Developing OData Services
## Content

1. **Developing an OData Service.** .......................................................... 4
   1.1 Supported OData Features. .......................................................... 6
   1.2 Creating an OData Service. ......................................................... 7
   1.3 Create an OData Model. ............................................................. 9
       Import from Data Source. ............................................................ 9
       Editing an OData Model. ............................................................ 15
   1.4 Viewing the OData Model Layout. ............................................... 19
       Graphical Model Viewer Features. ............................................. 20
   1.5 Bind to Data Source. ................................................................. 21
       Binding to SOAP. ................................................................. 23
       Binding to OData. ............................................................... 24
       Binding to REST. ............................................................... 25
       Binding to ODC. ............................................................... 26
   1.6 Managing Unused Bindings. ...................................................... 29
       Reconfigure Binding. ............................................................... 29
       Delete Binding. ................................................................. 30
   1.7 Editing an Integration Flow. ....................................................... 30
       Predefined Integration Flows. .................................................. 31
   1.8 Configuring an OData Service. .................................................. 40
   1.9 Deploying an OData Service. .................................................... 41

2. **Supported OData Features.** ......................................................... 42

3. **Creating an OData Service.** ....................................................... 44

4. **Create an OData Model.** ............................................................ 46
   4.1 Import from Data Source. .......................................................... 46
       Importing from SOAP. ............................................................ 47
       Importing from OData. ........................................................... 48
       Importing from ODC. ........................................................... 50
   4.2 Editing an OData Model. ........................................................... 52
       OData Model Editor Features. ................................................ 53

5. **Viewing the OData Model Layout.** ............................................ 57
   5.1 Graphical Model Viewer Features. .......................................... 57

6. **Bind to Data Source.** ............................................................... 59
   6.1 Binding to SOAP. ............................................................... 61
1 Developing an OData Service

The SAP Cloud Platform Integration Web application allows you to develop OData services that expose existing data sources, such as SOAP, as OData endpoints. These OData services can be consumed by SAP Fiori apps, SAP Cloud Platform Mobile Services, or any other custom app, to implement user-centric scenarios.

Differences Between OData Services and Integration Projects

When deciding whether to use OData services or integration projects to implement your business scenario, keep the following differences in mind:

Table 1:

<table>
<thead>
<tr>
<th>OData Services</th>
<th>Integration Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>An OData service can expose multiple entities and multiple associated operations at the same time.</td>
<td>An integration project can expose only one entity and one associated operation at a time.</td>
</tr>
<tr>
<td>You can configure any application to consume an OData service. As a result, any number of users with access to these applications can consume the OData service.</td>
<td>In an integration project, the credentials of a limited set of users are used to authenticate the integration scenario between two systems.</td>
</tr>
</tbody>
</table>

Service Development Process

As a service developer, you can create and deploy an OData service without going through the time-consuming process of writing code. You start off the process by creating an OData service artifact in an integration...
package. For more information, see Creating an OData Service [page 7]. Cloud Integration enables you to further develop the OData service with the following steps:

1. Import an OData model using simple modeling steps. If you prefer writing code, Cloud Integration also lets you handcraft an OData model or edit an existing one using an editor.
2. Connect the OData objects to a data source through integration flows.
3. Tweak the integration flows to enhance business logic.
4. Deploy and monitor the OData service.

- Import from Data Source [page 9]
- Bind to Data Source [page 21]
- Editing an Integration Flow [page 30]
- Deploying an OData Service [page 41]
- Editing an OData Model [page 15]

**Note**

You can develop OData services using the Chrome internet browser only.

**Authorization**

*User Role* is the default authorization used in OData services. Assign the ESBMessaging.send role to users in order for them to access the OData services.

**Related Information**

Supported OData Features [page 6]
1.1 Supported OData Features

The following operations are supported when developing an OData service in the SAP Cloud Platform Integration Web application:

Table 2:

<table>
<thead>
<tr>
<th>OData Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create</td>
<td>Maps to HTTP POST method</td>
</tr>
<tr>
<td>Read</td>
<td>Maps to HTTP GET method</td>
</tr>
<tr>
<td>Update</td>
<td>Maps to HTTP PUT method</td>
</tr>
<tr>
<td>Delete</td>
<td>Maps to HTTP DELETE method</td>
</tr>
<tr>
<td>Query</td>
<td>Maps to HTTP GET method</td>
</tr>
<tr>
<td>Function Imports</td>
<td>Custom operations that can be invoked through HTTP GET or POST method.</td>
</tr>
</tbody>
</table>

Note: Currently, Cloud Integration supports only function imports that can be invoked through HTTP GET method.

The following system query options are supported when calling an OData service developed in the SAP Cloud Platform Integration Web application:

Table 3:

<table>
<thead>
<tr>
<th>OData System Query Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$select</td>
<td>Specifies a subset of properties to return.</td>
</tr>
<tr>
<td>$top</td>
<td>Determines the maximum number of records to return.</td>
</tr>
<tr>
<td>$skip</td>
<td>Sets the number of records to skip before it retrieves records in a collection.</td>
</tr>
<tr>
<td>$filter</td>
<td>Specifies an expression or function that must evaluate to true for a record to be returned in the collection.</td>
</tr>
<tr>
<td>$count</td>
<td>Returns the number of records in a collection.</td>
</tr>
<tr>
<td>$orderby</td>
<td>Determines which values are used to order a collection of records.</td>
</tr>
</tbody>
</table>
### $expand

Specifies that related records must be retrieved in line with the record or collection being retrieved. For example, you can use $expand to retrieve a customer and all orders placed by that customer in a single query.

<table>
<thead>
<tr>
<th>OData System Query Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$expand</td>
<td></td>
</tr>
</tbody>
</table>

**Note**

The system query options $select and $count are supported out of the box. You do not have to write a script to enable functionality for these system query options.

### Related Information

- Developing an OData Service [page 4]

### 1.2 Creating an OData Service

#### Context

You use this procedure to create an integration package that contains an OData service artifact.

#### Procedure

1. Launch the SAP Cloud Platform Integration Web application by entering the URL provided by SAP in your local browser.
2. Enter your credentials and choose **Log On**.
3. Go to the **Design** tab.
4. Choose **Create**.
5. Enter the required data for the integration package.
6. In the **Artifacts** section, choose **Add OData Service**.
7. In the **Add OData Service Artifact** dialog box, enter the OData service details as listed in the following table.
Table 4:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create</td>
<td>Select this radio button if you want to create a new OData service artifact.</td>
</tr>
<tr>
<td>Upload</td>
<td>Select this radio button if you want to upload an OData service artifact from your file location.</td>
</tr>
<tr>
<td>Name</td>
<td>Name of the OData service.</td>
</tr>
<tr>
<td>Namespace</td>
<td>Logical grouping of the OData service. Default is SAP.</td>
</tr>
<tr>
<td>ID</td>
<td>Uniquely identifies the OData service and is a combination of the name and namespace of the OData service in the format &lt;name&gt;_&lt;namespace&gt;_1. This field is not editable.</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the OData service and its purpose.</td>
</tr>
<tr>
<td>OData Service</td>
<td>Select a ZIP file containing the OData service artifact. This field is available and mandatory when you are uploading an OData service artifact.</td>
</tr>
<tr>
<td>OData Version</td>
<td>Currently, OData v2 is the only supported OData version.</td>
</tr>
</tbody>
</table>

**Note**

You can edit some of the OData service details such as Name, Description, and Namespace after the service is created. For more information, see *Configuring an OData Service [page 40]*.  

8. Choose **OK**. The Service Designer page opens.  
   You can further develop the OData service by:  
   1. [Creating an OData model [page 9]]
   2. [Binding to a data source [page 21]]
   3. [Editing an Integration Flow [page 30]]

9. Choose **Save** on the Service Designer page to save the OData service artifact.  
   **Note**
   To retain a copy of the current artifact, choose **Save as version**.  

10. Deploy the OData service. For more information, see *Deploying an OData Service [page 41]*.  
11. Save the integration package.
1.3 Create an OData Model

You can create an OData model in SAP Cloud Platform Integration (Cloud Integration) by:

- Importing from a data source
- Editing a model from the ground up

Cloud Integration gives you the flexibility to create an OData model in an iterative fashion. For example, you can create an OData model by importing the model definition from a SOAP WSDL file. You can add to this OData model by editing it. Or, you can add to the model definition by importing from a different WSDL file or even a different data source, for example, an EDMX file.

Related Information

Import from Data Source [page 9]
Editing an OData Model [page 15]

1.3.1 Import from Data Source

Cloud Integration allows you to create an OData model by importing the model definition from an existing data source, such as SOAP and OData.

**Note**

When an OData service is generated not all attributes are copied from source metadata. However, once the OData service is generated, you can configure these attributes as needed in the OData Model Editor.

An import wizard guides you in this process with the following steps:

1. Select a data source type.
2. Select a structure.
3. Verify and finalize the EDMX structure.

Related Information

Importing from SOAP [page 10]
Importing from OData [page 11]
Importing from ODC [page 13]
1.3.1.1 Importing from SOAP

Context

You have created an OData service artifact in an integration package. You can use this procedure to create an OData model or build up an existing one by importing the model definition from SOAP web services.

Procedure

1. If you are creating a new OData service artifact, proceed to the next step. If you are editing an existing OData service artifact, choose Edit on the Service Designer page.

2. Choose Import Model Wizard.

3. In the Select Data Source Type page, select SOAP as data source type.

4. Choose Browse and select the WSDL file from your file system. You can also select the WSDL from the list of existing WSDL files (if any) used in the OData service artifact.

i Note

Composite WSDLs that are flattened are supported. However, multipart WSDLs and other file types are not supported.

5. Choose Step 2. In the Select Structure page, you can see the elements from the WSDL listed in a table.

6. Select elements that you want to add to the OData model from the WSDL Hierarchy column. You can expand or collapse WSDL elements to view the hierarchy better.

7. Deselect any child element that you do not want to include in the OData model. You can see that the corresponding parent elements are in a mixed state (neither checked nor unchecked), if they have other child elements that are selected.

8. Select the OData type that each WSDL element has to be converted to in the OData model. To see options available for various types of WSDL elements, refer the following table:

Table 5:

<table>
<thead>
<tr>
<th>WSDL Type</th>
<th>OData Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation</td>
<td>Complex Type</td>
</tr>
<tr>
<td></td>
<td>Entity Type</td>
</tr>
<tr>
<td></td>
<td>Function Import</td>
</tr>
<tr>
<td>Complex Type</td>
<td>Complex Type</td>
</tr>
<tr>
<td>WSDL Type</td>
<td>OData Type</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Input Message</td>
<td>Complex Type</td>
</tr>
<tr>
<td>Output Message</td>
<td>Complex Type</td>
</tr>
<tr>
<td>Part</td>
<td>Complex Type</td>
</tr>
</tbody>
</table>

**Note**

You can select Flatten from the Import As dropdown for all WSDL types other than Operation. Flatten is not an OData type that a WSDL element can be converted to. It is an instruction to the system to skip importing the specified WSDL element into the OData model without affecting the import of its children in the hierarchy.

9. Choose Step 3. You can review the mapping of WSDL elements to OData objects in this step.
10. In the Review and Finish EDMX Structure page, select a primary key from the properties of entity types.
11. Verify and perform the following additional changes, if required, to the OData model definition:
   - Rename the OData objects listed under EDM Name.
   - Select a different EDM type.
   - Update the Documentation column.
   - Restore default definition of the OData model by choosing Restore Default Values.
12. Once you have completed all the changes, choose Finish. The Service Designer page displays the entity sets and function imports from the OData model in a table.
13. Choose Save to save your changes to the OData service artifact.

**Note**

To retain a copy of the current artifact, choose Save as version.

### 1.3.1.2 Importing from OData

**Context**

You have created an OData service artifact in an integration package. You can use this procedure to create an OData model or build up an existing one by importing the model definition from an existing OData service.
Procedure

1. If you are creating a new OData service artifact, proceed to the next step. If you are editing an existing OData service artifact, choose Edit on the Service Designer page.

2. Choose Import Model Wizard.

3. In the Select Data Source Type page, select ODATA as data source type.

4. Choose Browse and select the EDMX file from your file system. You can also select the EDMX from the list of existing EDMX files (if any) used in the OData service artifact. metadata.edmx is the file used internally in Cloud Integration Web application to store the OData model definition of the OData service artifact.

   Note
   The OData Model Editor supports only OData SAP V2 annotations.

5. Choose Step 2. In the Select Structure page, you can see entity types and function imports from the EDMX listed in a table. The table also displays the EDM type, primary key and documentation information associated with the OData objects.

6. Select OData objects that you need to add to the model from the OData Hierarchy column. You can expand or collapse OData objects to view its properties better.

7. Choose Step 3.

8. In the Review and Finish EDMX Structure page, verify and perform the following additional changes, if required, to the OData model definition:
   ○ Rename the OData objects listed under EDM Name.
   ○ Select another primary key.
   ○ Select a different EDM type.
   ○ Update the Documentation column.
   ○ Restore default definition of the OData model by choosing Restore Default Values.

9. Once you have completed all the changes, choose Finish. The Service Designer page displays the entity sets and function imports from the OData model in a table.

10. Choose Save to save your changes to the OData service artifact.

   Note
   To retain a copy of the current artifact, choose Save as version.
1.3.1.3 Importing from ODC

Prerequisites

You have created an OData service artifact in an integration package.

Context

SAP Cloud Platform Integration helps you import a model definition from an OData service, which is created on an on-premise SAP Gateway system. SAP Gateway has a component called Backend Provisioning Component or IW_BEP. This component resides on SAP Business Suite and is used to build OData services from data sources like RFCs, BAPIs, and so on. SAP Cloud Platform Integration communicates with the IW_BEP component in SAP Gateway through OData Channel or ODC protocol, which is an SAP proprietary protocol.

Note

SAP Cloud Platform Integration does not support connecting to an SAP Gateway system based on SAP NetWeaver 7.02 where the IW_BEP component version is SP 06 or lower.

SAP Cloud Platform Integration also does not support connecting to an SAP Gateway system based on SAP NetWeaver 7.40 where the SAP_GWFND component is SP 02 or lower.

You can use this procedure to create an OData model or build up an existing one by importing the model definition from an OData service created in IW_BEP (ODC).

Procedure

1. If you are creating a new OData service artifact, proceed to the next step. If you are editing an existing OData service artifact, choose Edit on the Service Designer page.

2. Choose Import Model Wizard.

3. In the Select Data Source Type page, select ODC as data source type.

4. Enter the connection details as listed in the following table.

<table>
<thead>
<tr>
<th>Connection Details</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>Endpoint URI of the target system</td>
</tr>
<tr>
<td>Client</td>
<td>Client of the target system</td>
</tr>
</tbody>
</table>
### Connection Details

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Namespace</td>
</tr>
<tr>
<td>Service Name</td>
</tr>
<tr>
<td>Version</td>
</tr>
<tr>
<td>Proxy Type</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Authentication</td>
</tr>
<tr>
<td>Credential Name</td>
</tr>
</tbody>
</table>

5. Choose **Step 2**.

In the **Select Structure** page, you can see entity types and function imports from the EDMX listed in a table. The table also displays the associated EDM type, primary key, and documentation information.

6. Select entity types and function imports that you need to add to the model from the **OData Hierarchy** column. You can expand or collapse entity types and function imports to view its properties better.

7. Choose **Step 3**.

8. In the **Review and Finish EDMX Structure** page, verify and perform the following additional changes, if required, to the OData model definition:
   ○ Rename the OData objects listed under **EDM Name**.
   ○ Select a different EDM type.
   ○ Select another primary key.
   ○ Update the **Documentation** column.
   ○ Restore default definition of the OData model by choosing **Restore Default Values**.

9. Once you have completed all the changes, choose **Finish**. The Service Designer page displays the entity sets and function imports from the OData model in a table.

10. Choose **Save** to save your changes to the OData service artifact.

**Note**

To retain a copy of the current artifact, choose **Save as version**.
1.3.2 Editing an OData Model

Context

You have created an integration package and added an OData service artifact to the package.

You can use this procedure to create an OData model from the ground up. You can also use it to edit an existing OData model.

Procedure

1. If you are creating a new OData service artifact, proceed to the next step. If you are editing an existing OData service artifact, choose Edit on the Service Designer page.

2. Choose OData Model Editor on the Service Designer page.

   If you are creating an OData model from the ground up, the OData Model Editor opens a predefined structure based on the selected OData version. The root is a Schema element. Below the root, subelements such as EntityType, ComplexType, Association, EntityContainer, and so on, can be added if required.

   **Note**

   In the case of an existing OData service artifact, you can only view the OData model in the OData Model Editor unless you explicitly choose Edit on the Service Designer page.

   For more information on the features of the OData Model Editor, see OData Model Editor Features [page 16].

3. Once you have finished editing the OData model, choose OK.

4. Choose Back to get back to the Service Designer page.

5. Choose Save to save your changes to the OData service artifact.

   **Note**

   To retain a copy of the current artifact, choose Save as version.
1.3.2.1 OData Model Editor Features

The OData Model Editor in the SAP Cloud Platform Integration Web application enables you to write an Entity Data Model XML (EDMX). This model serves as metadata for your OData service. OData Model Editor provides you with the following basic features:

- **Code Assist**: Provides you with code suggestions that help speed up your development time. There are two types of code assist:
  - Schema-based
  - Model-based
- **Autocomplete**: Predicts the code element you are entering based on the pattern of letters entered.

**Note**

You can use the OData Model Editor to edit and fine-tune your OData model, after you have imported it from a data source. However, we recommend that you keep the overall end-to-end scenario in mind when you edit the model using this tool. The OData Model Editor can only provide semantic validations. It cannot validate the correctness of the OData model in the context of the OData provisioning scenario that you are working in.

**Related Information**

- Using Schema-Based Code Assist [page 16]
- Using Model-Based Code Assist [page 18]
- Using Autocomplete [page 18]
- Validation in the OData Model Editor [page 19]

1.3.2.1.1 Using Schema-Based Code Assist

**Context**

The schema-based code assist feature provides you with code suggestions that adhere to the Conceptual Schema Definition Language (CSDL) specification. These code suggestions are provided based on the current cursor position.

Follow this procedure to use schema-based code assist in the OData Model Editor.
Procedure

1. Choose OData Model Editor on the Service Designer page. The OData Model Editor page opens.
2. Place your cursor in the context where you require assistance, and press Ctrl + Spacebar.

Based on the cursor location, a list of the elements that can be used appears.

Each element in the dropdown list is represented by an icon. The following image shows the list of elements and their corresponding icons:

<table>
<thead>
<tr>
<th>Icons</th>
<th>Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>📊</td>
<td>Property Ref, Return Type, On Delete, Association Set</td>
</tr>
<tr>
<td>🔗</td>
<td>Association</td>
</tr>
<tr>
<td>📦</td>
<td>Complex Type</td>
</tr>
<tr>
<td>🇺🇸</td>
<td>Entity Container</td>
</tr>
<tr>
<td>🇺🇸</td>
<td>Entity Type</td>
</tr>
<tr>
<td>🇺🇸</td>
<td>Entity Set</td>
</tr>
<tr>
<td>🇺🇸</td>
<td>Function</td>
</tr>
<tr>
<td>🇺🇸</td>
<td>Using</td>
</tr>
<tr>
<td>🇺🇸</td>
<td>Documentation</td>
</tr>
<tr>
<td>❌</td>
<td>End</td>
</tr>
<tr>
<td>🔒</td>
<td>Referential Constraint</td>
</tr>
<tr>
<td>🌐</td>
<td>Parameter</td>
</tr>
<tr>
<td>🌐</td>
<td>Navigation Property</td>
</tr>
<tr>
<td>🔒</td>
<td>Key</td>
</tr>
<tr>
<td>🌐</td>
<td>Function Import</td>
</tr>
<tr>
<td>🌐</td>
<td>Property</td>
</tr>
<tr>
<td>🌐</td>
<td>Dependent</td>
</tr>
<tr>
<td>🌐</td>
<td>Default image</td>
</tr>
</tbody>
</table>

3. Scroll through the list, select the required element, and press Enter. The OData Model Editor inserts a full snippet of code in the desired location.
### 1.3.2.1.2 Using Model-Based Code Assist

**Context**

The model-based code assist feature provides code suggestions based on the OData model you are working on.

Follow this procedure to use model-based code assist in the OData Model Editor.

**Procedure**

1. Choose **OData Model Editor** on the Service Designer page. The OData Model Editor page opens.
2. Place the cursor between the quotes of the attribute of an element such as PropertyRef, EntitySet, or AssociationSet and press Ctrl + Spacebar to see the relevant suggestions.
3. Scroll through the list and select the required suggestion.

### 1.3.2.1.3 Using Autocomplete

**Context**

The autocomplete feature predicts the word you are entering based on the pattern of letters entered and provides you with suggestions. The suggestions are context-sensitive. You can get suggestions for elements, attributes, attribute types, properties, and property values.

Follow this procedure to use the autocomplete feature in the OData Model Editor.

**Procedure**

1. Choose **OData Model Editor** on the Service Designer page. The OData Model Editor page opens.
2. Place the cursor in the context where you need assistance and start entering letters. Autocomplete suggestions appear next to the letters in context.
3. Use the Up arrow and Down arrow keys on the keyboard to select the required suggestions and press Enter.
1.3.2.1.4 Validation in the OData Model Editor

Validation in the OData Model Editor ensures that your OData model adheres to the CSDL schema and does not deviate from logical or semantic specifications.

After validation, lines of code with errors are highlighted with a red icon in the code editor.

Let us look at an example of how validation works.

Example

In any OData model, remove the value corresponding to the Name attribute in an EntityContainer element.

Since Name is a mandatory attribute, the corresponding line of code is highlighted with a red icon in the code editor. You can see the error message in a tooltip when you hover the mouse over the red icon.

1.4 Viewing the OData Model Layout

Context

You have created an OData model either by importing from a data source or by editing the model manually using the SAP Cloud Platform Integration Web application.

You can use this procedure to view the overall layout of an OData model in the Graphical Model Viewer. This helps you verify the OData model and determine if any additional changes are necessary.

Procedure

1. Choose Graphical Model Viewer on the Service Designer page. The OData model is rendered in a graphical format.

   You can use the navigation control and viewing options to customize your viewing preferences. For more information, see Graphical Model Viewer Features [page 20].
2. Choose Back ( ) to get back to the Service Designer page.

1.4.1 Graphical Model Viewer Features

The Graphical Model Viewer in the SAP Cloud Platform Integration Web application provides you with the following navigation controls and viewing options to customize your viewing preferences:

**Fit to Screen**

There might be instances where the OData model objects of a big model extend beyond the scope of the Graphical Model Viewer space. In such cases, you can choose *Fit to Screen* to view the entire model within the boundaries of the Graphical Model Viewer. This feature automatically aligns the model to fit the Graphical Model Viewer space.

**Expanding and Collapsing OData Objects**

The Graphical Model Viewer lets you expand or collapse all the section headers of the objects in the OData model. You can also expand and collapse individual OData model objects.

**Collapsed State**

- To collapse all OData objects in the Graphical Model Viewer, choose the *Collapse All* button on the toolbar. The *Collapse All* button toggles to *Expand All* once the model is collapsed.
- If you want to collapse an individual OData object, choose the arrowhead button in the expanded object.

**Expanded State**

- To expand all OData objects in the Graphical Model Viewer, choose the *Expand All* button on the toolbar. The *Expand All* button toggles to *Collapse All* once the model is expanded.
- If you want to expand an individual OData object, choose the arrowhead button in the collapsed object.

**Navigation Controls**

Depending on the number of OData objects in the model, objects might appear too small or be too big for the Graphical Model Viewer to accommodate. In such cases, the navigation controls help you view the model better.
Table 7:

<table>
<thead>
<tr>
<th>Button</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Button" /></td>
<td>Use the compass to control the perspective of the model:</td>
</tr>
<tr>
<td><img src="image" alt="Button" /></td>
<td>- Choose to move right.</td>
</tr>
<tr>
<td><img src="image" alt="Button" /></td>
<td>- Choose to move left.</td>
</tr>
<tr>
<td><img src="image" alt="Button" /></td>
<td>- Choose to move down.</td>
</tr>
<tr>
<td><img src="image" alt="Button" /></td>
<td>- Choose to move up.</td>
</tr>
<tr>
<td><img src="image" alt="Button" /></td>
<td>- Choose to zoom the model to 100%.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Button</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Button" /></td>
<td>- Choose to zoom in to the model.</td>
</tr>
<tr>
<td><img src="image" alt="Button" /></td>
<td>- Choose to zoom out of the model.</td>
</tr>
<tr>
<td><img src="image" alt="Button" /></td>
<td>- The zoom level indicator shows the zoom percentage.</td>
</tr>
</tbody>
</table>

1.5 **Bind to Data Source**

You have created an OData model in the SAP Cloud Platform Integration Web application. As a result, you can see OData objects associated with the model listed in a table on the Service Designer page.

Next, you need to bind or connect function imports and operations of entity sets to a data source. When you bind an OData object to a data source, a predefined integration flow is autogenerated and set up for you. You can modify this integration flow to suit your scenario.

**Note**

The externalise option is not visible for these autogenerated integration flows.

You can bind one or all five CRUDQ operations of an entity set to a data source. This means that each entity set can have multiple integration flows connecting to the same data source. You can also bind entity sets and function imports in an OData service artifact to different data sources.

The following data sources are currently supported:

- SOAP
- OData
- REST
- ODC
Binding Status

To identify the graphical representations of the OData objects and their binding status on the Service Designer page, refer to the following table:

Table 8:

<table>
<thead>
<tr>
<th>OData Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image.png" alt="Entity Set" /></td>
<td>Represents an entity set in an OData model.</td>
</tr>
<tr>
<td><img src="image.png" alt="Operation" /></td>
<td>Represents an operation associated with an entity set in an OData model.</td>
</tr>
<tr>
<td><img src="image.png" alt="Operation with Errors" /></td>
<td>Represents an operation bound to a data source with errors in the associated integration flow. You cannot deploy the OData service unless the errors are corrected.</td>
</tr>
<tr>
<td><img src="image.png" alt="Operation with Warnings" /></td>
<td>Represents an operation bound to a data source with warnings in the associated integration flow. We recommend that you verify the warnings and take corrective measures where necessary before deploying the OData service.</td>
</tr>
<tr>
<td><img src="image.png" alt="Operation with No Errors or Warnings" /></td>
<td>Represents an operation bound to a data source with no errors or warnings.</td>
</tr>
<tr>
<td><img src="image.png" alt="Function Import" /></td>
<td>Represents a function import in an OData model.</td>
</tr>
<tr>
<td><img src="image.png" alt="Function Import with Errors" /></td>
<td>Represents a function import bound to a data source with errors in the associated integration flow. You cannot deploy the OData service unless the errors are corrected.</td>
</tr>
<tr>
<td><img src="image.png" alt="Function Import with Warnings" /></td>
<td>Represents a function import bound to a data source with warnings in the associated integration flow. We recommend that you verify the warnings and take corrective measures where necessary before deploying the OData service.</td>
</tr>
<tr>
<td><img src="image.png" alt="Function Import with No Errors or Warnings" /></td>
<td>Represents a function import bound to a data source with no errors or warnings.</td>
</tr>
</tbody>
</table>

Unused Bindings

After binding an entity set or function import to a data source, if you rename or delete it in the OData Model Editor, the binding is no longer associated with it and becomes unused. You cannot deploy the OData service unless you update the binding. For more information, see Managing Unused Bindings [page 29].
1.5.1 Binding to SOAP

Prerequisites

You are editing an OData service artifact and you have created an OData model. OData objects in the model are listed in a table on the Service Designer page.

Context

You can use this procedure to bind function imports and operations of entity sets to the corresponding SOAP Web service operations.

Procedure

1. Select SOAP as the data source for an entity set or function import.

2. Choose Bind (определен) for the operation associated with the entity set or function import that you need to bind. The Configure SOAP Data Source dialog box appears.

3. Choose Browse and select the WSDL file from your file location. If the OData model was imported from a WSDL file, the corresponding file name appears by default. You can change this to any other file name as required.

   **Note**

   Composite WSDLs that are flattened are supported. However, multipart WSDLs and other file types are not supported.

4. Select an operation. If the OData model was imported from a WSDL file, operations defined in the WSDL appear in the dropdown list.

5. Enter the Endpoint URI where the operation is located. If the OData model was imported from a WSDL file and an operation was selected from this WSDL, the corresponding Endpoint URI is auto-populated.
6. Choose OK. The Service Designer page appears and the Action column for this operation now displays

*Navigate to Integration Flow Editor* ( ) and Delete Binding ( ) buttons.

A predefined integration flow with mapping steps is generated for the operation. For more information, see Predefined Integration Flows for SOAP [page 32].

You can update the integration flow to better suit your business scenario by choosing Navigate to Integration Flow Editor ( ). For more information, see Editing an Integration Flow [page 30].

In case you need to update the binding information, you can do it in one of the following ways:

○ Choose Navigate to Integration Flow Editor ( ) and update the Endpoint URI information in the receiver channel.

○ Choose Delete Binding ( ) and repeat the binding procedure.

7. Choose Save to save your changes to the OData service artifact.

i Note
To retain a copy of the current artifact, choose Save as version.

### 1.5.2 Binding to OData

#### Prerequisites

You are editing an OData service artifact and you have created an OData model. OData objects in the model are listed in a table on the Service Designer page.

#### Context

You can use this procedure to bind function imports and operations of entity sets to the corresponding OData services.

#### Procedure

1. Select ODATA as the data source for an entity set or function import.

2. Choose Bind ( ) for the function import or operation associated with the entity set that you need to bind. The Configure OData Data Source dialog box appears.
3. Choose Browse and select the EDMX file from your file location. You can also select the file from the Existing Files list. The Existing Files list is populated with EDMX file names used in other bindings in the same OData service artifact.

4. Select an entity set from the Entity Sets dropdown list. This list displays all entity sets from the selected EDMX file.

5. Enter the Endpoint URI corresponding to the OData service that you are binding to.

6. Choose OK. The Service Designer page appears and the Action column for this operation now displays Navigate to Integration Flow Editor and Delete Binding buttons.

A predefined integration flow with mapping steps is generated for the operation or function import. For more information, see Predefined Integration Flows for OData [page 35].

You can update the integration flow to better suit your business scenario by choosing Navigate to Integration Flow Editor. For more information, see Editing an Integration Flow [page 30].

In case you need to update the binding information, you can do it in one of the following ways:

○ Choose Navigate to Integration Flow Editor and update the Endpoint URI information in the receiver channel.

○ Choose Delete Binding and repeat the binding procedure.

7. Choose Save to save your changes to the OData service artifact.

<table>
<thead>
<tr>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>To retain a copy of the current artifact, choose Save as version.</td>
</tr>
</tbody>
</table>

### 1.5.3 Binding to REST

#### Prerequisites

You are editing an OData service artifact and you have created an OData model. OData objects in the model are listed in a table on the Service Designer page.

#### Context

You can use this procedure to bind function imports and operations of entity sets to the corresponding REST services.
Procedure

1. Select REST as the data source for an entity set or function import.

2. Choose Bind (🔗) for the function import or operation associated with the entity set that you need to bind. The Configure REST Data Source dialog box appears.

3. Enter the Endpoint URI corresponding to the REST service that you are binding to.

4. Choose OK. The Service Designer page appears and the Action column for this operation or function import now displays Navigate to Integration Flow Editor (🔗) and Delete Binding (🗑️) buttons.

A predefined integration flow with mapping steps is generated for the operation or function import. For more information, see Predefined Integration Flows for REST [page 37].

You can update the integration flow to better suit your business scenario by choosing Navigate to Integration Flow Editor (🔗). For more information, see Editing an Integration Flow [page 30].

In case you need to update the binding information, you can do it in one of the following ways:

○ Choose Navigate to Integration Flow Editor (🔗) and update the Endpoint URI information in the receiver channel.

○ Choose Delete Binding (🗑️) and repeat the binding procedure.

5. Choose Save to save your changes to the OData service artifact.

<table>
<thead>
<tr>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>To retain a copy of the current artifact, choose Save as version.</td>
</tr>
</tbody>
</table>

1.5.4 Binding to ODC

Prerequisites

You are editing an OData service artifact and you have created an OData model. OData objects in the model are listed in a table on the Service Designer page.

Context

SAP Cloud Platform Integration helps you bind or connect to an OData service, which is created on an on-premise SAP Gateway system. SAP Gateway has a component called Backend Provisioning Component or IW_BEP. This component resides on SAP Business Suite and is used to build OData services from data sources.
like RFCs, BAPIs, and so on. Cloud Integration communicates with the IW_BEP component in SAP Gateway through OData Channel or ODC protocol, which is an SAP proprietary protocol.

**Note**

Cloud Integration does not support connecting to an SAP Gateway system based on SAP NetWeaver 7.02 where the IW_BEP component version is SP 06 or lower.

Cloud Integration also does not support connecting to an SAP Gateway system based on SAP NetWeaver 7.40 where the SAP_GWFND component is SP 02 or lower.

You can use this procedure to bind function imports and operations of entity sets to the corresponding OData services created in IW_BEP (ODC).

**Procedure**

1. Select **ODC** as the data source for an entity set or function import.

2. Choose **Bind** (🔗) for the function import or operation associated with the entity set that you want to bind.

   The **Configure ODC Data Source** dialog box appears. The entity set and CRUDQ operation information is prepopulated.

3. Enter the following information related to the ODC destination.

   **Table 9:**

<table>
<thead>
<tr>
<th>Connection Details</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Sets</td>
<td>A list of entity sets that is populated from the ODC source when importing the OData model. If you have coded the OData model, then the list is populated directly from your model. Select an entity set that you want to bind to.</td>
</tr>
<tr>
<td>Address</td>
<td>Endpoint URI of the target system</td>
</tr>
<tr>
<td>Client</td>
<td>Client of the target system</td>
</tr>
<tr>
<td>Namespace</td>
<td>Logical grouping of the service implementation</td>
</tr>
<tr>
<td>Service Name</td>
<td>Name of the service implementation</td>
</tr>
<tr>
<td>Version</td>
<td>Version of the service implementation</td>
</tr>
</tbody>
</table>
### Connection Details

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
</table>
| **Proxy Type** | The type of proxy that you are using to connect to the target system:  
- Select **On-Premise** if you are connecting to an on-premise system using the SAP Cloud Connector. This option is only available if you have chosen *HANA Cloud Integration* as the product profile. If you select this option, the **Address** field of the adapter refers to a virtual address, which has to be configured in the SAP Cloud Connector settings.  
| **Authentication** | Select **Basic** if you want to use user credentials to connect to the target system.  
| **Credential Name** | Name of the user credentials artifact deployed on the tenant |

4. Choose **OK**. The Service Designer page appears and the **Action** column for this operation now displays **Navigate to Integration Flow Editor** and **Delete Binding** buttons.

A predefined integration flow with mapping steps is generated for the operation or function import. For more information, see [Predefined Integration Flows for ODC](#page-37).

You can update the integration flow to better suit your business scenario by choosing **Navigate to Integration Flow Editor**. For more information, see [Editing an Integration Flow](#page-30).

If you want to update the binding information, you can do it in one of the following ways:

- Choose **Navigate to Integration Flow Editor** and update the Endpoint URI information in the receiver channel.
- Choose **Delete Binding** and repeat the binding procedure.

5. Choose **Save** to save your changes to the OData service artifact.

**Note**

To retain a copy of the current artifact, choose **Save as version**.
1.6 Managing Unused Bindings

Context

You are editing an OData service artifact and you have bound a function import or operations of an entity set to a data source. At this point, let us assume you changed the name of the function import or entity set, or even deleted it in the OData Model Editor. The binding associated with the renamed or deleted entity set or function import is now unused.

The unused bindings appear in a separate table on the Service Designer page. You cannot deploy the OData service unless you update the unused bindings.

SAP Cloud Platform Integration Web application helps you manage unused bindings by providing you with the following options:

Reconfigure Binding

Context

You can use this procedure to reconfigure the binding from the earlier entity set or function import to one in the updated OData model.

Procedure

1. Choose Bind for the entity set or function import associated with the unused binding. The Reconfigure Binding dialog box appears.
2. Select the entity set or function import in the updated OData model to which the unused binding needs to be associated.
3. Choose OK. The selected entity or function import in the updated OData model is now associated with the binding. The Unused Bindings table disappears if there are no other unused bindings.

SAP Cloud Platform Integration updates the sender channel with the new binding information. However, you have to update other integration flow components that references the earlier entity set or function import, such as mapping and scripts.
4. Choose Save to save your changes to the OData service artifact.

Note
To retain a copy of the current artifact, choose Save as version.
Delete Binding

Context

You can use this procedure to delete the unused binding so that it is no longer associated with any entity set or function import in the OData model.

Procedure

1. Choose Delete Binding ( ) for the associated entity set or function import.
2. Confirm the deletion. The Unused Bindings table disappears if there are no other unused bindings.
3. Choose Save to save your changes to the OData service artifact.

Note

To retain a copy of the current artifact, choose Save as version.

1.7 Editing an Integration Flow

Context

You are editing an OData service artifact and you have created an OData model. You have provided binding information for at least one OData object in the model.

Once a function import or operation of an entity set is bound to a data source, a predefined integration flow is set up for you. Within this integration flow, the OData adapter is selected as the default for the sender channel. The data source to which the operation is bound is selected as the default adapter for the receiver channel.

You can use this procedure to update the predefined integration flow to suit your business scenario.

Procedure

1. Choose Navigate to Integration Flow Editor ( ) for an operation or function import that is bound to a data source. The associated integration flow opens in the Integration Flow Editor.
2. Edit the integration flow according to your requirements.
Note

The following points need to be kept in mind when editing integration flows in an OData service artifact:
○ The OData sender channel is not editable.
○ The OData sender channel, participant and start event cannot be deleted.
○ A new sender participant cannot be dragged and dropped into the integration flow.
○ If you modify the default name of the receiver adapter, you need to make sure it is unique across all integration flows in an OData service artifact.
○ You need to ensure that the Namespace Mapping entered in the runtime configuration, is consistent across all integration flows in an OData service artifact.


For more information about configuring integration flow components, see .

3. Once you have finished editing the integration flow, choose **OK**.

4. Choose **Back** (←) to get back to the Service Designer page.

5. Choose **Save** to save your changes to the OData service artifact.

Note

To retain a copy of the current artifact, choose **Save as version**.

Related Information

Predefined Integration Flows [page 31]

1.7.1 Predefined Integration Flows

The SAP Cloud Platform Integration Web application sets up predefined integration flows when OData objects are bound to a data source. These integration flows have the sender channel preconfigured to the OData adapter type.

A predefined integration flow has the minimum setup required for an integration scenario. You have to configure this integration flow to suit the specific requirements of your scenario.
1.7.1.1 Predefined Integration Flows for SOAP

The SAP Cloud Platform Integration Web application sets up predefined integration flows when the CRUDQ (Create, Read, Update, Delete, and Query) operations of entity sets and function imports are bound to a SOAP Web service.

Related Information

Predefined Integration Flows for Create and Read Operations [page 32]
Predefined Integration Flows for Update and Delete Operations [page 33]
Predefined Integration Flow for Query Operations [page 34]
Predefined Integration Flow for Function Imports [page 35]

1.7.1.1.1 Predefined Integration Flows for Create and Read Operations

This integration flow is autogenerated for Create and Read operations. In order to complete the binding to the data source, you have to edit at least the following elements:

- Request Mapping
- Response Mapping

To understand this integration flow, let’s look at an example for the Create operation.
Example

You open an app on your smartphone to book a hotel room in the city of New York. This application sends the booking information to a hotel booking system. The hotel booking system receives this data with the help of a SOAP Web service, stores it, and sends the booking status back to you.

The mobile application communicates with the hotel booking system by calling a URI to create a booking record and passes the booking information as the payload. When the OData sender receives the request, it passes it to the Request Mapping element. This mapping element contains information that maps the fields from the OData structure to the SOAP structure and passes the create request to the SOAP receiver.

The SOAP receiver processes the request and creates a booking record in the back end. If the operation is successful, it returns the status to the Response Mapping element. The Response Mapping element contains information that maps the fields from the SOAP structure to the OData structure. It passes the booking status and other mapped details such as the confirmation number to the OData sender. OData sender in turn passes the response to the mobile application.

1.7.1.1.2 Predefined Integration Flows for Update and Delete Operations

This integration flow is autogenerated for Update and Delete operations. In order to complete the binding to the data source, you have to edit at least the Request Mapping element.

To understand this integration flow, let’s look at an example of an Update operation.

Example

You open an app on your smartphone to extend your stay in a hotel room in the city of New York. This application sends the booking information to a hotel booking system. The hotel booking system receives this data with the help of a SOAP Web service and updates it.

The mobile application communicates with the hotel booking system by calling a URI to update a booking record and passes the booking information as the payload. When the OData sender receives the request, it passes it to the Request Mapping element. This mapping element contains information that maps the fields from the OData structure to the SOAP structure and passes the update request to the SOAP receiver.
The SOAP receiver processes the request and updates the booking record in the back-end. The status of the update operation is then sent to the OData sender. The OData sender in turn sends it to the mobile application.

1.7.1.1.3 Predefined Integration Flow for Query Operations

This integration flow is autogenerated for Query operations. In order to complete the binding to the data source, you have to edit at least the following elements:

- Content Modifier
- Response Mapping

**Note**

If you want to enable system query options such as $top, $skip, and $filter, you can replace the Content Modifier element with a Script element.

To understand this integration flow, let's look at an example.

**Example**

A mobile application needs to get a list of hotels available in the city of New York from a hotel booking system. The hotel booking system publishes this data in the form of a SOAP Web service. The mobile application places this request in the form of a URI, where the city of New York is passed as a parameter.

When the OData sender channel receives the URI request, it passes it to the Content Modifier element. The Content Modifier element adds additional information to the head and body of the incoming message so that the SOAP receiver can understand it.

The SOAP receiver processes the request and, if there are no issues with the request, returns a list of hotels as the response. The Response Mapping element contains information that maps the fields from the SOAP structure to the OData structure. The mapped data is then sent to the mobile application through the OData sender channel.
1.7.1.4 Predefined Integration Flow for Function Imports

This integration flow is autogenerated for function imports. In order to complete the binding to the SOAP Web service, you have to edit at least the Script elements for request and response mapping.

To understand this integration flow, let’s look at an example.

**Example**

You open an app on your smartphone to check the most frequently visited hotel in the city of New York and the number of guests that visited this hotel in the previous year. The hotel booking system publishes this data in the form of a SOAP Web service. The mobile application places this request in the form of a URI, where the city of New York is passed as a parameter.

When the OData sender channel receives the URI request, it passes it to the Script element. The Script element maps the request so that the SOAP receiver understands it.

The SOAP receiver processes the request and, if there are no issues with the request, returns the most frequently visited hotel and the guest count as the response. The Script element contains information that maps the fields from the SOAP structure to the return type defined in the OData model. The mapped data is then sent to the mobile application through the OData sender channel.

1.7.1.2 Predefined Integration Flows for OData

The SAP Cloud Platform Integration Web application sets up predefined integration flows when the Create, Read, Update, and Query operations of entity sets are bound to an OData service.

**Note**

Function imports are not available in the OData receiver channel. We recommend that you use another adapter like HTTP for it.
Predefined Integration Flows for Query and Read Operations

This integration flow is autogenerated for Query and Read operations. Script step in this integration flow takes the query options from the incoming URI and sets it into an exchange header `<odataURI>`. In the case of Query operations, you can use this exchange header to dynamically pass the query options to the OData receiver channel. And in the case of Read operations, you can use this exchange header to dynamically pass the key fields of the entity set to the OData receiver channel.

Predefined Integration Flows for Create and Update Operations

This integration flow is autogenerated for Create and Update operations. Script step in this integration flow takes the query options from the incoming URI and sets it into an exchange header `<odataURI>`. In the case of Update operations, you can use this exchange header to dynamically pass the key fields of the entity set to the OData receiver channel.
1.7.1.3 Predefined Integration Flows for REST

The SAP Cloud Platform Integration Web application sets up predefined integration flows when the CRUDQ (Create, Read, Update, Delete, and Query) operations of entity sets and function imports are bound to a REST web service.

This integration flow is autogenerated for all CRUDQ operations when the receiver is a REST service. You can edit the Script elements in order to define the request and response mappings.

1.7.1.4 Predefined Integration Flows for ODC

The SAP Cloud Platform Integration Web application sets up predefined integration flows when the CRUDQ (Create, Read, Update, Delete, and Query) operations of entity sets and function imports are bound to an ODC service.

Predefined Integration Flow for Query and Read Operations
This integration flow is autogenerated for Query and Read operations. Script step in this integration flow takes the query options and key fields of entity sets from the incoming URI and sets it into an exchange header `<odataURI>`.

**Note**
Currently, the ODC receiver channel does not support any query options or key fields of entity sets that are dynamically passed using the exchange header.

You can map the fields from the ODC structure to the OData structure in the Response Mapping element.

**Predefined Integration Flow for Create Operation**

This integration flow is autogenerated for Create operation. You can map the fields from the OData structure to the ODC structure in the Request Mapping element. Similarly, map the fields from the ODC structure to the OData structure in the Response Mapping element.
Predefined Integration Flow for Update Operation

This integration flow is autogenerated for Update operation. Script step in this integration flow takes the key fields of entity sets from the incoming URI and sets it into an exchange header <odataURI>.

**Note**
Currently, the ODC receiver channel does not support any key fields of entity sets that are dynamically passed using the exchange header.

You can map the fields from the OData structure to the ODC structure in the Request Mapping element.

Predefined Integration Flow for Delete Operation

This integration flow is autogenerated for Delete operation. Script step in this integration flow takes the key fields of entity sets from the incoming URI and sets it into an exchange header <odataURI>.

**Note**
Currently, the ODC receiver channel does not support any key fields of entity sets that are dynamically passed using the exchange header.
1.8 Configuring an OData Service

Context

You have created an OData service artifact.

You can use this procedure to modify the parameters that you entered when creating the artifact. These parameters identify the OData service.

Procedure

1. If the OData service is newly created, proceed directly to the next step. If it is an existing one, choose Edit on the Service Designer page.

2. Choose Service Configuration (⚙️) on the Service Designer page.

3. Edit or modify the parameters listed in the following table, as required.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the OData service.</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the OData service and its purpose.</td>
</tr>
<tr>
<td>Namespace</td>
<td>Logical grouping of the OData service.</td>
</tr>
</tbody>
</table>

4. Choose OK.

5. Choose Save on the Service Designer page to save your changes to the OData service artifact.

Note

To retain a copy of the current artifact, choose Save as version.
1.9 Deploying an OData Service

Context

You have created an OData service artifact with a valid OData model. You have provided binding information for at least one OData object in the model. You have verified that all integration flows are valid.

You can use this procedure to deploy the OData service.

Procedure

2. Confirm that you do want to deploy. If there are no validation errors, a message is displayed confirming that the service is triggered for deployment.
3. Choose OK. The OData service is now deployed.

The OData service starts on the tenant, after deployment. You can confirm that the status of the OData service is Started in the Integration Content Monitor. For more information, see .

It may take a few moments for the OData service to get active. Once it is active, it is ready for consumption.
2 Supported OData Features

The following operations are supported when developing an OData service in the SAP Cloud Platform Integration Web application:

Table 11:

<table>
<thead>
<tr>
<th>OData Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create</td>
<td>Maps to HTTP POST method</td>
</tr>
<tr>
<td>Read</td>
<td>Maps to HTTP GET method</td>
</tr>
<tr>
<td>Update</td>
<td>Maps to HTTP PUT method</td>
</tr>
<tr>
<td>Delete</td>
<td>Maps to HTTP DELETE method</td>
</tr>
<tr>
<td>Query</td>
<td>Maps to HTTP GET method</td>
</tr>
<tr>
<td>Function Imports</td>
<td>Custom operations that can be invoked through HTTP GET or POST method.</td>
</tr>
</tbody>
</table>

Note: Currently, Cloud Integration supports only function imports that can be invoked through HTTP GET method.

The following system query options are supported when calling an OData service developed in the SAP Cloud Platform Integration Web application:

Table 12:

<table>
<thead>
<tr>
<th>OData System Query Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$select</td>
<td>Specifies a subset of properties to return.</td>
</tr>
<tr>
<td>$top</td>
<td>Determines the maximum number of records to return.</td>
</tr>
<tr>
<td>$skip</td>
<td>Sets the number of records to skip before it retrieves records in a collection.</td>
</tr>
<tr>
<td>$filter</td>
<td>Specifies an expression or function that must evaluate to true for a record to be returned in the collection.</td>
</tr>
<tr>
<td>$count</td>
<td>Returns the number of records in a collection.</td>
</tr>
<tr>
<td>$orderby</td>
<td>Determines which values are used to order a collection of records.</td>
</tr>
</tbody>
</table>
**OData System Query Option**

<table>
<thead>
<tr>
<th>OData System Query Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$expand</td>
<td>Specifies that related records must be retrieved in line with the record or collection being retrieved. For example, you can use $expand to retrieve a customer and all orders placed by that customer in a single query.</td>
</tr>
</tbody>
</table>

**Note**

The system query options $select and $count are supported out of the box. You do not have to write a script to enable functionality for these system query options.

**Related Information**

Developing an OData Service [page 4]
3  Creating an OData Service

Context

You use this procedure to create an integration package that contains an OData service artifact.

Procedure

1. Launch the SAP Cloud Platform Integration Web application by entering the URL provided by SAP in your local browser.
2. Enter your credentials and choose Log On.
3. Go to the Design tab.
4. Choose Create.
5. Enter the required data for the integration package.
6. In the Artifacts ( ) section, choose Add OData Service.
7. In the Add OData Service Artifact dialog box, enter the OData service details as listed in the following table.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create</td>
<td>Select this radio button if you want to create a new OData service artifact.</td>
</tr>
<tr>
<td>Upload</td>
<td>Select this radio button if you want to upload an OData service artifact from your file location.</td>
</tr>
<tr>
<td>Name</td>
<td>Name of the OData service.</td>
</tr>
<tr>
<td>Namespace</td>
<td>Logical grouping of the OData service. Default is SAP.</td>
</tr>
<tr>
<td>ID</td>
<td>Uniquely identifies the OData service and is a combination of the name and namespace of the OData service in the format &lt;name&gt;_&lt;namespace&gt;_1. This field is not editable.</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the OData service and its purpose.</td>
</tr>
</tbody>
</table>
### OData Service

Select a ZIP file containing the OData service artifact. This field is available and mandatory when you are uploading an OData service artifact.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OData Version</td>
<td>Currently, OData v2 is the only supported OData version.</td>
</tr>
</tbody>
</table>

**Note**
You can edit some of the OData service details such as Name, Description, and Namespace after the service is created. For more information, see Configuring an OData Service [page 40].

8. Choose **OK**. The Service Designer page opens.

   You can further develop the OData service by:
   1. Creating an OData model [page 9]
   2. Binding to a data source [page 21]
   3. Editing an Integration Flow [page 30]

9. Choose **Save** on the Service Designer page to save the OData service artifact.

   **Note**
   To retain a copy of the current artifact, choose **Save as version**.

10. Deploy the OData service. For more information, see Deploying an OData Service [page 41].

11. Save the integration package.
4  Create an OData Model

You can create an OData model in SAP Cloud Platform Integration (Cloud Integration) by:

- Importing from a data source
- Editing a model from the ground up

Cloud Integration gives you the flexibility to create an OData model in an iterative fashion. For example, you can create an OData model by importing the model definition from a SOAP WSDL file. You can add to this OData model by editing it. Or, you can add to the model definition by importing from a different WSDL file or even a different data source, for example, an EDMX file.

Related Information

Import from Data Source [page 9]
Editing an OData Model [page 15]

4.1  Import from Data Source

Cloud Integration allows you to create an OData model by importing the model definition from an existing data source, such as SOAP and OData.

i  Note

When an OData service is generated not all attributes are copied from source metadata. However, once the OData service is generated, you can configure these attributes as needed in the OData Model Editor.

An import wizard guides you in this process with the following steps:

1. Select a data source type.
2. Select a structure.
3. Verify and finalize the EDMX structure.

Related Information

Importing from SOAP [page 10]
Importing from OData [page 11]
Importing from ODC [page 13]
4.1.1 Importing from SOAP

Context

You have created an OData service artifact in an integration package. You can use this procedure to create an OData model or build up an existing one by importing the model definition from SOAP web services.

Procedure

1. If you are creating a new OData service artifact, proceed to the next step. If you are editing an existing OData service artifact, choose Edit on the Service Designer page.

2. Choose Import Model Wizard.

3. In the Select Data Source Type page, select SOAP as data source type.

4. Choose Browse and select the WSDL file from your file system. You can also select the WSDL from the list of existing WSDL files (if any) used in the OData service artifact.

   **Note**
   Composite WSDLs that are flattened are supported. However, multipart WSDLs and other file types are not supported.

5. Choose Step 2. In the Select Structure page, you can see the elements from the WSDL listed in a table.

6. Select elements that you want to add to the OData model from the WSDL Hierarchy column. You can expand or collapse WSDL elements to view the hierarchy better.

7. Deselect any child element that you do not want to include in the OData model. You can see that the corresponding parent elements are in a mixed state (neither checked nor unchecked), if they have other child elements that are selected.

8. Select the OData type that each WSDL element has to be converted to in the OData model. To see options available for various types of WSDL elements, refer the following table:

   **Table 14:**
<table>
<thead>
<tr>
<th>WSDL Type</th>
<th>OData Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation</td>
<td>Complex Type</td>
</tr>
<tr>
<td></td>
<td>Entity Type</td>
</tr>
<tr>
<td></td>
<td>Function Import</td>
</tr>
<tr>
<td>Complex Type</td>
<td>Complex Type</td>
</tr>
</tbody>
</table>
### 4.1.2 Importing from OData

#### Context

You have created an OData service artifact in an integration package. You can use this procedure to create an OData model or build up an existing one by importing the model definition from an existing OData service.

<table>
<thead>
<tr>
<th>WSDL Type</th>
<th>OData Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Message</td>
<td>Complex Type</td>
</tr>
<tr>
<td>Output Message</td>
<td>Complex Type</td>
</tr>
<tr>
<td>Part</td>
<td>Complex Type</td>
</tr>
</tbody>
</table>

**Note**

You can select *Flatten* from the *Import As* dropdown for all WSDL types other than *Operation*. *Flatten* is not an OData type that a WSDL element can be converted to. It is an instruction to the system to skip importing the specified WSDL element into the OData model without affecting the import of its children in the hierarchy.

9. Choose **Step 3**. You can review the mapping of WSDL elements to OData objects in this step.
10. In the **Review and Finish EDMX Structure** page, select a primary key from the properties of entity types.
11. Verify and perform the following additional changes, if required, to the OData model definition:
    - Rename the OData objects listed under *EDM Name*.
    - Select a different EDM type.
    - Update the *Documentation* column.
    - Restore default definition of the OData model by choosing *Restore Default Values*.
12. Once you have completed all the changes, choose **Finish**. The Service Designer page displays the entity sets and function imports from the OData model in a table.
13. Choose **Save** to save your changes to the OData service artifact.

**Note**

To retain a copy of the current artifact, choose **Save as version**.
Procedure

1. If you are creating a new OData service artifact, proceed to the next step. If you are editing an existing OData service artifact, choose Edit on the Service Designer page.

2. Choose Import Model Wizard.

3. In the Select Data Source Type page, select ODATA as data source type.

4. Choose Browse and select the EDMX file from your file system. You can also select the EDMX from the list of existing EDMX files (if any) used in the OData service artifact. metadata.edmx is the file used internally in Cloud Integration Web application to store the OData model definition of the OData service artifact.

   **Note**

   The OData Model Editor supports only OData SAP V2 annotations.

5. Choose Step 2. In the Select Structure page, you can see entity types and function imports from the EDMX listed in a table. The table also displays the EDM type, primary key and documentation information associated with the OData objects.

6. Select OData objects that you need to add to the model from the OData Hierarchy column. You can expand or collapse OData objects to view its properties better.

7. Choose Step 3.

8. In the Review and Finish EDMX Structure page, verify and perform the following additional changes, if required, to the OData model definition:

   - Rename the OData objects listed under EDM Name.
   - Select another primary key.
   - Select a different EDM type.
   - Update the Documentation column.
   - Restore default definition of the OData model by choosing Restore Default Values.

9. Once you have completed all the changes, choose Finish. The Service Designer page displays the entity sets and function imports from the OData model in a table.

10. Choose Save to save your changes to the OData service artifact.

   **Note**

   To retain a copy of the current artifact, choose Save as version.
4.1.3 Importing from ODC

Prerequisites

You have created an OData service artifact in an integration package.

Context

SAP Cloud Platform Integration helps you import a model definition from an OData service, which is created on an on-premise SAP Gateway system. SAP Gateway has a component called Backend Provisioning Component or IW_BEP. This component resides on SAP Business Suite and is used to build OData services from data sources like RFCs, BAPIs, and so on. SAP Cloud Platform Integration communicates with the IW_BEP component in SAP Gateway through OData Channel or ODC protocol, which is an SAP proprietary protocol.

Note

SAP Cloud Platform Integration does not support connecting to an SAP Gateway system based on SAP NetWeaver 7.02 where the IW_BEP component version is SP 06 or lower.

SAP Cloud Platform Integration also does not support connecting to an SAP Gateway system based on SAP NetWeaver 7.40 where the SAP_GWFND component is SP 02 or lower.

You can use this procedure to create an OData model or build up an existing one by importing the model definition from an OData service created in IW_BEP (ODC).

Procedure

1. If you are creating a new OData service artifact, proceed to the next step. If you are editing an existing OData service artifact, choose Edit on the Service Designer page.

2. Choose Import Model Wizard.

3. In the Select Data Source Type page, select ODC as data source type.

4. Enter the connection details as listed in the following table.

<table>
<thead>
<tr>
<th>Table 15: Connection Details</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>Endpoint URI of the target system</td>
</tr>
<tr>
<td>Client</td>
<td>Client of the target system</td>
</tr>
</tbody>
</table>
### Connection Details

<table>
<thead>
<tr>
<th>Description</th>
<th>Connection Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical grouping of the service</td>
<td>Namespace</td>
</tr>
<tr>
<td>Name of the service implementation</td>
<td>Service Name</td>
</tr>
<tr>
<td>Version</td>
<td>Version</td>
</tr>
<tr>
<td>The type of proxy that you are</td>
<td>Proxy Type</td>
</tr>
<tr>
<td>using to connect to the target</td>
<td></td>
</tr>
<tr>
<td>system:</td>
<td></td>
</tr>
<tr>
<td>○ Select On-Premise if you are</td>
<td></td>
</tr>
<tr>
<td>connecting to an on-premise</td>
<td></td>
</tr>
<tr>
<td>system using the SAP Cloud</td>
<td></td>
</tr>
<tr>
<td>Connector. This option is only</td>
<td></td>
</tr>
<tr>
<td>available if you have chosen</td>
<td></td>
</tr>
<tr>
<td>SAP Cloud Platform Integration</td>
<td></td>
</tr>
<tr>
<td>as the product profile. If you</td>
<td></td>
</tr>
<tr>
<td>select this option, the Address</td>
<td></td>
</tr>
<tr>
<td>field of the adapter refers to a</td>
<td></td>
</tr>
<tr>
<td>virtual address that has to be</td>
<td></td>
</tr>
<tr>
<td>configured in the SAP Cloud</td>
<td></td>
</tr>
<tr>
<td>Connector settings.</td>
<td></td>
</tr>
<tr>
<td>Select Basic if you want to use</td>
<td>Authentication</td>
</tr>
<tr>
<td>user credentials to connect to</td>
<td></td>
</tr>
<tr>
<td>the target system.</td>
<td></td>
</tr>
<tr>
<td>Name of the user credentials</td>
<td>Credential Name</td>
</tr>
<tr>
<td>artifact deployed on the tenant</td>
<td></td>
</tr>
</tbody>
</table>

5. Choose **Step 2**.
   
   In the **Select Structure** page, you can see entity types and function imports from the EDMX listed in a table. The table also displays the associated EDM type, primary key, and documentation information.

6. Select entity types and function imports that you need to add to the model from the **OData Hierarchy** column. You can expand or collapse entity types and function imports to view its properties better.

7. Choose **Step 3**.

8. In the **Review and Finish EDMX Structure** page, verify and perform the following additional changes, if required, to the OData model definition:
   ○ Rename the OData objects listed under **EDM Name**.
   ○ Select a different EDM type.
   ○ Select another primary key.
   ○ Update the **Documentation** column.
   ○ Restore default definition of the OData model by choosing **Restore Default Values**.

9. Once you have completed all the changes, choose **Finish**. The Service Designer page displays the entity sets and function imports from the OData model in a table.

10. Choose **Save** to save your changes to the OData service artifact.

---

**Note**

To retain a copy of the current artifact, choose **Save as version**.
4.2 Editing an OData Model

Context

You have created an integration package and added an OData service artifact to the package. You can use this procedure to create an OData model from the ground up. You can also use it to edit an existing OData model.

Procedure

1. If you are creating a new OData service artifact, proceed to the next step. If you are editing an existing OData service artifact, choose *Edit* on the Service Designer page.

2. Choose *OData Model Editor* on the Service Designer page.
   
   If you are creating an OData model from the ground up, the OData Model Editor opens a predefined structure based on the selected OData version. The root is a *Schema* element. Below the root, subelements such as *EntityType*, *ComplexType*, *Association*, *EntityContainer*, and so on, can be added if required.

   **Note**
   
   In the case of an existing OData service artifact, you can only view the OData model in the OData Model Editor unless you explicitly choose *Edit* on the Service Designer page.

   For more information on the features of the OData Model Editor, see [OData Model Editor Features](page 16).

3. Once you have finished editing the OData model, choose *OK*.

4. Choose *Back* to get back to the Service Designer page.

5. Choose *Save* to save your changes to the OData service artifact.

   **Note**
   
   To retain a copy of the current artifact, choose *Save as version*. 
4.2.1 OData Model Editor Features

The OData Model Editor in the SAP Cloud Platform Integration Web application enables you to write an Entity Data Model XML (EDMX). This model serves as metadata for your OData service. OData Model Editor provides you with the following basic features:

- **Code Assist**: Provides you with code suggestions that help speed up your development time. There are two types of code assist:
  - Schema-based
  - Model-based
- **Autocomplete**: Predicts the code element you are entering based on the pattern of letters entered.

**Note**

You can use the OData Model Editor to edit and fine-tune your OData model, after you have imported it from a data source. However, we recommend that you keep the overall end-to-end scenario in mind when you edit the model using this tool. The OData Model Editor can only provide semantic validations. It cannot validate the correctness of the OData model in the context of the OData provisioning scenario that you are working in.

### Related Information

- Using Schema-Based Code Assist [page 16]
- Using Model-Based Code Assist [page 18]
- Using Autocomplete [page 18]
- Validation in the OData Model Editor [page 19]

#### 4.2.1.1 Using Schema-Based Code Assist

**Context**

The schema-based code assist feature provides you with code suggestions that adhere to the Conceptual Schema Definition Language (CSDL) specification. These code suggestions are provided based on the current cursor position.

Follow this procedure to use schema-based code assist in the OData Model Editor.
Procedure

1. Choose **OData Model Editor** on the Service Designer page. The OData Model Editor page opens.
2. Place your cursor in the context where you require assistance, and press `Ctrl + Spacebar`.

   Based on the cursor location, a list of the elements that can be used appears.

   Each element in the dropdown list is represented by an icon. The following image shows the list of elements and their corresponding icons:

   ![Element Icons](image)

3. Scroll through the list, select the required element, and press `Enter`. The OData Model Editor inserts a full snippet of code in the desired location.
4.2.1.2 Using Model-Based Code Assist

**Context**

The model-based code assist feature provides code suggestions based on the OData model you are working on.

Follow this procedure to use model-based code assist in the OData Model Editor.

**Procedure**

1. Choose [OData Model Editor](#) on the Service Designer page. The OData Model Editor page opens.
2. Place the cursor between the quotes of the attribute of an element such as PropertyRef, EntitySet, or AssociationSet and press `Ctrl + Spacebar` to see the relevant suggestions.
3. Scroll through the list and select the required suggestion.

4.2.1.3 Using Autocomplete

**Context**

The autocomplete feature predicts the word you are entering based on the pattern of letters entered and provides you with suggestions. The suggestions are context-sensitive. You can get suggestions for elements, attributes, attribute types, properties, and property values.

Follow this procedure to use the autocomplete feature in the OData Model Editor.

**Procedure**

1. Choose [OData Model Editor](#) on the Service Designer page. The OData Model Editor page opens.
2. Place the cursor in the context where you need assistance and start entering letters. Autocomplete suggestions appear next to the letters in context.
3. Use the `Up arrow` and `Down arrow` keys on the keyboard to select the required suggestions and press `Enter`.
4.2.1.4 Validation in the OData Model Editor

Validation in the OData Model Editor ensures that your OData model adheres to the CSDL schema and does not deviate from logical or semantic specifications.

After validation, lines of code with errors are highlighted with a red icon in the code editor.

Let us look at an example of how validation works.

Example

In any OData model, remove the value corresponding to the Name attribute in an EntityContainer element.

Since Name is a mandatory attribute, the corresponding line of code is highlighted with a red icon in the code editor. You can see the error message in a tooltip when you hover the mouse over the red icon.
5 Viewing the OData Model Layout

Context

You have created an OData model either by importing from a data source or by editing the model manually using the SAP Cloud Platform Integration Web application. You can use this procedure to view the overall layout of an OData model in the Graphical Model Viewer. This helps you verify the OData model and determine if any additional changes are necessary.

Procedure

1. Choose Graphical Model Viewer on the Service Designer page. The OData model is rendered in a graphical format. You can use the navigation control and viewing options to customize your viewing preferences. For more information, see Graphical Model Viewer Features [page 20].

2. Choose Back to get back to the Service Designer page.

5.1 Graphical Model Viewer Features

The Graphical Model Viewer in the SAP Cloud Platform Integration Web application provides you with the following navigation controls and viewing options to customize your viewing preferences:

Fit to Screen

There might be instances where the OData model objects of a big model extend beyond the scope of the Graphical Model Viewer space. In such cases, you can choose Fit to Screen to view the entire model within the boundaries of the Graphical Model Viewer. This feature automatically aligns the model to fit the Graphical Model Viewer space.
Expanding and Collapsing OData Objects

The Graphical Model Viewer lets you expand or collapse all the section headers of the objects in the OData model. You can also expand and collapse individual OData model objects.

Collapsed State

- To collapse all OData objects in the Graphical Model Viewer, choose the Collapse All button on the toolbar. The Collapse All button toggles to Expand All once the model is collapsed.
- If you want to collapse an individual OData object, choose the arrowhead button in the expanded object.

Expanded State

- To expand all OData objects in the Graphical Model Viewer, choose the Expand All button on the toolbar. The Expand All button toggles to Collapse All once the model is expanded.
- If you want to expand an individual OData object, choose the arrowhead button in the collapsed object.

Navigation Controls

Depending on the number of OData objects in the model, objects might appear too small or be too big for the Graphical Model Viewer to accommodate. In such cases, the navigation controls help you view the model better.

Table 16:

<table>
<thead>
<tr>
<th>Button</th>
<th>Usage</th>
</tr>
</thead>
</table>
| ![Compass Button](image) | Use the compass to control the perspective of the model:  
- Choose ![Right Arrow](image) to move right.  
- Choose ![Left Arrow](image) to move left.  
- Choose ![Down Arrow](image) to move down.  
- Choose ![Up Arrow](image) to move up.  
- Choose ![Zoom In](image) to zoom the model to 100%. |
| ![Zoom Buttons](image) |  
- Choose ![Zoom In](image) to zoom in to the model.  
- Choose ![Zoom Out](image) to zoom out of the model.  
- The zoom level indicator ![Zoom Percentage](image) shows the zoom percentage. |
6  Bind to Data Source

You have created an OData model in the SAP Cloud Platform Integration Web application. As a result, you can see OData objects associated with the model listed in a table on the Service Designer page.

Next, you need to bind or connect function imports and operations of entity sets to a data source. When you bind an OData object to a data source, a predefined integration flow is autogenerated and set up for you. You can modify this integration flow to suit your scenario.

Note

The externalise option is not visible for these autogenerated integration flows.

You can bind one or all five CRUDQ operations of an entity set to a data source. This means that each entity set can have multiple integration flows connecting to the same data source. You can also bind entity sets and function imports in an OData service artifact to different data sources.

The following data sources are currently supported:

- SOAP
- OData
- REST
- ODC

Binding Status

To identify the graphical representations of the OData objects and their binding status on the Service Designer page, refer to the following table:

<table>
<thead>
<tr>
<th>OData Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![grid icon]</td>
<td>Represents an entity set in an OData model.</td>
</tr>
<tr>
<td>![gear icon]</td>
<td>Represents an operation associated with an entity set in an OData model.</td>
</tr>
<tr>
<td>![lock icon]</td>
<td>Represents an operation bound to a data source with errors in the associated integration flow. You cannot deploy the OData service unless the errors are corrected.</td>
</tr>
</tbody>
</table>
### OData Object Description

<table>
<thead>
<tr>
<th>OData Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🔄⚠️</td>
<td>Represents an operation bound to a data source with warnings in the associated integration flow. We recommend that you verify the warnings and take corrective measures where necessary before deploying the OData service.</td>
</tr>
<tr>
<td>⚙️</td>
<td>Represents an operation bound to a data source with no errors or warnings.</td>
</tr>
<tr>
<td>⚙️</td>
<td>Represents a function import in an OData model.</td>
</tr>
<tr>
<td>⚙️⚠️</td>
<td>Represents a function import bound to a data source with errors in the associated integration flow. You cannot deploy the OData service unless the errors are corrected.</td>
</tr>
<tr>
<td>⚙️⚠️</td>
<td>Represents a function import bound to a data source with warnings in the associated integration flow. We recommend that you verify the warnings and take corrective measures where necessary before deploying the OData service.</td>
</tr>
<tr>
<td>⚙️</td>
<td>Represents a function import bound to a data source with no errors or warnings.</td>
</tr>
</tbody>
</table>

### Unused Bindings

After binding an entity set or function import to a data source, if you rename or delete it in the OData Model Editor, the binding is no longer associated with it and becomes unused. You cannot deploy the OData service unless you update the binding. For more information, see Managing Unused Bindings [page 29].

### Related Information

- Binding to SOAP [page 23]
- Binding to OData [page 24]
- Binding to REST [page 25]
- Binding to ODC [page 26]
6.1 Binding to SOAP

Prerequisites

You are editing an OData service artifact and you have created an OData model. OData objects in the model are listed in a table on the Service Designer page.

Context

You can use this procedure to bind function imports and operations of entity sets to the corresponding SOAP Web service operations.

Procedure

1. Select SOAP as the data source for an entity set or function import.

2. Choose Bind ( ) for the operation associated with the entity set or function import that you need to bind. The Configure SOAP Data Source dialog box appears.

3. Choose Browse and select the WSDL file from your file location. If the OData model was imported from a WSDL file, the corresponding file name appears by default. You can change this to any other file name as required.

   **Note**

   Composite WSDLs that are flattened are supported. However, multipart WSDLs and other file types are not supported.

4. Select an operation. If the OData model was imported from a WSDL file, operations defined in the WSDL appear in the dropdown list.

5. Enter the Endpoint URI where the operation is located. If the OData model was imported from a WSDL file and an operation was selected from this WSDL, the corresponding Endpoint URI is auto-populated.

6. Choose OK. The Service Designer page appears and the Action column for this operation now displays Navigate to Integration Flow Editor ( ) and Delete Binding ( ) buttons.

   A predefined integration flow with mapping steps is generated for the operation. For more information, see Predefined Integration Flows for SOAP [page 32].

   You can update the integration flow to better suit your business scenario by choosing Navigate to Integration Flow Editor ( ). For more information, see Editing an Integration Flow [page 30].
In case you need to update the binding information, you can do it in one of the following ways:

○ Choose **Navigate to Integration Flow Editor** and update the Endpoint URI information in the receiver channel.

○ Choose **Delete Binding** and repeat the binding procedure.

7. Choose **Save** to save your changes to the OData service artifact.

### Note
To retain a copy of the current artifact, choose **Save as version**.

---

### 6.2 Binding to OData

#### Prerequisites

You are editing an OData service artifact and you have created an OData model. OData objects in the model are listed in a table on the Service Designer page.

#### Context

You can use this procedure to bind function imports and operations of entity sets to the corresponding OData services.

#### Procedure

1. Select **ODATA** as the data source for an entity set or function import.

2. Choose **Bind** for the function import or operation associated with the entity set that you need to bind. The **Configure OData Data Source** dialog box appears.

3. Choose **Browse** and select the EDMX file from your file location. You can also select the file from the **Existing Files** list. The **Existing Files** list is populated with EDMX file names used in other bindings in the same OData service artifact.

4. Select an entity set from the **Entity Sets** dropdown list. This list displays all entity sets from the selected EDMX file.

5. Enter the Endpoint URI corresponding to the OData service that you are binding to.
6. Choose **OK**. The Service Designer page appears and the **Action** column for this operation now displays *Navigate to Integration Flow Editor* ( ) and *Delete Binding* ( ) buttons.

A predefined integration flow with mapping steps is generated for the operation or function import. For more information, see *Predefined Integration Flows for OData* [page 35].

You can update the integration flow to better suit your business scenario by choosing *Navigate to Integration Flow Editor* ( ). For more information, see *Editing an Integration Flow* [page 30].

In case you need to update the binding information, you can do it in one of the following ways:

- Choose *Navigate to Integration Flow Editor* ( ) and update the Endpoint URI information in the receiver channel.
- Choose *Delete Binding* ( ) and repeat the binding procedure.

7. Choose **Save** to save your changes to the OData service artifact.

**Note**

To retain a copy of the current artifact, choose **Save as version**.

### 6.3 Binding to REST

**Prerequisites**

You are editing an OData service artifact and you have created an OData model. OData objects in the model are listed in a table on the Service Designer page.

**Context**

You can use this procedure to bind function imports and operations of entity sets to the corresponding REST services.

**Procedure**

1. Select **REST** as the data source for an entity set or function import.
2. Choose **Bind** (🔗) for the function import or operation associated with the entity set that you need to bind. The **Configure REST Data Source** dialog box appears.

3. Enter the Endpoint URI corresponding to the REST service that you are binding to.

4. Choose **OK**. The Service Designer page appears and the **Action** column for this operation or function import now displays **Navigate to Integration Flow Editor** (🔗) and **Delete Binding** (🗑️) buttons.

   A predefined integration flow with mapping steps is generated for the operation or function import. For more information, see **Predefined Integration Flows for REST** [page 37].

   You can update the integration flow to better suit your business scenario by choosing **Navigate to Integration Flow Editor** (🔗). For more information, see **Editing an Integration Flow** [page 30].

   In case you need to update the binding information, you can do it in one of the following ways:

   - Choose **Navigate to Integration Flow Editor** (🔗) and update the Endpoint URI information in the receiver channel.
   - Choose **Delete Binding** (🗑️) and repeat the binding procedure.

5. Choose **Save** to save your changes to the OData service artifact.

   **Note**

   To retain a copy of the current artifact, choose **Save as version**.

### 6.4 Binding to ODC

#### Prerequisites

You are editing an OData service artifact and you have created an OData model. OData objects in the model are listed in a table on the Service Designer page.

#### Context

SAP Cloud Platform Integration helps you bind or connect to an OData service, which is created on an on-premise SAP Gateway system. SAP Gateway has a component called Backend Provisioning Component or IW_BEP. This component resides on SAP Business Suite and is used to build OData services from data sources like RFCs, BAPIs, and so on. Cloud Integration communicates with the IW_BEP component in SAP Gateway through OData Channel or ODC protocol, which is an SAP proprietary protocol.
Note

Cloud Integration does not support connecting to an SAP Gateway system based on SAP NetWeaver 7.02 where the IW_BEP component version is SP 06 or lower.

Cloud Integration also does not support connecting to an SAP Gateway system based on SAP NetWeaver 7.40 where the SAP_GWFND component is SP 02 or lower.

You can use this procedure to bind function imports and operations of entity sets to the corresponding OData services created in IW_BEP (ODC).

Procedure

1. Select **ODC** as the data source for an entity set or function import.

2. Choose **Bind** (🔗) for the function import or operation associated with the entity set that you want to bind.

   The **Configure ODC Data Source** dialog box appears. The entity set and CRUDQ operation information is prepopulated.

3. Enter the following information related to the ODC destination.

   Table 18:

<table>
<thead>
<tr>
<th>Connection Details</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Sets</td>
<td>A list of entity sets that is populated from the ODC source when importing the OData model. If you have coded the OData model, then the list is populated directly from your model. Select an entity set that you want to bind to.</td>
</tr>
<tr>
<td>Address</td>
<td>Endpoint URI of the target system</td>
</tr>
<tr>
<td>Client</td>
<td>Client of the target system</td>
</tr>
<tr>
<td>Namespace</td>
<td>Logical grouping of the service implementation</td>
</tr>
<tr>
<td>Service Name</td>
<td>Name of the service implementation</td>
</tr>
<tr>
<td>Version</td>
<td>Version of the service implementation</td>
</tr>
</tbody>
</table>
### Connection Details

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
</table>
| **Proxy Type** | The type of proxy that you are using to connect to the target system:  
  ○ Select **On-Premise** if you are connecting to an on-premise system using the SAP Cloud Connector. This option is only available if you have chosen **HANA Cloud Integration** as the product profile. If you select this option, the **Address** field of the adapter refers to a virtual address, which has to be configured in the SAP Cloud Connector settings. |
| **Authentication** | Select **Basic** if you want to use user credentials to connect to the target system. |
| **Credential Name** | Name of the user credentials artifact deployed on the tenant. |

4. Choose **OK**. The Service Designer page appears and the **Action** column for this operation now displays **Navigate to Integration Flow Editor** and **Delete Binding** buttons.

A predefined integration flow with mapping steps is generated for the operation or function import. For more information, see [Predefined Integration Flows for ODC](page 37).

You can update the integration flow to better suit your business scenario by choosing **Navigate to Integration Flow Editor**. For more information, see [Editing an Integration Flow](page 30).

If you want to update the binding information, you can do it in one of the following ways:

○ Choose **Navigate to Integration Flow Editor** and update the Endpoint URI information in the receiver channel.

○ Choose **Delete Binding** and repeat the binding procedure.

5. Choose **Save** to save your changes to the OData service artifact.

**Note**

To retain a copy of the current artifact, choose **Save as version**.
7 Managing Unused Bindings

Context

You are editing an OData service artifact and you have bound a function import or operations of an entity set to a data source. At this point, let us assume you changed the name of the function import or entity set, or even deleted it in the OData Model Editor. The binding associated with the renamed or deleted entity set or function import is now unused.

The unused bindings appear in a separate table on the Service Designer page. You cannot deploy the OData service unless you update the unused bindings.

SAP Cloud Platform Integration Web application helps you manage unused bindings by providing you with the following options:

Reconfigure Binding

Context

You can use this procedure to reconfigure the binding from the earlier entity set or function import to one in the updated OData model.

Procedure

1. Choose Bind (🔗) for the entity set or function import associated with the unused binding. The Reconfigure Binding dialog box appears.
2. Select the entity set or function import in the updated OData model to which the unused binding needs to be associated.
3. Choose OK. The selected entity or function import in the updated OData model is now associated with the binding. The Unused Bindings table disappears if there are no other unused bindings.
   - SAP Cloud Platform Integration updates the sender channel with the new binding information. However, you have to update other integration flow components that references the earlier entity set or function import, such as mapping and scripts.
4. Choose Save to save your changes to the OData service artifact.
Delete Binding

Context

You can use this procedure to delete the unused binding so that it is no longer associated with any entity set or function import in the OData model.

Procedure

1. Choose Delete Binding (trash can) for the associated entity set or function import.
2. Confirm the deletion. The Unused Bindings table disappears if there are no other unused bindings.
3. Choose Save to save your changes to the OData service artifact.

Note

To retain a copy of the current artifact, choose Save as version.
8 Editing an Integration Flow

Context

You are editing an OData service artifact and you have created an OData model. You have provided binding information for at least one OData object in the model.

Once a function import or operation of an entity set is bound to a data source, a predefined integration flow is set up for you. Within this integration flow, the OData adapter is selected as the default for the sender channel. The data source to which the operation is bound is selected as the default adapter for the receiver channel.

You can use this procedure to update the predefined integration flow to suit your business scenario.

Procedure

1. Choose Navigate to Integration Flow Editor for an operation or function import that is bound to a data source. The associated integration flow opens in the Integration Flow Editor.
2. Edit the integration flow according to your requirements.

   Note

   The following points need to be kept in mind when editing integration flows in an OData service artifact:
   ○ The OData sender channel is not editable.
   ○ The OData sender channel, participant and start event cannot be deleted.
   ○ A new sender participant cannot be dragged and dropped into the integration flow.
   ○ If you modify the default name of the receiver adapter, you need to make sure it is unique across all integration flows in an OData service artifact.
   ○ You need to ensure that the Namespace Mapping entered in the runtime configuration, is consistent across all integration flows in an OData service artifact.


   However, it is perfectly valid to enter `xmlns:test=http://sapcd.com/testABC` in one integration flow and `xmlns:test2=http://sapab.com` in another.

   For more information about configuring integration flow components, see .
3. Once you have finished editing the integration flow, choose OK.
4. Choose Back to get back to the Service Designer page.
5. Choose Save to save your changes to the OData service artifact.

Note
To retain a copy of the current artifact, choose Save as version.

Related Information

Predefined Integration Flows [page 31]

8.1 Predefined Integration Flows

The SAP Cloud Platform Integration Web application sets up predefined integration flows when OData objects are bound to a data source. These integration flows have the sender channel preconfigured to the OData adapter type.

A predefined integration flow has the minimum setup required for an integration scenario. You have to configure this integration flow to suit the specific requirements of your scenario.

Related Information

Predefined Integration Flows for SOAP [page 32]
Predefined Integration Flows for OData [page 35]
Predefined Integration Flows for REST [page 37]
Predefined Integration Flows for ODC [page 37]

8.1.1 Predefined Integration Flows for SOAP

The SAP Cloud Platform Integration Web application sets up predefined integration flows when the CRUDQ (Create, Read, Update, Delete, and Query) operations of entity sets and function imports are bound to a SOAP Web service.
8.1.1.1 Predefined Integration Flows for Create and Read Operations

This integration flow is autogenerated for Create and Read operations. In order to complete the binding to the data source, you have to edit at least the following elements:

- Request Mapping
- Response Mapping

To understand this integration flow, let’s look at an example for the Create operation.

**Example**

You open an app on your smartphone to book a hotel room in the city of New York. This application sends the booking information to a hotel booking system. The hotel booking system receives this data with the help of a SOAP Web service, stores it, and sends the booking status back to you.

The mobile application communicates with the hotel booking system by calling a URI to create a booking record and passes the booking information as the payload. When the OData sender receives the request, it passes it to the Request Mapping element. This mapping element contains information that maps the fields from the OData structure to the SOAP structure and passes the create request to the SOAP receiver.

The SOAP receiver processes the request and creates a booking record in the back end. If the operation is successful, it returns the status to the Response Mapping element. The Response Mapping element contains information that maps the fields from the SOAP structure to the OData structure. It passes the booking status and other mapped details such as the confirmation number to the OData sender. OData sender in turn passes the response to the mobile application.
8.1.1.2 Predefined Integration Flows for Update and Delete Operations

This integration flow is autogenerated for Update and Delete operations. In order to complete the binding to the data source, you have to edit at least the Request Mapping element.

To understand this integration flow, let’s look at an example of an Update operation.

**Example**

You open an app on your smartphone to extend your stay in a hotel room in the city of New York. This application sends the booking information to a hotel booking system. The hotel booking system receives this data with the help of a SOAP Web service and updates it.

The mobile application communicates with the hotel booking system by calling a URI to update a booking record and passes the booking information as the payload. When the OData sender receives the request, it passes it to the Request Mapping element. This mapping element contains information that maps the fields from the OData structure to the SOAP structure and passes the update request to the SOAP receiver.

The SOAP receiver processes the request and updates the booking record in the back-end. The status of the update operation is then sent to the OData sender. The OData sender in turn sends it to the mobile application.

8.1.1.3 Predefined Integration Flow for Query Operations
This integration flow is autogenerated for Query operations. In order to complete the binding to the data source, you have to edit at least the following elements:

- Content Modifier
- Response Mapping

**Note**

If you want to enable system query options such as $top, $skip, and $filter, you can replace the Content Modifier element with a Script element.

To understand this integration flow, let’s look at an example.

**Example**

A mobile application needs to get a list of hotels available in the city of New York from a hotel booking system. The hotel booking system publishes this data in the form of a SOAP Web service. The mobile application places this request in the form of a URI, where the city of New York is passed as a parameter.

When the OData sender channel receives the URI request, it passes it to the Content Modifier element. The Content Modifier element adds additional information to the head and body of the incoming message so that the SOAP receiver can understand it.

The SOAP receiver processes the request and, if there are no issues with the request, returns a list of hotels as the response. The Response Mapping element contains information that maps the fields from the SOAP structure to the OData structure. The mapped data is then sent to the mobile application through the OData sender channel.

### 8.1.1.4 Predefined Integration Flow for Function Imports

This integration flow is autogenerated for function imports. In order to complete the binding to the SOAP Web service, you have to edit at least the Script elements for request and response mapping.

To understand this integration flow, let’s look at an example.
Example

You open an app on your smartphone to check the most frequently visited hotel in the city of New York and the number of guests that visited this hotel in the previous year. The hotel booking system publishes this data in the form of a SOAP Web service. The mobile application places this request in the form of a URI, where the city of New York is passed as a parameter.

When the OData sender channel receives the URI request, it passes it to the Script element. The Script element maps the request so that the SOAP receiver understands it.

The SOAP receiver processes the request and, if there are no issues with the request, returns the most frequently visited hotel and the guest count as the response. The Script element contains information that maps the fields from the SOAP structure to the return type defined in the OData model. The mapped data is then sent to the mobile application through the OData sender channel.

8.1.2 Predefined Integration Flows for OData

The SAP Cloud Platform Integration Web application sets up predefined integration flows when the Create, Read, Update, and Query operations of entity sets are bound to an OData service.

Note

Function imports are not available in the OData receiver channel. We recommend that you use another adapter like HTTP for it.

Predefined Integration Flows for Query and Read Operations

This integration flow is autogenerated for Query and Read operations. Script step in this integration flow takes the query options from the incoming URI and sets it into an exchange header <odataURI>. In the case of
Query operations, you can use this exchange header to dynamically pass the query options to the OData receiver channel. And in the case of Read operations, you can use this exchange header to dynamically pass the key fields of the entity set to the OData receiver channel.

**Predefined Integration Flows for Create and Update Operations**

![Diagram of integration flow](image)

This integration flow is autogenerated for Create and Update operations. Script step in this integration flow takes the query options from the incoming URI and sets it into an exchange header `<odataURI>`. In the case of Update operations, you can use this exchange header to dynamically pass the key fields of the entity set to the OData receiver channel.

### 8.1.3 Predefined Integration Flows for REST

The SAP Cloud Platform Integration Web application sets up predefined integration flows when the CRUDQ (Create, Read, Update, Delete, and Query) operations of entity sets and function imports are bound to a REST web service.
This integration flow is autogenerated for all CRUDQ operations when the receiver is a REST service. You can edit the Script elements in order to define the request and response mappings.

### 8.1.4 Predefined Integration Flows for ODC

The SAP Cloud Platform Integration Web application sets up predefined integration flows when the CRUDQ (Create, Read, Update, Delete, and Query) operations of entity sets and function imports are bound to an ODC service.

#### Predefined Integration Flow for Query and Read Operations

This integration flow is autogenerated for Query and Read operations. Script step in this integration flow takes the query options and key fields of entity sets from the incoming URI and sets it into an exchange header `<odataURI>`.

**Note**

Currently, the ODC receiver channel does not support any query options or key fields of entity sets that are dynamically passed using the exchange header. You can map the fields from the ODC structure to the OData structure in the Response Mapping element.
Predefined Integration Flow for Create Operation

This integration flow is autogenerated for Create operation. You can map the fields from the OData structure to the ODC structure in the Request Mapping element. Similarly, map the fields from the ODC structure to the OData structure in the Response Mapping element.

Predefined Integration Flow for Update Operation

This integration flow is autogenerated for Update operation. Script step in this integration flow takes the key fields of entity sets from the incoming URI and sets it into an exchange header <odataURI>.

Note
Currently, the ODC receiver channel does not support any key fields of entity sets that are dynamically passed using the exchange header.

You can map the fields from the OData structure to the ODC structure in the Request Mapping element.
Predefined Integration Flow for Delete Operation

This integration flow is autogenerated for Delete operation. Script step in this integration flow takes the key fields of entity sets from the incoming URI and sets it into an exchange header `<odataURI>`.

**Note**
Currently, the ODC receiver channel does not support any key fields of entity sets that are dynamically passed using the exchange header.
9 Configuring an OData Service

Context

You have created an OData service artifact. You can use this procedure to modify the parameters that you entered when creating the artifact. These parameters identify the OData service.

Procedure

1. If the OData service is newly created, proceed directly to the next step. If it is an existing one, choose Edit on the Service Designer page.

2. Choose Service Configuration on the Service Designer page.

3. Edit or modify the parameters listed in the following table, as required.

Table 19:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the OData service.</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the OData service and its purpose.</td>
</tr>
<tr>
<td>Namespace</td>
<td>Logical grouping of the OData service.</td>
</tr>
</tbody>
</table>

4. Choose OK.

5. Choose Save on the Service Designer page to save your changes to the OData service artifact.

Note

To retain a copy of the current artifact, choose Save as version.
10  Deploying an OData Service

Context

You have created an OData service artifact with a valid OData model. You have provided binding information for at least one OData object in the model. You have verified that all integration flows are valid.

You can use this procedure to deploy the OData service.

Procedure

1. Choose **Deploy** on the Service Designer page.
2. Confirm that you do want to deploy. If there are no validation errors, a message is displayed confirming that the service is triggered for deployment.
3. Choose **OK**. The OData service is now deployed.

   The OData service starts on the tenant, after deployment. You can confirm that the status of the OData service is **Started** in the Integration Content Monitor. For more information, see .

   It may take a few moments for the OData service to get active. Once it is active, it is ready for consumption.
Important Disclaimers and Legal Information

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. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, unless damages were caused by SAP intentionally or by SAP’s gross negligence.

Accessibility

The information contained in the SAP documentation represents SAP’s current view of accessibility criteria as of the date of publication; it is in no way intended to be a binding guideline on how to ensure accessibility of software products. SAP in particular disclaims any liability in relation to this document. This disclaimer, however, does not apply in cases of willful misconduct or gross negligence of SAP. Furthermore, this document does not result in any direct or indirect contractual obligations of SAP.

Gender-Neutral Language

As far as possible, SAP documentation is gender neutral. Depending on the context, the reader is addressed directly with “you”, or a gender-neutral noun (such as “sales person” or “working days”) is used. If when referring to members of both sexes, however, the third-person singular cannot be avoided or a gender-neutral noun does not exist, SAP reserves the right to use the masculine form of the noun and pronoun. This is to ensure that the documentation remains comprehensible.

Internet Hyperlinks

The SAP documentation may contain hyperlinks to the Internet. These hyperlinks are intended to serve as a hint about where to find related information. SAP does not warrant the availability and correctness of this related information or the ability of this information to serve a particular purpose. SAP shall not be liable for any damages caused by the use of related information unless damages have been caused by SAP’s gross negligence or willful misconduct. All links are categorized for transparency (see: https://help.sap.com/viewer/disclaimer).