <xsd:documentation>ContentI
          </xsd:annotation>
        </xsd:element>
        <xsd:element maxOccurs="1" minC
          <xsd:annotation>
            <xsd:documentation>Content
          </xsd:annotation>
          <xsd:simpleType>
            <xsd:restriction base="xsd:strin
              <xsd:minLength value="0"/>
              <xsd:maxLength value="133
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element maxOccurs="1" minC
          <xsd:annotation>
            <xsd:documentation>Content
          </xsd:annotation>
        </xsd:element>
        <xsd:element maxOccurs="1" minC
          <xsd:annotation>
            <xsd:documentation>Title</xs
          </xsd:annotation>
          <xsd:simpleType>
            <xsd:restriction base="xsd:strin
              <xsd:minLength value="0"/>
```

Figure 11.47 XSD Generator for Content Version, Salesforce Object – Create

3. Import the XSD into the SAP CI mapping object to observe and map to this target XSD structure.



sObjects

Search

Structure	Occurrence
PathOnClient	0..1
ContentModifiedDate	0..1
OwnerId	0..1
TagCsv	0..1
VersionData	0..1
FirstPublishLocationId	0..1

Figure 11.48 Content Version, Salesforce Object - Create XSD in Message Mapping

4. Determine which attributes are specific to the scenario to populate in the sObjects XML. Please find an example payload below. The VersionData must be mapped as binary Base 64 encoded data.
   An option is to encode the data with the Base64 Encode in SAP CI.
   Note that (for Salesforce API version 50.0), the attribute Title is mandatory as part of the XSD. To properly insert the VersionData, the ContentDocumentId, ReasonForChange, PathOnClient, OwnerId, and the VersionData itself are needed.

```
<?xml version="1.0" encoding="UTF-8"?>
<sObjects>
  <ContentDocumentId>0695E000002vymCQAQ</ContentDocumentId>
  <Title>str1234</Title>
  <ReasonForChange>Marketing materials updated</ReasonForChange>
  <PathOnClient>Q1 Sales Brochure.pdf</PathOnClient>
  <VersionData>{in.body}</VersionData>
  <OwnerId>0055E00000AZjLe</OwnerId>
</sObjects>
```

5. Configure the Salesforce Receiver Adapter accordingly:

Example Configuration	
API Version	50.0
Operation	Salesforce Object - Create
Payload Format	Application/XML
Salesforce Object	Content Version
Pretty Print	Enable

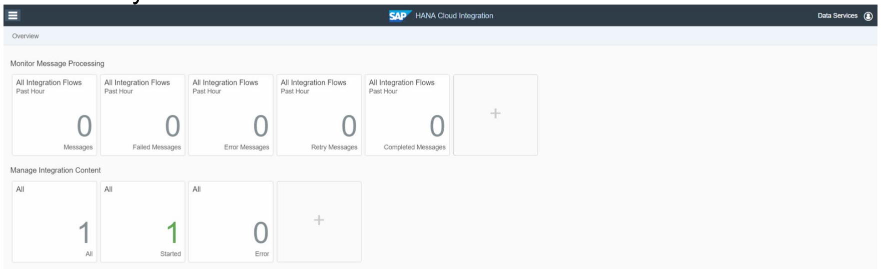
6. In the response the id of the Content Version will be returned:

```
<?xml version="1.0" encoding="UTF-8"?>
<Result>
  <id>0685E000002njNpQAI</id>
  <success>true</success>
</Result>
```

11.7 Monitor and Test Integration Flow in SAP Cloud Integration with Salesforce

11.7.1 Monitor Integration Flow in SAP Cloud Integration

1. After deploying an iFlow select ☰ and choose the Monitor Tab to verify if the integration flow started successfully.



Monitor Message Processing

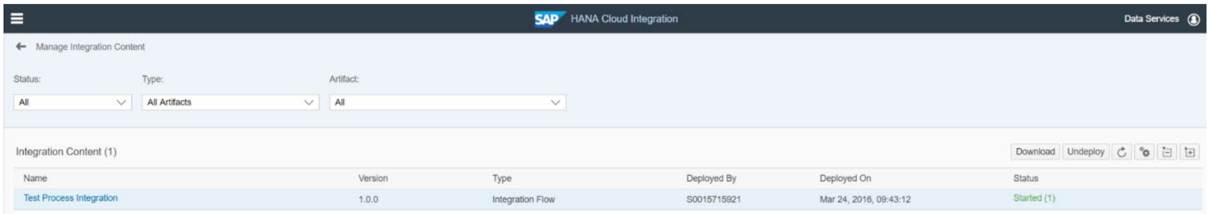
All Integration Flows Past Hour	All Integration Flows Past Hour	All Integration Flows Past Hour	All Integration Flows Past Hour	All Integration Flows Past Hour
0	0	0	0	0
Messages	Failed Messages	Error Messages	Retry Messages	Completed Messages

Manage Integration Content

All	All	All
1	1	0
All	Started	Error

Figure 11.49 SAP Cloud Integration – Monitor

2. Select the started Integration Flow in Manage Integration Content to see details. This provides an overview of all the messages and iFlows that successfully started or and those that encountered errors.



Status: All
Type: All Artifacts
Artifact: All

Integration Content (1)

Name	Version	Type	Deployed By	Deployed On	Status
Test Process Integration	1.0.0	Integration Flow	S0015715921	Mar 24, 2016, 09:43:12	Started (1)

Figure 11.50 SAP Cloud Integration - Integration Content

3. A Groovy Script can be added after logging the body payload in the Message Monitor. Select the Groovy Script. Click on Create and add the following script:

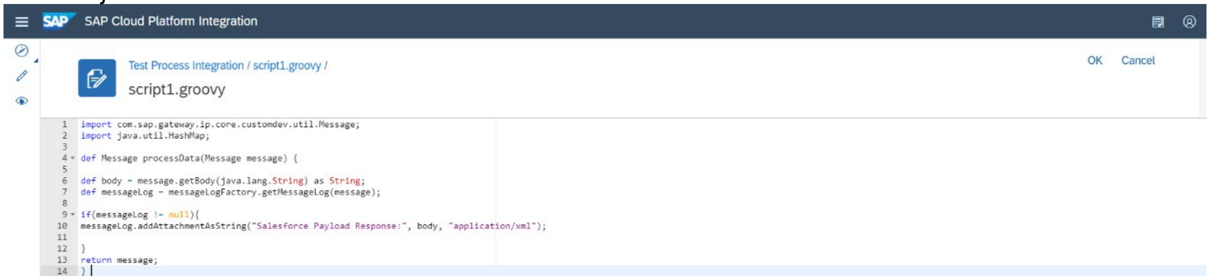
```
import com.sap.gateway.ip.core.customdev.util.Message;
import java.util.HashMap;

def Message processData(Message message) {

def body = message.getBody(java.lang.String) as String;
def messageLog = messageLogFactory.getMessageLog(message);

if (messageLog != null){
messageLog.addAttachmentAsString("Salesforce Payload Response:", body, "application/xml");

}
return message;
}
```


SAP Cloud Platform Integration
Test Process Integration / script1.groovy /
script1.groovy
OK
Cancel

```
import com.sap.gateway.ip.core.customdev.util.Message;
import java.util.HashMap;

def Message processData(Message message) {

def body = message.getBody(java.lang.String) as String;
def messageLog = messageLogFactory.getMessageLog(message);

if(messageLog != null){
messageLog.addAttachmentAsString("Salesforce Payload Response:", body, "application/xml");

}
return message;
}
```

Figure 11.51 Groovy Script

11.7.2 Monitor Message Processing in SAP Cloud Integration

1. After deploying an integration flow select ☰ and choose the Monitor tab to verify whether the integration flow started successfully or not.

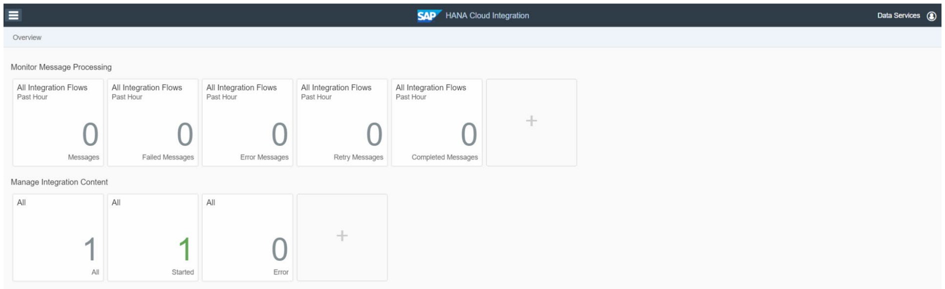
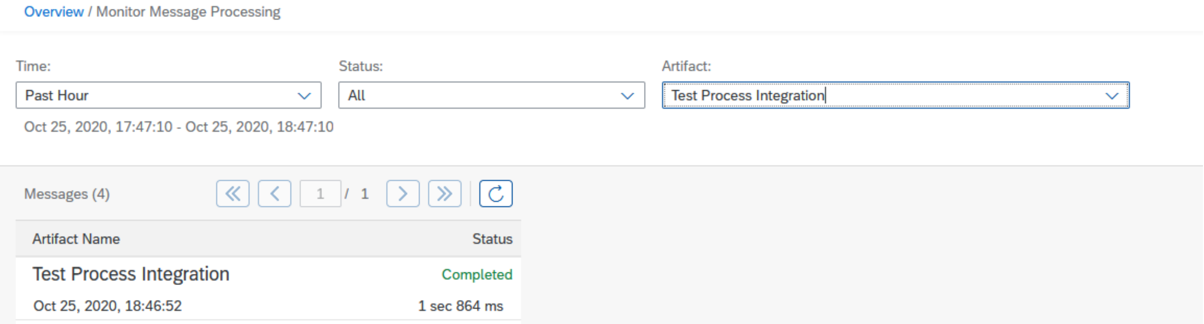


Figure 11.52 SAP Cloud Integration – Monitor

2. Filter on the started integration flow in Manage Message Processing to view the message details. This provides an overview of all the messages for the selected iFlow.



Overview / Monitor Message Processing

Time: Past Hour

Status: All

Artifact: Test Process Integration

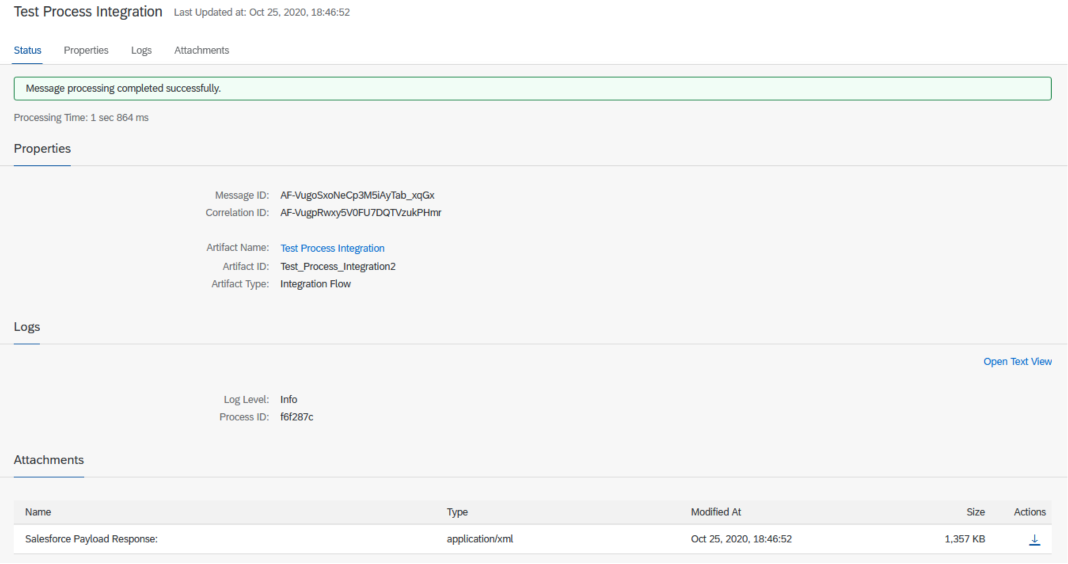
Oct 25, 2020, 17:47:10 - Oct 25, 2020, 18:47:10

Messages (4)

Artifact Name	Status
Test Process Integration	Completed
Oct 25, 2020, 18:46:52	1 sec 864 ms

Figure 11.53 SAP Cloud Integration - Message Processing

3. Here the Status, Properties, Logs, and Attachments of the Message are visible.



Test Process Integration Last Updated at: Oct 25, 2020, 18:46:52

Message processing completed successfully.

Processing Time: 1 sec 864 ms

Properties

Message ID: AF-VugoSxoNeCp3M5iAyTab_xqGx
Correlation ID: AF-VugpRwxy5V0FU7DQTVzukPHmr

Artifact Name: Test Process Integration
Artifact ID: Test_Process_Integration2
Artifact Type: Integration Flow

Logs

Log Level: Info
Process ID: f6f287c

Attachments

Name	Type	Modified At	Size	Actions
Salesforce Payload Response:	application/xml	Oct 25, 2020, 18:46:52	1,357 KB	⤓

Figure 11.54 Message Monitor – Message processing details

12 REFERENCES

12.1 Initialize Username Password Credentials in Salesforce

1. The **Login URL** is the login URL that is used by the authorization server of Salesforce. In the documentation of Salesforce, this URL is specified as https://login.salesforce.com. This link directs the Salesforce adapter to the login page of Salesforce to authorize.
2. The **Basic Credential Name** refers to an alias of a Username and Password combination stored in the SAP Cloud Integration Secure Store. This is the exact Username and Password used to log in to the Salesforce console.
3. Open the **Monitor** tab in SAP Cloud Integration.
4. Select **Security Material** in the section **Manage Security Material**.

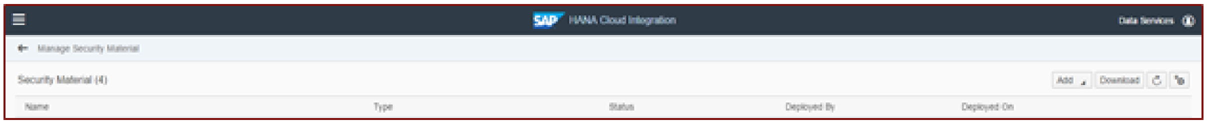
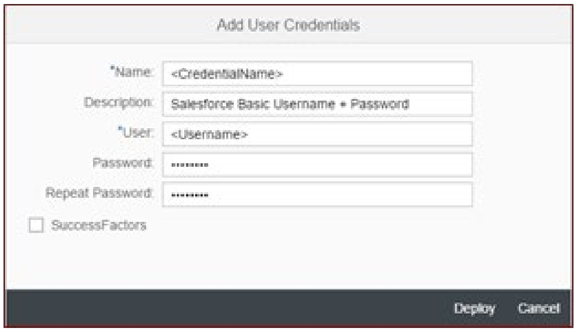


Figure 12.1 SAP Cloud Integration - Security Material

5. Select **Add**, select **User Credentials**, and enter a **Name** that should be the same as the Basic Credential Name configured in the Connection Tab of the Salesforce Adapter. In the User and Password fields, fill in the username and password used to log in to the Salesforce console. Click on Deploy.



Add User Credentials

*Name: <CredentialName>

Description: Salesforce Basic Username + Password

*User: <Username>

Password: ••••••••

Repeat Password: ••••••••

□ SuccessFactors

Figure 12.2 Add User Credentials for Basic Authentication

6. For the Security Token and OAuth Credentials, an app is required to be created in the Salesforce tenant.
7. Login to the Salesforce console and select **Setup**. On the left panel of the Build overview, select **Create**. Click on **Apps** and select **New** in the **Connected Apps** section.

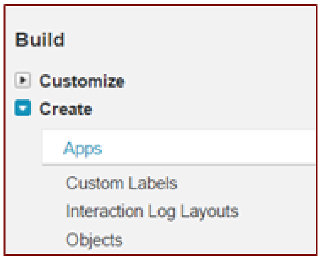
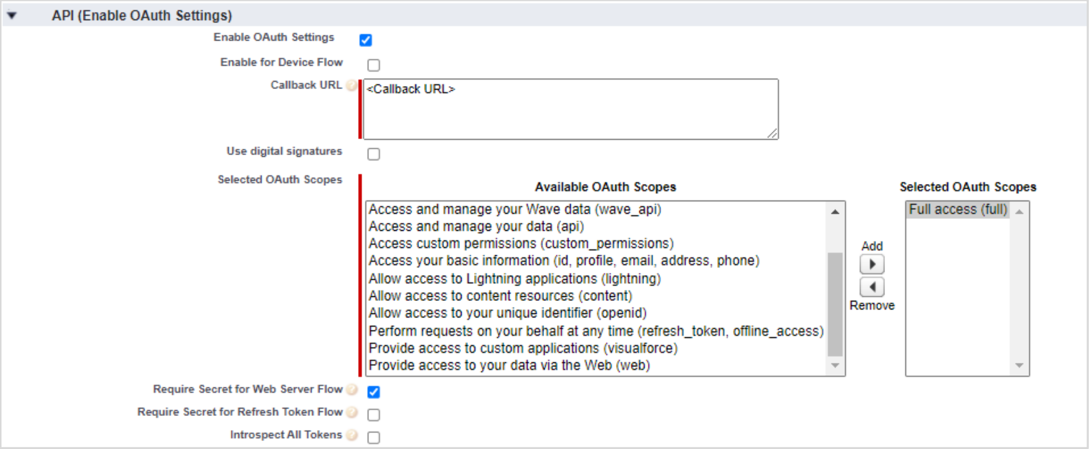


Figure 12.3 Create Salesforce App

8. In the next screen, fill in basic details such as **App Name**, **API Name**, and **Contact Email**.
9. In the **API (Enable OAuth Settings)** section, select **Enable OAuth Settings**. Figure 12.4 displays a sample configuration of the OAuth Settings.
10. Click on **Save** to complete the creation of the app.



▼ API (Enable OAuth Settings)

Enable OAuth Settings ☑

Enable for Device Flow □

Callback URL: <Callback URL>

Use digital signatures □

Selected OAuth Scopes

Available OAuth Scopes:
- Access and manage your Wave data (wave_api)
- Access and manage your data (api)
- Access custom permissions (custom_permissions)
- Access your basic information (id, profile, email, address, phone)
- Allow access to Lightning applications (lightning)
- Allow access to content resources (content)
- Allow access to your unique identifier (openid)
- Perform requests on your behalf at any time (refresh_token, offline_access)
- Provide access to custom applications (visualforce)
- Provide access to your data via the Web (web)

Selected OAuth Scopes:
- Full access (full)

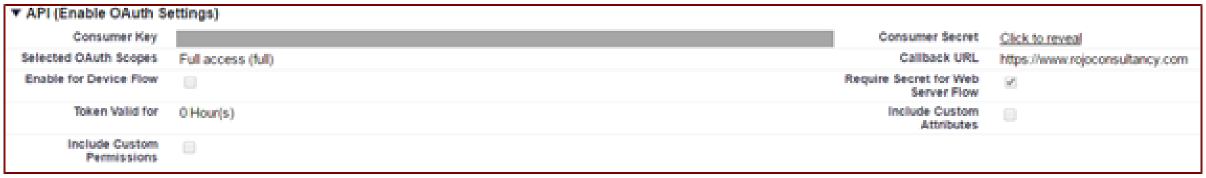
Require Secret for Web Server Flow ☑

Require Secret for Refresh Token Flow □

Introspect All Tokens □

Figure 12.4 Salesforce API Enable OAuth Settings

11. The next overview displays details for a **Connected App** which is selected. The **Consumer Key** and **Consumer Secret** can be seen in the respective Consumer Key and Consumer Secret fields.



▼ API (Enable OAuth Settings)

Consumer Key	[redacted]	Consumer Secret	Click to reveal
Selected OAuth Scopes	Full access (full)	Callback URL	https://www.rojoconsultancy.com
Enable for Device Flow	□	Require Secret for Web Server Flow	☑
Token Valid for	0 Hour(s)	Include Custom Attributes	□
Include Custom Permissions	□		

Figure 12.5 OAuth Setting of the app

12. Deploy the **Consumer Key** and **Consumer Secret** as User Credentials similarly as done in Step 2. Create a new Credential Name and fill in Consumer Key as User and Consumer Secret as Password.
13. In case the IP address of SAP Cloud Integration is not whitelisted in the settings of Salesforce, a **Security Token** is needed.
If the Security Token has not been received via email yet, navigate to **My Settings** and select **Personal**. Select **Reset My Security Token** to receive a new security token.

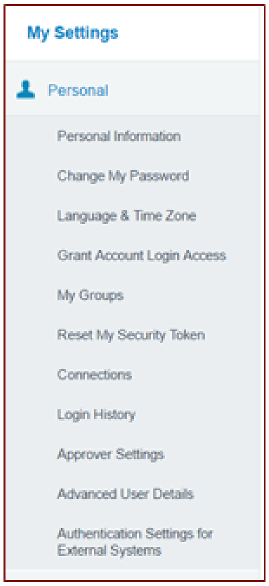
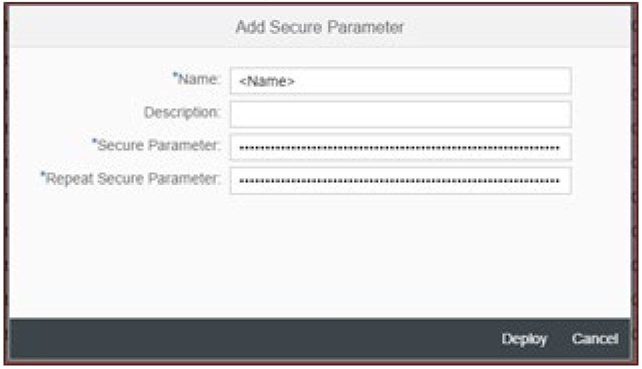


Figure 12.6 Obtaining the Security Token

14. To deploy the Security Token, select **Security Material** in the section **Manage Security Material**. Select **Add** and then select **Secure Parameter**.
15. Fill in the Secure Parameter **Name**. Note that this should match the **Security Token Name** used in the Salesforce Adapter. Fill in the Security Token in the **Secure Parameter** field and select **Deploy**.



Add Secure Parameter

*Name: <Name>

Description:

*Secure Parameter: ••

*Repeat Secure Parameter: ••

Figure 12.7 Deploy Security Token as a Secure Parameter

16. Security Details successfully deployed in SAP Cloud Integration ensure a secure OAuth 2.0 Autonomous Client connection for the Salesforce Adapter.

    The names configured for Basic Credential Name, Security Token, and OAuth Credential Name must match the names of the deployed artifacts in the SAP Secure Stores. With the authentication mechanisms in place, the Salesforce Adapter can be configured for specific Salesforce integration scenarios.

12.2 Initialize OAuth JWT Bearer in Salesforce

When using the OAuth JWT Bearer, the following properties need to be maintained in the Connection Tab:

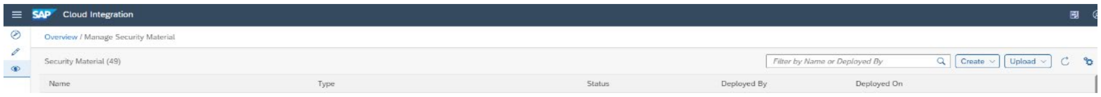
1. The **Audience** is the login URL used by the authorization server of Salesforce. In the documentation of Salesforce, this URL is specified as https://login.salesforce.com. This link directs the Salesforce Adapter to the login page of Salesforce.
2. The **Subject Alias** refers to an alias of a Secure Parameter stored in the SAP Cloud Integration Security Material. It specifies the username of the Salesforce user. Refer to Section 12.2.1, to add Subject Alias as a Secure Parameter.
3. The **Issuer Alias** refers to an alias of a Secure Parameter stored in the SAP Cloud Integration Security Material. It specifies the OAuth Consumer Key of the connected app for which the certificate was registered. Refer to Section 12.2.2, to create a Connected App in Salesforce and capture Consumer Key. Refer to Section 12.2.1, to add Issuer Alias as a Secure Parameter.
4. The **Keystore Alias** refers to the added JKS file in Keystore as a Key Pair. It consists of a key and certificate to sign the JWT. Refer to Section 12.2.3, to generate a JKS file using OpenSSL. Refer to Section 12.2.4, to add the created JKS file in Keystore.

Refer to Salesforce documentation for more information.

12.2.1 Deploy Secure Parameter in Cloud Integration

To deploy a Secure Parameter, follow the steps below:

1. Navigate to the **Monitor** Tab in SAP Cloud Integration.
2. Select **Security Material** in the section **Manage Security**.

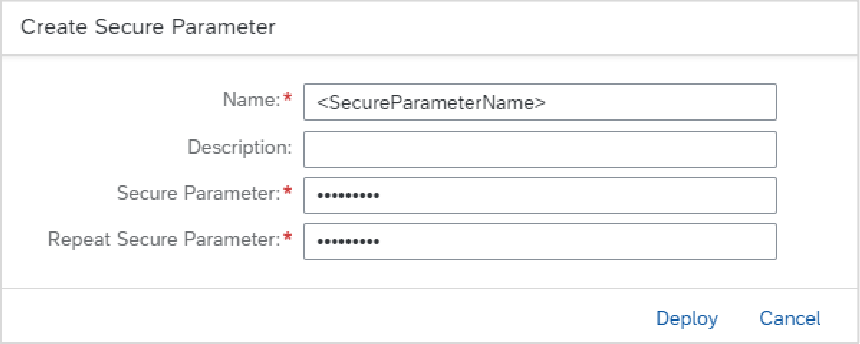

SAP Cloud Integration
Overview / Manage Security Material
Filter by Name or Deployed By
Create
Upload

Security Material (49)

Name	Type	Status	Deployed By	Deployed On

Figure 12.8 Cloud Integration - Security Material

3. Select **Create** and choose **Secure Parameter**. Supply a **Name** that matches the Alias configured in the Salesforce Adapter Authentication tab. In the Secure Parameter, fill in the **Secure Parameter** value. Click on **Deploy**.



Create Secure Parameter

Name:* <SecureParameterName>
Description:
Secure Parameter:* •••••••••
Repeat Secure Parameter:* •••••••••

Figure 12.9 Cloud Integration - Secure Parameter

12.2.2 Create a Connected App in Salesforce

To create a Connected App, follow the steps below:

1. Login to the Salesforce console and select **Setup**.
2. On the left panel in the **Build** overview, select **Create**. Click on **Apps** and select **New** for the **Connected Apps** section.

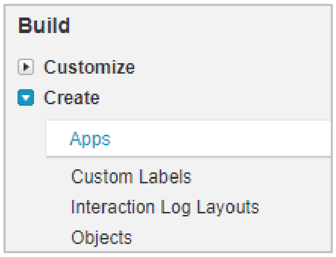
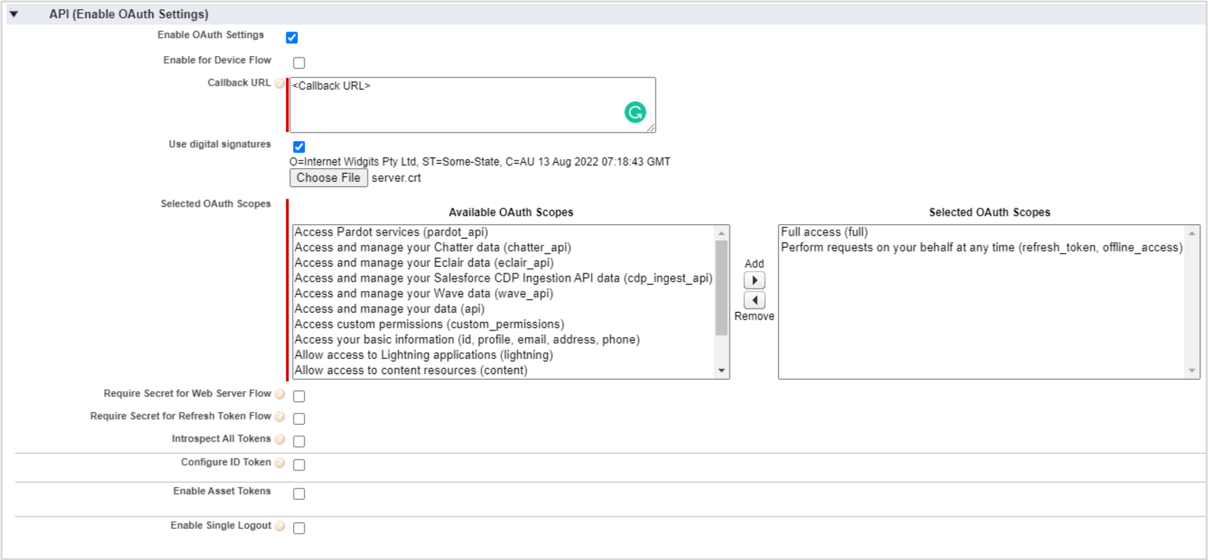


Figure 12.10 Create Salesforce App

3. In the next screen, fill in basic details such as **Connected App Name**, **API Name**, and **Contact Email**.
4. In the API (Enable OAuth Settings), select **Enable OAuth Settings**.
5. Select **Use Digital Signatures** and upload **Security Certificate(*.crt)**. Refer to section 12.2.3, on how to create a certificate. Figure 12.11 shows a sample configuration of the **OAuth Settings** section.



▼ API (Enable OAuth Settings)

Enable OAuth Settings	☑
Enable for Device Flow	□
Callback URL	<Callback URL>
Use digital signatures	☑ O=Internet Widgits Pty Ltd, ST=Some-State, C=AU 13 Aug 2022 07:18:43 GMT Choose File server.crt
Selected OAuth Scopes	
Require Secret for Web Server Flow	□
Require Secret for Refresh Token Flow	□
Introspect All Tokens	□
Configure ID Token	□
Enable Asset Tokens	□
Enable Single Logout	□

Available OAuth Scopes

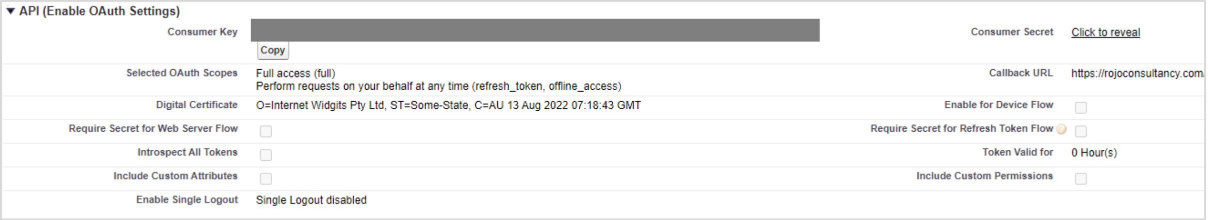
- Access Pardot services (pardot_api)
- Access and manage your Chatter data (chatter_api)
- Access and manage your Eclair data (eclair_api)
- Access and manage your Salesforce CDP Ingestion API data (cdp_ingest_api)
- Access and manage your Wave data (wave_api)
- Access and manage your data (api)
- Access custom permissions (custom_permissions)
- Access your basic information (id, profile, email, address, phone)
- Allow access to Lightning applications (lightning)
- Allow access to content resources (content)

Selected OAuth Scopes

- Full access (full)
- Perform requests on your behalf at any time (refresh_token, offline_access)

Figure 12.11 Salesforce API Enable OAuth Settings

6. Click on **Save** to complete the creation of the App.
7. In the next view, the Consumer Key can be seen in the respective field.

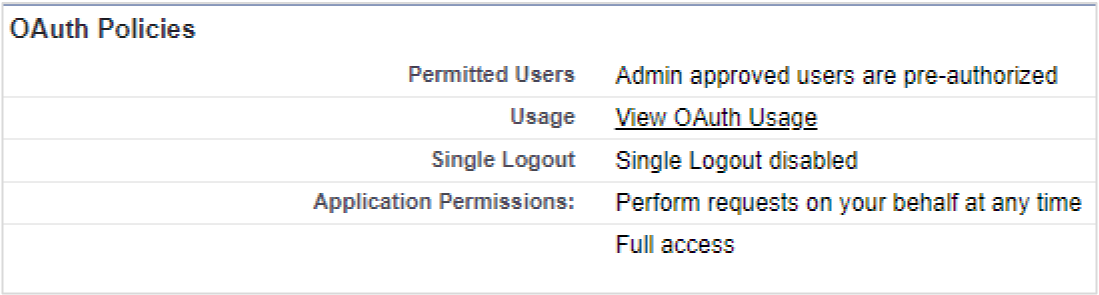


▼ API (Enable OAuth Settings)

Consumer Key	[redacted] Copy	Consumer Secret	Click to reveal
Selected OAuth Scopes	Full access (full) Perform requests on your behalf at any time (refresh_token, offline_access)	Callback URL	https://rojoconsultancy.com
Digital Certificate	O=Internet Widgits Pty Ltd, ST=Some-State, C=AU 13 Aug 2022 07:18:43 GMT	Enable for Device Flow	□
Require Secret for Web Server Flow	□	Require Secret for Refresh Token Flow	□
Introspect All Tokens	□	Token Valid for	0 Hour(s)
Include Custom Attributes	□	Include Custom Permissions	□
Enable Single Logout	Single Logout disabled		

Figure 12.12 Salesforce OAuth Setting indicating Consumer Key of the App

8. In the created App overview, select **Manage** and choose **Edit Policies**.
9. Change the **Permitted Users** to **Admin approved users are pre-authorized** from **OAuth Policies**. See Figure 12.13 for a sample configuration.

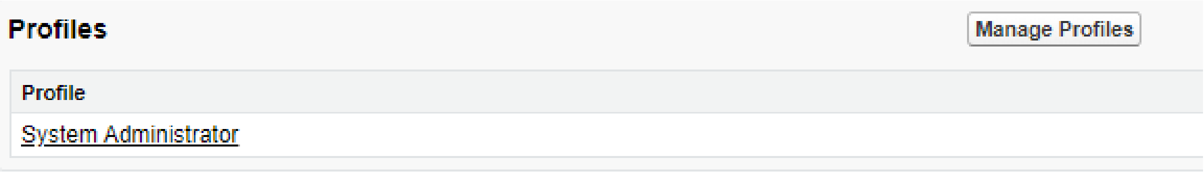


OAuth Policies

Permitted Users	Admin approved users are pre-authorized
Usage	View OAuth Usage
Single Logout	Single Logout disabled
Application Permissions:	Perform requests on your behalf at any time
	Full access

Figure 12.13 Salesforce App OAuth Policies

10. From the **Edit Policies** overview, select **Manage Profiles** then select **System Administrator** as shown in Figure 12.14.


Profiles

Profile
System Administrator

Figure 12.14 Salesforce App Manages Profiles

12.2.3 Create Security Certificate and JKS file using OpenSSL

To create Security Certificate and JKS file using OpenSSL, follow the steps below:

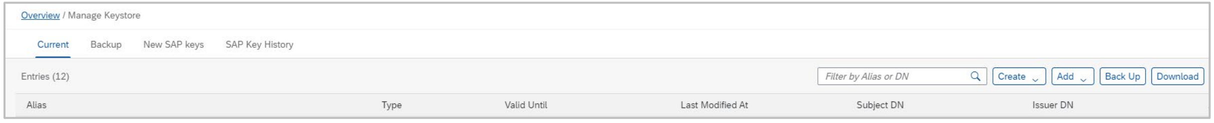
1. Generate a private key and store it in a file called server.key
   OpenSSL Command:
   - openssl genrsa -out server.pass.key 2048
   - openssl rsa -in server.pass.key -out server.key
4. Generate a certificate signing request by using the server.key file created in the previous step and store it in a file called server.csr. Enter the information about the organization when prompted.
   OpenSSL Command:
   - openssl req -new -key server.key -out server.csr
5. Generate a self-signed digital certificate from the server.key and server.csr files and store the certificate in a file called server.crt. This server.crt is uploaded when creating the Connected App in Salesforce.
   OpenSSL Command:
   - openssl x509 -req -sha256 -days 365 -in server.csr -signkey server.key -out server.crt
6. Generate Java Key Store (JKS) file as salesforcejwt.jks and name/alias as salesforcejwt by providing the files server.key and server.crt created in previous steps.
   OpenSSL Command:
   - openssl pkcs12 -export -in server.crt -inkey server.key -out salesforcejwt.jks -name salesforcejwt
   - Password:<PASSWORD>

Refer to Salesforce documentation for more information.

12.2.4 Add Key Pair in Cloud Integration Keystore

To add Key Pair to Cloud Integration Keystore, follow the steps below:

1. Open the **Monitor** Tab in SAP Cloud Integration.
2. Select **Keystore** in the Manage Security Section.

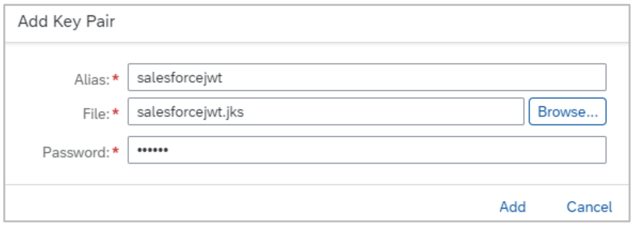

Overview / Manage Keystore

Entries (12)

Alias	Type	Valid Until	Last Modified At	Subject DN	Issuer DN

Figure 12.15 Cloud Integration – Keystore

3. Select **Add** and choose **Key Pair**. Supply an **Alias** and the **File** and **Password** combination created in Step 5 of Section 12.2.3.


Add Key Pair

Alias:* salesforcejwt

File:* salesforcejwt.jks

Password:* ••••••

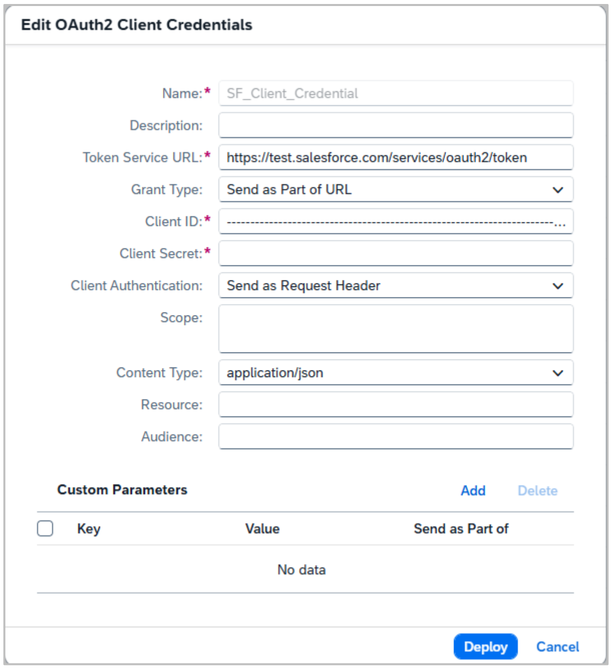
Figure 12.16 Cloud Integration – JKS file in Keystore

12.3 Create OAuth Client Credentials

- For more information on how to create OAuth Client Credentials, see OAuth 2.0 Client Credentials Flow.
- As per the instructions in the above link, you must create an external an external client app. For more information, see Create an External Client App.
- After completing the above setup, fetch **Consumer Key** and **Consumer Secret** for your app using **External Client App Manager**. You will require those details in the next step
- For creating OAuth2 Client Credentials in SAP Cloud Integration, see 12.3.1.

12.3.1 Creating OAuth2 Client Credentials in Security Material

1. Open the **Monitor** Tab in SAP Cloud Integration.
2. Select **Security Material** in the **Manage Security** section.
3. On the right of your screen, click **Create** and select **OAuth2 Client Credentials.**
4. Populate the following details
    a. `https://yourDomain.my.salesforce.com/services/oauth2/token` **as Token Service URL**.
    b. Consumer Key as **Client ID.**
    c. Consumer Secret as **Client Secret.**
    d. Retain Client Authentication as **Send as Request Header**.
    e. Retain Content Type as **application/json.**



Edit OAuth2 Client Credentials

Name:*	SF_Client_Credential
Description:	
Token Service URL:*	https://test.salesforce.com/services/oauth2/token
Grant Type:	Send as Part of URL
Client ID:*	--...
Client Secret:*	
Client Authentication:	Send as Request Header
Scope:	
Content Type:	application/json
Resource:	
Audience:	

Custom Parameters

□	Key	Value	Send as Part of
		No data	

5. Click **Deploy** to finish.

www.sap.com/contactsap

The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.