Oracle® Cloud

Using the REST Adapter

Release 16.4

E66630-02

December 2016

ORACLE

Oracle Cloud Using the REST Adapter, Release 16.4

E66630-02

Primary Author: Mark Kennedy

Contents

Preface

Using the REST Adapter describes how to configure the REST Adapter as a connection in an integration in Oracle Integration Cloud Service.

Topics:

Audience

Using the REST Adapter is intended for developers who want to use the REST Adapter in integrations in Oracle Integration Cloud Service.

Related Resources

For more information, see these Oracle resources:

- Oracle Cloud

  `http://cloud.oracle.com`
- *Using Oracle Integration Cloud Service*
- *Using the Oracle Mapper*

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

Getting Started with the REST Adapter

Review the following conceptual topics to learn about the REST Adapter and how to use it as a connection in integrations in Oracle Integration Cloud Service. A typical workflow of adapter and integration tasks is also provided.

Topics

REST Adapter Capabilities

The REST Adapter enables you to expose an Oracle Integration Cloud Service integration flow as a REST service and to invoke an external REST application.

The REST Adapter provides the following benefits:

- Trigger: When configured as a trigger, it acts as a generic inbound REST Adapter for exposing an Integration Cloud Service integration flow as a REST resource. A client can connect to the integration using a REST endpoint.
- Invoke: When configured as an invoke, it acts as a generic outbound REST Adapter for connecting to any external REST-exposed SaaS application.

The message in the integration flow is always in XML format. Messages sent to Integration Cloud Service through the REST Adapter in JSON or URL-encoded format are converted to XML. If messages sent from Integration Cloud Service through the REST Adapter are in XML message format, they can be converted to JSON or URL-encoded format, depending on the configured media type.

The REST Adapter provides design-time support for REST metadata catalog-compliant REST APIs. There is also top-level array support in JSON documents.

The REST Adapter provides the following additional features:

- Configuration Parameters
- Standard and Custom Header Support

- Message Exchange Patterns
- Security Roles
- Extensibility Support for Multiple OAuth Providers
- Role-Based Connections
- Cross-Origin Resource Sharing (CORS)
- Swagger and RAML Payload Support
- Multipart Attachment Support for Trigger and Invoke Connections
- Transport Layer Security (TLS) Version Support
- Homogenous Multidimensional Array Support in JSON Documents
- On-Premises REST API Support with the Agent
- View the Inbound Invocation as a Swagger Format

Configuration Parameters

The REST Adapter is not configured from an existing WSDL or WADL file. Instead, you configure the following parameters using the REST endpoint wizard to expose or consume the REST service:

- Relative resource path URI
- HTTP method (actions) to perform
- Template and query parameters
- Request/response message structure

Standard and Custom Header Support

The REST adapter supports standard and custom HTTP request and response headers in the invoke and trigger directions.

- Outbound (Invoke) direction

  HTTP headers enable you to use an outbound invocation to specify header properties. Many REST APIs expect certain properties to be specified in the HTTP headers (similar to SOAP APIs where you can specify header properties such as the WS address). Use the standard HTTP headers to specify these properties. You can also use the custom HTTP headers to specify properties. The REST APIs can expect the client application to pass properties in the custom headers, which can influence the behavior of the APIs. You can map the header properties in the Oracle Mapper.

- Inbound (trigger) direction

  You can expose integration flows as REST endpoints and enable client applications to access the properties in the standard and custom headers. You can use these properties to create routing expressions in your integrations. You can map the header properties in the Oracle Mapper. For more information about creating routing expressions, see Creating Routing Paths for Two Different Invoke Endpoints in Integrations and Creating an Orchestrated Integration.

Note:

- If you want to send multiple values of a header, use a comma separated value (CSV) file. This is considered as one header and one value that consists of:

  ```
  val1 comma val2 comma val3 ...
  ```

  The same value is propagated across the mapper and then to the outbound service. The outbound service must then interpret the CSVs of the header to be used as multiple values.

- You cannot store multiple headers with the same name. The WSDL can only store one element with one unique name.

Message Exchange Patterns

The REST Adapter supports the following message exchange patterns:

- Synchronous request and response patterns
- Synchronous one-way request patterns
- Asynchronous inbound patterns

  With this pattern, the REST service takes a request and immediately returns a blank response with a status of `202` (accepted). This is a fire request and forget scenario. Any PUT, POST, and DELETE operations without a response payload or a response header are assumed to be asynchronous. Asynchronous GET operations are not applicable with this message exchange pattern. GET endpoints are treated as synchronous and the generated artifacts remain unchanged. The summary page of the Adapter Endpoint Configuration Wizard indicates that you configured a one-way operation. For new integrations, this is the default behavior.

Security Roles

The REST Adapter supports the following security policies:

- Basic Authentication
- OAuth Client Credentials (two-legged flow)
- OAuth Resource Owner Password Credentials (two-legged flow)
- OAuth Authorization Code Credentials (three-legged flow)
- OAuth Custom Three Legged Flow
- OAuth Custom Two Legged Flow

See Configuring Connection Security for more information about these security policies.

Extensibility Support for Multiple OAuth Providers

You can use the extensibility framework of the REST Adapter to access the OAuth-protected resource of endpoints. This framework enables you to access endpoints that have implemented their own variations of OAuth.

The OAuth standard provides flexibility for endpoints to define specific aspects of their OAuth flows. For example:

- Create their own properties.
- Decide when to use these properties in an OAuth flow. For example, some custom properties may be required with the authorization request, while others may be required for the access token request or for the refresh of the access token after its expiration.
- Decide how to pass these properties in an OAuth flow. For example, whether a property is passed as a header, query parameter, or payload.

To address these challenges, Oracle Integration Cloud Service provides two custom security policies that enable you to specify each step in the OAuth flow when you create the REST Adapter connection:

- OAuth custom two-legged flow: The client application directly interacts with the authorization server on behalf of a resource owner.
- OAuth custom three-legged flow: The client application redirects the owner to a separate resource URL where the resource owner authenticates and provides consent for the flow to continue.

This enables you to adapt to most OAuth framework scenarios and integrate with many third-party applications without writing additional code.

- During design-time, the access token is obtained, validated, and stored in the CSF. The security token is also stored in the CSF.
- During runtime, the access token is retrieved, applied, and managed. A valid access token is applied to the request before invoking the REST endpoint.

For information about specifying the OAuth custom two-legged flow and three-legged flow security policies, see Configuring Connection Security.

Note: This extensibility feature is an advanced feature, and not for business users. Users of this feature should use a tool such as postman to configure the necessary properties.

Role-Based Connections

The REST adapter is bidirectional. You can configure the REST Adapter depending on the context in which you want to use the connection.

- Trigger: The REST Adapter is used to create a REST endpoint to trigger an integration. You select **Trigger** from the **Connection Role** list on the New Connection - Information dialog. When configured as a trigger, a base URI is not required. The security policy defined in the inbound direction accepts credentials configured in the identity domain. Therefore, you are not required to provide the applicable credentials. When configuring security on the Connections page, you only provide the security policy that must be attached to the inbound endpoint. Basic authentication is the only security policy available. Agent configuration is not applicable on a connection with the trigger role.
- Invoke: The REST Adapter is used to invoke external REST endpoints. A base URI and security configuration for accessing external protected resources are required. You are prompted for these additional details on the Connections page. You cannot use an invoke connection on the trigger side.

- Trigger and invoke: The REST Adapter is used in both the trigger and invoke directions of an integration. This connection requires invoke and trigger values.

Cross-Origin Resource Sharing (CORS)

CORS defines a way in which a browser and server can interact to determine safely whether or not to allow the cross-origin request. CORS provides for more flexibility than same-origin requests, but is more secure than simply permitting all cross-origin requests.

CORS is supported by browsers based on the following layout engines:

- Blink- and Chromium-based browsers (Chrome 28, Opera 15, Amazon Silk, Android's 4.4+ WebView, and Qt's WebEngine).
- Gecko 1.9.1 (Firefox 3.5, SeaMonkey 2.0, and Camino 2.1) and above.
- MSHTML/Trident 6.0 (Internet Explorer 10) has native support. MSHTML/Trident 4.0 & 5.0 (Internet Explorer 8 & 9) provide partial support through the XDomainRequest object.
- Presto-based browsers (Opera) implement CORS as of Opera 12.00 and Opera Mobile 12, but not Opera Mini.
- WebKit (Safari 4 and above, Google Chrome 3 and above, possibly earlier).

The following browsers do not support CORS:

- Camino does not implement CORS in the 2.0.x release series because these versions are based on Gecko 1.9.0
- As of version 0.10.2, Arora exposes WebKit's CORS-related APIs, but attempted cross-origin requests fail.[16]

Supports cross origin resource sharing (CORS) in the inbound direction.

Swagger and RAML Payload Support

The REST Adapter provides support for the following payload types:

- RESTful API Modeling Language (RAML): A language for describing RESTful APIs. RAML provides the information necessary to describe RESTful or practically-RESTful APIs (APIs that do not obey all REST constraints).
- Swagger: A specification for describing, producing, consuming, and visualizing RESTful web services.

The following example shows a Swagger 2.0 file. This file contains two main resources:

- `/Book`. This resource contains `get` and `post` methods and `/Book/{id}`, `/Book/hello`, and `/Book/search` subresources.
- `/Author`. This resource contains a `get` method and an `/Author/{id}` subresource.

When configuring an invoke (outbound) REST Adapter in the Adapter Endpoint Configuration Wizard, the resources and subresources are displayed for selection as business objects and the methods are displayed for selection as operations to perform on the business objects.

When creating the REST Adapter connection, you select **Swagger Definition URL** in the **Connection Type** field and specify the URL in the **Connection URL** field of the Connection Properties dialog.

```
{
  "swagger" : "2.0",
  "info" : {
    "version" : "1.0",
    "title" : "RestServiceForBooks"
  },

  "host" : "host_name:8080",
  "basePath" : "/Test/rest",
  "schemes" : ["http"],
  "paths" : {
    "/Book" : {
      "get" : {
        "operationId" : "getBooks",
        "description" : "Returns all the available books in teh store",
        "produces" : [ "application/xml", "application/json" ],
        "responses" : {
          "default" : {
            "schema" : {
              "$ref" : "#/definitions/Books"
            }
          }
        }

      },
      "post" : {
        "operationId" : "postBook",
        "description" : "Creates a new book item",
        "produces" : [ "application/xml", "application/json" ],
        "consumes" : [ "application/xml", "application/json" ],
        "parameters" : [
          {
            "name" : "Book",
            "in" : "body",
            "required" : true,
            "schema" : { "$ref" : "#/definitions/Book" }
          }
        ],
        "responses" : {
          "default" : {
            "schema" : { "$ref" : "#/definitions/Book" }
          }
        }
      }
    },
    "/Book/{id}" : {
      "get" : {
        "operationId" : "getSingleBook",
        "description" : "Returns a book with specific id",
        "produces" : [ "application/xml", "application/json" ],
        "parameters" : [
          {
            "name": "id",
            "in": "path",
            "required" : true,
            "type" : "string"
          }
```

```
    ],
    "responses" : {
      "default" : {
        "schema" : { "$ref" : "#/definitions/Book" }
      }
    }
  }
},
"/Book/hello" : {
  "get" : {
    "operationId" : "sayHelloToBook",
    "description" : "says hello to a book",
    "produces" : [ "application/xml", "application/json" ],
    "responses" : {
      "default" : {
        "schema" : { "type" : "string" }
      }
    }
  }
},
"/Book/search" : {
  "get" : {
    "operationId" : "searchBook",
    "description" : "Returns a list of books that match query param",
    "produces" : [ "application/xml", "application/json" ],
    "parameters" : [
      {
        "name": "name",
        "in": "query",
        "required" : false,
        "type" : "string"
      }
    ],
    "responses" : {
      "default" : {
        "schema" : {
          "$ref" : "#/definitions/Books"
        }
      }
    }
  }
},
"/Author" : {
  "get" : {
    "operationId" : "getAuthors",
    "description": "Returns a list of authors",
    "produces": [
      "application/xml",
      "application/json"
    ],
    "responses": {
      "default": {
        "schema": {
          "$ref" : "#/definitions/Authors"
        }
      }
    }
  }
},
"/Author/{id}" : {
  "get" : {
```

```
      "operationId" : "getSingleAuthor",
      "description" : "Returns a Author with specific id",
      "produces" : [ "application/xml", "application/json" ],
      "parameters" : [
        {
          "name": "id",
          "in": "path",
          "required" : true,
          "type" : "string"
        }
      ],
      "responses" : {
        "default" : {
          "schema" : { "$ref" : "#/definitions/Author" }
        }
      }
    }
  }
},
"definitions" : {
  "Author" : {
    "type" : "object",
    "properties" : {
      "id" : { "type" : "string" },
      "firstName" : { "type"  : "string"},
      "lastName" : { "type" : "string" }
    },
    "required" : [ "id", "firstName", "lastName"]
  },
  "Authors" : {
    "type" : "object",
    "properties" : {
      "items" : {
        "type" : "array",
        "items" : {
          "$ref" : "#/definitions/Author"
        }
      }
    }
  },
  "Publisher" : {
    "type" : "object",
    "properties" : {
      "id" : { "type" : "string" },
      "name" : { "type"  : "string"},
      "location" : { "type" : "string" }
    },
    "required" : [ "id", "name", "location"]
  },
  "Publishers" : {
    "type" : "object",
    "properties" : {
      "items" : {
        "type" : "array",
        "items" : {
          "$ref" : "#/definitions/Publisher"
        }
      }
    }
  },
  "Book" : {
```

```
      "type" : "object",
      "properties" : {
        "id" : { "type" : "string" },
        "name" : { "type" : "string" },
        "ISBN" : { "type" : "string" },
        "price" : { "type" : "integer" },
        "author" : { "type" : "array", "items" :{ "$ref" : "#/definitions/
Author" } },
        "publisher" : { "$ref" : "#/definitions/Publisher"  }
      },
      "required" : ["id", "name", "ISBN", "price", "author", "publisher" ]
    },
    "Books" : {
      "type" : "object",
      "properties" : {
        "items" : {
          "type" : "array",
          "items" : {
            "$ref" : "#/definitions/Book"
          }
        }
      }
    }
  }
}
```

The following example shows a RAML file. The file contains the schemas that use the service. This file contains two main resources:

- `/Author`. This resource contains a `get` method and an `/Author/{id}` subresource.
- `/Book`. This resource contains `get` and `post` methods and `/Book/{id}` and `/Book/search` subresources.

When configuring an invoke (outbound) REST Adapter in the Adapter Endpoint Configuration Wizard, the resources and subresources are displayed for selection as business objects and the methods are displayed for selection as operations to perform on the business objects.

When creating your REST Adapter connection, you select **RAML Definition URL** in the **Connection Type** field and specify the URL in the **Connection URL** field of the Connection Properties dialog.

```
 #%RAML 0.8
title: API for Books
version: v1
baseUri: "http://host_name:8080/Test/rest"
protocols: [ HTTP ]
schemas:
 - authors-jsonschema: |
    {
      "$schema" : "http://json-schema.org/draft-03/schema",
       "type":"object",
       "properties":{
          "items":{
           "type":"array",
           "items":{
                 "type":"object",
                 "properties":{
                    "id":{
```

```
                        "type":"string"
                     },
                     "firstName":{
                        "type":"string"
                     },
                     "lastName":{
                        "type":"string"
                     }
                  },
                  "required":[
                     "id",
                     "firstName",
                     "lastName"
                  ]
               }
            }
         }
      }
   - author-jsonschema: |
      {
         "$schema":"http://json-schema.org/draft-03/schema",

               "type":"object",
               "properties":{
                  "id":{
                     "type":"string"
                  },
                  "firstName":{
                     "type":"string"
                  },
                  "lastName":{
                     "type":"string"
                  }
               },
               "required":[
                  "id",
                  "firstName",
                  "lastName"
               ]
      }

   - books-jsonschema: |
      {
         "$schema":"http://json-schema.org/draft-03/schema",

               "type":"object",
               "properties":{
                  "items":{
                     "type":"array",
                     "items":{
                           "type":"object",
                           "properties":{
                              "id":{
                                 "type":"string"
                              },
                              "name":{
                                 "type":"string"
                              },
                              "ISBN":{
                                 "type":"string"
                              },
```

```
                        "price":{
                           "type":"integer"
                        },
                        "author":{
                           "type":"array",
                           "items":{
                              "type":"object",
                              "properties":{
                                 "id":{
                                    "type":"string"
                                 },
                                 "firstName":{
                                    "type":"string"
                                 },
                                 "lastName":{
                                    "type":"string"
                                 }
                              },
                              "required":[
                                 "id",
                                 "firstName",
                                 "lastName"
                              ]
                           }
                        },
                        "publisher":{
                           "type":"object",
                           "properties":{
                              "id":{
                                 "type":"string"
                              },
                              "name":{
                                 "type":"string"
                              },
                              "location":{
                                 "type":"string"
                              }
                           },
                           "required":[
                              "id",
                              "name",
                              "location"
                           ]
                        }
                     },
                     "required":[
                        "id",
                        "name",
                        "ISBN",
                        "price",
                        "author",
                        "publisher"
                     ]
                  }
               }
            }

   }
- book-jsonschema: |
   {
      "$schema":"http://json-schema.org/draft-03/schema",
```

```
"type":"object",
"properties":{
   "id":{
      "type":"string"
   },
   "name":{
      "type":"string"
   },
   "ISBN":{
      "type":"string"
   },
   "price":{
      "type":"integer"
   },
   "author":{
      "type":"array",
      "items":{
         "type":"object",
         "properties":{
            "id":{
               "type":"string"
            },
            "firstName":{
               "type":"string"
            },
            "lastName":{
               "type":"string"
            }
         },
         "required":[
            "id",
            "firstName",
            "lastName"
         ]
      }
   },
   "publisher":{
      "type":"object",
      "properties":{
         "id":{
            "type":"string"
         },
         "name":{
            "type":"string"
         },
         "location":{
            "type":"string"
         }
      },
      "required":[
         "id",
         "name",
         "location"
      ]
   }
},
"required":[
   "id",
   "name",
   "ISBN",
   "price",
```

```
              "author",
              "publisher"
            ]
    }
/Author:
  get:
    responses:
      200:
        body:
          application/xml:
            schema: authors-jsonschema
            example: |
              <?xml version="1.0" encoding="UTF-8"?>
              <authors></authors>
          application/json:
            schema: authors-jsonschema
            example: |
              {
                "authors" : ""
              }
  /{id}:
    get:
      responses:
        200:
          body:
            application/xml:
              schema: author-jsonschema
              example: |
                <?xml version="1.0" encoding="UTF-8"?>
                <author></author>
            application/json:
              schema: author-jsonschema
              example: |
                {
                  "author" : ""
                }
/Book:
  post:
    body:
      application/xml:
        schema: book-jsonschema
      application/json:
        schema: book-jsonschema
    responses:
      200:
        body:
          application/xml:
            schema: book-jsonschema
            example: |
              <?xml version="1.0" encoding="UTF-8"?>
              <book>
                <price></price>
              </book>
          application/json:
            schema: book-jsonschema
            example: |
              {
                "book" : {
                  "price" : ""
                }
              }
```

```
get:
  responses:
    200:
      body:
        application/xml:
          schema: books-jsonschema
          example: |
            <?xml version="1.0" encoding="UTF-8"?>
            <book>
              <price></price>
            </book>
        application/json:
          schema: books-jsonschema
          example: |
            {
              "book" : {
                "price" : ""
              }
            }
/search:
  get:
    queryParameters:
      name:
    responses:
      200:
        body:
          application/xml:
            schema: books-jsonschema
            example: |
              <?xml version="1.0" encoding="UTF-8"?>
              <book>
                <price></price>
              </book>
          application/json:
            schema: books-jsonschema
            example: |
              {
                "book" : {
                  "price" : ""
                }
              }
/{id}:
  get:
    responses:
      200:
        body:
          application/xml:
            schema: book-jsonschema
            example: |
              <?xml version="1.0" encoding="UTF-8"?>
              <book>
                <price></price>
              </book>
          application/json:
            schema: book-jsonschema
            example: |
              {
                "book" : {
                  "price" : ""
                }
              }
```

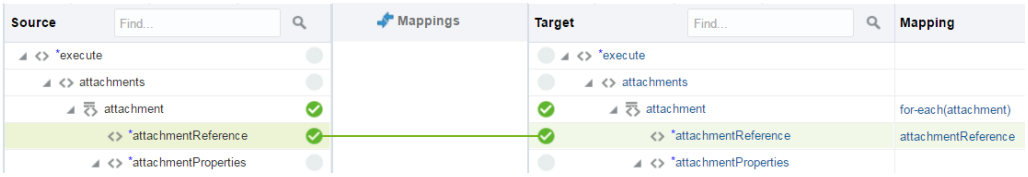
Multipart Attachment Support for Trigger and Invoke Connections

The REST Adapter supports the following multipart attachments for trigger (inbound) requests and invoke (outbound) requests. You can send the following media types with attachments to your endpoint:

- Multipart/mixed: Send XML or JSON payload with attachments.
- Multipart/form-data: Send XML or JSON payload with attachments.
- Multipart/form-data: Send an HTML form payload type, with the rest of the body consisting of attachments.

For example, you can send a review document attachment with the trigger (inbound) REST Adapter to an invoke (outbound) Adobe eSign Adapter or DocuSign Adapter for delivery to the downstream endpoint for signing.

If you want to send attachments from inbound to outbound (in request messages) or to download attachments from outbound to inbound (in response messages), then for each attachment you must map the `attachmentReference` from source to target in the mapper.



If you do not map `attachmentReference` in the mapper for a request, the outbound REST Adapter does not receive attachments from the inbound direction (multipart request). Similarly, if you do not map `attachmentReference` in the mapper for a response, the inbound REST Adapter does not receive attachments from the outbound REST Adapter (multipart response).

Transport Layer Security (TLS) Version Support

TLS version support of the target server in provided. The TLS protocol provides privacy and data integrity between two communicating computer applications. For more information, see Configuring Connection Properties.

Homogenous Multidimensional Array Support in JSON Documents

You can select a JSON sample with homogenous multidimensional arrays when configuring the REST Adapter in the Adapter Configuration Wizard.

All JSON messages must be converted to XML before they can be processed by Oracle Integration Cloud Service at runtime. Semantically, there is no equivalent of multidimensional arrays in XML. To support multidimensional arrays, intermediate XML elements are generated that denote the beginning and ending of a nested array. When receiving a JSON message containing multidimensional arrays, these reserved elements are injected into the generated XML to denote the beginning and ending of a nested array. While converting XML elements back into JSON, the injected elements are converted into JSON with nested arrays.

The following JSON document consists of a multidimensional array (`@ref "rercordsData”`).

```
{
    "studentData": {
        "fieldNames": [ "id","mobile_number" ],
```

```
        "recordsData": [ ["21","23"], ["+91123456789", "+91987654321" ]  ],
        "name": "jack"
    },
    "schoolData": {
        "Name": "ABCInternations",
        "StudentNumbers": 1300,
        "Address": "YYY streets Sector-44 India"
    }
}
```

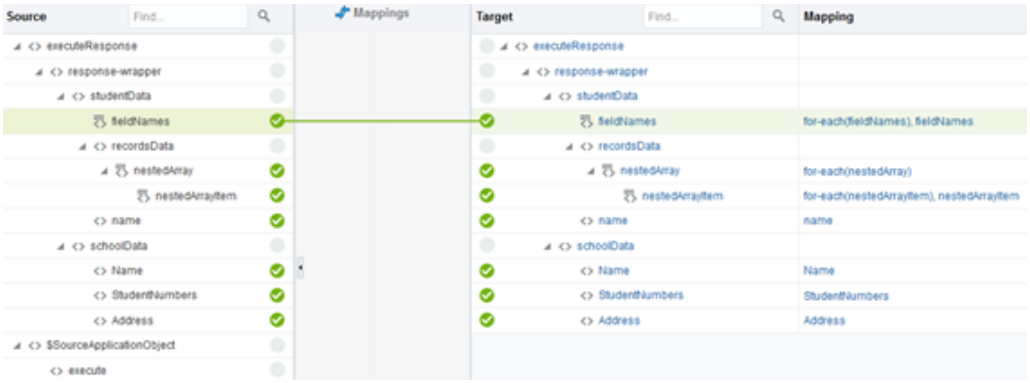
The sample generated schema XML for the JSON document looks as follows:

```
<?xml version = '1.0' encoding = 'UTF-8'?>

<ns0:executeResponse xmlns:ns1="http://xmlns.oracle.com/cloud/adapter/REST/test/
types"
xmlns:ns0="http://xmlns.oracle.com/cloud/adapter/REST/test_REQUEST/types">

<ns1:response-wrapper>
  <ns1:studentData>
    <ns1:fieldNames>id</ns1:fieldNames>
    <ns1:fieldNames>mobile_number</ns1:fieldNames>
    <ns1:recordsData>
         <ns1:nestedArray>
                <ns1:nestedArrayItem>21</ns1:nestedArrayItem>
                <ns1:nestedArrayItem>23</ns1:nestedArrayItem>
         </ns1:nestedArray>
                        <ns1:nestedArray>
                                        <ns1:nestedArrayItem>+91123456789</
ns1:nestedArrayItem>
                                        <ns1:nestedArrayItem>+91987654321</
ns1:nestedArrayItem>
                        </ns1:nestedArray>
         </ns1:recordsData>
    <ns1:name>jack</ns1:name>
  </ns1:studentData>
  <ns1:schoolData>
        <ns1:Name>ABCInternations</ns1:Name>
        <ns1:StudentNumbers>1300</ns1:StudentNumbers>
        <ns1:Address>YYY streets Sector-44 India</ns1:Address>
   </ns1:schoolData>
</ns1:response-wrapper>
</ns0:executeResponse>
```

Elements in the nested array appear as `nestedArray` in the mapper and items in the elements appear as `nestedArrayItem`. You must map `nestedArray` as a `for-each` statement and `nestedArrayItem` as a `for-each` statement.



On-Premises REST API Support with the Agent

Oracle Integration Cloud Service provides an agent framework that enables you to create integrations and exchange messages between on-premises applications and Oracle Integration Cloud Service. You can integrate on-premises REST APIs with Oracle Integration Cloud Service through use of the on-premises agent. Once you create an agent group and install the on-premises agent, you can create and configure a REST Adapter connection as follows:

- Select **REST API Base URL** or **Swagger Definition URL** from the **Connection Type** list and enter the appropriate URL in the **Connection URL** field of the Connection Properties dialog. No other connection types are supported.
- Select **Basic Authentication** or **No Security Policy** from the **Security Policy** list of the Credentials dialog. No other security policies are supported.
- Select the previously-created agent group in the Select an Agent Group dialog.

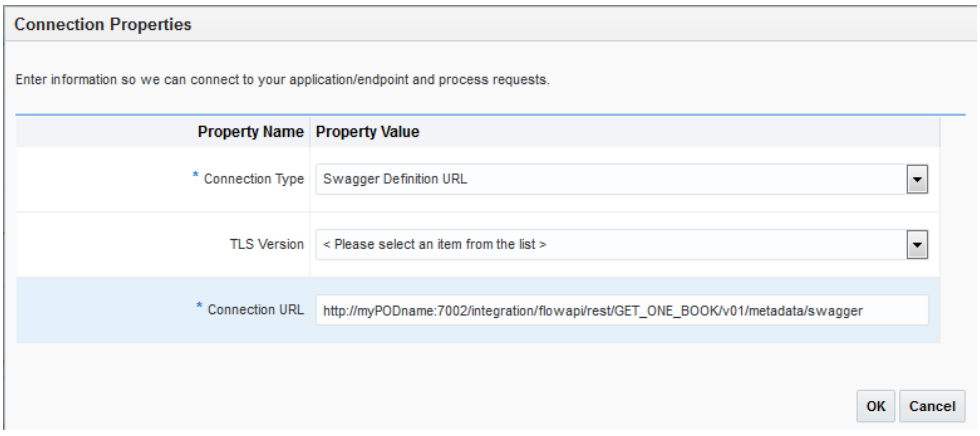
For conceptual information about the on-premises agent, see About Agents and Integrations Between On-Premises Applications and Oracle Integration Cloud Service. For information about creating an agent group and installing the on-premises agent, see Managing Agent Groups and the On-Premises Agent.

View the Inbound Invocation as a Swagger Format

You can view the metadata of an activated REST integration and then append `/swagger` to the metadata URL to view the swagger format for the integration. The inbound REST integration can then be exposed as a swagger connection.

1. On the Integrations page, find the integration whose endpoint URL you want to use.
2. Click the **Details** icon at the far right.
3. Click the **Endpoint URL** value (for example, `http://myPODname:7002/integration/flowapi/rest/GET_ONE_BOOK/v01/metadata`).
4. Append `/swagger` to the end of the URL, and press **Enter**.

   Appending `/swagger` to the URL generates a swagger file for the inbound integration. This URL can also be used to create a new swagger connection in the Connection Properties dialog. You enter the swagger URL in the **Connection URL** field and select **Swagger Definition URL** from the **Connection Type** field.



What Application Version Does the REST Adapter Support?

The REST Adapter supports OAuth 2.0.

About Oracle Integration Cloud Service

Oracle Integration Cloud Service is a complete, secure, but lightweight integration solution that enables you to connect your applications in the cloud. It simplifies connectivity between your applications and connects both your applications that live in the cloud and your applications that still live on premises. Oracle Integration Cloud Service provides secure, enterprise-grade connectivity regardless of the applications you are connecting or where they reside.

Oracle Integration Cloud Service provides native connectivity to Oracle Software as a Service (SaaS) applications, such as Oracle Sales Cloud, Oracle RightNow Cloud, and so on. Oracle Integration Cloud Service *adapters* simplify connectivity by handling the underlying complexities of connecting to applications using industry-wide best practices. You only need to create a *connection* that provides minimal connectivity information for each system. Oracle Integration Cloud Service *lookups* map the different codes or terms used by the applications you are integrating to describe similar items (such as country or gender codes). Finally, the visual data mapper enables you to quickly create direct mappings between the trigger and invoke data structures. From the mapper, you can also access lookup tables and use standard XPath functions to map data between your applications.

Once you integrate your applications and activate the integrations to the runtime environment, the dashboard displays information about the running integrations so you can monitor the status and processing statistics for each integration. The dashboard measures and tracks the performance of your transactions by capturing and reporting key information, such as throughput, the number of messages processed successfully, and the number of messages that failed processing. You can also manage business identifiers that track fields in messages and manage errors by integrations, connections, or specific integration instances.

About Oracle Integration Cloud Service Connections

Connections define information about the instances of each configuration you are integrating. Oracle Integration Cloud Service includes a set of predefined *adapters*, which are the types of applications on which you can base your connections, such as Oracle Sales Cloud, Oracle Eloqua Cloud, Oracle RightNow Cloud, and others. A connection is based on an adapter. A connection includes the additional information required by the adapter to communicate with a specific instance of an application (this can be referred to as metadata or as connection details). For example, to create a connection to a specific RightNow Cloud application instance, you must select the Oracle RightNow adapter and then specify the WSDL URL, security policy, and security credentials to connect to it.

 Video

About Oracle Integration Cloud Service Integrations

Integrations are the main ingredient of Oracle Integration Cloud Service. An integration includes at the least a trigger (source) connection (for requests sent to Oracle Integration Cloud Service) and invoke (target) connection (for requests sent

from Oracle Integration Cloud Service to the target) and the field mapping between those two connections.

When you create your integrations, you build on the connections you already created by defining how to process the data for the trigger (source) and invoke (target) connections. This can include defining the type of operations to perform on the data, the business objects and fields against which to perform those operations, required schemas, and so on. To make this easier, the most complex configuration tasks are handled by Oracle Integration Cloud Service. Once your trigger (source) and invoke (target) connections are configured, the mappers between the two are enabled so you can define how the information is transferred between the trigger (source) and invoke (target) data structures for both the request and response messages.

Video

Typical Workflow for Creating and Including an Adapter Connection in an Integration

You follow a very simple workflow to create a connection with an adapter and include the connection in an integration in Integration Cloud Service.

Step	Description	More Information
1	Create the adapter connections for the applications you want to integrate. The connections can be reused in multiple integrations and are typically created by the administrator.	Creating a REST Adapter Connection
2	Create the integration. When you do this, you add trigger and invoke connections to the integration.	Creating an Integration and Adding the REST Adapter Connection to an Integration
3	Map data between the trigger connection data structure and the invoke connection data structure.	Mapping Integration Cloud Service Data of *Using Oracle Integration Cloud Service*
4	(Optional) Create lookups that map the different values used by those applications to identify the same type of object (such as gender codes or country codes).	Creating Lookups of *Using Oracle Integration Cloud Service*
5	Activate the integration.	Managing Integrations of *Using Oracle Integration Cloud Service*
6	Monitor the integration on the dashboard.	Monitoring Integration Cloud Services of *Using Oracle Integration Cloud Service*
7	Track payload fields in messages during runtime.	Assigning Business Identifiers for Tracking Fields in Messages and Managing Business Identifiers for Tracking Fields in Messages of *Using Oracle Integration Cloud Service*

Step	Description	More Information
8	Manage errors at the integration level, connection level, or specific integration instance level.	Managing Errors of *Using Oracle Integration Cloud Service*

2

Creating a REST Adapter Connection

A connection is based on an adapter. You define connections to the specific cloud applications that you want to integrate. The following topics describe how to define connections.

Topics

Prerequisites for Creating a Connection

You must satisfy the following prerequisites to create a connection with the REST Adapter:

- If you are using one of the OAuth security policies, you must already have registered your client application to complete the necessary fields on the Connections page. The Basic Authentication and No Security Policy security policies are exempted.

  Before a client application can request access to resources on a resource server, the client application must first register with the authorization server associated with the resource server.

  The registration is typically a one-time task. Once registered, the registration remains valid, unless the client application registration is revoked.

  At registration time, the client application is assigned a client ID and a client secret (password) by the authorization server. The client ID and secret are unique to the client application on that authorization server. If a client application registers with multiple authorization servers (for example, Facebook, Twitter, and Google), each authorization server issues its own unique client ID to the client application.

  ```
  @ref: http://tutorials.jenkov.com/oauth2/authorization.html
  ```

  For OAuth configuration, read the provider documentation carefully and provide the relevant values.

- For SSL endpoints, obtain and upload a server certificate. For more information, see Uploading an SSL Certificate.

Uploading an SSL Certificate

Certificates are used to validate outbound SSL connections. If you make an SSL connection in which the root certificate does not exist in Oracle Integration Cloud Service, an exception is thrown. In that case, you must upload the appropriate certificate. A certificate enables Oracle Integration Cloud Service to connect with external services. If the external endpoint requires a specific certificate, request the certificate and then upload it into Oracle Integration Cloud Service.

To upload a certificate:

1. From the Oracle Integration Cloud Service home page, click the **Administration** tab in the upper right corner.

   All certificates currently uploaded to the trust store are displayed in the Certificates dialog. The **Filter By** > **Type** list displays the following details:

   - **Preinstalled**: Displays the certificates automatically installed in Oracle Integration Cloud Service. These certificates cannot be deleted.
   - **Uploaded**: Displays the certificates uploaded by individual users. These certificates can be deleted and updated.

   You can also search for certificates in the **Search** field. The search results are limited to a maximum of ten records sorted by name for performance and usability reasons. To ensure that your search results are more granular, enter as much of the certificate name as possible.

2. Click **Upload** at the top of the page.
3. In the Upload Certificate dialog box, enter a unique identifier for the certificate.

   This is a name you can use to identify the certificate.

4. Click **Browse** to locate the certificate file (`.cer`).
5. Click **Upload**.
6. Click the certificate name to view details such as the subject of the certificate, the issuer of the certificate, the date the certificate was issued, and the date the certificate expires.

Creating a Connection

The first step in creating an integration is to create the connections to the applications with which you want to share data.

1. In the Integration Cloud Service toolbar, click **Designer**.
2. On the Designer Portal, click **Connections**.
3. Click **New Connection**.

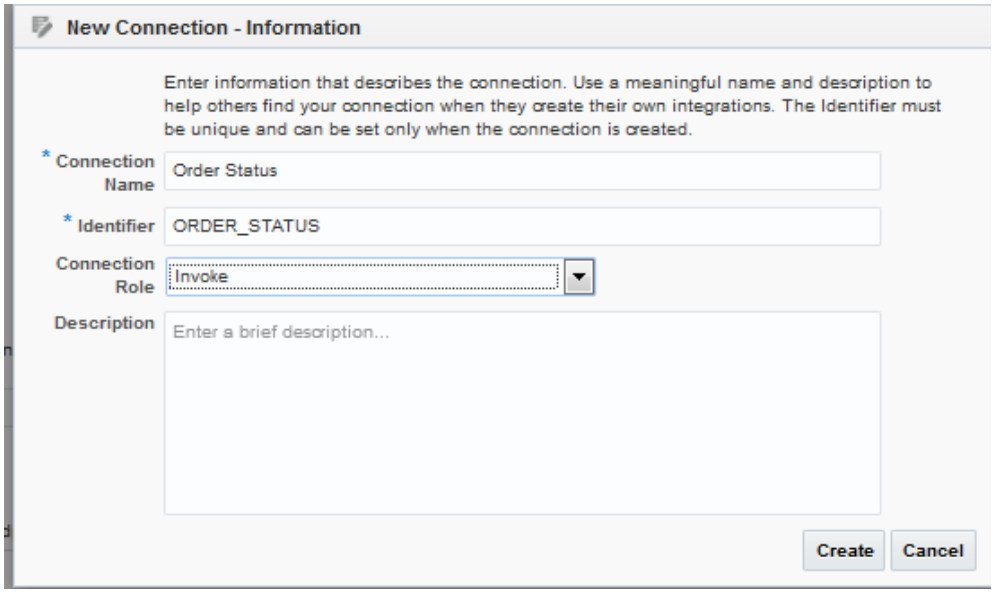
   The Create Connection — Select Adapter dialog is displayed.

4. Select an adapter from the dialog. You can also search for the type of adapter to use by entering a partial or full name in the Search field, and clicking **Search**.

   The New Connection — Information dialog is displayed.

5. Enter the information to describe the connection.

   - Enter a meaningful name to help others find your connection when they begin to create their own integrations. The name you enter is automatically added in capital letters to the **Identifier** field. If you modify the identifier name, do not include a blank space (for example, `OSC Inbound`).
   - Select the role (direction) in which to use this connection (trigger, invoke, or both). Only the roles supported by this adapter are displayed for selection. When you select a role, only the connection properties and security policies appropriate to that role are displayed on the Connections page. If you select an adapter that supports both invoke and trigger, but select only one of those roles, then try to drag the adapter into the section you did not select, you receive an error (for example, configure an Oracle RightNow Cloud Adapter as only an invoke, but drag the adapter to the trigger section).
   - Enter an optional description of the connection.



6. Click **Create**.

   Your connection is created and you are now ready to configure connection details, such as email contact, connection properties, security policies, and connection login credentials.

Adding a Contact Email

From the Connection Administrator section of the connection, you can add a contact email address for notifications.

1. In the **Email Address** field, enter an email address to receive email notifications when problems occur.
2. In the upper right corner, click **Save**.

Configuring Connection Properties

Enter connection information so your application can process requests.

1. Click **Configure Connectivity**.

   The Connection Properties dialog is displayed.

2. From the **TLS Version** list, optionally specify the Transport Layer Security (TLS) version of the target server. The TLS protocol provides privacy and data integrity between two communicating computer applications. If no version is selected, the REST Adapter uses TLSV1 by default. The selected version is used for SSL/TLS negotiation and SSL handshake in all outbound invocations of the REST API. Existing integrations and connections are not impacted.

   - **TLSv1**
   - **TLSv1.1**
   - **TLSv1.2**

3. From the **Connection Type** list, select the type to use:

   - **REST API Base URL**
   - **Metadata Catalog URL**
   - **Swagger Definition URL**
   - **RAML Definition URL**

4. In the **Connection URL** field, specify the endpoint URL to use based on your selection in Step 2. The connection URL can be both HTTP and HTTPS.

Type	Endpoint Example
REST API Base URL	`https://hostname:port/integration/flowapi/rest/RESTTORESTINTEGRATIONPUT_TEST/v01/`
Metadata Catalog URL	`https://hostname:port/Test/mdcatalogmain.json`
Swagger Definition URL	`https://hostname:port/Test/application.json`
RAML Definition URL	`https://hostname:port//Test/fullapi2.raml`

5. Click **OK**.

6. Configure connection security.

Configuring Connection Security

Configure security for your REST Adapter connection by selecting the security policy and security token.

1. Click **Configure Credentials**.
2. Select the security policy to use. Based on your selection, the page is referenced to display various login credential fields. You must already have created your client application to complete the necessary fields.

Selected Security Policy	Fields
Basic Authentication	• **Username** — The name of a user who has access to the destination web service. • **Password** — Enter the password. • **Confirm Password** — Reenter the password.
OAuth Client Credentials	• **Access Token URI** — The URL from which to obtain the access token. • **Client Id** — The client identifier issued to the client during the registration process. • **Client Secret** — The client secret. • **Confirm Client Secret** — Reenter the client secret. • **Scope** — The scope of the access request. Scopes enable you to specify which type of access you need. Scopes limit access for the OAuth token. They do not grant any additional permission beyond that which the user already possesses. • **Auth Request Media Type** — The format of the data you want to receive. This is an optional parameter that can be kept blank. For example, if you are invoking Twitter APIs, you do not need to select any type.
OAuth Resource Owner Password Credentials	• **Access Token URI** — The URL from which to obtain the access token. • **Client Id** — The client identifier issued to the client during the registration process. • **Client Secret** — The client secret. • **Confirm Client Secret** — Reenter the client secret. • **Scope** — The scope of the access request. Scopes enable you to specify which type of access you need. Scopes limit access for the OAuth token. They do not grant any additional permission beyond that which the user already possesses. • **Auth Request Media Type** — The format of the data you want to receive. • **Username** — The resource owner's user name. • **Password** — The resource owner's password. • **Confirm Password** — Reenter the password.

Selected Security Policy	Fields
OAuth Authorization Code Credentials	• **Client Id** — The client identifier issued to the client during the registration process. • **Client Secret** — The client secret. • **Confirm Client Secret** — Reenter the client secret. • **Authorization Code URI** — The URI from which to request the authorization code. • **Access Token URI** — URI to use for the access token. • **Scope** — The scope of the access request. Scopes enable you to specify which type of access you need. Scopes limit access for the OAuth token. They do not grant any additional permission beyond that which the user already possesses.
OAuth Custom Three Legged Flow	• **Authorization Request** — The client application URL to which you are redirected when you provide consent. The authorization server sends a callback to Oracle Integration Cloud Service to obtain an access token for storage. When you create your client application, you must register a redirect URI where the client application is listening. • **Access Token Request** — The access token request to use to fetch the access token. Specify the request using CURL syntax. For example: `-X POST` *`method`* `-H` *`headers`* `-d` *`string_data access_token_uri?query_parameters`* • **Refresh Token Request** — The refresh token request to use to fetch the access token. This request refreshes the access token if it expires. Specify the request using CURL syntax. For example `-X POST` *`method`* `-H` *`headers`* `-d` *`string_data access_token_uri?query_parameters`* • **Sauth_code** — Use regex to identify the authorization code. • **Saccess_token** — Use a regular expression (regex) to retrieve the access token. • **Srefresh_token** — Use regex to retrieve the refresh token. • **Sexpiry** — Use regex to identify when the access token expires. • **Stoken_type** — Use regex to identify the access token type. • **access_token_usage** — Specify how to pass the access token to access a protected resource. You can pass the token as a bearer token or as a query parameter. For example: `-H Authorization: Bearer ${access_token}`

Selected Security Policy	Fields

OAuth Custom Two Legged Flow	• **Access Token Request** — The access token request to use to fetch the access token. Specify the request using CURL syntax. For example:
	`-X POST method -H headers -d string_data access_token_uri?query_parameters`
	• **Refresh Token Request** — The refresh token request to use to fetch the access token. This request refreshes the access token if it expires. Specify the request using CURL syntax. For example
	`-X POST method -H headers -d string_data access_token_uri?query_parameters`
	• **Saccess_token** — Use regex to identify the access token.
	• **Srefresh_token** — Use regex to identify the refresh token.
	• **Sexpiry** — Use regex to identify when the access token expires.
	• **Stoken_type** — Use regex to identify the access token type.
	• **access_token_usage** — Specify how to pass the access token to access a protected resource. You can pass the token as a bearer token or as a query parameter. For example:
	`-H Authorization: Bearer ${access_token}`

No Security Policy	If you select this security policy, no additional fields are displayed.

3. Click **OK**.

Note: OAuth Authorization Code Credentials, OAuth Custom Three Legged Flow, and OAuth Custom Two Legged Flow security types, the connection is only successful after you click the **Provide Consent** button. Configuring all the details alone is not sufficient.

Configuring an Agent Group

Configure an agent group for accessing your on-premises application.

1. Click **Configure Agents**.

   The Select an Agent Group page appears.

2. Click the name of the agent group.
3. Click **Use**.
4. Test the connection. See Testing the Connection.

Related Topics:

About Agents and Integrations Between On-Premises Applications and Oracle Integration Cloud Service

Managing Agent Groups and the On-Premises Agent

Monitoring Agents

Testing the Connection

Test your connection to ensure that it is successfully configured.

1. In the upper right corner of the page, click **Test**.

   If successful, the following message is displayed and the progress indicator shows 100%.

   ```
   The connection test was successful!
   ```

2. If your connection was unsuccessful, an error message is displayed with details. Verify that the configuration details you entered are correct.

3. When complete, click **Save**.

Editing a Connection

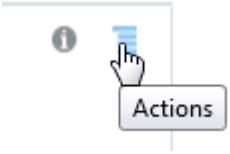
You can edit connection settings after creating a new connection.

1. In the Oracle Integration Cloud Service toolbar, click **Designer**.

2. On the Designer Portal, click **Connections**.

3. On the Connections page, search for the connection name.

4. Select **Edit** from the connection **Actions** menu or click the connection name.



   The Connection page is displayed.

5. To edit the notification email contact, change the email address in the **Email Address** field.

6. To edit the connection properties, click **Configure Connectivity**. Note that some connections do not include this button. If your connector does not include a **Configure Connectivity** button, then click the **Configure Credentials** button.

Cloning a Connection

You can clone a copy of an existing connection. It is a quick way to create a new connection.

1. In the Oracle Integration Cloud Service toolbar, click **Designer**.

2. On the Designer Portal, click **Connections**.

3. On the Connections page, search for the connection name.

4. Select **Clone** from the connection **Actions** menu.

The Clone Connection dialog is displayed.

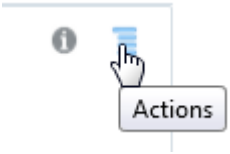
5. Enter the connection information.
6. Click **Clone**.
7. Click **Edit** to configure the credentials of your cloned connection. Cloning a connection does not copy the credentials.

See Editing a Connection for instructions.

Deleting a Connection

You can delete a connection from the connection menu.

1. In the Oracle Integration Cloud Service toolbar, click **Designer**.
2. On the Designer Portal, click **Connections**.
3. On the Connections page, search for the connection name.
4. Click **Delete** from the connection **Actions** menu.



The Delete Connection dialog is displayed if the connection is not used in an integration.

5. Click **Yes** to confirm deletion.

3

Creating an Integration

Integrations use the adapter connections you created to your applications, and define how information is shared between those applications. You can create, import, modify, or delete integrations; create integrations to publish or subscribe to messages; add and remove request and response enrichment triggers; and create routing paths for different invoke endpoints in integrations. Click the following topics for more information.

Topic

- Creating Integrations (in *Using Oracle Integration Cloud Service*)

4

Adding the REST Adapter Connection to an Integration

When you drag the REST Adapter into the trigger and invoke area of an integration, the Adapter Endpoint Configuration Wizard is invoked. This wizard guides you through configuration of the REST Adapter endpoint properties.

The following sections describe the wizard pages that guide you through configuration of the REST Adapter as a trigger or invoke in an integration.

Topics

For more information about the REST Adapter, see REST Adapter Capabilities.

Configuring REST Adapter Basic Information Properties

Enter the REST Adapter user name, description, relative resource URI, and endpoint action. You can also select to add query and template parameters or configure a request and/or response for the endpoint.

Topics

What You Can Do from the REST Adapter Basic Info Page

You can specify the following values on the trigger (source) or invoke (target) REST Adapter Basic Info page. The REST Adapter Basic Info page is the initial wizard page

that is displayed when you drag a REST Adapter to the trigger (source) or invoke (target) area.

- Specify a meaningful name.
- Add a description of endpoint responsibilities.
- Specify the relative resource URI of the endpoint.
- Select the HTTP action for the endpoint to perform.
- Add endpoint parameters, configure the endpoint request payload, or configure the endpoint to receive the response. Based on the HTTP action selected, you can configure multiple options.
- Select to configure standard and custom HTTP request and response headers.
- Select to configure cross origin resource sharing (CORS).

What You See on the REST Adapter Basic Info Page

The following table describes the key information on the REST Adapter Basic Info page.

Element	Description
What do you want to call your endpoint?	Provide a meaningful name so that others can understand the connection. For example, if you are creating a source Oracle REST connection, you may want to name it `ExposeFlowAsRESTResource`. You can include English alphabetic characters, numbers, underscores, and dashes in the name. You cannot include the following: • Blank spaces (for example, `My REST Connection`) • Special characters (for example, `#;83& or res(t)4)` • Multibyte characters
What does this endpoint do?	Enter an optional description of the connection's responsibilities (for example,`This inbound REST connection exposes this integration flow as a REST resource`).

Element	Description

What is the endpoint's relative resource URI?	Specify the relative path associated with the resource. The path can contain template parameters specified with curly braces (for example, `{order-id}`). A resource is any source of specific information that can be addressed. The resource path follows a fixed, prefixed URL appended with the specified relative path. By default, the URL is prefixed with the following path:
	
	`http://host:port/integration/flowapi/rest/INTEGRATION_NAME`
	
	For example, if the integration name is `ExposeFlowAsRESTResource`, the URL becomes:
	
	`http://host:port/integration/flowapi/rest/EXPOSEFLOWASRESTRESOURCE`
	
	You can override the URL, except for the fixed part at the beginning:
	
	`host:port/integrations`

What action does the endpoint perform?	Select a single HTTP action (method) for the endpoint to perform: • **GET**: Retrieves (reads) information (for example, makes queries). If you select this option, you cannot configure a request payload for this endpoint. • **PUT**: Updates information. • **POST**: Creates information. • **DELETE**: Deletes information. If you select this option, you cannot configure a request payload for this endpoint. **PATCH**: Partially updates existing resources (for example, when you only need to update one attribute of the resource). **Note**: The **PATCH** verb does not work with a non-SSL REST service.

Element	Description
Based on your selections, you can add parameters or configure a request and/or response for this endpoint	Select the options that you want to configure: • **Add and review parameters for this endpoint**: Click to specify the query parameters and view the template request parameters created as part of the resource URI for this endpoint. If you select this option and click **Next**, the Request Parameters page is displayed. • **Configure a request payload for this endpoint**: Click to configure the request payload for this endpoint, including specifying the schema location and payload type with which you want the endpoint to reply. You can also select this option if you want to include an attachment with the inbound request. If you select this option and click **Next**, the Request page is displayed. • **Configure this endpoint to receive the response**: Click to configure the response payload for this endpoint, including specifying the schema location and payload type that you want the endpoint to receive. If you select this option and click **Next**, the Response page is displayed.
Configure Request Headers?	Select the type of request header to configure: • **Standard**: Select to configure standard HTTP headers for the request message. • **Custom**: Select to configure custom HTTP headers for the request message.
Configure Response Headers?	Select the type of response header to configure: • **Standard**: Select to configure standard HTTP headers for the response message. • **Custom**: Select to configure custom HTTP headers for the response message.
Configure CORS (Cross Origin Resource Sharing) (available only in the trigger (inbound) direction)	Select to configure CORS parameters. CORS enables restricted resources (for example,. custom HTTP headers that introduce cross-site Java scripting security issues) on a web page to be requested from another domain outside of the domain from which the resource originated.

Configuring REST Adapter Request Parameters Properties

Enter the REST Adapter request parameters for this endpoint.

Topics

What You Can Do from the REST Adapter Request Parameters Page

You can view and configure the following request parameters on the trigger or invoke REST Adapter Request Parameters page.

- Specify query parameters.
- View the template parameters in the relative resource URI.

What You See on the REST Adapter Request Parameters Page

The following table describes the key information on the REST Adapter Request Parameters page.

Element	Description
Resource URI	Displays the endpoint relative resource URI entered on the Basic Info page.
Specify Query Parameters	Specify query parameters for the REST endpoint. Click the **Add** icon to display a row for entering the parameter name and selecting its data type. For example, specify `state` and select a data type of **string**. Click the **Delete** icon to delete a selected row.
Template Parameters	Displays the template parameters in the relative resource URI. Template parameters are based on details you specified on the Basic Info page and cannot be edited. Template parameters must be defined as part of a path with curly braces around them. For example, the URL `default/customers/{cust-id}/{ship-id}` has `cust-id` and `ship-id` template parameters. You can change the data type for the parameters. **Note**: Both query and template parameter values are displayed in the mapper through use of an XML element. For query, the XML element name is `query_parameters` For template, the XML element name is `template_parameters`.

Configuring REST Adapter Request Properties

Enter the REST Adapter request payload details for the endpoint.

Topics

What You Can Do from the REST Adapter Request Page

You can configure the following request payload details on the trigger or invoke REST Adapter Request page.

- Select to include attachments with inbound requests
- Specify the schema or JSON sample request payload file location
- Select the type of payload with which you want the endpoint to reply

What You See on the REST Adapter Request Page

The following table describes the key information on the REST Adapter Request page.

Element	Description
Select the attachment processing options	Configure the following options based on whether the request is inbound or outbound. For inbound (trigger) requests, select the multipart attachment type to include. This option is only available if you selected the POST action on the Basic Info page. • **Accept attachments from request**: Select for the REST endpoint to process attachments from the inbound multipart request. This selection refreshes the page to display the **Select the type of payload that you want the endpoint to receive** field at the bottom of the page. • **Request is HTML form**: Select for the REST endpoint to accept to configure an HTML form. You must first select the **Accept attachments from request** option before you can select this option. This selection assumes that the media type is multipart/form-data. For outbound (invoke) requests, select the multipart attachment type to include. This option is only available if you selected the POST action on the Basic Info page. • **Send attachments in request**: Select for the REST endpoint to process attachments from the outbound multipart request. This selection refreshes the page to display the **Select the type of payload that you want the endpoint to receive** field at the bottom of the page. • **Request is HTML form**: Select for the REST endpoint to accept to configure an HTML form. You must first select the **Send attachments in request** option before you can select this option. This selection assumes that the media type is multipart/form-data.

Element	Description
Select the request payload file	Select the type of request payload file to use. The request payload body must be defined by the XSD element that defines the structure of this representation. • **XML Schema** • **JSON Sample** (Creates a sample JSON file.). Select this option to use Swagger and RAML files. Empty arrays in JSON sample files are not supported. For information, see Empty Arrays Are Not Supported in Sample JSON Files.
Schema Location	Specify the schema file in either of the following ways: • Click **Browse** to select the request schema file to use. • Click **<<inline>>** to copy and paste the JSON payload or URL into a text field. Click **OK** when complete.
Element	Select the element that defines the payload structure. This field is not displayed until you import the request payload file. Once you browse for and select the schema or JSON sample file, the schema is displayed automatically. It also displays a combination box that selects the root element by default.

Element	Description
Select the type of payload with which you want the endpoint to receive	Select the payload type with which you want the request payload to reply. • **None**: Select if no payload type is required. • **XML**: Displays the payload in XML format. • **XML (text)**: Displays the payload in XML text format. • **JSON**: Displays the payload in JavaScript Object Notation (JSON) format. • **URL-encoded**: Displays the payload in URL-encoded format. • **Other Media Type**: Select to display the payload in another format (for example, `application/oracle.cloud+json`). You can only specify the media types that end with `+json` or `+xml`. The following media types are supported implicitly and cannot be configured. At runtime, the request media type is in the form of an `http Content-Type` header. The expected response media type is specified through an `Accept` header. Any service can be accessed through either of these media types. – Application/XML – Application/JSON Select the multipart attachment type for the endpoint to receive. This field is displayed if you selected an option in **Select the attachment processing options** field. • **multipart/mixed**: Send an XML or JSON payload type with an attachment. For example, send a PDF document for review as a link in an email. • **multipart/form-data**: Send an XML or JSON payload type with an attachment. For example, you create an HTML form to upload and send an image. In the HTML form, the `method` is defined as `post` and the `enctype` (encoding type) is defined as `multipart/form-data`. You can also send the attachment alone without a payload when using this attachment type.

Configuring REST Adapter Request Header Properties

Enter the REST Adapter request header properties for this endpoint.

Topics

What You Can Do from the REST Adapter Request Headers Page

You can configure the following request header details on the trigger or invoke REST Adapter Request Headers page.

- Specify standard HTTP request headers.

- Specify custom HTTP request headers.

> **Note:** If you specify a custom header name that is the same as a standard header name, an error occurs. Ensure that you specify unique names for your custom headers.

What You See on the REST Adapter Request Headers Page

The following tables describe the key information on the REST Adapter Request Headers page. Your request header selections on the Basic Info page determine whether you can select standard HTTP headers, customer HTTP headers, or both on this page. The headers are available for mapping in the Oracle Mapper.

Specify the standard HTTP request headers to use.

Element	Description
Add Standard Request Headers	Select the standard HTTP request header to use from the default dropdown list. • Click the **Add** icon to add an additional row, then select the standard HTTP request header to use from the dropdown list. • Click the **Delete** icon to delete the row of a selected standard HTTP request header.
HTTP Header Name	Perform the following tasks: • From the list, select the header to use.

Specify the custom HTTP request headers to use.

Element	Description
Add Custom Request Headers	Perform the following custom request header tasks: • Click the **Add** icon to add custom HTTP request headers and optional descriptions. • Click the **Delete** icon to delete the selected custom HTTP request headers.
Custom Header Name	Enter the custom header name.
Custom Header Description	Enter an optional description.

Configuring REST Adapter CORS Configuration Properties

Enter the REST Adapter CORS configuration properties for this endpoint.

Topics

What You Can Do from the REST Adapter CORS Configuration Page

You can configure the following CORS information on the REST Adapter CORS Configuration page.

- Specify the allowable domains from which to make CORS requests.
- View the allowable action (method) selected on the Basic Info page.

What You See on the REST Adapter CORS Configuration Page

The following table describes the key information on the REST Adapter CORS Configuration page.

Element	Description
Allowed Origins	Specify the allowable domains from which to make CORS requests. Requests coming from these domains are accepted. Enter an asterisk (`*`) for all domains to make the requests. Enter comma-separated values for specific domains to make the requests (for example, `http://localhost:8080` , `https://myhost.example.com:7002`).
Allowed Methods	The allowed method displayed is based on your selection in the **What action does the endpoint perform?** list on the Basic Info page. Requests are only accepted from the allowable domains that perform the allowable actions (methods). You cannot configure the method name listed in the CORS configuration.

For more information about CORS, see REST Adapter Capabilities.

Configuring REST Adapter Response Properties

Enter the REST Adapter response payload details for the endpoint.

Topics

What You Can Do from the REST Adapter Response Page

You can configure the following response payload details on the trigger or invoke REST Adapter Response page.

- Specify attachment processing options.
- Specify the schema or JSON sample response payload file location.
- Select the type of payload for the endpoint to receive.

What You See on the REST Adapter Response Page

The following table describes the key information on the REST Adapter Response page.

Element	Description
Select the attachment processing options	Configure the following options based on whether the request is inbound or outbound. For inbound (trigger) responses, select the multipart attachment type to include. • **Accept attachments from response**: Select to receive the response from the payload. This selection refreshes the page to display the **Select the type of payload with which you want the endpoint to reply** field at the bottom of the page. • **Response is HTML form**: Select for the REST endpoint to accept to configure an HTML form. You must first select the **Accept attachments from response** option before you can select this option. This selection assumes that the media type is multipart/form-data. For outbound (invoke) responses, select the multipart attachment type to include. • **Process attachments from response**: Select for the REST endpoint to process attachments from the outbound multipart request. This selection refreshes the page to display the **Select the type of payload with which you want the endpoint to reply** field at the bottom of the page. • **Response is HTML form**: Select for the REST endpoint to accept to configure an HTML form. You must first select the **Process attachments from response** option before you can select this option. This selection assumes that the media type is multipart/form-data.
Select the response payload file	Select the type of response payload file to use. The response payload body must be defined by the XSD element that defines the structure of this representation. • **XML Schema** • **JSON Sample** (Creates a sample JSON file.). Select this option to use Swagger and RAML files. Empty arrays in JSON sample files are not supported. For information, see Empty Arrays Are Not Supported in Sample JSON Files.

Element	Description
Schema Location	Specify the schema file in either of the following ways: • Click **Browse** to select the response schema file to use. • Click **<<inline>>** to copy and paste the JSON payload or URL into a text field. Click **OK** when complete.
Element	Select the element that defines the payload structure. This field is not displayed until you import the response payload file. Once you browse for and select the schema file, it displays a combination box that selects the root element by default.
Select the type of payload with which you want the endpoint to reply	Select the payload type with which you want the endpoint to reply. • **XML**: Displays the payload in XML format. • **XML (text)**: Displays the payload in XML text. • **JSON**: Displays the payload in JavaScript Object Notation (JSON) format. • **Other Media Type**: Select to display the payload in another format (for example, `application/oracle.cloud+json`). You can only specify media types that end with `+json` or `+xml`. The following media types are supported implicitly and cannot be configured. At runtime, the request media type is in the form of an `http Content-Type` header. The expected response media type is specified through an `Accept` header. Any service can be accessed through either of these media types. – Application/XML – Application/JSON Select the multipart attachment type for the endpoint to receive. This field is displayed if you selected an option in **Select the attachment processing options** field. • **multipart/mixed**: Send an XML or JSON payload type with an attachment. For example, send a PDF document for review as a link in an email. • **multipart/form-data**: Send an XML or JSON payload type with an attachment. For example, you create an HTML form to upload and send an image. In the HTML form, the `method` is defined as `post` and the `enctype` (encoding type) is defined as `multipart/form-data`.

Configuring REST Adapter Response Header Properties

Enter the REST Adapter response header properties for this endpoint.

Topics

What You Can Do from the REST Adapter Response Headers Page

You can configure the following response header details on the trigger or invoke REST Adapter Response Headers page.

- Specify standard HTTP response headers.
- Specify custom HTTP response headers.

Note: If you specify a custom header name that is the same as a standard header name, an error occurs. Ensure that you specify unique names for your custom headers.

What You See on the REST Adapter Response Headers Page

The following table describes the key information on the REST Adapter Response Headers page. Your response header selections on the Basic Info page determine whether you can select standard HTTP headers, customer HTTP headers, or both on this page. The headers are available for mapping in the Oracle Mapper.

Specify the standard HTTP response headers to use.

Element	Description
Add Standard Response Headers	Select the standard HTTP response header to use from the default dropdown list. • Click the **Add** icon to add an additional row, then select the standard HTTP response header to use from the dropdown list. • Click the **Delete** icon to delete the row of a selected standard HTTP response header.
HTTP Header Name	Perform the following tasks: • From the list, select the header to use.

Specify the custom HTTP response headers to use.

Element	Description
Add Custom Response Headers	Perform the following custom response header tasks: • Click the **Add** icon to add custom HTTP response headers and optional descriptions. • Click the **Delete** icon to delete the selected custom HTTP response headers.
Custom Header Name	Enter the custom header name.
Custom Header Description	Enter an optional description.

Configuring Oracle REST Adapter Invoke Operation Selection Properties

Enter the REST Adapter invoke operation selection parameters for this endpoint.

Topics

What You Can Do from the REST Adapter Operation Selection Page

You can view and configure the following request parameters on the invoke REST Adapter Operation Selection page.

- Select the business object.
- Select the operation to perform on the business object.

What You See on the REST Adapter Operation Selection Page

The following table describes the key information on the REST Adapter Operation Selection page. The business objects and operations that are displayed for selection are based on the RAML Definition URL, Swagger Definition URL, or Metadata Catalog URL you specified in the Connection Properties dialog of the Connections page.

Element	Description
Business Object	Select the business object (resource) to use in this connection.
Operations	Select the operation (method) to perform on the business object in this connection.

Reviewing Configuration Values on the Summary Page

You can review the specified adapter configuration values on the Summary page.

Topics

What You Can Do from the Summary Page

You can review configuration details from the Summary page. The Summary page is the final wizard page for each adapter after you have completed your configuration.

- View the configuration details you defined for the adapter. For example, if you have defined an inbound trigger (source) adapter with a request business object and immediate response business object, specific details about this configuration are displayed on the Summary page.
- Click **Done** if you want to save your configuration details.

- Click a specific tab in the left panel or click **Back** to access a specific page to update your configuration definitions.
- Click **Cancel** to cancel your configuration details.

What You See on the Summary Page

The following table describes the key information on the Summary page.

Element	Description
Summary	Displays a summary of the configuration values you defined on previous pages of the wizard. The information that is displayed can vary by adapter. For some adapters, the selected business objects and operation name are displayed. For adapters for which a generated XSD file is provided, click the XSD link to view a read-only version of the file. To return to a previous page to update any values, click the appropriate tab in the left panel or click **Back**.

5

Creating Mappings and Lookups in Integrations

You must map data between trigger connections and invoke connections in integrations. You can also optionally create lookups in integrations.

Topics

- Mapping Integration Cloud Service Data (in *Using Oracle Integration Cloud Service*)
- Creating Lookups (in *Using Oracle Integration Cloud Service*)

6

Administering Integrations

Oracle Integration Cloud Service provides you with the information and tools required to activate, monitor, and manage your integrations in the runtime environment.

Topic

- Administering Integration Cloud Service (in *Using Oracle Integration Cloud Service*)

7

Troubleshooting the REST Adapter

Review the following topics to learn about troubleshooting issues with the REST Adapter.

Topics

Troubleshooting SSL Certification Issues

For SSL certificate errors, perform the following tasks.

Topics

- Go to the **Administration** tab and upload the server certificate.
- For exception errors that occur when configuring a connection with OAuth Client Credentials or OAuth Resource Owner Password Credentials:

  Carefully review the OAuth documentation and use the Custom Two-Legged security policy.
- For exception errors that occur when configuring a connection OAuth Authorization:

  Carefully review the OAuth documentation and use the Custom Three-Legged Security Policy.

Defining Fault and Response Pipelines in Basic Map Data Integrations

You can define REST Adapter fault and response pipelines in Basic Map Data integrations.

The REST Adapter on the trigger (inbound) side exposes an HTTP endpoint that HTTP clients can request for using an HTTP request, and returns an HTTP response.

If successful, the REST Adapter returns a success response. The REST Adapter returns an error response with an HTTP status belonging to the error family of codes depending on the situation. This table describes the possible cause and the REST Adapter response.

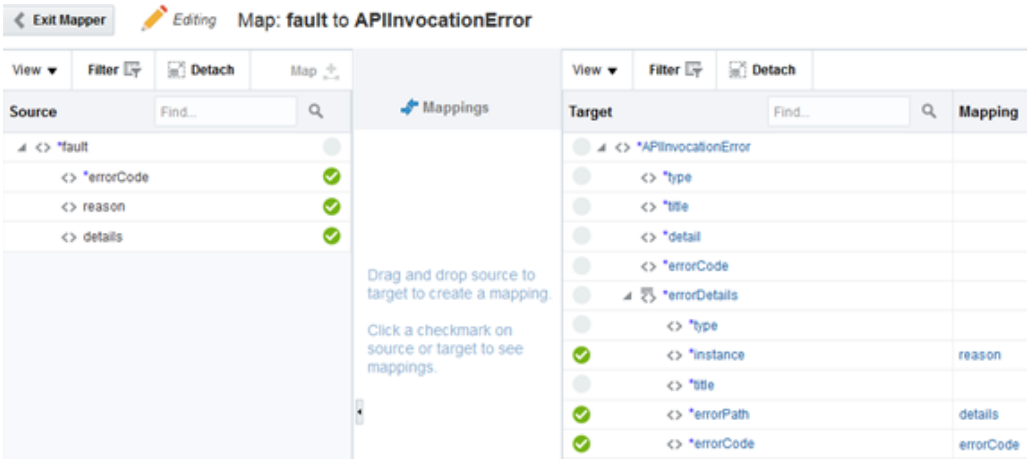
Condition	HTTP Status	Details
Invalid client request	4*xx*	There are several conditions that can cause client side failures, including: • Invalid resource URL • Incorrect query parameters • Unsupported method type • Unsupported media type • Bad data
Downstream processing errors	5*xx*	All other errors that can occur within the integration, including: • Invalid target • HTTP error response • General processing errors.

In addition, the REST Adapter also returns an error response with additional details about the error and possible steps for troubleshooting. The standard error response format is returned according to the configured response media type. The following is a sample JSON response structure:

```
{
  "type" : "http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1",
  "title" : "Internal Server Error",
  "detail" : "An internal error occurred while processing the request. Please see
the fault details for the nested error details.",
  "o:errorCode" : "500",
  "o:errorDetails" : [ {
    "type" : "http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1",
    "instance" : "{\n   \"error_message\" : \"Invalid request. Missing the 'origin'
parameter.\",\n   \"routes\" : [],\n
    \"status\" : \"INVALID_REQUEST\"\n}\n",
    "title" : "Bad Request",
    "o:errorPath" : "GET http://maps.googleapis.com/maps/api/directions/json?
destination=Montreal returned a response status of
      400 Bad Request",
    "o:errorCode" : "APIInvocationError"
  } ]
}
```

The `errorDetails` section is reserved for the actual cause. You must configure the fault pipelines to map the target faults into this element. The top portion is used to add any integration-specific details to the fault. This is typically not necessary, but if you want to control the HTTP status, title, and details, then set these values appropriately. If not entered, sufficient default values are provided by the adapter.

The suggested mappings to map faults raised by an outbound system to the trigger (inbound) REST Adapter are as follows:



The top section is left out in this mapping and so these are appropriately assigned by the adapter in the previously described sample.

Unmapped faults are propagated as system faults by Oracle Integration Cloud Service to the inbound adapter. They may not communicate the appropriate details. Therefore, it is recommended that you define the fault pipelines.

Note: Fault pipelines are only available with Basic Map Data integrations.

Empty Arrays Are Not Supported in Sample JSON Files

When configuring the REST Adapter, if a JSON property in the included JSON sample file has an empty array, you receive the following error message. Note the last part of the message. Modify the JSON sample file to include a value for the JSON property.