Oracle® Construction and Engineering

Using the Oracle Primavera Cloud Data Service with the REST Adapter

G17663-01
January 2025

Oracle Construction and Engineering Using the Oracle Primavera Cloud Data Service with the REST Adapter, G17663-01

Primary Author: Oracle Corporation

Contents

Preface

This document describes how to create, configure, and run this integration in Oracle Integration 3.

Topics:

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `https://www.oracle.com/corporate/accessibility/`.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit `https://support.oracle.com/portal/` or visit `Oracle Accessibility Learning and Support` if you are hearing impaired.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Related Resources

See these Oracle resources:

- Oracle Cloud at `http://cloud.oracle.com`
- *Using Integrations in Oracle Integration 3*
- *Using the REST Adapter with Oracle Integration 3*
- Oracle Primavera Cloud REST API
- Oracle Integration documentation
- Oracle Primavera Cloud documentation

Conventions

The following text conventions are used in this document.

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

About This Guide

This guide describes how to create an integration flow to fetch all in-progress activities across all projects within a workspace in Oracle Primavera Cloud.

This integration uses the Oracle Primavera Cloud Data Service REST API Adapter to fetch the in-progress activities from all projects in a workspace. After the activities are fetched, the output data from the Oracle Primavera Cloud Data Service API is saved in a file. This file is then stored at a Secure File Transfer Protocol (SFTP) location.

This integration requires two connections: an Oracle Primavera Cloud Data Service connection and an SFTP connection. The connection to the Oracle Primavera Cloud Data Service is established through the REST Adapter, while the connection to the SFTP service is established through the FTP Adapter in Oracle Integration.

The process of fetching all in-progress activities across all projects within a workspace includes the following:

- Creating an Oracle Primavera Cloud Data Service connection using the REST Adapter
- Creating an SFTP connection using the FTP Adapter
- Constructing a JSON request body for the runQuery() API
- Creating an Oracle Primavera Cloud Data Service integration flow
- Adding the required mappings

2

Create the Connections

Before building this integration, you must create two connections: an Oracle Primavera Cloud Data Service connection and an SFTP connection.

The connection to the Oracle Primavera Cloud Data Service is established through the REST Adapter, and the connection to the SFTP service is established through the FTP Adapter.

Create an Oracle Primavera Cloud Data Service REST Connection

1. In the navigation pane, select **Design**, then **Connections**.
2. Click **Create**.
3. In the **Create connection** panel, do the following
   a. Select **REST**.
   b. Enter the information that describes this connection:
      i. In the **Name** field, enter `Oracle Primavera Cloud Data Service Connection`.

         The value you enter in the **Name** field is automatically added to the **Identifier** field. Any spaces or dashes in the name are automatically replaced with an underscore character in the **Identifier** field. For example: `Oracle_Primavera_Cloud_Data Service_Connection`. If you choose to modify the identifier, do not include spaces.

      ii. From the **Role** list, select **Trigger and Invoke**.
   c. Click **Create**.
4. On the **Configure connection properties** page, enter the connection details:
   a. In the **Properties** section, do the following:
      i. From the **Connection Type** list, select **REST API Base URL**.
      ii. In the **Connection URL** field, enter the base URL of your Oracle Primavera Cloud Data Service REST API.
      For example:

      `https://primavera.oraclecloud.com/data/rest`.
   b. In the **Security** section, do the following:
      i. From the **Security policy** list, select **OAuth Custom Two Legged Flow**.
      ii. In the **Access Token Request** field, enter the OAuth access token generation request for accessing the API.

```
X POST -H "Content-Type: application/x-www-form-urlencoded" -H
"Authorization: Basic <base64 encoded code>" -d
'grant_type=client_credentials' https://primavera.oraclecloud.com/
primediscovery/apitoken/request?scope=http://primavera.oraclecloud.com/
dataPOST -H "Content-Type: application/x-www-form-urlencoded" -H
```

```
"Authorization: Basic <base64 encoded code>" -d
'grant_type=client_credentials'
```

> **Note:**
>
> Use a base64 converter to generate the authcode. For example, you can use https://base64.guru/converter. Enter the input in your base64 converter in the *username.password* format. For example: `testuser@oracle.com:Primavera@1`

iii. Expand **Optional security**, and then do the following:

i. In the **Refresh Token Request** field, enter `NA`.

ii. In the **access_token** field, enter `access.[tT]oken`.

iii. In the **refresh_token** field, enter `refresh.[tT]oken`.

iv. In the **expiry** field, enter `expires_in`.

v. In the **token_type** field, enter `token.?[tT]ype`.

vi. In the **access_token_usage** field, enter the following:

```
-H Authorization: Bearer ${access_token} -H x-prime-tenant:
<tenant-id> -H x-prime-identity-app: <tenant-appid> -H x-prime-
tenant-code: <tenant-code> -H x-prime-region: <region>
```

> **Note:**
>
> Update the `x-prime-tenant`, `x-prime-identity-app`, `x-prime-tenant-code`, `x-prime-region` values before setting the **access_token_usage** value.

5. Click **Save**. If prompted, click **Save** again.
6. Click **Test** to ensure that your connection is successfully configured. In the resulting dialog, click **Test** again.

   A message confirms if your test is successful.

Create an SFTP Service REST Connection

1. In the navigation pane, select **Design**, then **Connections**.
2. Click **Create**.
3. In the **Create connection** panel, do the following:
   a. Select **FTP**.
   b. Enter the information that describes this connection:
      i. In the **Name** field, enter `SFTP Connection`.

      The value you enter in the **Name** field is automatically added to the **Identifier** field. Any spaces or dashes in the name are automatically replaced with an underscore character in the **Identifier** field. For example: `SFTP_Connection`. If you choose to modify the identifier, do not include spaces.

      ii. From the **Role** list, select **Invoke**.

c. Click **Create**.

4. On the **Configure connection properties** page, enter the connection details:
   a. In the **Properties** section, do the following:
      i. In the **FTP Server Host Address** field, enter the host address of the FTP/FTPS server.
      ii. In the **FTP Server Port** field, enter the FTP server port number.
      iii. Expand **Optional properties**, and then from the **SFTP Connection** list, select **Yes**.
   b. In the **Security** section, do the following:
      i. From the **Security policy** list, select **FTP Server Access Policy**.
      ii. In the **User Name** field, enter your user name.
      iii. In the **Password** field, enter your password.
5. Click **Save**. If prompted, click **Save** again.
6. Click **Test** to ensure that your connection is successfully configured. In the resulting dialog, click **Test** again.

   A message confirms if your test is successful.

3

Construct a JSON Request Body for runQuery() API

The Oracle Primavera Cloud Data Service runQuery() API is required to fetch the in-progress activities from a workspace in Oracle Primavera Cloud.

A POST request runs queries against the Oracle Primavera Cloud Data Service. The following query is used to fetch the in-progress activities from a workspace:

```
Select a.ACTIVITY_NAME, a.ACTIVITY_CODE, a.PROJ_ID, a.STATUS from SM_ACTIVITY a,
CO_PROJECT b, WS_WORKSPACE c where a.status='IN_PROGRESS' and a.PROJ_ID=b.PROJ_ID
and b.WORKSPACE_ID = c.WORKSPACE_ID and c.WORKSPACE_CODE = "OICWS";
```

Convert the query to a JSON request to send it to the runQuery() API. Your JSON request should contain the following:

- **Tables**: The query should consist of one or more objects specifying the application database tables. For example, in the above query, `Select a.*` indicates the `SM_ACTIVITY` table. The `CO_PROJECT` and `WS_WORKSPACE` tables are also part of the query, but the requested columns are only from the `SM_ACTIVITY` table. The other tables are used for joining the condition. Therefore, the table name is set as:
  `"tables": [{ "tableName": "SM_ACTIVITY"}]`
- **Columns**: The query should contain a list of database table columns. The API returns only the specified columns. The other table columns are not included in the response. The above query includes the following columns: `ACTIVITY_NAME`, `ACTIVITY_CODE`, `PROJ_ID`, and `STATUS`. The columns are listed as:

  ```
  ACTIVITY_NAME, ACTIVITY_CODE, PROJ_ID and STATUS"columns": [
          "ACTIVITY_NAME",
          "ACTIVITY_CODE",
          "PROJ_ID",
          "STATUS"
      ]
  ```

- **Conditions**: The query should contain a set of conditions that the column data must meet in order to be included in the response. The above query has the following conditions: `status='IN_PROGRESS'` and `WORKSPACE_CODE = "OICWS"`. To fetch the in-progress activities from a workspace, the status code should be *in-progress* and the workspace code should be the user-specified code of the workspace from where the activities are fetched.

Your JSON query should look like this:

```
{
    "name": "Project Activity",
    "pageSize": "10",
    "nextTableName": "SM_ACTIVITY",
    "tables": [
        {
            "tableName": "SM_ACTIVITY",
            "columns": [
```

```
                "ACTIVITY_NAME",
                "ACTIVITY_CODE",
                "PROJ_ID",
                "STATUS"
            ],
            "condition": {
                "operator": "AND",
                "conditions": [
                    {
                        "columnName": "STATUS",
                        "operator": "EQUALS",
                        "value1": "IN_PROGRESS"
                    },
                    {
                        "columnName": "PROJ_ID",
                        "operator": "IN",
                        "val-ue1": "(SELECT PROJ_ID FROM CO_PROJECT PROJ,
WS_WORKSPACE WS WHERE PROJ.WORKSPACE_ID=WS.WORKSPACE_ID AND WS.NAME='OICWS')"
                    } ]            }
        } ]
}
```

4

Create an Integration Workflow

Integrations use the connections you created to your applications and define how information is shared between those applications.

Create an integration flow by creating an integration, and then dragging the required adapters into the invoke area of the integration. Map data between the invoke and the target connections' data structures to define how data will be transferred.

Topics:

Create the Integration

After you create your connections, you can create the integration.

To create the integration:

1. In the navigation pane, select **Design**, then **Integrations**.
2. Click **Create**.
3. In the **Create Integration** panel, do the following:
   a. Click **Schedule**.
   b. Enter the following mandatory schedule details:
      i. In the **Name** field, enter a name for the schedule.
      You can include English alphabetic characters, numbers, underscores, and dashes in the identifier.
      ii. In the **Identifier** field, accept the default value or change it if needed.
      The name you enter in the Name field is automatically added to the Identifier field in upper case. If you want to modify the identifier, do not include blank spaces.
   c. Click **Create**.

Add an Invoke for Oracle Primavera Cloud Data Service

To fetch all in-progress activities from all projects across a workspace, add the Oracle Primavera Cloud Data Service connection to the integration. When you drag the adapter into the invoke area of an integration, the Adapter Endpoint Configuration Wizard opens. This wizard guides you through the configuration of the adapter endpoint properties.

To add the invoke:

1. On the integration page, hover over an arrow and click **Add** .
2. From the **Available connections** list, select **Oracle Primavera Cloud Data Service Connection**.
3. In the Adapter Endpoint Configuration Wizard, do the following:
   a. On the **Basic Info** page:
      i. In the **What do you want to call your endpoint?** field, enter `runQuery`.
      ii. In the **What does this endpoint do?** field, enter an optional description. For example: `Runs a query against the Oracle Primavera Cloud Data Service runQuery endpoint.`
      iii. In the **What is the endpoint's relative resource URI?** field, enter `/dataservice/runquery/`.
      iv. From the **What action do you want to perform on the endpoint?** list, select **POST**.
      v. Select the **Configure a request payload at this endpoint** and **Configure this endpoint to receive a response** check boxes.
   b. On the **Request** page:
      i. From the **Select the request payload format** list, select **JSON sample**, and then enter the following JSON sample:

```
{
    "name": "Project Activity",
    "pageSize": "10",
    "nextTableName": "SM_ACTIVITY",
    "tables": [
        {
            "tableName": "SM_ACTIVITY",
            "columns": [
                "ACTIVITY_NAME",
                "ACTIVITY_CODE",
                "PROJ_ID",
                "STATUS"
            ],
            "condition": {
                "operator": "AND",
                "conditions": [
                    {
                        "columnName": "STATUS",
                        "operator": "EQUALS",
                        "value1": "IN_PROGRESS"
                    },
                    {
                        "columnName": "PROJ_ID",
                        "operator": "IN",
                        "val-ue1": "(SELECT PROJ_ID FROM CO_PROJECT
PROJ, WS_WORKSPACE WS WHERE PROJ.WORKSPACE_ID=WS.WORKSPACE_ID AND
WS.NAME='OICWS')"
                    } ]            }
        } ]
}
```

c. On the **Response** page:

i. From the **Select the response payload format** list, select **JSON sample**, and then enter the following JSON sample:

```
{
    "data": {
        "SM_ACTIVITY": [
            {
                "ACTIVITY_NAME": "Act1",
                "ACTIVITY_CODE": "A1000",
                "PROJ_ID": "124101",
                "STATUS": "IN_PROGRESS
            }
        ],
        "pagination": [
            {
                "nextTableName": "-1",
                "queryName": "Inprogress Activities",
                "nextKey": "0",
                "rowCount": "8",
                "sinceDate": "20230720T142903.000Z"
            }
        ],
        "safetyDate": [
            {
                "queryName": "Inprogress Activities",
                "sinceDate": "20230720T142903.000Z"
            }
        ],
        "tableList": [
            {
                "rowCount": "2",
                "tableName": "SM_ACTIVITY"
            }
        ]
    }
}
```

d. On the **Summary** page, view the summary and click **Finish**.

4. Click **Save**.

Map Data to Fetch In-Progress Activities

One of the key tasks to your integration is defining how data is transferred, or mapped, between the source and target projects and workspaces.

Mapper in Oracle Integration enables you to map element nodes between applications by dragging source element nodes onto target element nodes. When you open the mapper for a request or response message in an integration, the data structures are automatically populated with the information pulled from the source and target connections. You can expand and load data structure levels on demand to display additional levels. There is no limit on the levels of display.

To map data to get the source project by the workspace code and project code:

1. On the integration page, select **Map (runQuerys)**, click **Actions** ..., and then select **Edit**.

   On the mapper, you can create the mappings either by linking **Source** fields to **Target** fields or by entering values directly on the **Target** section using an expression builder.

2. In the **Target** section, do the following:

   a. Right-click the **Name** field and select **Create Target Node**, then in the Expression Builder window, enter `InProgress Activities` and click **Save** ✓.

   b. Right-click the **Page Size** field and select **Create Target Node**, then in the Expression Builder window, enter `100` and click **Save** ✓.

   c. Right-click the **Next Table Name** field and select **Create Target Node**, then in the Expression Builder window, enter `SM_ACTIVITY` and click **Save** ✓.

   d. Expand **Tables**, right-click the **Table Name** field and select **Create Target Node**, then in the **Developer Toolbar** panel, enter `SM_ACTIVITY` and then click **Save** ✓.

   e. Expand **Columns** under **Tables**, right-click and select **Create Target Node**, then in the **Developer Toolbar** panel, enter `ACTIVITY_NAME` and click **Save** ✓.

      Previously, in the Request Wrapper, you requested the columns in the following format; therefore, you need to add `ACTIVITY_CODE`, `PROJ_ID`, and `STATUS` to the columns field.

      ```
      "tableName": "SM_ACTIVITY",
                  "columns":
                   [
                      "ACTIVITY_NAME",
                      "ACTIVITY_CODE",
                      "PROJ_ID",
                      "STATUS"
                   ]
      ```

   f. Right-click **Columns** and select **Repeat Node**. A **2 of 2 Columns** node is created and the Expression Builder window opens. In the Expression Builder, enter `ACTIVITY_CODE` and click **Save** ✓.

   g. Right-click the **2 of 2 Columns** node and select **Repeat Node**. A **3 of 3 Columns** node is created and the Expression Builder window opens. In the Expression Builder, enter `PROJ_ID` and click **Save** ✓.

   h. Right-click the **3 of 3 Columns** node and select **Repeat Node**. A **4 of 4 Columns node** is created and the Expression Builder window opens. In the Expression Builder, enter `STATUS` and click **Save** ✓.

   i. Expand **Condition**, and then right-click **Operator** and select **Create Target Node**. In the Expression Builder, enter `AND` and click **Save** ✓.

   j. Expand **Condition**, and then **Conditions**. Add the following mappings for the child nodes under the **Conditions** node:

i. For **Column Name**, add `STATUS`.

ii. For **Operator**, add `EQUALS`.

iii. For **Value1**, add `InProgress`.

k. Right-click **Conditions**, and then select **Repeat Node**. A **2 of 2 Conditions** node is created with three child nodes, **Column Name**, **Operator**, and **Value1**. Add the following mappings for the child nodes under the **2 of 2 Conditions** node:

- For **Column Name**, add `PROJ_ID`.
- For **Operator**, add `IN`.
- For **Value1**, add `"(SELECT PROJ_ID FROM CO_PROJECT PROJ, WS_WORKSPACE WS WHERE PROJ.WORKSPACE_ID=WS.WORKSPACE_ID AND WS.NAME='OICWS')"`. Replace `WS.NAME='OICWS'` with your workspace name.

3. Click **Save** , then click **Validate**.

4. Click **Go Back** , then click **Save**.

Add an Invoke for FTP Write to File

Add another invoke to the integration for the File Transfer Protocol (FTP) connection.

To add the invoke:

1. On the integration page, hover over an arrow and click **Add** .
2. From the **Available connections** list, select **SFTP Connection**.
3. In the Adapter Endpoint Configuration Wizard, do the following:
   a. On the **Basic Info** page:
      i. In the **What do you want to call your endpoint?** field, enter a name for the connection.
      ii. In the **What does this endpoint do?** field, enter an optional description.
   b. On the **Configure Operations** page:
      i. In the **Output Directory** field, enter `OICIntegrations`.
      ii. In the **File Name Pattern** field, enter `ActivitiesInformation.json`.
   c. On the **Configure Schema** page, from the **Which one of the following choices would be used to describe the structure of the file contents?** list, select **Sample JSON document**.
   d. On the **File Contents - Definition** page, upload a JSON file with the following JSON code:

```
{
    "data": {
        "SM_ACTIVITY": [
            {
                "ACTIVITY_NAME": "Act1",
                "ACTIVITY_CODE": "A1000",
                "PROJ_ID": "124101",
                "STATUS": "IN_PROGRESS"
```

```
            },
            {
                "ACTIVITY_NAME": "Act2",
                "ACTIVITY_CODE": "A1010",
                "PROJ_ID": "124101",
                "STATUS": "IN_PROGRESS"
            }
        ],
        "pagination": [
            {
                "nextTableName": "-1",
                "queryName": "Inprogress Activities",
                "nextKey": "0",
                "rowCount": "8",
                "sinceDate": "20230720T142903.000Z"
            }
        ],
        "safetyDate": [
            {
                "queryName": "Inprogress Activities",
                "sinceDate": "20230720T142903.000Z"
            }
        ],
        "tableList": [
            {
                "rowCount": "2",
                "tableName": "SM_ACTIVITY"
            }
        ]
    }
}
```


e. On the **Summary** page, view the summary and click **Finish**.

4. Click **Save**.

Map Data to Write to File

Map data to Write to File to specify the source and destination of the data the integration fetches.

To map data to write to file:

1. On the integration page, select **Map (WriteToFile)**, click **Actions** ..., and then select **Edit**.

   On the mapper, you can create the mappings either by linking **Source** fields to **Target** fields or by entering values directly on the **Target** section using an expression builder.

2. In the **Sources** section, expand **runQuery Response (REST)**, then **Execute Response**, then **Response Wrapper**, then **Data**, then **SM ACTIVITY**.

3. In the **Target** section, expand **Outbound FTP Header Type**, then **Data**, then **SM ACTIVITY**.

4. Map all fields between the source **SM ACTIVITY** and target **SM ACTIVITY** tables.

5. Click **Save** ✓, then click **Validate**.

6. Click **Go Back** , then click **Save**.

5

Activate and Run the Integration

After you configure the connections and your integration design is complete, you can activate and run the integration.

1. In the navigation pane, click **Design**, then **Integrations**.
2. Activate the integration:
   a. Hover over the integration, then click **Activate** .
   b. In the **Activate Integration** panel, choose the appropriate level of tracing, then click **Activate**.

      A message confirms that the integration has been activated. Refresh the page to view the updated status of the integration.
3. Run the integration:
   - Hover over the integration, click **Actions** , then select **Run**.
4. After a successful run, connect to the SFTP server through Winscp and locate the file `ActivitiesInformation.json` in the folder path specified in the FTP connection.

Related Topics

- Activating and Deactivating Integrations