Informatica

Informatica® Application Integration
October 2024

SAP BAPI Connector Guide

Informatica Application Integration SAP BAPI Connector Guide
October 2024

Publication Date: 2024-10-04

Table of Contents

Preface

Read the *SAP BAPI Connector Guide* to learn how to set up and use connections with SAP BAPI.

This guide assumes that you are familiar with SAP BAPI and how to create connections and processes in Application Integration.

CHAPTER 1

Introduction to SAP BAPI Connector

This chapter includes the following topics:

SAP BAPI Connector

Business Application Programming Interfaces (BAPI) provide a way for third-party applications to synchronously integrate with SAP at the object level. You can use BAPIs to read, create, change, or delete data in SAP.

SAP provides the ability to remotely call business functions through Remote Function Call (RFC). BAPIs are a subset of the RFC-enabled functions published by SAP in the Business Object Repository. BAPIs are published as methods of SAP business object types that perform specific business functions. BAPIs allow access to the SAP system objects through methods for the business object types. Together with the business object types, BAPIs define and document the interface standard at the business level.

The SAP BAPI Connector is available as a service call in Application Integration. For example, to update the sales order data in SAP, you can configure an SAP BAPI connection to access the BAPI_SALESORDER_CHANGE function.

You can view and test the BAPI interface definitions in SAP using transaction SE37.

SAP BAPI Connector Implementation

You can create an SAP BAPI connection to connect to the SAP system and access a specific BAPI function. You can also access table type parameters.

You can configure a service call in a process to pass data to and from the SAP system.

BAPI/RFC functions use function parameter values to perform tasks.

BAPI/RFC functions can have the following parameters:

Function Parameter	Description
Import	Input values. Some BAPI functions require input values to perform tasks such as changing data.
Export	Output values that a BAPI function returns after performing a task.
Table	SAP structures with more than one row. Table parameters can be input, output, or both. Input table parameters pass table input values to a BAPI/RFC function. For example, some BAPI/RFC functions require table inputs to change data.

Based on the task you want to perform, you can configure the SAP BAPI connection in a service call in the process and map the input and output parameters to the event sources and targets required for the BAPI.

SAP BAPI Connector Administration

Before you can use an SAP BAPI connection to process data through BAPIs, an administrator must perform the following tasks:

- Verify that the BAPI package is registered.
- Download and configure the SAP libraries.
- Configure SAP user authorizations.

Downloading and Configuring SAP Libraries for BAPI

Download the SAP JCo libraries and configure them on the machine where the Secure Agent runs. Contact SAP Customer Support if you encounter any issues with downloading the libraries.

1. Go to the SAP Service Marketplace: http://service.sap.com.

   Note: You must have SAP credentials to access the Service Marketplace.
2. Download the most recent version of the 64-bit SAP JCo libraries based on the operating system on which the Secure Agent runs::

Secure Agent System	SAP File Names
Windows	`sapjco3.jar` `sapjco3.dll`
Linux	`sapjco3.jar` `libsapjco3.so`

3. Copy the libraries to the following directory:

   `<Informatica Secure Agent installation directory>\apps\process-engine\ext`
4. Restart the Secure Agent.

Configuring SAP User Authorizations

An SAP administrator needs to create a profile in the development, test, and production SAP system so that you can use the integration features. This profile name must include authorization for the objects and related activities. The profile on the test system should be the same as the profile on the production system.

The setup of the user and profiles is done within SAP using the SAP GUI. This activity is external to Informatica Cloud®.

The following table describes the authorization an SAP user requires to execute tasks using the BAPI/RFC functions:

Authorization Object	Authorization Value
S_RFC	SYST, SDTX, SDIFRUNTIME, RFC1, RFC2

Note: In addition to the above authorization, the user needs access to any BAPI/RFC function that needs to be executed.

CHAPTER 2

SAP BAPI Connections

This chapter includes the following topics:

SAP BAPI connections overview

You can create an SAP BAPI connection to connect to the SAP system and access a specific BAPI function. You can also access table type parameters.

After you create an SAP BAPI connection, you can validate, save, and test the connection.

You can then publish the SAP BAPI connection and click the **Metadata** tab to view the generated process objects for the connection.

Basic connection properties

The following table describes the basic properties that you can configure on the **Properties** tab of the connection creation page:

Property	Description
Name	Required. Unique name for the SAP BAPI connection that identifies it in the Process Designer. The name must start with an alphabet and can contain only alphabets, numbers, or hyphens (-).
Location	The location of the project or folder where you want to save the connection. Click **Browse** to select a location. If the **Explore** page is currently active and a project or folder is selected, the default location for the connection is the selected project or folder. Otherwise, the default location is the location of the most recently saved asset.

Property	Description
Description	Optional. Description of the connection.
Type	Required. The type of connection you want to use for the connector or service connector. Select **SAP BAPI**.
Run On	Required. The name of the Secure Agent group or the Secure Agent machine where the connection must run.
Connection Test	Indicates whether the connection test was successful or not. By default, the property displays the results of the connection test.
OData-Enabled	Not supported for SAP BAPI Connector.

Along with these basic properties, you must define the properties applicable to the SAP BAPI connection type.

The **Metadata** tab displays the process objects generated when you publish the SAP BAPI connection.

SAP BAPI connection properties

The following table defines the basic SAP BAPI connection properties that you must configure:

Property	Description
Username	Required. SAP user name with authorization on S_DATASET, S_TABU_DIS, S_PROGRAM, and B_BTCH_JOB objects.
Password	Required. SAP password.
Host Name	Required. Logical name of the SAP system that you want to connect to.
Client	Required. SAP client number.
Language	Optional. Language code that corresponds to the SAP language.
System Number	Required. SAP system number.

The following table defines the advanced SAP BAPI connection properties that you can configure:

Property	Description

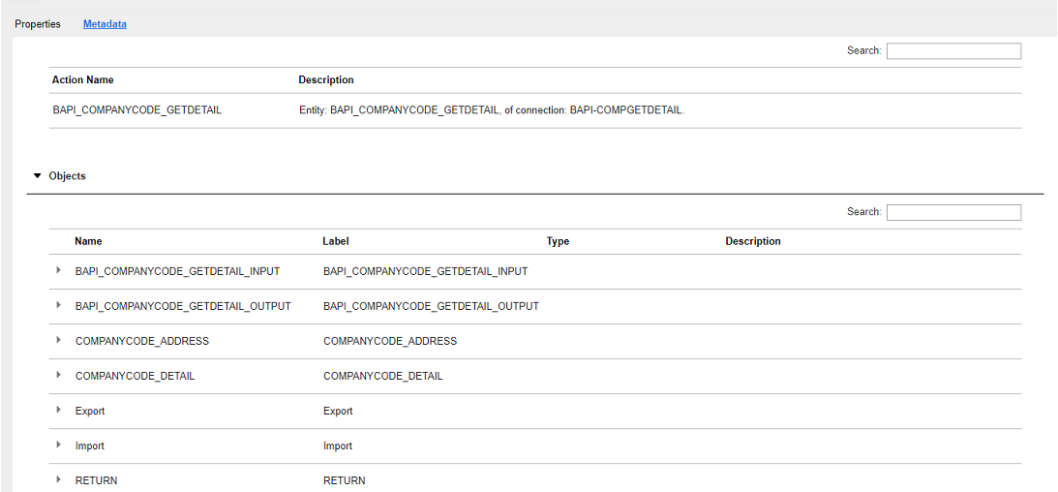
SAP Additional Parameters	Optional. Additional SAP parameters that the Secure Agent must use to connect to the SAP system as an RFC client.
	For example, you can define the load balancing parameters as shown in the following sample:
	`GROUP=interfaces`
	`MSHOST=<Message server hostname>`
	`R3NAME=<System ID or name of SAP system>`
	SAP infers the connection type based on the parameters that you specify. For example, if you define the GROUP, MSHOST, and R3NAME parameters, SAP infers the connection type as a load balancing connection. The GROUP parameter defines the group name of the SAP application server. The MSHOST parameter defines the host name of the SAP message server. The R3NAME parameter defines the system ID or name of the SAP system.
	You can define the `DOCOMMIT=true` parameter to commit data to the SAP system with each BAPI/RFC call.
	For more information about SAP parameters, see the SAP documentation.
	You can also enable the Secure Network Communication (SNC) protocol and secure communications between Application Integration and SAP by adding properties in the **SAP Additional Parameters** field. For more information about configuring the SNC parameters, see How to configure the SAP Secure Network Communication Protocol in Application Integration.
	Note: If you configure a parameter in both the **SAP Additional Parameters** field and in other connection property fields, the values in the **SAP Additional Parameters** field override the values in the connection property fields.

Jco Trace	Optional. Enables or disables JCo trace. Default is **No**. If you enable JCo trace, you can access the JCo trace file from the following directory: `<Secure Agent installation directory>\apps\process-engine\<latest_version>`
Object Filter	Optional. Fully qualified name of the BAPI function that you want to connect to. You can specify a single BAPI name.

SAP BAPI connection metadata

After you create or update an SAP BAPI connection, you can save and publish the connection. You can view the connection metadata on the **Metadata** tab.

The following image shows the published metadata of an SAP BAPI connection:



When you publish an SAP BAPI connection, the **Metadata** tab is refreshed by default. You can click **Publish - Skip Metadata Refresh** to skip the metadata refresh and reduce the publishing time.

Best practices for metadata refresh

- When you create an SAP BAPI connection for the first time, click **Publish** to download the metadata. Subsequently, you can click **Publish - Skip Metadata Refresh** if there are no metadata changes.
- If you want to change credentials in an SAP BAPI connection but you know that the metadata has not changed, skip the metadata refresh while publishing the connection to apply just the credential changes.
- If you know that the metadata has changed, you must click **Publish** to retrieve the metadata changes.

Understanding SAP BAPI connection metadata

You can view the import, export, and table parameters of the BAPI function as process objects when you publish the SAP BAPI connection.

BAPI Input and Output Process Objects

For any BAPI function, the input parameters are published in the <BAPI name>_INPUT process object and the output parameters are published in the <BAPI name>_OUTPUT process object. For example, if the BAPI name is BAPI_COMPANYCODE_GETDETAIL, you can view BAPI_COMPANYCODE_GETDETAIL_INPUT and BAPI_COMPANYCODE_GETDETAIL_OUTPUT process objects in the object list.

In a BAPI function, based on the requirement, you can specify the table parameters as input, output, or both. So, the table structure is replicated in both the input and the output process objects. The <BAPI name>_INPUT process object contains the Table_Input parameter and the <BAPI name>_Output process object contains the Table_Output parameter.

Note: Although the table input and table output parameters have the same structure, you can view individual process objects for the table input and output parameters. For example, for the BAPI_COMPANYCODE_GETLIST BAPI function, the table parameter COMPANYCODE_LIST is listed as COMPANYCODE_LIST and COMPANYCODE_LIST1 process objects. In this scenario, the type of the Table_Input process object is COMPANYCODE_LIST and the type of the Table_Output process object is COMPANYCODE_LIST_1.

The <BAPI name>_INPUT process object contains the following objects:

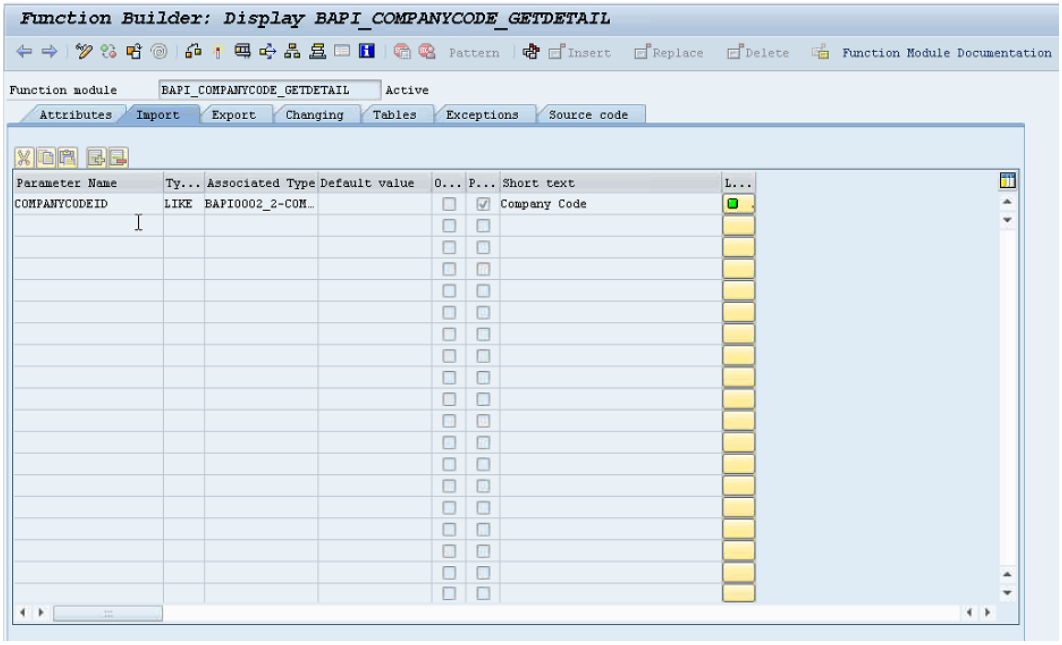
- BAPI import parameters
- BAPI Table input parameters

The <BAPI name>_OUTPUT process object contains the following objects:

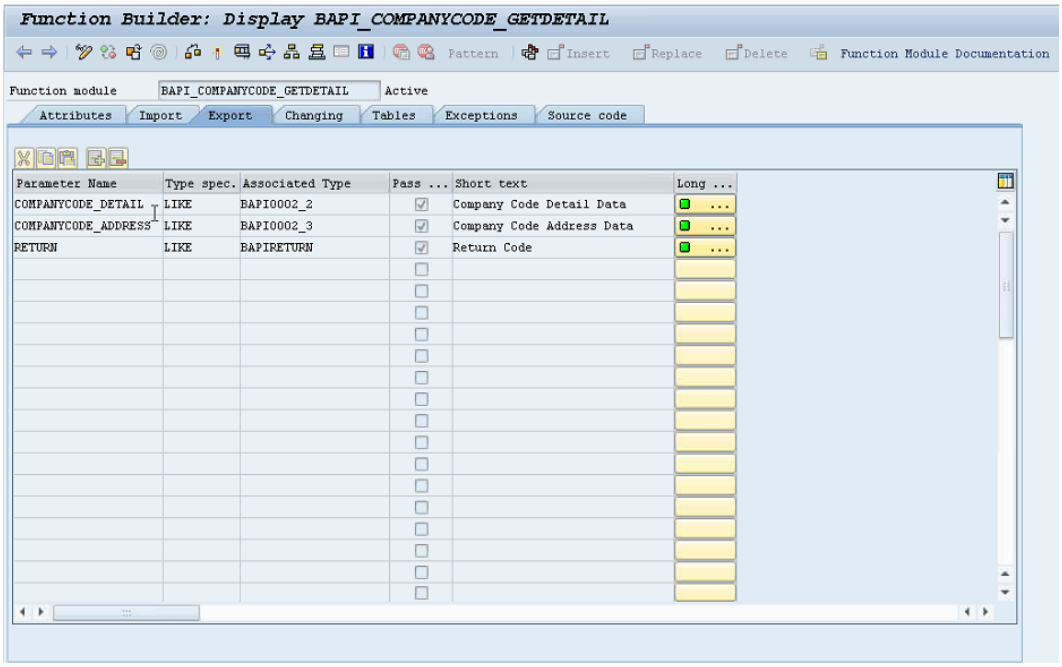
- BAPI export parameters
- BAPI Table output parameters

To view the field details in a process object, you can expand the process object. Based on the BAPI structure, the fields in a process object can be associated with a data type or with another hierarchical structure. All the fields in the process object are listed alphabetically.

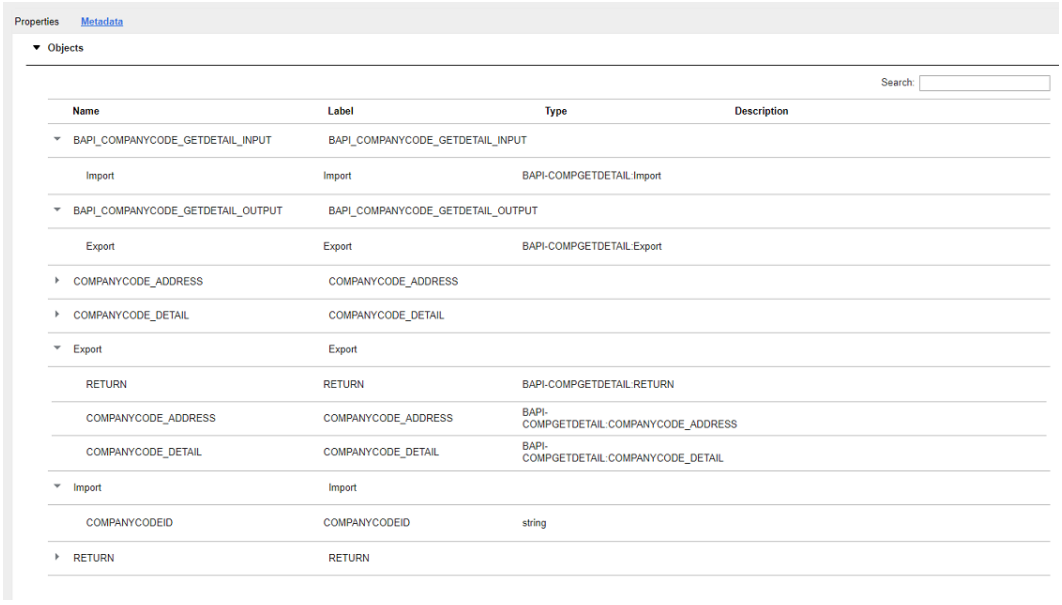
The following image shows the import parameters of the BAPI_COMPANYCODE_GETDETAIL function in the SAP system:



The following image shows the export parameters of the BAPI_COMPANYCODE_GETDETAIL function in the SAP system:



The following image shows the published import, export, and table metadata of the SAP BAPI connection that connects to the BAPI_COMPANYCODE_GETDETAIL function:



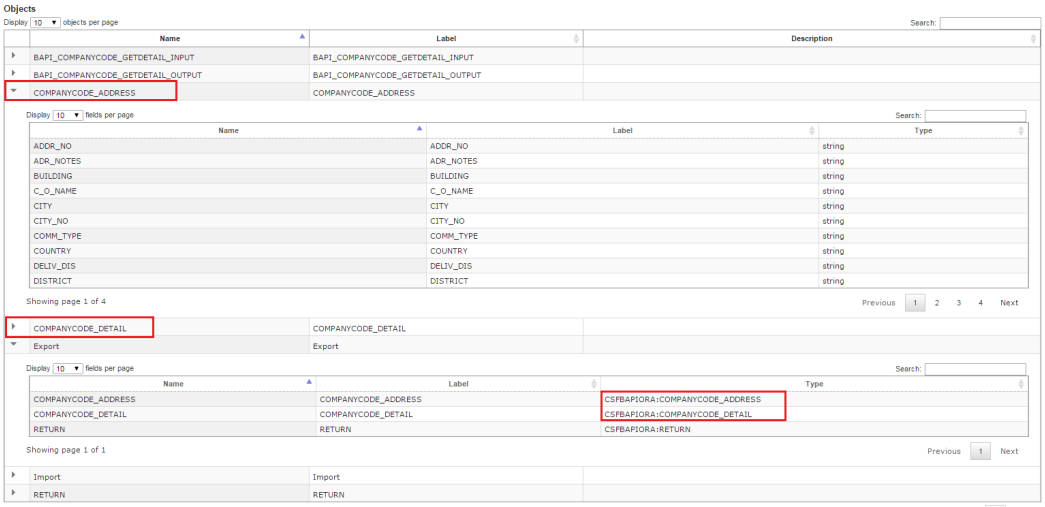
Nested BAPI Structures

A BAPI function parameter can be associated with another structure or a data type, When a BAPI function parameter is associated with another structure, you can view the name of the structure in the **Type** field. Each structure in the BAPI function is listed as a process object in the published metadata.

For example, when you expand the Export process object, you can view the COMPANYCODE_ADDRESS, COMPANYCODE_DETAIL, and RETURN parameters. All these parameters are nested structures and are

associated with another structure. Each nested structure is listed as individual process objects in the Objects list. So, you can view COMPANYCODE_ADDRESS, COMPANYCODE_DETAIL, and RETURN structures in the Object list. You can expand the process object to view all the fields and data types in the structure.

The following image shows the process objects in the published metadata of the SAP BAPI connection:



Creating an SAP BAPI connection

You can create an SAP BAPI connection to connect to the SAP system and access a specific BAPI function. You can also access table type parameters.

1. In Application Integration, click **New** > **App Connections** > **App Connection** > **Create**.

   The **Connection** page appears.
2. Select **SAP BAPI** as the connection type.
3. Enter a name and description for the connection.
4. Select a Secure Agent group or Secure Agent machine on which the connection must run.
5. Enter the SAP BAPI connection properties.

   The **OData-Enabled** option is not applicable to the SAP BAPI connection.
6. Test the connection.

   Important: You must ensure that you enter a valid SAP BAPI function name and connection properties. Also, enable Jco trace to view any connection issues.
7. Save and publish the connection.

   You can click the **Metadata** tab to view the BAPI metadata that the connection accesses in the SAP system.

CHAPTER 3

SAP BAPI Connector Processes

This chapter includes the following topics:

SAP BAPI Connector processes overview

You can use Process Designer in Application Integration to read, create, change, or delete data in SAP.

The SAP BAPI Connector is available as a service call in Application Integration. In a process, you can configure the SAP BAPI connection in a service call and map the input and output parameters to the event sources and targets required for the BAPI.

Process with SAP BAPI Connector example

You can read the company code ID from an event source, connect to the SAP system through a service call to the BAPI_COMPANYCODE_GETDETAIL function, and write the company code list to a relational data source.

In this example, you can configure a service call to access the BAPI_COMPANYCODE_GETDETAIL function, read the company code ID from a Salesforce source, and write the company address and details to an Oracle database.

Perform the following steps to read the company code ID from a Salesforce source and write the company address and details to an Oracle database:

1. Create the connections in Application Integration.
2. Create the variables in the process properties.
3. Read from Salesforce and assign the variables in the process.
4. Configure the BAPI service call in the process.
5. Write the BAPI output parameters to Oracle tables.
6. Run the process.

Step 1: Create the connections

Create multiple connections in Application Integration to connect to the native systems.

Create the following connections:

- Salesforce connection to read company code.
- SAP BAPI connection to access the BAPI_COMPANYCODE_GETDETAIL function in the SAP system.
- JDBC connection to Oracle to write the list of company codes to an Oracle database.

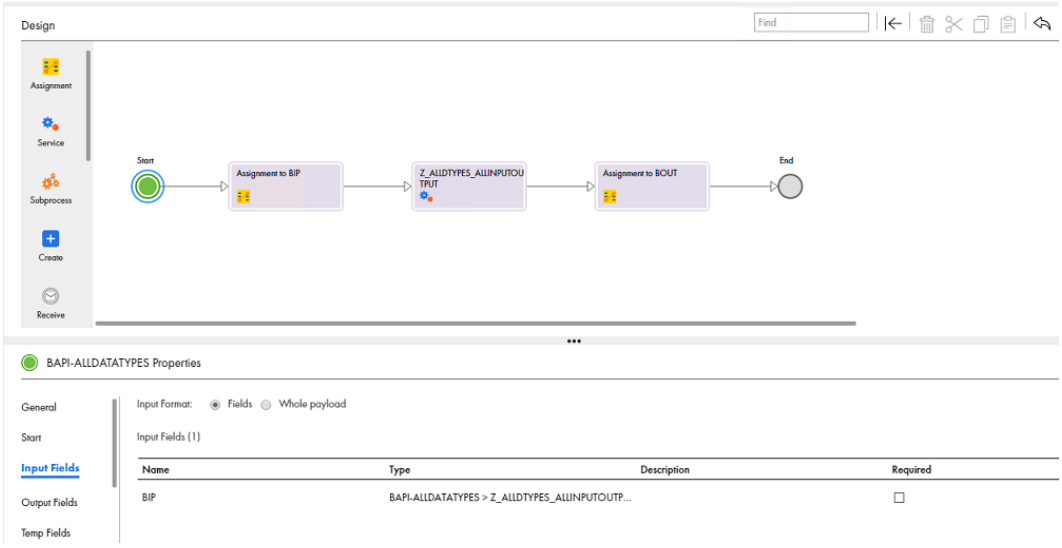
Step 2: Create the variables

Create variables in the process and assign the input and output fields.

For this example, create the following variables:

- A Salesforce input variable to read company code ID from Salesforce.
- A BAPI input variable to capture the BAPI_COMPANYCODE_GETDETAIL_Input parameter and read the company code from the Salesforce source.
- A BAPI output variable to capture the BAPI_COMPANYCODE_GETDETAIL_Output parameter and write the company code details to the Oracle tables.

The following image shows the multiple variables you need to create:

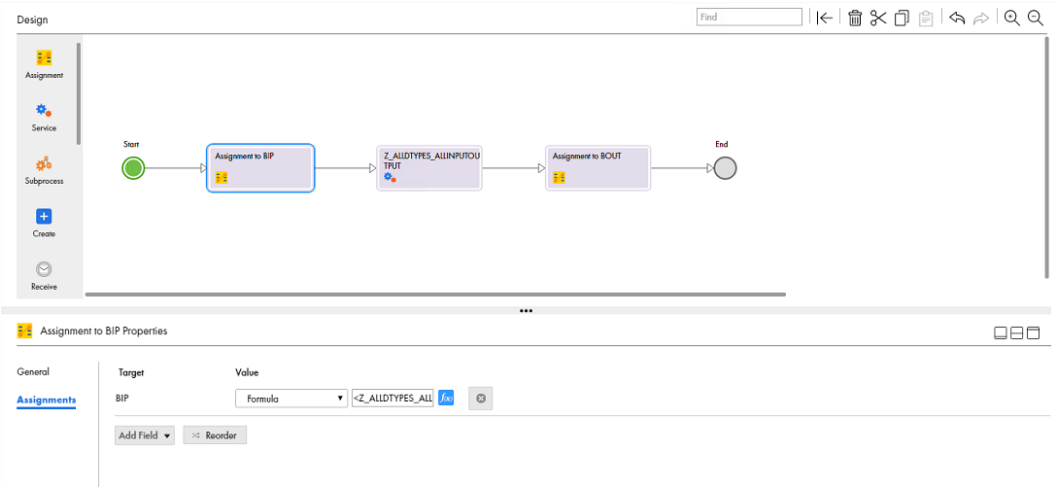


Step 3: Assign the variables

After you create the variables, assign the required input and output variables.

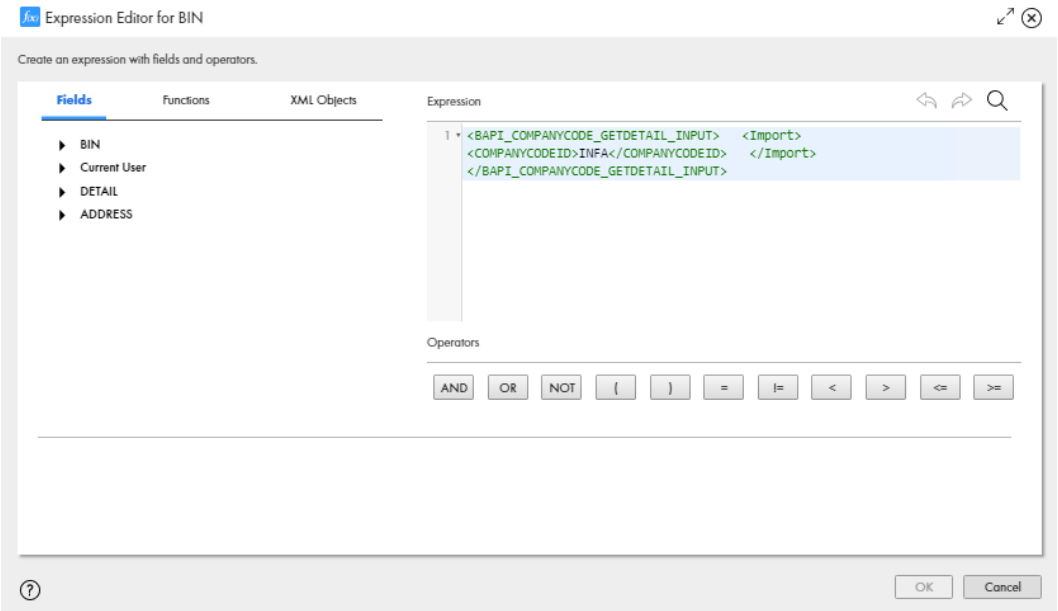
Assign the Saleforce input variable to read the company code ID from Salesforce. Configure the Assignment step to add the Salesforce input variable field.

The following image shows the assignment of the Salesforce input field:



Assign the BAPI input variable to read the company code IDs from the Salesforce input variable. Configure the Assignment step to add the BAPI input variable field. Select Formula as the source of the input field. You can enter a formula read all the input fields from the Salesforce input variable.

The following image shows the assignment of the BAPI input field:

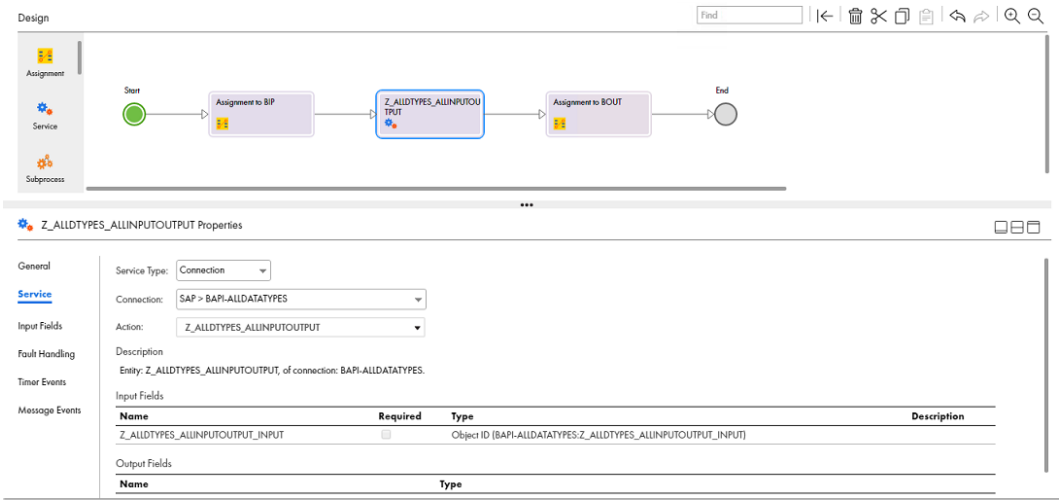


Step 4: Configure the BAPI service call

Configure a Service call in the process to configure the input to the BAPI input parameter.

Configure the BAPI service call to use the SAP BAPI connection that accesses the Z_ALLDTYPES_ALLINPUTOUTPUT_INPUT function.

The following image shows the service call configuration:



Configure the input field in the service call to read from the Z_ALLDTYPES_ALLINPUTOUTPUT_INPUT input variable. In the **Input** tab, select **Field** as the value of the BAPI input parameter and select the **Z_ALLDTYPES_ALLINPUTOUTPUT_INPUT** input field.

The following image shows the configuration of the input field in the service call:

Step 5: Write the BAPI output parameters to Oracle tables

The BAPI_COMPANYCODE_GETDETAIL_Output parameter returns the COMPANYCODE_ADDRESS, COMPANYCODE_DETAIL, and Return parameters. You can write the output parameters to Oracle tables.

Perform the following steps to write the BAPI output to Oracle tables:

Configure parallel paths in the process.

Configure a Parallel Paths step in the process and add three paths to write each of the output parameters to individual Oracle tables.

Configure a Create object in the process.

For each path, configure a Create object. Perform the following steps to configure a Create object:

- Select an Oracle connection to specify the object type as an Oracle table.
- In the **Input** tab, map a field in the Oracle table with the associated field in the BAPI output parameter.
- Repeat the field mapping for all the output fields that you want to write to the Oracle table.

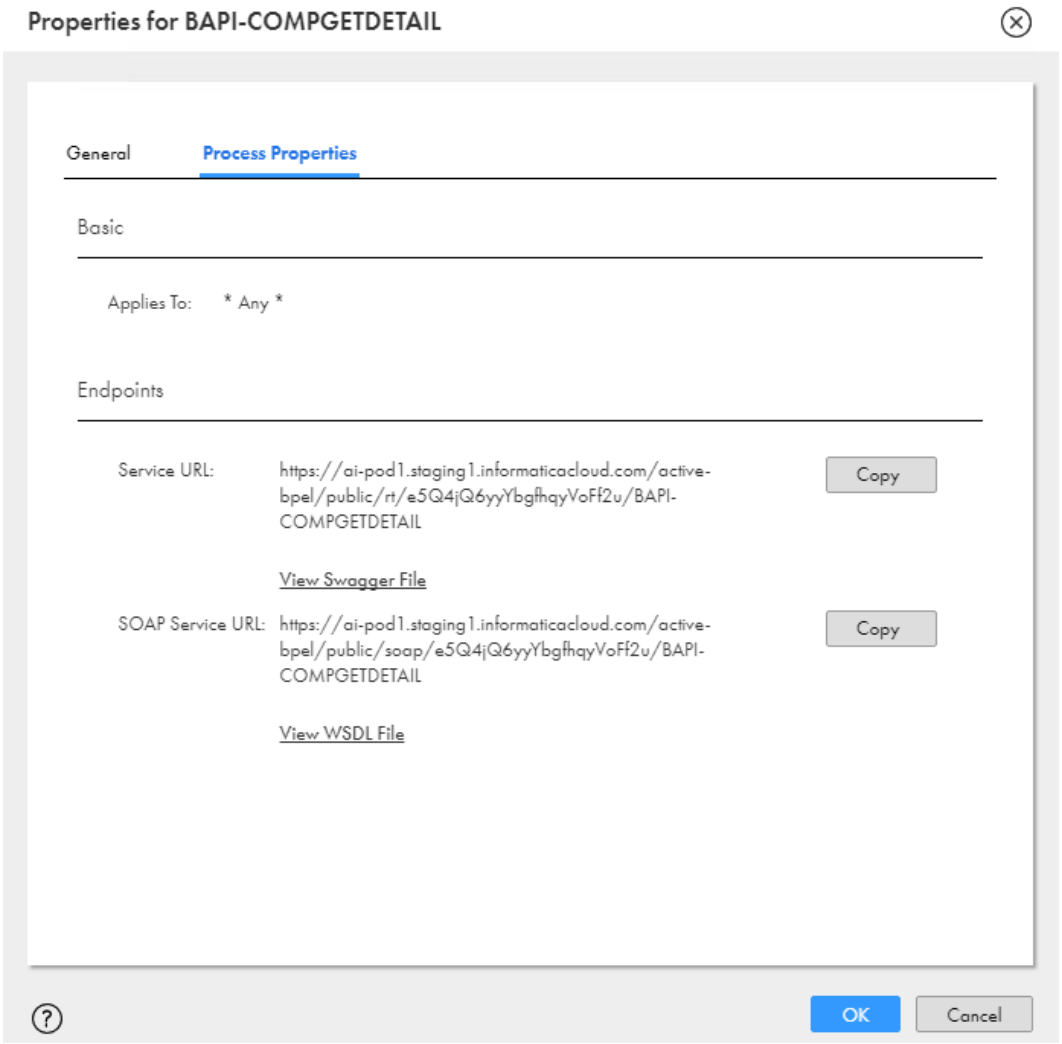
Step 6: Run the process

After you save the process, the new process appears in the **Processes** tab.

Click the **Start** tab in the **Process Properties** dialog box and enable the **Allow anonymous access** option.

Select the process in the Design Home page and copy the Service URL. Use the Service URL to run the process. The process runs and the Secure Agent writes the company address and details in the Oracle tables.

The following image displays the Service URL under the **Process Properties** tab:



Process with BAPI_SALESORDER_CREATEFROMDATA1 example

You can create a sales order document using the BAPI_SALESORDER_CREATEFROMDATA1 function.

Use the function to read from the Import fields and the Table input fields in the function. You specify the customer details and the ordered items to generate the sales order document.

As input, pass the order header, item details, and partner details. Order header contains fields such as number, doc type, sales organization, and division. The item details Table input contains fields such as item number, material, required quantity, sales unit, and required date. Assign the multiple input fields to a BAPI input field and pass the value to the BAPI_SALESORDER_CREATEFROMDATA1 function. SAP generates an unique sales number and creates the sales order based on your input details. The output of the BAPI is an unique sales document. You also get other details such as the party to whom the items are sold and the party that was billed for the items.

In this example, you can configure a service call to access the BAPI_SALESORDER_CREATEFROMDATA1 function and read the input details from an Oracle database. SAP generates multiple outputs such as the sales document, sold to party, ship to party, and bill to party details.

Perform the following steps to create the sales order:

1. Create the SAP BAPI and JDBC connections in Application Integration.
2. Create the input variables in the process properties. Create input fields for item details, order header, and partner details. Also, create a variable to provide the consolidated input the BAPI function.
3. Create the output variables in the process properties. Create output fields such as sales document, sold to party, ship to party, bill to party to capture the BAPI output.
4. Read the order header, item details, and partner details from the Oracle database and assign to the variables in the process.
5. Because Table input can have multiple inputs, create a formula using the For loop to read all the items from the order items input table object and assign to the BAPI input field.
6. To assign the partner details to the BAPI input field, create a formula using the For loop to read all the partner details.
7. Configure the BAPI service call to connect to the BAPI using the SAP BAPI connection.
8. Assign the BAPI input field to the BAPI_SALESORDER_CREATEFROMDATA1 input.
9. Assign the output fields to the BAPI output.
10. Run the process to print the sales order document.

INDEX