Oracle® Cloud

Using the SOAP Adapter with Oracle Integration 3

F45611-10
April 2024

Oracle Cloud Using the SOAP Adapter with Oracle Integration 3,

F45611-10

Primary Author: Oracle Corporation

Contents

Preface

This guide describes how to configure this adapter as a connection in an integration in Oracle Integration.

Note:

The use of this adapter may differ depending on the features you have, or whether your instance was provisioned using Standard or Enterprise edition. These differences are noted throughout this guide.

Topics:

Audience

This guide is intended for developers who want to use this adapter in integrations in Oracle Integration.

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 Oracle Mapper with Oracle Integration 3*
- Oracle Integration documentation on the Oracle Help Center.

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

Understand the SOAP Adapter

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

Topics:

- SOAP Adapter Capabilities
- SOAP Adapter Restrictions
- What Application Version Is Supported?
- Workflow to Create and Add a SOAP Adapter Connection to an Integration

Note:

There are overall service limits with Oracle Integration. A service limit is the quota or allowance set on a resource. See Service Limits.

SOAP Adapter Capabilities

The SOAP Adapter can consume an external SOAP API in an integration in Oracle Integration. The message received from Oracle Integration can be passed as payload to an external SOAP endpoint by the SOAP Adapter. Any response received from the endpoint can be sent to the next action in the integration for further processing.

The SOAP Adapter can expose inbound SOAP endpoints for accepting SOAP requests that are addressed to a specific URI. The request body is passed to the next activity present in the integration as the message payload, along with the SOAP and HTTP headers.

Note:

The SOAP Adapter treats all endpoints as they are exposed. The SOAP Adapter does not filter or change any of the APIs exposed by the application to which you are connecting. If there is a native adapter for the application to which you are connecting, use that adapter instead. If you choose to use the SOAP Adapter instead of the native adapter, the API restrictions and deprecation policies apply as specified in the respective application’s documentation. To connect to the Oracle HCM Cloud SOAP APIs, see Oracle HCM Cloud Adapter Capabilities.

The SOAP Adapter provides the following capabilities:

- SOAP Adapter Capabilities When Configured as a Trigger
- SOAP Adapter Capabilities When Configured as an Invoke

SOAP Adapter Capabilities When Configured as a Trigger

> **Caution:**
>
> Integrations exposed as SOAP APIs (using a SOAP Adapter-specific connection configured as a trigger) cannot accept attachments.

- Ensures that an incoming structured payload (XML) from a client does not exceed the size limit. If the size of the payload exceeds the limit, an HTTP error code message is returned to the client. See Service Limits in *Provisioning and Administering Oracle Integration 3*.
- Allows configuring only HTTPS protocol-based SOAP endpoints for accepting incoming SOAP requests.
- Allows triggering of integrations using OAuth 2.
- Supports configuring the inbound SOAP endpoints using the following security policies: HTTP Basic Authentication, WS-Username token-based authentication, OAuth 2.0, and Security Assertion Markup Language (SAML) (see SAML Policy Security Support in the Trigger (Inbound) Direction and OAuth 2.0 Policy Security Support in the Trigger (Inbound) Direction).
- Supports accessing of standard and custom SOAP/HTTP header properties present in the incoming SOAP request and making them available as part of an Oracle Integration message for any processing in subsequent actions (see Support for Adding Standard and Custom SOAP and HTTP Headers).
- Enables you to implement the following message exchange patterns on the inbound SOAP endpoint: synchronous request/response, one-way request, and asynchronous request with callback support. See Asynchronous Trigger Support in Integrations.
- Supports TLS server v1.2 in the trigger (inbound) direction.

SOAP Adapter Capabilities When Configured as an Invoke

- Ensures that an outgoing structured payload (XML) does not exceed the size limit. If the size of the payload exceeds the limit, an HTTP error code message is returned. See Service Limits in *Provisioning and Administering Oracle Integration 3*.
- Ensures that an incoming unstructured payload (MTOM) from a client does not exceed the size limit. If the size of the payload exceeds the limit, an HTTP error code message is returned to the client. See Service Limits in *Provisioning and Administering Oracle Integration 3*.
- Supports connecting to private resources that are in your virtual cloud network (VCN) with a private endpoint. See Connect to Private Resources in *Provisioning and Administering Oracle Integration 3* and Configure the Endpoint Access Type. The SOAP Adapter does not support private endpoints with trigger connections. Only invoke connections are supported.
- Allows invocation of an HTTPS protocol-based external SOAP endpoint, thereby encrypting the communications using transport layer security (TLS). See Transport Layer Security Version Support.

- Allows invocation of HTTP protocol-based SOAP endpoints.
- Allows invocation of external SOAP endpoints that are unprotected and protected using HTTP Basic Authentication and WS-Username token-based authentication.
- Allows invocation of external SOAP endpoints protected using OAuth Client Credentials and OAuth Authorization Code Credentials.
- Allows invocation of external SOAP endpoints hosted on TLS servers v1.1 and v1.2.
- Supports invocation of two-way, SSL-enabled external SOAP endpoints (see Two-Way SSL Support for Outbound Connections).
- Supports configuration of standard and custom SOAP/HTTP header properties available to the outbound SOAP request (see Support for Adding Standard and Custom SOAP and HTTP Headers).
- Supports invocation of external SOAP endpoints that implement the following message exchange patterns: synchronous request/response, one-way request, and asynchronous request with callback support (using WS-Addressing) (see Asynchronous Callback Response Support in the Invoke (Outbound) Direction).
- Supports propagation of the subject between co-located modules (for example, integrations to processes and processes to integrations). This enables the module to provide custom features and restrictions based on the current subject. When an integration invokes another process or integration, the subject is propagated using a JWT token. Similarly, when a process invokes an integration, it propagates the subject using JWT (see Support for Invoking Co-located SOAP Endpoints).
- Supports the dynamic discovery of endpoints. This is useful for scenarios in which the endpoint invoked by the SOAP Adapter must be dynamically configured based on runtime logic (see Support for Dynamic Endpoints).
- Supports the following:
  - Sending binary and nonbinary content as an MTOM attachment (up to 1 GB) as part of a request message while invoking external SOAP APIs.
  - Receiving binary and nonbinary content as an MTOM attachment (up to 1 GB) as part of a response message while invoking external SOAP APIs.

SOAP Adapter Restrictions

Note the following SOAP Adapter restrictions.

- The SOAP Adapter does not support private endpoints with trigger connections. Only invoke connections are supported. Attempting to use a trigger connection results in the following error:

  ```
  Connection with PrivateEndpoint access type is not supported for trigger.
  Please choose a valid
  access type for the trigger.
  ```

- Transport Layer Security (TLS) version 1.3 is not supported.
- You cannot invoke a SOAP endpoint from Oracle Integration that is using OAuth and SAML-based security. This is by design. The OAuth and SAML security policies are only supported with trigger connections.

- Two-way SSL is not supported for calls to external services through the connectivity agent. Two-way SSL requires direct connectivity from Oracle Integration without the connectivity agent.
- The SOAP Adapter does not support RPC-style WSDL binding.
- Integrations exposed as SOAP APIs (using a SOAP Adapter-specific connection configured as a trigger) cannot accept attachments.
- Without specifying a header, multiple parts in a document-style WSDL are not supported.
- You cannot switch from an asynchronous trigger/callback invoke to nonasynchronous trigger/invoke.
- NT LAN Manager (NTLM) authentication is not supported.
- SOAP WSDL 1.2 binding is not supported in the inbound direction. That version is only supported in the outbound direction.
- Operation overloading in the WSDL file is not supported with the SOAP Adapter. For example, assume your WSDL file includes the following operations with the same name, but different cases:
  - getDocumentStatus
  - GetDocumentStatus

  Each operation is also using a different request payload.

  If you select the **GetDocumentStatus** operation in the Adapter Endpoint Configuration Wizard, it is invoked at runtime, but the request payload corresponds to the **getDocumentStatus** operation. The mapper also shows the payload corresponding to the **getDocumentStatus** operation. If you change the order in the WSDL and upload it on the Connections page, this leads to activation failure. Oracle recommends the following:
  - Use unique operation names.
  - Comment out the other operation in the uploaded WSDL when creating a connection.

What Application Version Is Supported?

For information about which application version is supported by this adapter, see the Connectivity Certification Matrix.

Workflow to Create and Add a SOAP Adapter Connection to an Integration

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

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.	Create a SOAP Adapter Connection
2	Create the integration. When you do this, you add trigger and invoke connections to the integration.	Create Integrations and Add the SOAP Adapter Connection to an Integration
3	Map data between the trigger connection data structure and the invoke connection data structure.	Map Data in *Using Integrations in Oracle Integration 3*
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).	Manage Lookups in *Using Integrations in Oracle Integration 3*
5	Activate the integration.	Manage Integrations in *Using Integrations in Oracle Integration 3*
6	Monitor the integration on the dashboard.	Monitor Integrations During Runtime in *Using Integrations in Oracle Integration 3*
7	Track payload fields in messages during runtime.	Assign Business Identifiers for Tracking Fields in Messages and Track Integration Instances in *Using Integrations in Oracle Integration 3*
8	Manage errors at the integration level, connection level, or specific integration instance level.	Manage Errors in *Using Integrations in Oracle Integration 3*

2

SOAP Adapter Concepts

The following sections describe SOAP Adapter capabilities in more detail.

Topics:

SOAP Specifications

The following specifications are supported:

- SOAP 1.2
- WS-I Security (for SSL, TLS, and ciphers)
- SOAP 1.1 binding for MTOM
- WS-Addressing
- WS-Security Username Token

Transport Layer Security Version Support

Specifying the transport Layer Security (TLS) version of the target server is supported. The TLS protocol provides privacy and data integrity between two communicating computer applications. See Configure Connection Properties.

Version Suppression of the Timestamp in the WS-Security Header

Version suppression of the timestamp in the WS-Security header is supported. Suppression applies to the Username Password Token security policy in the invoke (outbound) direction. In secure Web Services transactions, a WS-Utility (WSU) timestamp can be inserted into a WS-Security header to define the lifetime of the message in which it is placed. See Configure Connection Properties.

Ability to Specify if the Timestamp is Not Required in the Response Message

You can specify if the timestamp is not required in the response message. See Configure Connection Properties.

SOAP Action Validation Disabling for Inbound Requests

Disabling SOAP action validation for inbound requests on the Operations page of the Adapter Endpoint Configuration Wizard is supported. This is useful for environments in which your WSDL includes custom code and you want to bypass validation. See Invoke Operation Page and Callback Integrations Fail with a Configured SOAP Action Mismatch Error.

Asynchronous Callback Response Support in the Invoke (Outbound) Direction

An asynchronous callback response in the invoke (outbound) direction is supported. This feature is enabled when the WSDL used in the connection defines a one-way operation in the selected service/port. The callback response endpoint must be specified through a different integration flow. The callback endpoint can be secured with any policy supported by the SOAP Adapter on the trigger side.

Based on the operation selection, if it is a one-way operation, you are asked to select the expected response type (no response or delayed response) on the Callback Operation page of the Adapter Endpoint Configuration Wizard.

- No Response: One-way invocation.
- Delayed Response: Specify the callback port type, operation, callback flow identifier, and version.

The callback flow identifier and version are used to determine the callback endpoint and sent in the ReplyTo header while sending a request to the outbound endpoint.

Support for Adding Standard and Custom SOAP and HTTP Headers

Adding standard and custom SOAP and HTTP headers to outbound and inbound requests and handling the responses with headers to propagate back to the user are supported. This configuration enables header configuration for the inbound service and header propagation for the outbound service. WS-Addressing headers propagation is not supported (for example, MessageId, ReplyTo, FaultTo, and so on). All header information and body elements are encapsulated under a single element so the mapper can display request and response information. See Add the SOAP Adapter Connection to an Integration.

Support for Multiple Part Messages in Document-Style WSDLs

Note:

The SOAP Adapter does *not* support RPC-style WSDL bindings. Only document-style WSDL bindings are supported.

- Multiple part messages in document-style WSDLs is supported. The support is provided for both inbound and outbound adapter configurations.

  Standard SOAP headers can be defined in a WSDL in two ways:

  – Implicit headers:

    With this type, the request header and body part are in different message types. In the binding section of the WSDL, the header uses the part name within the message type and message type name. The body does not have any part names explicitly defined in it.

    ```
    <wsdl:message name="CreateUserRequestHeader">
    <wsdl:part name="requestHeader" element=" tns:UserCreate"/>
    </wsdl:message>
    <wsdl:message name="CreateUserRequest">
    <wsdl:part element="tns:UserCreateHeader" name="parameters"/>
    </wsdl:message>
    <wsdl:binding name="UserBinding" type="tns:UserEndPoint">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/
    soap/http" />
    <wsdl:operation name="CreateUser">
    <soap:operation soapAction="http://example.com/CreateUser" />
    <wsdl:input> <soap:body use="literal" />
    <soap:header use="literal" part="requestHeader"
    message="tns:CreateUserRequestHeader"/>
    </wsdl:input>
    <wsdl:output>
    <soap:body use="literal" />
    </wsdl:output>
    </wsdl:operation>
    <wsdl:binding/>
    ```

  – Explicit headers:

    With this type, there are multiple parts in a single message type in the WSDL: one for the header and one for the body payload. The header is specified by its part name. The body uses its own name.

    ```
    <wsdl:message name="CreateUserRequest">
    <wsdl:part element="tns:UserCreateHeader" name="parameters"/>
    <wsdl:part name="requestHeader" element=" tns:UserCreate"/>
    </wsdl:message>
    <wsdl:binding name="UserBinding" type="tns:UserEndPoint">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/
    soap/http" />
    <wsdl:operation name="CreateUser">
    ```

```
<soap:operation soapAction="http://example.com/CreateUser" />
<wsdl:input> <soap:body use="literal" />
<soap:header use="literal" part="requestHeader"
message="tns:CreateUserRequest"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal" />
</wsdl:output>
</wsdl:operation>
<wsdl:binding/>
```

Note:

Without specifying a header, multiple parts in a document-style WSDL are not supported.

When you invoke the Adapter Endpoint Configuration Wizard to configure the SOAP Adapter as a trigger or invoke, the Operations page detects that the WSDL includes defined SOAP request and/or response headers and automatically enables the button to configure SOAP headers for the endpoint. You can select **No** to remove the headers for the endpoint. You cannot modify these headers. The subsequent Request Header and Response Header pages of the WSDL load and show the specific headers defined in the WSDL. See Add the SOAP Adapter Connection to an Integration.

Two-Way SSL Support for Outbound Connections

The use of two-way SSL for outbound communications is supported. This feature enables an integration to invoke web services hosted on a two-way, SSL-enabled server and receive a response in return.

You must satisfy the following requirements to use this feature:

- Upload the following certificates in the Upload Certificate dialog in Oracle Integration. See Upload a Certificate to Connect with External Services.
  - Upload a two-way SSL identity certificate type. This certificate is created from the client server on which two-way SSL must be enabled
  - Upload a trust certificate type for the outbound call. This is the certificate for the client server that hosts your web service.
- Specify a WSDL URL with secure HTTP (`https`) on the Connection Properties dialog. This WSDL must use the web service URL hosted on the two-way, SSL-enabled server.
- Two-way SSL is only supported with the JCA transport mechanism of the SOAP Adapter. The HTTP transport mechanism of the SOAP Adapter is not supported.
- Configure the server that hosts the web service for two-way SSL communication.
- Configure the SOAP Adapter as both a trigger and invoke. When you create the integration, you configure this same SOAP Adapter connection as both the trigger and invoke.

SAML Policy Security Support in the Trigger (Inbound) Direction

The Security Assertion Markup Language (SAML) is an XML-based, open-standard data format for exchanging authorization and authentication information between two different systems, typically an identity provider and a service provider. A SOAP Adapter trigger-specific connection can be configured to protect inbound SOAP endpoints using SAML token-based authentication. This configuration can be used to implement use cases that involve receiving callback messages from Oracle ERP Cloud upon completion of file-based data import (FBDI) jobs and upon completion of asynchronous operations in any Oracle ERP Cloud application, such as fusion order management (FOM). However, any client that supports SAML bearer token authentication can use this policy.

OAuth 2.0 Policy Security Support in the Trigger (Inbound) Direction

Integrations exposing SOAP endpoints using the SOAP Adapter as a trigger connection can be OAuth 2.0-protected. OAuth 2.0 is an industry-standard protocol for authorization. OAuth 2.0 focuses on client developer simplicity while providing specific authorization flows for web applications.

Asynchronous Trigger Support in Integrations

When a call is made to an asynchronous service, the response is expected at a later time and possibly to a different endpoint. If the response is expected at a different endpoint, the endpoint information must be passed to the asynchronous service during the request using the WS-Addressing ReplyTo header. In this case, the Oracle Integration endpoint with the SOAP Adapter configured as the (trigger) inbound connection acts as an asynchronous service. Oracle Integration determines if the selected operation is asynchronous and then enables you to provide callback endpoint details through a ReplyTo standard header and to use this information to invoke the callback response.

The following requirements must be satisfied to use this feature:

- This feature is only available when the SOAP Adapter is included in integrations.
- The asynchronous service uses only the SOAP Adapter-supported OWSM policies. The callback endpoint being specified in the ReplyTo header *must* support one of the security policies available on the Connections page for the invoke-only role.
- The corresponding callback invoke must also be configured when the trigger is configured for an asynchronous response.
- The request payload contains a Reply-To header that contains the value of the endpoint to which to send the asynchronous response.

In the trigger direction, you must configure the Adapter Endpoint Configuration Wizard as follows:

- On the Callback Operations Page, you select **Delayed Response** because a callback response is expected.
- On the Headers page, the **Do you want to configure headers for this Endpoint** option is automatically enabled. The **SOAP Headers** option is automatically selected in the **Request Headers** section and cannot be changed.
- On the Request Headers page, the WS-Addressing `ReplyTo`, `MessageID`, and `Action` headers are automatically populated in the **Standard SOAP Headers** tab.

In the invoke direction, you must configure the Adapter Endpoint Configuration Wizard as follows:

- On the Welcome page, select **Yes** to configure the SOAP Adapter as a callback invoke.
- On the Operations page, the list of port types is displayed (instead of the service and port). You must select the callback port type and callback operation. The Callback Operations page is disabled as it is not an outbound asynchronous case.
- On the Headers page, the headers configuration option is automatically enabled. The **SOAP Headers** option is automatically selected in the **Request Headers** section and cannot be changed.
- On the Request Headers page, the WS-Addressing `ReplyTo`, `MessageID`, `Action` headers are populated in the Standard SOAP Headers tab.

During integration creation, you must explicitly map the WS-Addressing headers from **Inbound Request** to **Callback Invoke Request** in addition to the other required mapping options. During runtime, when Oracle Integration is invoked with the request having a wsa:ReplyTo header, the service invokes the endpoint sent in the header with the response.

Note:

You cannot switch from an asynchronous trigger/callback invoke to nonasynchronous trigger/invoke.

Support for Invoking Co-located SOAP Endpoints

You can propagate the subject between co-located modules (for example, Integrations to Processes and Processes to Integrations). This enables the module to provide custom features and restrictions based on the current subject.

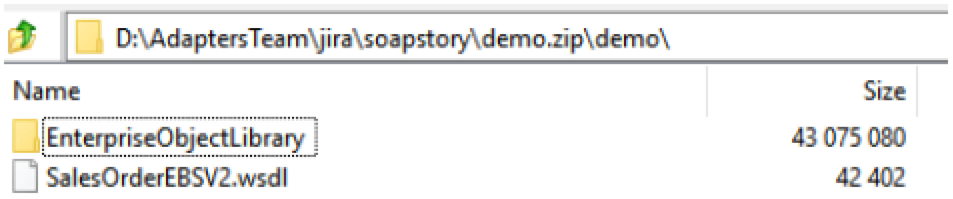
Oracle Integration can automatically determine if an outbound (invoke) SOAP endpoint being invoked by an integration is local (co-located) or remote to Oracle Integration and then optimize the invoke call to the endpoint. Co-located means the integrations are running on the same host instance or in the same domain. If the outbound endpoint is co-located, the endpoint is invoked using an optimized HTTP request using a JSON Web Token (JWT) token for authorization. The optimized HTTP request is a plain HTTP request (non-SSL) sent directly to the managed server. The currently configured security policy is overwritten by a JWT token. JWT is a JSON-based open standard (RFC 7519) for creating access tokens that assert some number of claims. For example, a server can generate a token that has the claim "logged in as admin" and provide that to a client. The client can then use that token to prove that it is logged in as admin. The tokens are signed by the server's key. Therefore, the client and server can both verify that the token is legitimate.

Support for Uploading a WSDL with Schemas in a ZIP File

When creating a connection, you can upload a ZIP file with a WSDL and dependencies such as other WSDLs and XSDs nested inside the ZIP. This can be useful in scenarios in which you adopt the standard canonical model (OAGIS) in integrations. If you upload a ZIP, you must ensure that the WSDLs are available in the top two directory levels. This makes the WSDLs available for selection in the **Service WSDL** dropdown list on the Connections page. There can be any number of WSDLs at these two levels. Any WSDLs below these levels do not appear for you to select in the **Service WSDL** dropdown list.

The following use cases are supported:

- The ZIP file contains the main WSDL (for this example, named `SalesOrderEBSV2.wsdl`) in the immediate folder and its dependencies (EBM, other XSDs, and so on) nested deeply inside a folder.



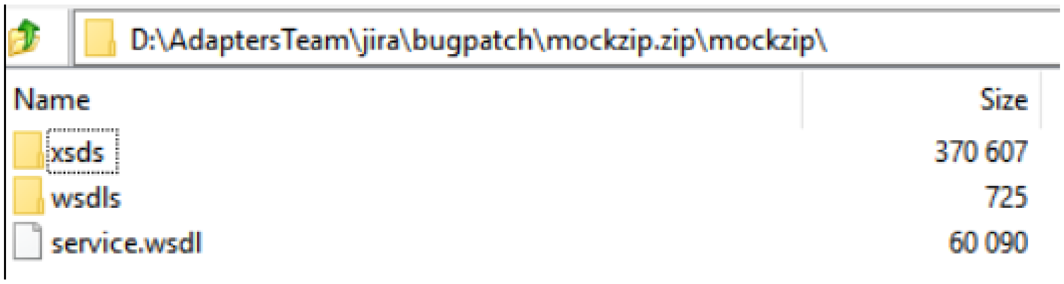
D:\AdaptersTeam\jira\soapstory\demo.zip\demo\

Name	Size
EnterpriseObjectLibrary	43 075 080
SalesOrderEBSV2.wsdl	42 402

For this example, the main WSDL reference must be corrected to reference `EnterpriseObjectLibrary` as shown.

```
<xsd:import namespace="http://xmlns.oracle.com/EnterpriseObjects/Core/EBO/
SalesOrder/V2" schemaLocation="./EnterpriseObjectLibrary/Core/EBO/
SalesOrder/V2/SalesOrderEBM.xsd"/>
<xsd:import namespace="http://xmlns.oracle.com/EnterpriseObjects/Core/
Common/V2" schemaLocation="./EnterpriseObjectLibrary/Core/Common/V2/
Meta.xsd"/>
```

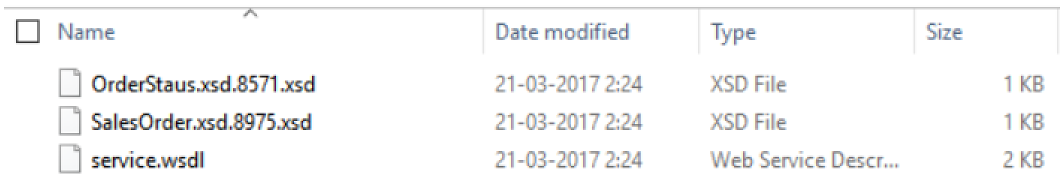
- The ZIP file contains a main WSDL (for this example, named `service.wsdl`) and references another WSDL and XSDs:



D:\AdaptersTeam\jira\bugpatch\mockzip.zip\mockzip\

Name	Size
xsds	370 607
wsdls	725
service.wsdl	60 090

- The ZIP file contains a WSDL (for this example, named `service.wsdl`) and any number of XSD dependencies in the same directory.



Name	Date modified	Type	Size
OrderStaus.xsd.8571.xsd	21-03-2017 2:24	XSD File	1 KB
SalesOrder.xsd.8975.xsd	21-03-2017 2:24	XSD File	1 KB
service.wsdl	21-03-2017 2:24	Web Service Descr...	2 KB

Support for Using MTOM to Transfer Large Binary Payloads

Message Transmission Optimization Mechanism (MTOM) support is provided. MTOM is a W3C Message Transmission Optimization Mechanism, a method for efficiently sending binary data to and from web services. Binary objects in SOAP are represented as base64 encoded messages, which essentially expands the data by about 33%. For large payloads, this can significantly impact performance and transmission time. MTOM provides a solution to transfer a large binary payload using optimization. MTOM/XOP optimizes a SOAP message and the XOP processing serializes it into a MIME multipart/related message. The XOP processing extracts the base64-encoded data from the SOAP message and packages it as separate binary attachments within the MIME message. See Configure MTOM Support in the SOAP Adapter.

MTOM is currently supported only on the invoke connection in an integration.

Note:

MTOM upload/download cannot be invoked asynchronously with a large payload and high concurrency. This scenario can result in an out-of-memory error depending upon payload size and concurrency. Take care in your design. For example, in a schedule integration with scheduling set to every 10 minutes, four flows can run consistently with 512 MB payload every 10 minutes on a two-node Oracle Integration cluster without any out-of-memory errors.

Support for Dynamic Endpoints

The SOAP Adapter supports the dynamic discovery of endpoints. This is useful for scenarios in which the endpoint that the SOAP Adapter invokes must be dynamically configured based on runtime logic. This feature is applicable for both new integrations and existing integrations (edited to add new invokes) that include the SOAP Adapter as an invoke connection. The endpoint invoked must support the same security policies as supported in Oracle Integration. However, WS-Addressing is not used. Instead, two types of properties in the mapper are provided for configuring dynamic invocation. These properties are used during runtime to override the properties configured on the Connections page during design time.

- Endpoint Properties: Override the endpoint details.
  - **EndpointURI**: Replaces the existing endpoint URI specified on the Connections page before invoking the endpoint.
  - **SoapAction**: Replaces the existing SOAP action validation setting before invoking the endpoint.
- Security Properties: Override the endpoint credential details, if required.
  - If the connection uses Username Password Token:
    - **Username**: Replaces the username credentials before invoking the endpoint.
    - **Password**: Replaces the password credentials before invoking the endpoint.
    - **ignoreNonce**: Accepts a boolean **true**/**false** value. The default is **false**. Setting this value to **true** in the request prevents the `Nonce` and `Created` headers from being sent in the Username Password Token.
    - **ignoreCreated**: Accepts a boolean **true**/**false** value. The default is **false**. Setting this value to **true** in the request prevents the `Nonce` and `Created` headers from being sent in the Username Password Token.

For Username Password Token, Oracle Integration supports the WS-Security 2004 version (SOAP Message Security 1.0 specification) in the outbound direction. This contains a SOAP message security header:

```
<wsse:Security soapenv:mustUnderstand="1" xmlns:wsse="http://
docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-wssecurity-utility-1.0.xsd">
         <wsu:Timestamp wsu:Id="TS-910919AD52F5349E1B16034288264664">
            <wsu:Created>2020-10-23T04:53:46.466Z</wsu:Created>
            <wsu:Expires>2020-10-23T04:54:46.466Z</wsu:Expires>
         </wsu:Timestamp>
         <wsse:UsernameToken
wsu:Id="UsernameToken-910919AD52F5349E1B16034288238563">
            <wsse:Username>username</wsse:Username>
            <wsse:Password Type="http://docs.oasis-open.org/wss/
2004/01/oasis-200401-wss-username-token-
profile-1.0#PasswordText">password</wsse:Password>
            <wsse:Nonce EncodingType="http://docs.oasis-open.org/wss/
2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary">VG04/
faWm6rFXSuu8kBGJg==</wsse:Nonce>
            <wsu:Created>2020-10-23T04:53:43.856Z</wsu:Created>
         </wsse:UsernameToken>
      </wsse:Security>
```

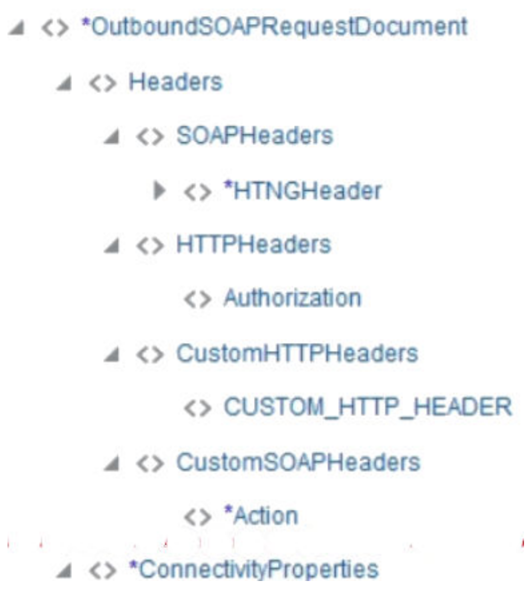
The specification says `Nonce` and `Created` are optional and useful to avoid distributed denial-of-service (DDOS) attacks. To support endpoints that only support Username Password Token without `Nonce` and `Created`, the **ignoreNonce** and **ignoreCreated** connectivity properties are provided in the outbound request mapper for SOAP Adapter connections that use Username Password Token.

– If the connection uses Basic Authentication Security:

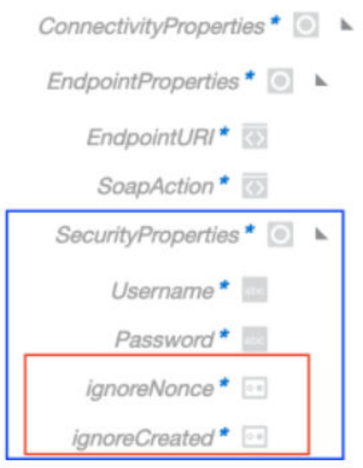
  * Authorization: Replaces the authorization header before invoking the endpoint.

The following dynamic properties are visible for configuration in the mapper and used during runtime to override the properties configured during design time.

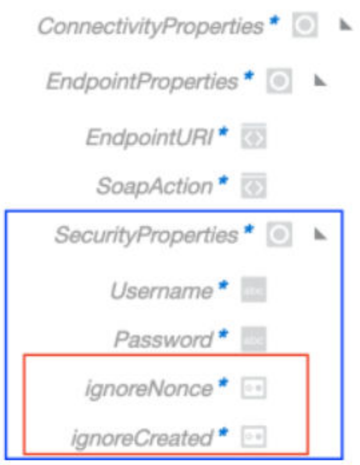
– Sample outbound request: The **ConnectivityProperties** section and the **Headers** and/or **Body** sections are displayed with properties for configuration. The **ConnectivityProperties** section is not visible if this is a callback invoke request.



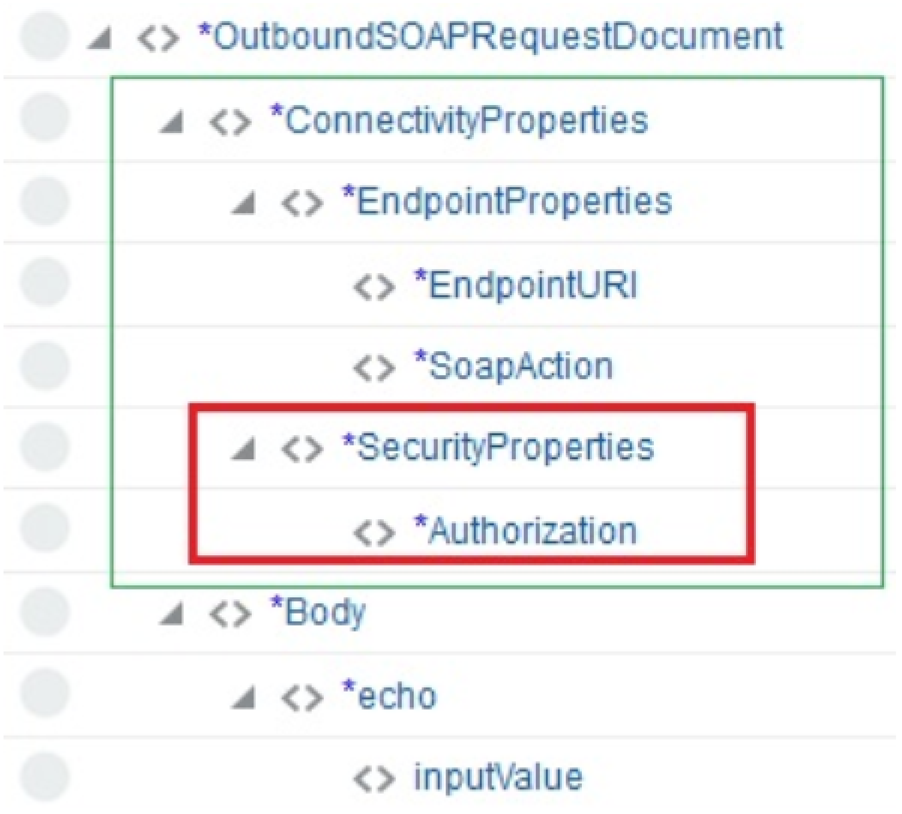
When expanded, **ConnectivityProperties** shows the following for an outbound request.



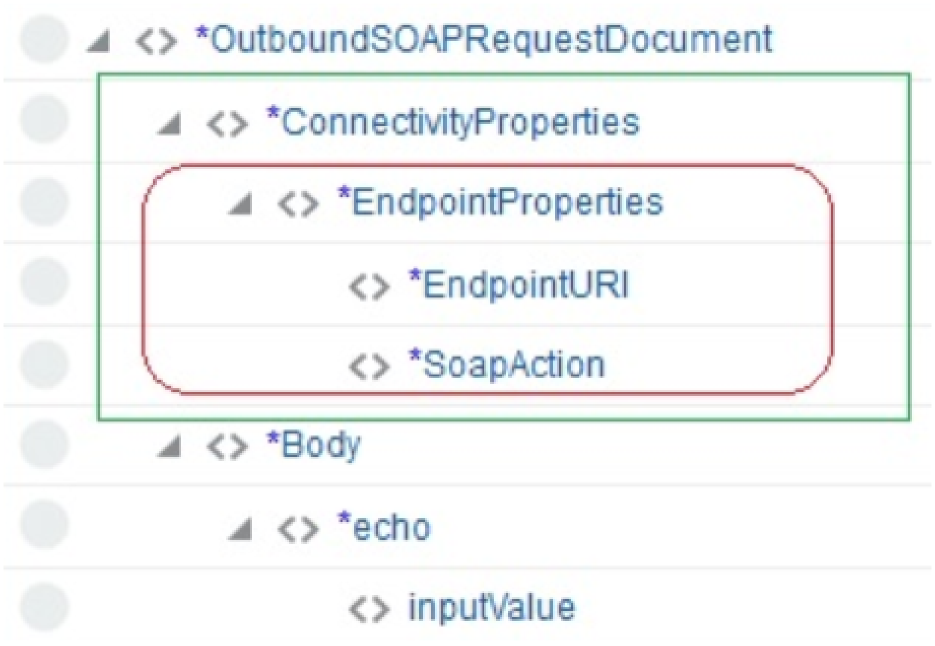
– Username Password Token security policy: The **EndpointProperties** section and **SecurityProperties** section are visible with properties for configuration under the main **ConnectivityProperties** section.



– Basic Authentication security policy: The **EndpointProperties** section and **SecurityProperties** section are visible with properties for configuration under the main **ConnectivityProperties** section.



– No security policy: The **EndpointProperties** section is visible with properties for configuration under the main **ConnectivityProperties** section.



Note:

- Overriding security properties with the dynamic endpoint invocation feature logs the details mapped when trace is enabled during activation of the integration.
- When using dynamic endpoints with the SOAP Adapter, be aware that if you activate an integration using this feature with **Enable tracing** and **Include payload** selected, the password used in the payload during runtime is exposed in clear text in the log file.

Authenticate Requests for Invoking Oracle Integration Flows

Integrations support multiple authentication methods suited to different applications and use cases. The adapters used as a trigger connection to stand up the endpoints/ listener for a specific integration can support one or multiple authentication methods.

The following sections discuss the use cases, pros and cons, prerequisites, and instructions necessary for sending a request for each of the supported authentication methods.

Topics:

- About Requests to Invoke Integrations
- About OAuth 2.0 Grants
- Use OAuth 2.0 Grants in Oracle Identity Cloud Service Environments
- Use OAuth 2.0 Grants in Identity Domain Environments

See OAuth Grant Types.

About Requests to Invoke Integrations

All integrations using this adapter as a trigger connection are protected by default using HTTP Basic Authentication and OAuth token-based authentication.

You currently can authenticate your requests to invoke integrations in either of the following ways:

- Using HTTP Basic Authentication by sending the credentials of the user (that is, created in Oracle Identity Cloud Service) through the HTTP authorization header
- Sending an OAuth access token in the header while invoking an Oracle Integration endpoint after acquiring the access token from Oracle Identity Cloud Service that serves as the OAuth authorization provider

You must have the ServiceUser role in Oracle Identity Cloud Service to invoke integrations.

Invoke Integration Endpoints Using HTTP Basic Authentication

This authentication method allows the credentials belonging to an Oracle Integration user to send the request to invoke an integration. You must create this user in the Oracle Integration identity provider Oracle Identity Cloud Service and ensure that the user was granted the role for invoking an integration.

The user can be:

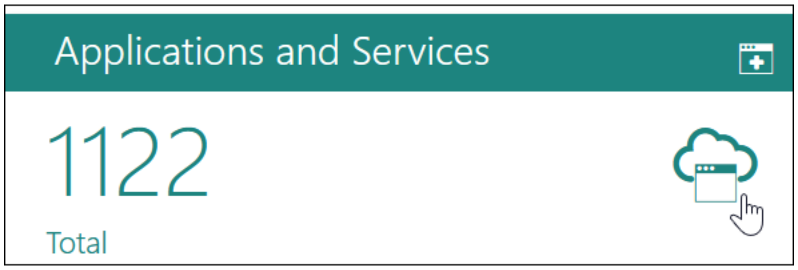
- Human - representing a business user such as a sales representative, technician, or any other person for invoking an integration
- Nonhuman - representing a service integration account used by an external client application for invoking an integration

Even though it's easy to implement the authentication scheme, this is the least secure way to send a request to Oracle Integration for invoking an integration. Also, Oracle Integration doesn't recommend this authentication scheme.

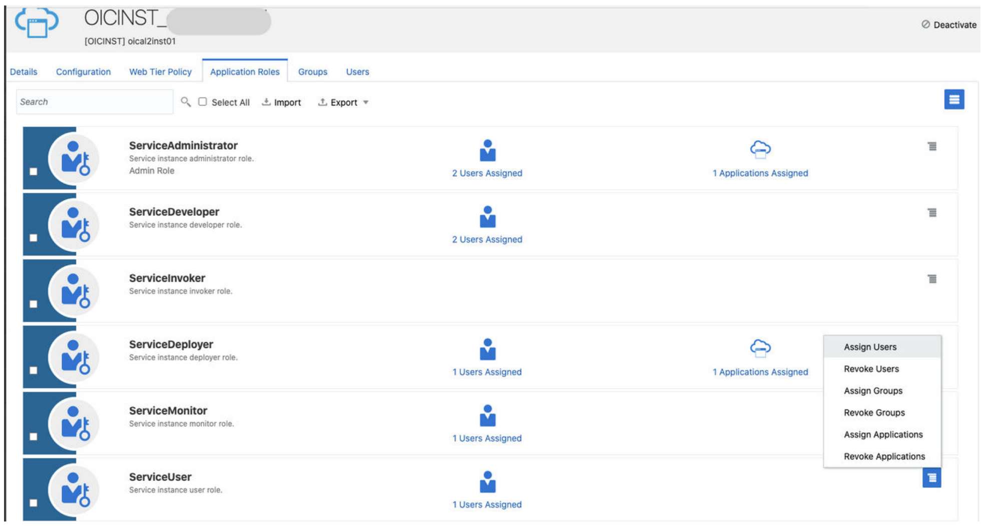
In addition, the customer must ensure the credentials, when reset, are provided to the client application that invokes the integration to ensure a new set of credentials are being used from then on.

Assign appropriate user(s) to the various Oracle Integration roles. For standard/production configurations, use the ServiceUser role. (See Oracle Integration Roles in *Provisioning and Administering Oracle Integration 3*.)

1. From the ☰ menu on the Oracle Cloud Infrastructure home page, select **Identity & Security**, then select **Federation**.
2. In the **Federation** table, click **OracleIdentityCloudService**.
3. In the **Oracle Identity Cloud Service Console** field, click the URL.
4. Click the applications page icon.



5. Click the application.
6. To assign a user, go to the **Application Roles** section of Oracle Identity Cloud Service.



ServiceAdministrator
Service instance administrator role.
Admin Role
2 Users Assigned
1 Applications Assigned

ServiceDeveloper
Service instance developer role.
2 Users Assigned

ServiceInvoker
Service instance invoker role.

ServiceDeployer
Service instance deployer role.
1 Users Assigned
1 Applications Assigned

ServiceMonitor
Service instance monitor role.
1 Users Assigned

ServiceUser
Service instance user role.
1 Users Assigned

7. Make a request to trigger an endpoint.

```
curl --location --request GET 'https://OIC host/OIC endpoint' \
--header 'Authorization: Basic <base64-encoded username:password>'
```

Invoke Integration Endpoints Using OAuth Token-Based Authentication

This authentication scheme allows the external client to acquire a token that is also sent as part of the request sent to invoke an integration.

The most important step for an application in the OAuth flow is how the application receives an access token (and optionally a refresh token). A grant type is the mechanism used to retrieve the token. OAuth defines several different access grant types that represent different authorization mechanisms.

Applications can request an access token to access protected endpoints in different ways, depending on the type of grant type specified in the Oracle Identity Cloud Service application. A grant is a credential representing the resource owner's authorization to access a protected resource.

The following sections discuss the various grant types and their pros/cons, along with instructions on how to configure the specific grant type.

About OAuth 2.0 Grants

There are several OAuth 2.0 grant types you can use in Oracle Integration. Review the following information to identify the grant type to use for your use case.

Grant Type	About the Grant Type	Use Cases and Risks
JWT user assertion (recommended)	A user assertion is a user token that contains identity information about the user. The user can either represent a human or a service integration account created for identifying a specific calling application. The user assertion is used directly as an authorization grant to obtain an access token. The client details are provided either as an authentication header in the request or as a client assertion. The user assertion grant is more secure than the resource owner password credentials grant because the user's credentials are never exposed. The user assertion workflow: • Is used with confidential clients. The OAuth clients are trusted to assert a user/service integration account identity on behalf of the user/service integration account. • The resource owner's credentials (Oracle Integration user) are never accessible to the client application. It just uses the assertion of the resource owner. • It isn't redirection-based. It takes a request only from the client application to the authorization server. The user is not redirected between interfaces to authorize the request. This user assertion grant works as follows: • The client requests an access token by providing a user assertion. The client details are provided either as an authentication header in the request or as a client assertion. • The OAuth service authenticates the client	This grant is used by applications that want to programmatically invoke integrations without any user intervention. The client application impersonates the user by sending the user assertion to Oracle Identity Cloud Service while requesting token access. An access token is returned in the user context. The user can either represent a human or a service integration account created for identifying a specific calling application. Oracle Integration recommends the use of this grant for acquiring an OAuth access token by the applications that must programmatically start the integration without any user intervention. **Risks** Carefully use this grant (only with first party/trusted clients) because it allows for trivial impersonation to more highly privileged accounts on services. **Usage** See Prerequisites for JWT User Assertions.

Grant Type	About the Grant Type	Use Cases and Risks
	and, if valid, supplies an access token. The JWT user assertion characteristics are as follows: • Does not require the client to have knowledge of user credentials. • There is no browser-based end user interaction. • A refresh token is allowed. • An access token is in the context of the end user. In this OAuth flow: • A user attempts to access a client application by sending a generated user assertion. • The client application requests an access token, and often a refresh token, by providing a user assertion or a third-party user assertion. • The Oracle Identity Cloud Service authorization server returns the access token to the client application. • The client application uses the access token in an API call to invoke the integration.	

Grant Type	About the Grant Type	Use Cases and Risks
Authorization code	The authorization code grant type is used by web and mobile applications. It differs from most of the other grant types by first requiring the application to launch a browser to begin the integration. At a high level, the integration consists of the following steps: • The application opens a browser to send the user to the OAuth server. • The user sees the authorization prompt and approves the application request. • The user is redirected back to the application with authorization code in the query string. • The application exchanges the authorization code for an access token. The authorization code has the following characteristics: • Does not require the client to have knowledge of user credentials. • Is a browser-based end user interaction. • A refresh token is allowed. • An access token is in the context of the end user. In this OAuth flow: • A user clicks a link in a web server client application to request access to protected resources. • The client application redirects the browser to the Oracle Identity Cloud Service authorization endpoint with a request for an authorization code: `oauth2/v1/authorize` • The Oracle Identity Cloud Service authorization	This grant is used by the applications such as web portals and mobile applications involving user interactions that may end up invoking the integrations. In this type of use case, the user signing in to the web portal/ mobile application explicitly provides the consent by authenticating against Oracle Integration to let their application start the integration. **Usage** See Prerequisites for Authorization Code.

Grant Type	About the Grant Type	Use Cases and Risks
	server returns an authorization code to the client application through a browser redirect after the resource owner gives consent. • The client application subsequently exchanges the authorization code for an access token, and often a refresh token. • The Oracle Identity Cloud Service authorization server returns the access token to the client application. • The client application uses the access token in an API call to invoke the integration.	

Grant Type	About the Grant Type	Use Cases and Risks
Resource owner password credential (ROPC)	The resource owner's password credentials (that is, the user name and password) can be used by the OAuth client directly as an authorization grant to obtain an access token. The resource owner password credentials grant type is suitable for cases where the resource owner has a trust relationship with the OAuth client. When using the resource owner password credentials grant, the user provides the credentials (user name and password) directly to the application. The application then uses the credentials to obtain an access token from the OAuth token service. The resource owner password credentials grant is a grant workflow where the client application, together with its client identifier and secret, sends the user name and password in exchange for an access token. Instead of the user having to log in and approve the authorization request in a web interface, the user can enter the user name and password in the client application user interface directly. This workflow has different security properties than other OAuth workflows. The primary difference is that the user's password is accessible to the application. This requires a strong trust of the application by the user. The resource owner password credentials grant has the following characteristics: • The client is required to have knowledge of user credentials. • Is not a browser-based end user interaction. • A refresh token is allowed. • An access token is in the context of the end user.	This grant can be used by applications that want to programmatically invoke the integration without any user intervention. Use this grant only with trusted first-party clients that securely handle user credentials. Even though this grant type can be used by client applications to acquire an OAuth access token to use for sending the request to invoke an integration in a programmatic manner, Oracle Integration does *not* recommend the resource owner password credential grant because of the following risks: **Risks** • This grant type carries a higher risk than other grant types because it maintains the password anti-pattern this protocol seeks to avoid. The client can abuse the password or the password can unintentionally be disclosed to an attacker (for example, through log files or other records kept by the client). • The application can request a scope with complete access to user resources once it possesses the password credential. • Passwords expire. • This grant is currently in a deprecated state. **Usage** See Prerequisites for Resource Owner Password Credentials.

Grant Type	About the Grant Type	Use Cases and Risks
	In this OAuth flow: • The user clicks a link in the client application requesting access to protected resources. • The client application requests the resource owner's user name and password. • The user logs in with their user name and password. • The client application exchanges those credentials for an access token, and often a refresh token, from the Oracle Identity Cloud Service authorization server. • The Oracle Identity Cloud Service authorization server returns the access token to the client application. • The client application uses the access token in an API call to invoke the integration.	

Grant Type	About the Grant Type	Use Cases and Risks
Client credentials	The client uses its client credentials (or other supported means of authentication) to request an access token when requesting access to protected resources: • Under its control • Those of another resource owner that have been previously arranged with the authorization server Only confidential clients must use this grant type. In this OAuth flow: • The client authenticates with the authorization server and requests an access token from the token endpoint. Because client authentication is used as the authorization grant, no additional authorization request is required. • The authorization server authenticates the client and, if valid, issues an access token. If the request fails client authentication or is invalid, the authorization server returns an error response.	This grant is typically used by clients to obtain an access token outside of the context of a user (for example, to access resources about themselves rather than to access a user's resources). **Usage** See Prerequisites for Client Credentials.

Use OAuth 2.0 Grants in Oracle Identity Cloud Service Environments

To use an OAuth 2.0 grant type with this adapter in an Oracle Identity Cloud Service environment of Oracle Integration, you must perform the following prerequisites.

- Prerequisites for All Grants
- Prerequisites for JWT User Assertion
- Prerequisites for Authorization Code
- Prerequisites for Resource Owner Password Credentials
- Prerequisites for Client Credentials

Prerequisites for All Grants

Perform the following tasks for each grant type you use.

- Obtain the Oracle Identity Cloud Service URL.
  1. Go to the URL for your Oracle Integration instance.

For example, if your Oracle Integration instance is `https://myhost.example.com/ic/home`, when you go to that URL, you are redirected to a URL such as:

```
https://idcs-c2881.identity.myhost.example.com/ui/v1/signin
```

2. Replace `/signin` with `/adminconsole` to access Oracle Identity Cloud Service. For example:

   ```
   https://idcs-c2881.identity.myhost.example.com/ui/v1/adminconsole
   ```

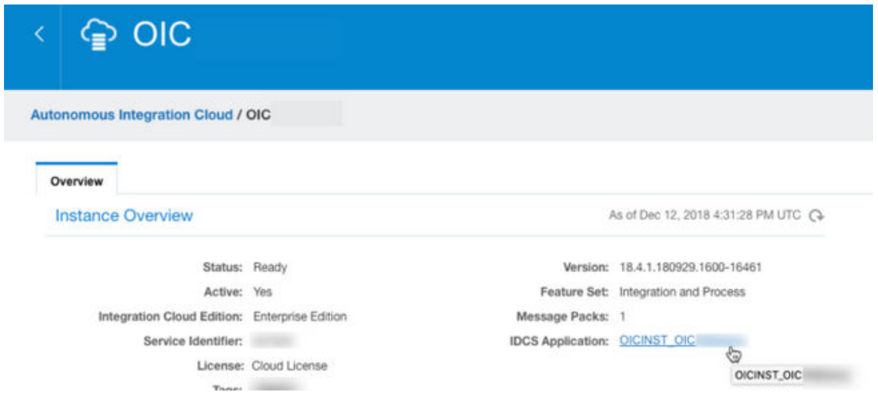
   You'll be prompted to sign in again to the Oracle Identity Cloud Service Console.
3. Log in to the Oracle Identity Cloud Service Console with your identity domain administrator credentials.

- Check the Oracle Integration application in Oracle Identity Cloud Service.
  When an Oracle Integration instance is provisioned, an Oracle Identity Cloud Service application is created for that Oracle Integration instance. The application name is `OICINST_`*`service_instance_name`*.
  1. Log in to the Oracle instance to get the service instance name.

     ```
     https://myhost.example.com/ic/home
     ```

  2. Log in to Oracle Identity Cloud Service to get the application.
  3. Go to **Applications** and find the application with the above name to access the application.

  Alternatively, you can find the application through the Oracle Cloud Dashboard. When you click the **IDCS Application** link on the details page of the Oracle Integration instance (for this example, named **OIC**), it opens the Oracle Identity Cloud Service application for Oracle Integration that is already created.

  As of Dec 12, 2018 4:31:28 PM UTC

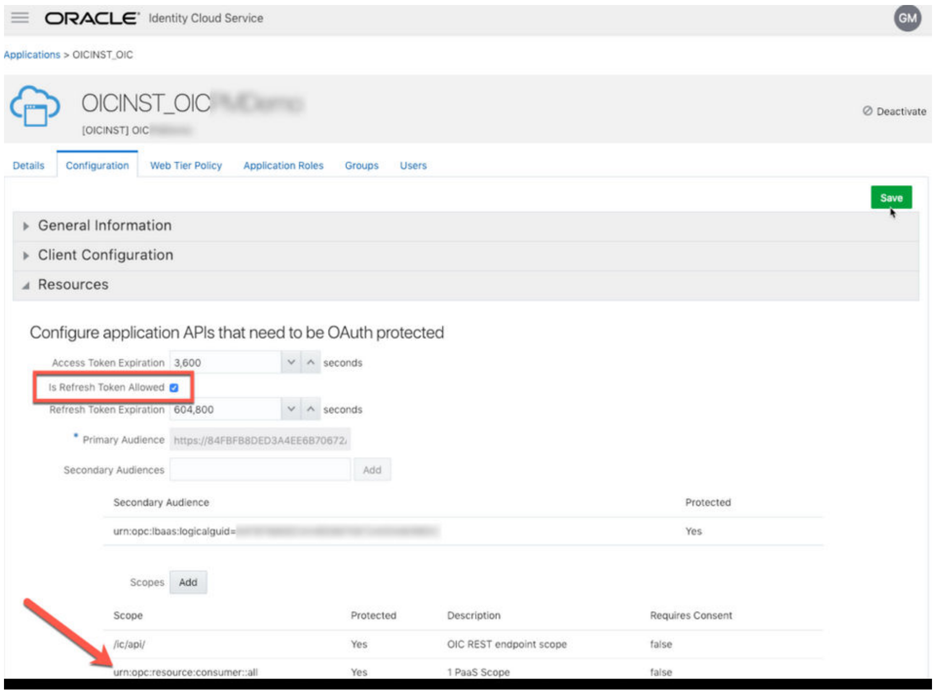
  Status: Ready
  Active: Yes
  Integration Cloud Edition: Enterprise Edition
  Service Identifier: [redacted]
  License: Cloud License
  Tags: [redacted]
  Version: 18.4.1.180929.1600-16461
  Feature Set: Integration and Process
  Message Packs: 1
  IDCS Application: OICINST_OIC[redacted]
  OICINST_OIC[redacted]

Prerequisites for JWT User Assertion

Perform the following tasks.

- Validate the Oracle Integration application and user roles.
  1. Verify that the **Is Refresh Token Allowed** option is enabled for the Oracle Identity Cloud Service application.

2. Check under the **Configuration** > **Resources** section of the application. Note also that there is a special scope predefined (**urn:opc:resource:consumer::all**), which can trigger integrations using OAuth.



OICINST_OIC[redacted]
[OICINST] OIC[redacted]

Configure application APIs that need to be OAuth protected

Access Token Expiration	3,600	seconds
Is Refresh Token Allowed	✓	
Refresh Token Expiration	604,800	seconds
* Primary Audience	https://84FBFB8DED3A4EE6B70672	
Secondary Audiences		Add

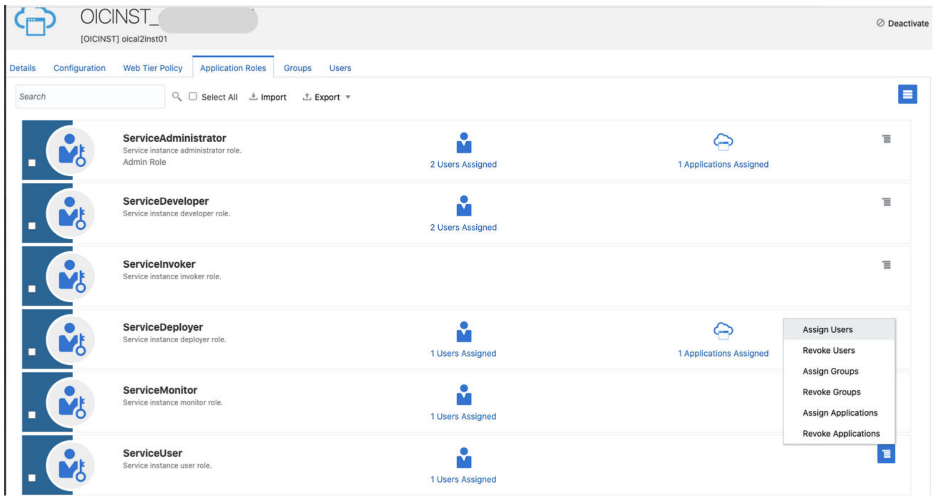
Secondary Audience	Protected
urn:opc:lbaas:logicalguid=[redacted]	Yes

Scopes Add

Scope	Protected	Description	Requires Consent
/ic/api/	Yes	OIC REST endpoint scope	false
urn:opc:resource:consumer::all	Yes	1 PaaS Scope	false

3. Add the appropriate users to the various Oracle Integration roles. For standard/production configurations, use the **ServiceUser** role. (See Oracle Integration Service Roles in *Provisioning and Administering Oracle Integration 3*.)

4. To assign the user, go to the **Application Roles** section of the application.



OICINST_[redacted]
[OICINST] oical2inst01

ServiceAdministrator Service instance administrator role. Admin Role	2 Users Assigned	1 Applications Assigned
ServiceDeveloper Service instance developer role.	2 Users Assigned	
ServiceInvoker Service instance invoker role.		
ServiceDeployer Service instance deployer role.	1 Users Assigned	1 Applications Assigned
ServiceMonitor Service instance monitor role.	1 Users Assigned	
ServiceUser Service instance user role.	1 Users Assigned	

- Generate the key:

1. Generate the self-signed key pair.

   ```
   keytool -genkey -keyalg RSA -alias <your_alias> -keystore
   <keystore_file> -storepass <password> -validity 365 -keysize 2048

   ##example
   keytool -genkey -keyalg RSA -alias assert -keystore
   sampleKeystore.jks -storepass samplePasswd -validity 365 -keysize 2048
   ```

2. Export the public key for signing the JWT assertion.

   ```
   keytool -exportcert -alias <your_alias> -file <filename> -keystore
   <keystore_file> -storepass <password>

   ##example
   keytool -exportcert -alias assert -file assert.cer -keystore
   sampleKeystore.jks -storepass samplePasswd

   ## This should show a success message e.g. Certificate stored in file
   <assert.cer>
   ```

3. Convert the keystore to P12 format.

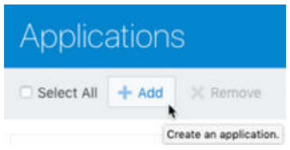
   ```
   keytool -importkeystore -srckeystore <filename> -srcstorepass
   <password> -srckeypass <password> -srcalias <your_alias> -destalias
   <your_alias> -destkeystore <destFileName> -deststoretype PKCS12 -
   deststorepass <password> -destkeypass <password>

   ##example
   keytool -importkeystore -srckeystore sampleKeystore.jks -srcstorepass
   samplePasswd -srckeypass samplePasswd -srcalias assert -destalias
   assert -destkeystore assert.p12 -deststoretype PKCS12 -deststorepass
   samplePasswd -destkeypass samplePasswd

   ## This should show a success message e.g. Importing keystore
   sampleKeystore.jks to assert.p12...
   ```

4. Export the private key from the P12 keystore.

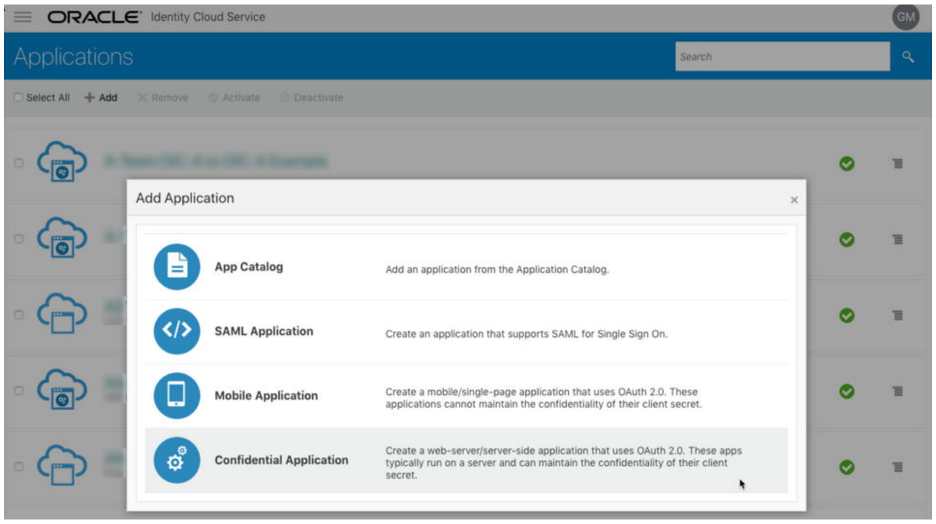
   ```
   openssl pkcs12 -in <destFileName> -nodes -nocerts -out <pem_file>

   ##example
   openssl pkcs12 -in assert.p12 -nodes -nocerts -out private_key.pem

   ## This should show a success message: MAC verified OK
   ```

- Configure the client application:
  To trigger the integration with OAuth, a client application is required.

  1. In the Oracle Identity Cloud Service Console, go to the **Applications** section to create a new application that allows you to trigger an integration with OAuth.

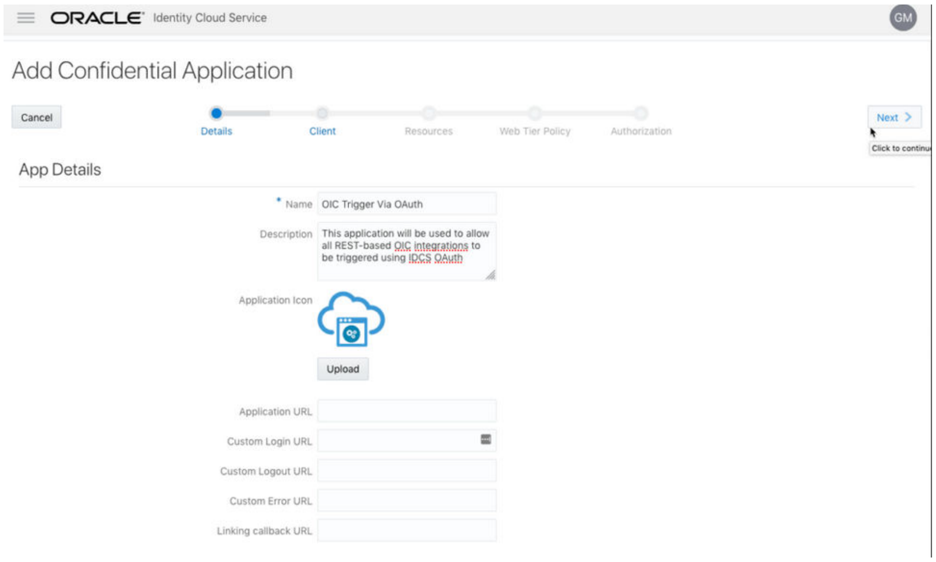


2. Click **Add**.
3. Select **Confidential Application**.



App Catalog — Add an application from the Application Catalog.
SAML Application — Create an application that supports SAML for Single Sign On.
Mobile Application — Create a mobile/single-page application that uses OAuth 2.0. These applications cannot maintain the confidentiality of their client secret.
Confidential Application — Create a web-server/server-side application that uses OAuth 2.0. These apps typically run on a server and can maintain the confidentiality of their client secret.

4. Complete the **Details** page, and go to the **Client** page.

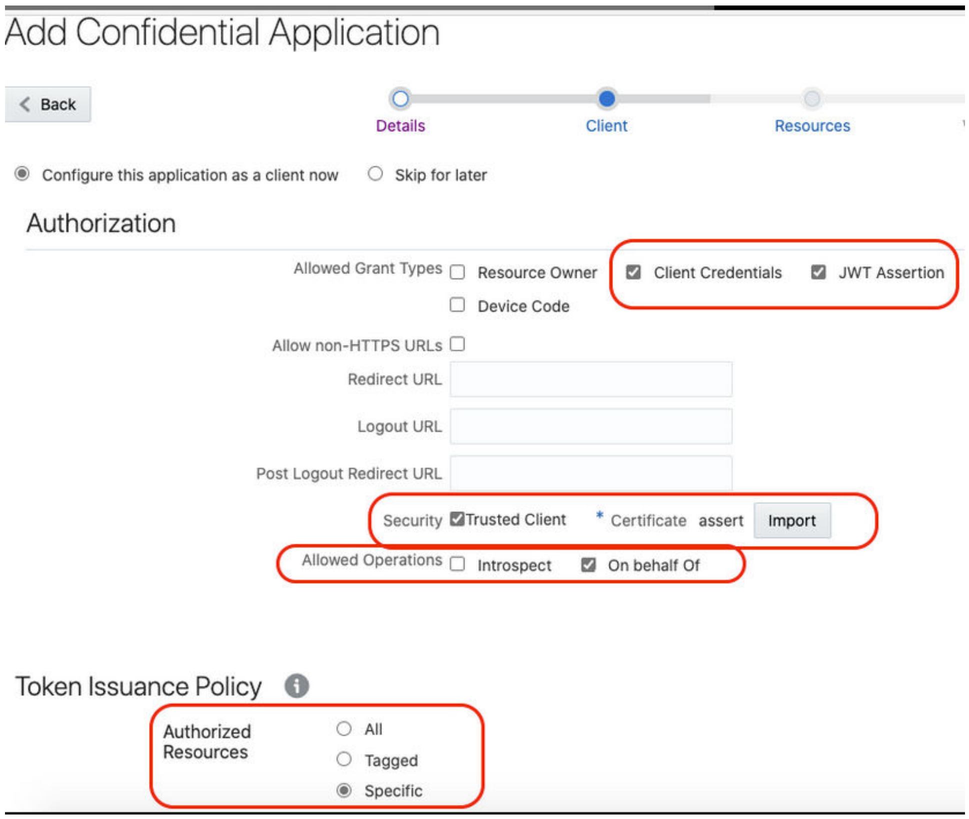


Name: OIC Trigger Via OAuth
Description: This application will be used to allow all REST-based OIC integrations to be triggered using IDCS OAuth
Application Icon
Application URL
Custom Login URL
Custom Logout URL
Custom Error URL
Linking callback URL

5. On the **Client** page, select **Configure this application as a client now** and add the following.
   a. Select **Client Credentials** and **JWT Assertion** for the **Allowed Grant Types**.

b. In the **Security** section, select **Trusted Client** and upload the certificate created in the previous section (Generate the key - Step 2).

c. Select **Specific** in the **Authorized Resources** section.


Add Confidential Application

Back | Details | Client | Resources

(● Configure this application as a client now) ○ Skip for later

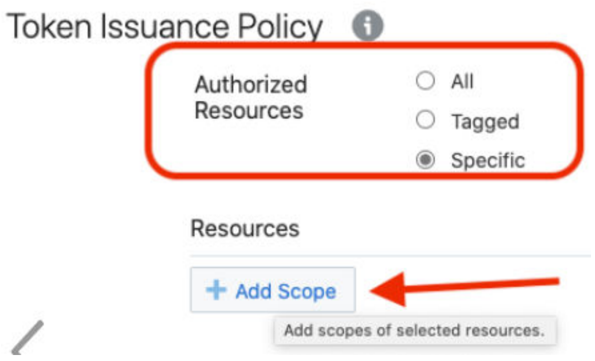
Authorization

Allowed Grant Types	☐ Resource Owner ☑ Client Credentials ☑ JWT Assertion ☐ Device Code
Allow non-HTTPS URLs	☐
Redirect URL	
Logout URL	
Post Logout Redirect URL	
Security	☑ Trusted Client * Certificate assert Import
Allowed Operations	☐ Introspect ☑ On behalf Of

Token Issuance Policy

Authorized Resources: ○ All ○ Tagged [✓ Specific]

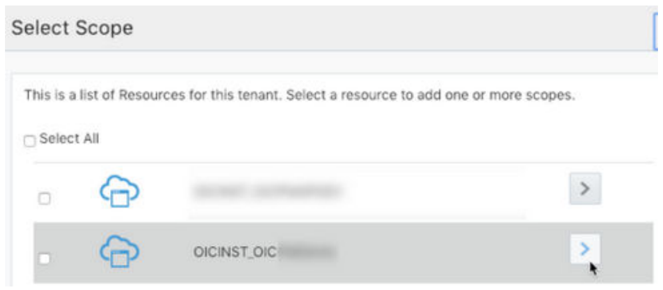
d. Click **Add Scope** under the **Resources** section.


Token Issuance Policy

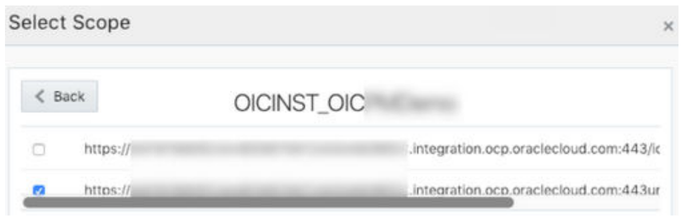
Authorized Resources: ○ All ○ Tagged [✓ Specific]

Resources

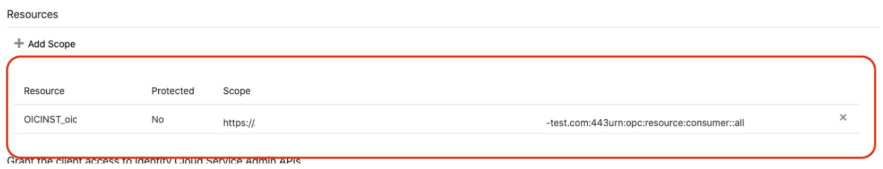
e. Find the Oracle Integration application, and click **>**.


Select Scope
This is a list of Resources for this tenant. Select a resource to add one or more scopes.
Select All
OICINST_OIC

f. Add the scope containing **urn:opc:resource:consumer::all**, and click **>**.


Select Scope
Back
OICINST_OIC
https:// .integration.ocp.oraclecloud.com:443/ic
https:// .integration.ocp.oraclecloud.com:443ur

The scope containing **urn:opc:resource:consumer::all** is added.


Resources
Add Scope
Grant the client access to Identity Cloud Service Admin APIs

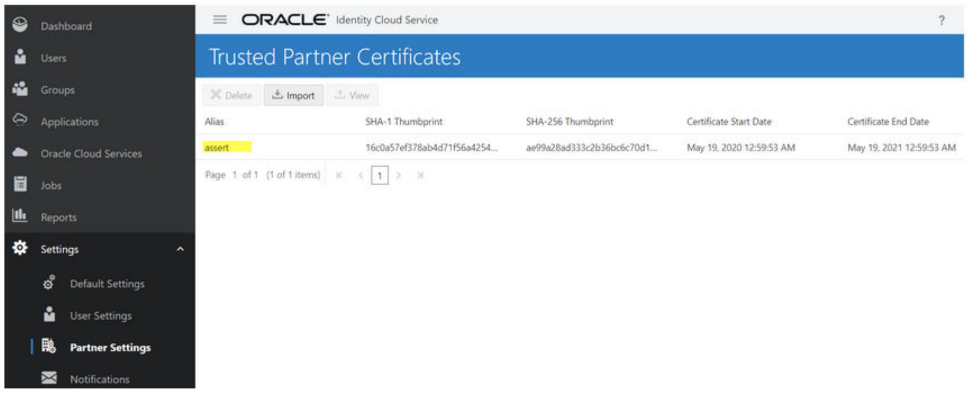
Resource	Protected	Scope
OICINST_oic	No	https:// -test.com:443urn:opc:resource:consumer::all

g. Save your changes.

6. Click through the remaining wizard pages without making changes and save the application.

7. Activate the application for use.

- Add a certificate as a trusted partner:
  Even though you imported the signing certificate in the application, Oracle Identity Cloud Service requires you to also have the certificate as a trusted partner certificate. Upload the certificate created in the previous section. (See Generate the key - Step 2.)



Alias	SHA-1 Thumbprint	SHA-256 Thumbprint	Certificate Start Date	Certificate End Date
assert	16c0a57ef378ab4d71f56a4254...	ae99a28ad333c2b36bc6c70d1...	May 19, 2020 12:59:53 AM	May 19, 2021 12:59:53 AM

- Generate the JWT user assertion:
  1. Generate the JWT user assertion using the generated private key and simple Java code.

     Note:

     You can use the https://github.com/jwtk/jjwt library to generate the user assertion. There are many libraries listed at https://jwt.io/ for multiple technologies.

     ```
     Sample:
     header:
     {
     "alg": "RS256",
     "typ": "JWT",
     "kid": "assert"
     }

     payload:
     {
     "sub": "ssaInstanceAdmin",
     "jti": "8c7df446-bfae-40be-be09-0ab55c655436",
     "iat": 1589889699,
     "exp": 1589909699,
     "iss": "d702f5b31ee645ecbc49d05983aaee54",
     "aud": "https://identity.oraclecloud.com/"
     }
     ```

  Where:
  - `sub` specifies the user name for whom user assertion is generated.
  - `jti` is a unique identifier
  - `iat` is issued (epoch seconds).
  - `exp` is the token expiry (epoch seconds).
  - `iss` is the client ID.
  - `aud` must include the Oracle Identity Cloud Service audience `https://identity.oracle.com/`. The signing algorithm must be RS256.
  - `kid` specifies the key to use to verify the signature. Therefore, it must match with the uploaded certificate alias in Oracle Identity Cloud Service.
- Validate the client application:
  1. Once you generate the JWT user assertion, generate the Oracle Identity Cloud Service access token as follows.

     ```
     ##Syntax
     curl -i -H 'Authorization: Basic <base64Encoded clientid:secret>' -H
     'Content-Type: application/x-www-form-urlencoded;charset=UTF-8' --
     request POST https://<IDCS-Service-Instance>.identity.oraclecloud.com/
     oauth2/v1/token -d 'grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-
     ```

```
type%3Ajwt-bearer&assertion=<user assertion>&scope=<app_scope>'

###where
#### grant type - urn:ietf:params:oauth:grant-type:jwt-bearer
#### <base64-clientid-secret> - Base 64 encode
clientId:ClientSecret
#### <user assertion> - User assertion generated
#### <app scope> - Scope added while creating application in
client configuration section (Ends with
urn:opc:resource:consumer::all)
```

2. Capture the `access_token` from the response.

```
{
    "access_token": "eyJ4NXQjG...dfsdfsFgets2ed",
    "token_type": "Bearer",
    "expires_in": 3600
}
```

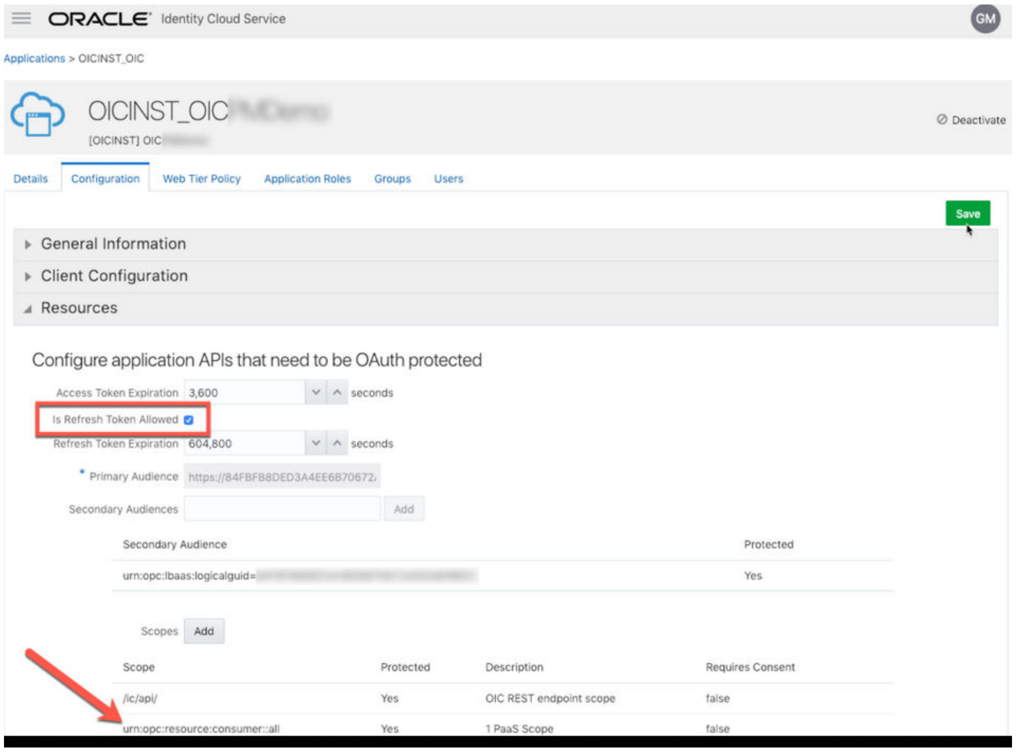
3. Use an `access_token` in the authorization header to invoke the Oracle Integration trigger endpoint.

```
curl --location --request GET 'https://OIC host/OIC endpoint' \
--header 'Authorization: Bearer eyJ4NXQjG...dfsdfsFgets2ed'
```

Prerequisites for Authorization Code

Perform the following tasks.

- Validate the Oracle Integration application and user roles:
  1. Verify that the **Is Refresh Token Allowed** option is enabled for the Oracle Identity Cloud Service application.
  2. Check the **Configuration** > **Resources** section of the application. Note also that there is a special predefined scope (**urn:opc:resource:consumer::all**) that permits triggering of the Oracle Integration integrations using OAuth.



Configure application APIs that need to be OAuth protected

Access Token Expiration	3,600	seconds
Is Refresh Token Allowed	✓	
Refresh Token Expiration	604,800	seconds
* Primary Audience	https://84FBFB8DED3A4EE6B70672.	
Secondary Audiences		Add

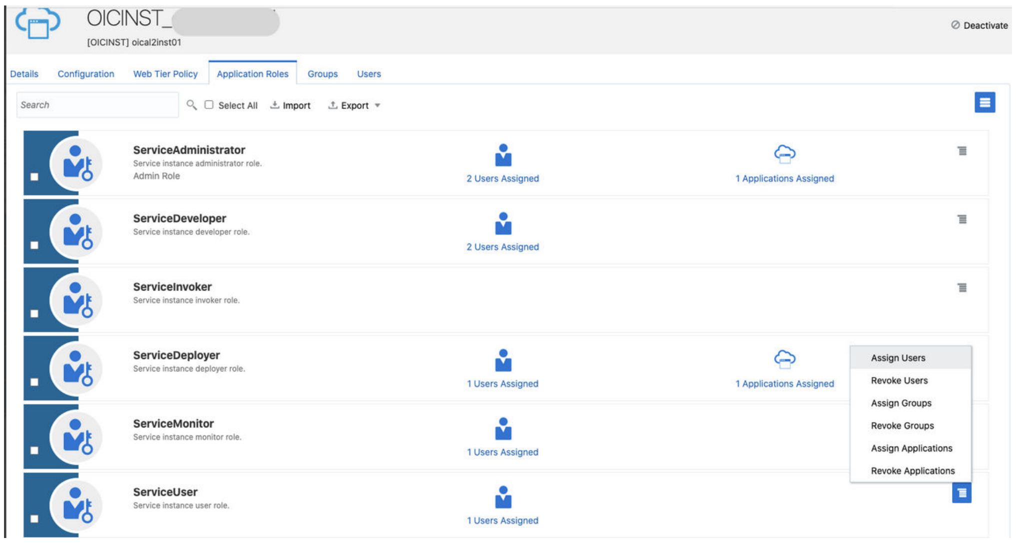
Secondary Audience	Protected
urn:opc:lbaas:logicalguid=	Yes

Scopes Add

Scope	Protected	Description	Requires Consent
/ic/api/	Yes	OIC REST endpoint scope	false
urn:opc:resource:consumer::all	Yes	1 PaaS Scope	false

3. Add the appropriate users to the various Oracle Integration roles. For standard/ production configurations, use the **ServiceUser** role. (See Oracle Integration Service Roles in *Provisioning and Administering Oracle Integration 3*.)

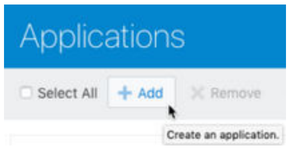
4. To assign the user, go to the **Application Roles** section of the application.



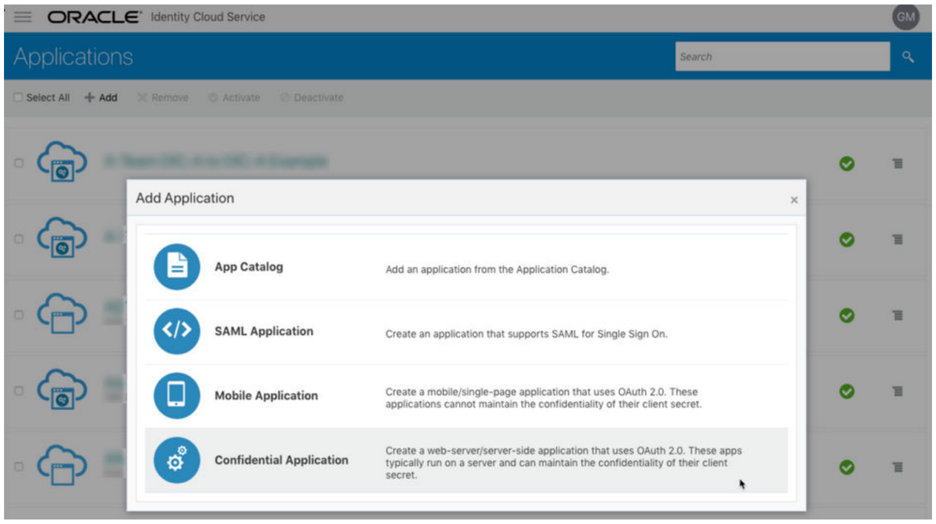
ServiceAdministrator Service instance administrator role. Admin Role	2 Users Assigned	1 Applications Assigned
ServiceDeveloper Service instance developer role.	2 Users Assigned	
ServiceInvoker Service instance invoker role.		
ServiceDeployer Service instance deployer role.	1 Users Assigned	1 Applications Assigned
ServiceMonitor Service instance monitor role.	1 Users Assigned	
ServiceUser Service instance user role.	1 Users Assigned	

- Configure the client application:
  To allow you to trigger the Oracle Integration integration with OAuth, the client application is required.

1. In the Oracle Identity Cloud Service Console, go to the **Applications** section to create a new application that allows you to trigger the Oracle Integration integration with OAuth.

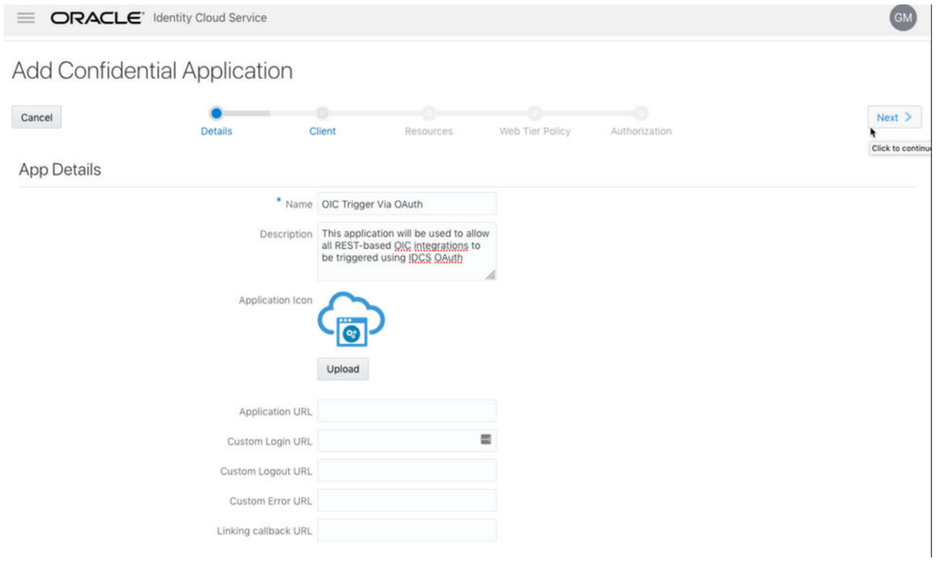


2. Select **Confidential Application**.



App Catalog — Add an application from the Application Catalog.
SAML Application — Create an application that supports SAML for Single Sign On.
Mobile Application — Create a mobile/single-page application that uses OAuth 2.0. These applications cannot maintain the confidentiality of their client secret.
Confidential Application — Create a web-server/server-side application that uses OAuth 2.0. These apps typically run on a server and can maintain the confidentiality of their client secret.

3. Complete the **Details** page, and go to the **Client** page.



* Name	OIC Trigger Via OAuth
Description	This application will be used to allow all REST-based OIC integrations to be triggered using IDCS OAuth
Application Icon	
Application URL	
Custom Login URL	
Custom Logout URL	
Custom Error URL	
Linking callback URL	

4. On the **Client** page, select **Configure this application as a client now** and add the following.

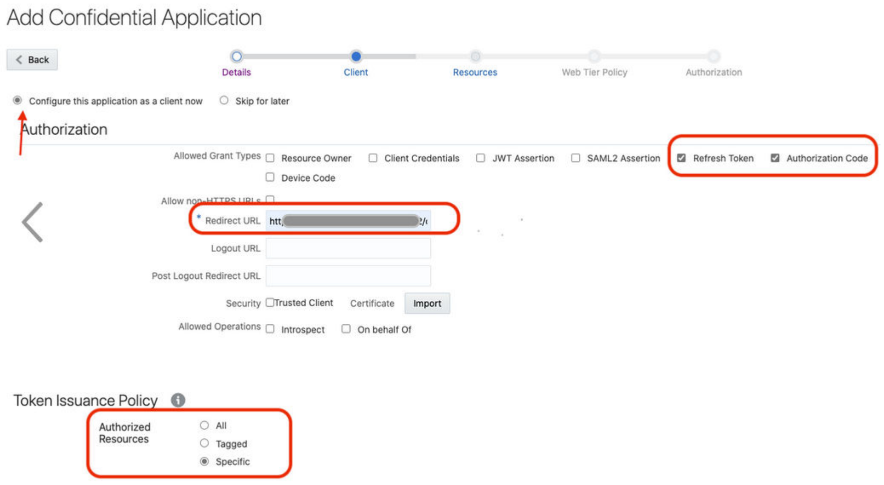
a. Select **Refresh Token** and **Authorization Code** for **Allowed Grant Types**.

b. Set the redirect URL to the URL of the client application. After user login, Oracle Identity Cloud Service redirects to this URL with the authorization code.

> **Note:**
>
> If you don't know the following information, check with your administrator:
>
> – If your instance is new or upgraded from Oracle Integration Generation 2 to Oracle Integration 3.
>
> – The complete instance URL with the region included (required for new instances).

For Connections…	Include the Region as Part of the Redirect URL?	Example of Redirect URL to Specify…
Created on new Oracle Integration 3 instances	Yes.	`https://`*`OIC_instance_URL`*`.`*`region`*`.ocp.oraclecloud.com/icsapis/agent/oauth/callback`
Created on instances upgraded from Oracle Integration Generation 2 to Oracle Integration 3	No. This applies to both: – New connections created after the upgrade – Existing connections that were part of the upgrade	`https://`*`OIC_instance_URL`*`.ocp.oraclecloud.com/icsapis/agent/oauth/callback`

c. Select **Specific** in the **Authorized Resources** section.



Add Confidential Application

Back | Details | Client | Resources | Web Tier Policy | Authorization

[✓ Configure this application as a client now] Skip for later

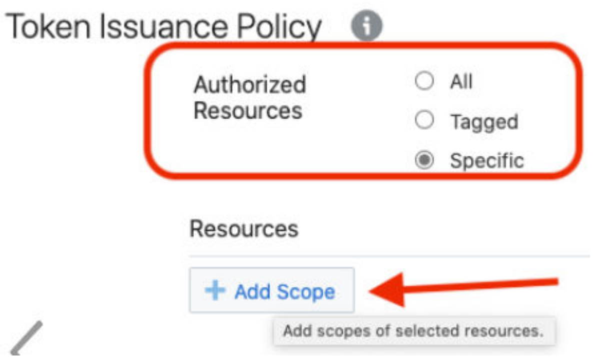
Authorization

Allowed Grant Types	☐ Resource Owner ☐ Client Credentials ☐ JWT Assertion ☐ SAML2 Assertion ☑ Refresh Token ☑ Authorization Code ☐ Device Code
Allow non-HTTPS URLs	☐
* Redirect URL	htt[redacted]2/(
Logout URL	
Post Logout Redirect URL	
Security	☐ Trusted Client Certificate Import
Allowed Operations	☐ Introspect ☐ On behalf Of

Token Issuance Policy

Authorized Resources: All Tagged [✓ Specific]

d. Click **Add Scope** under the **Resources** section.

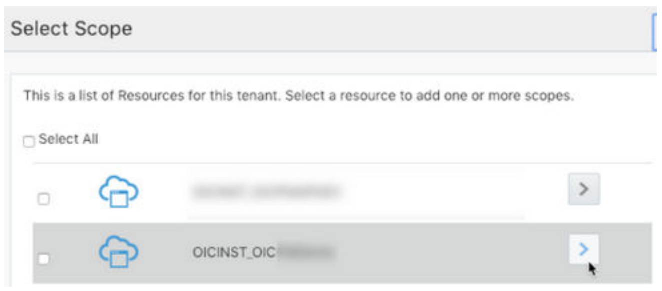


Token Issuance Policy

Authorized Resources: All Tagged [✓ Specific]

Resources

e. Find the Oracle Integration application, and click **>**.



Select Scope

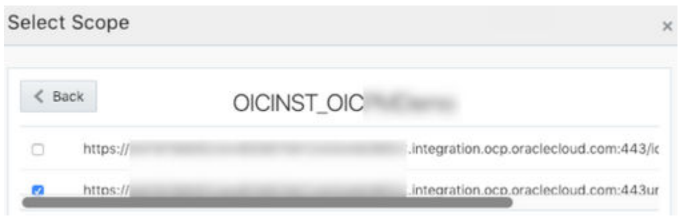
This is a list of Resources for this tenant. Select a resource to add one or more scopes.

☐ Select All

☐ [redacted] >

☐ OICINST_OIC[redacted] >

f. Add the scope containing **urn:opc:resource:consumer::all**, and click **>**.



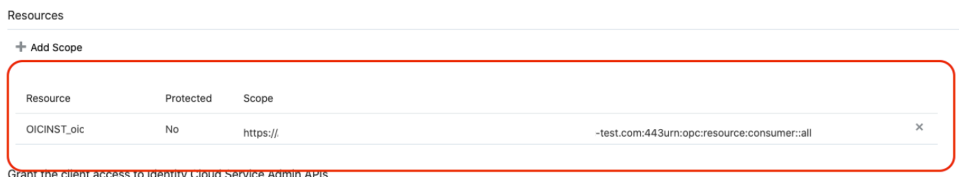
Select Scope ×

Back OICINST_OIC[redacted]

☐ https://[redacted].integration.ocp.oraclecloud.com:443/ic

☑ https://[redacted].integration.ocp.oraclecloud.com:443ur

The scope containing **urn:opc:resource:consumer::all** is added.



Resource	Protected	Scope
OICINST_oic	No	https://. -test.com:443urn:opc:resource:consumer::all

g. Save your changes.

5. Click through the remaining wizard pages without making changes and save the application.
6. Activate the application for use.

- Validate the client application:

1. To fetch the authorization code, make the following request from the browser.

```
##Syntax
GET https://<IDCS-Service-Instance>.identity.oraclecloud.com/
oauth2/v1/authorize?client_id=<client-
id>&response_type=code&redirect_uri=<client-redirect-
uri>&scope=<app_scope>%20offline_access&nonce=<nonce-
value>&state=<unique_value>

###where
#### <client-id> - ID of Client application generated.
#### <client-redirect-uri> - Redirect URI, in client application.
#### <app_scope> - scope added while creating application in client
configuration. (Ends with urn:opc:resource:consumer::all)
#### nonce - Optional, unique value to mitigate replay attacks
#### state - Recommended, Opaque to IDCS. Value used to maintain
state between the request and the callback
##Example
GET https://<idcs-host>/oauth2/v1/authorize?
client_id=<clientID>&response_type=code&redirect_uri=https://
app.getpostman.com/oauth2/callback&scope=https://
<Resource_APP_Audience>urn:opc:resource:consumer::all%20offline_access
&nonce=121&state=12345544
```

2. If the user is not already logged in, Oracle Identity Cloud Service challenges the user to authenticate. Oracle Identity Cloud Service checks the user's credentials. (For authentication, the user assigned the **ServiceUser** role must be used.)
After authentication is successful, Oracle Identity Cloud Service redirects back to the client redirect URL with the authorization code and state added to the URL.

```
##Response URL
https://<redirect_URL>?code=<code_value>=&state=<state_value>

###Client should validate state received is same as one sent in
request.
```

3. Capture the code value from the above response and make the following request to Oracle Identity Cloud Service to get the access token.

   ```
   ##Syntax
   curl -i -H 'Authorization: Basic <base64-clientid-secret>' -H
   'Content-Type: application/x-www-form-urlencoded;charset=UTF-8'
   --request POST https://<IDCS-Service-
   Instance>.identity.oraclecloud.com/oauth2/v1/token -d
   'grant_type=authorization_code&code=<authz-
   code>&redirect_uri=<client-redirect-uri>

   ###where
   #### <base64-clientid-secret> - BAse 64 encode
   clientId:ClientSecret
   #### <authz-code> - code value received as response on redirect.
   #### <client-redirect-uri> - Redirect URI, in client application.

   ##Example
   curl -i -H 'Authorization: Basic MDMx..NGY1' -H 'Content-Type:
   application/x-www-form-urlencoded;charset=UTF-8' --request POST
   https://<idcs_host>/oauth2/v1/token -d
   'grant_type=authorization_code&code=AQAg...3jKM4Gc=&redirect_uri=
   https://app.getpostman.com/oauth2/callback
   ```

4. Capture the `access_token` and `refresh_token` from the response.

   ```
   {
       "access_token": "eyJ4NXQjG...dfsdfsFgets2ed",
       "token_type": "Bearer",
       "expires_in": 3600,
       "refresh_token": "AQAgY2MzNjVlOTVhOTRh...vM5S0MkrFSpzc="
   }
   ```

5. Use the `access_token` in the authorization header to invoke the Oracle Integration trigger endpoint.

   ```
   curl --location --request GET 'https://OIC host/OIC endpoint' \
   --header 'Authorization: Bearer eyJ4NXQjG...dfsdfsFgets2ed'
   ```

6. To update the access token, use the refresh token and make the request to Oracle Identity Cloud Service.

7. Capture the `access_token` and `refresh_token` from a response for further use.

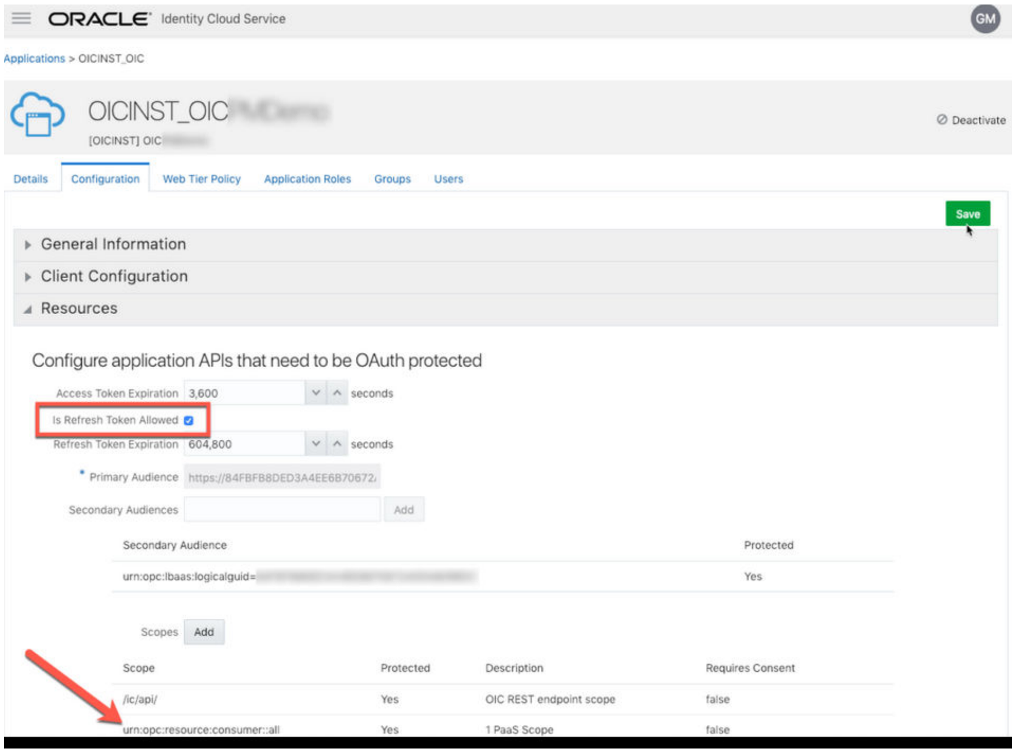
   ```
   curl -i -H 'Authorization: Basic <base64-clientid-secret>' -H
   'Content-Type: application/x-www-form-urlencoded;charset=UTF-8'
   --request POST https://<IDCS-Service-
   Instance>.identity.oraclecloud.com/oauth2/v1/token  -d
   'grant_type=refresh_token&refresh_token=<refresh_token>'


   ##Example
   curl -i -H 'Authorization: Basic OGQyM...ZDA0Mjcz' -H 'Content-
   Type: application/x-www-form-urlencoded;charset=UTF-8' --request
   ```

```
POST https://IDCS-Service-Instance.identity.oraclecloud.com/oauth2/v1/
token  -d
'grant_type=refresh_token&refresh_token=AQAgY2MzNjVlOTVhOTRh...vM5S0Mk
rFSpzc='
```

Prerequisites for Resource Owner Password Credentials

Perform the following tasks.

- Validate the Oracle Integration application and user roles:
  1. Verify that the **Is Refresh Token Allowed** option is enabled for the Oracle Integration Oracle Identity Cloud Service application.
  2. Check under the **Configuration** > **Resources** section of **Applications**. Note also that there is a special predefined scope (**urn:opc:resource:consumer::all**) that allows the triggering of integrations with OAuth.



Configure application APIs that need to be OAuth protected

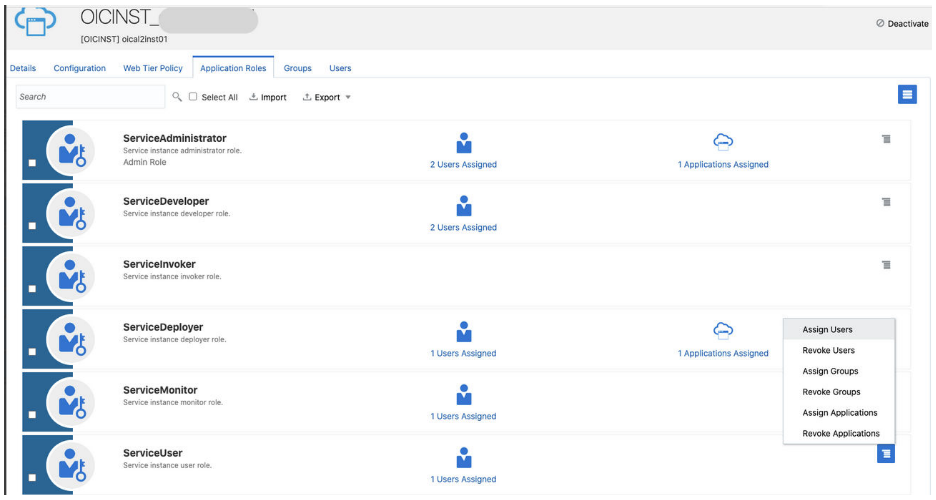
Access Token Expiration 3,600 seconds
Is Refresh Token Allowed ✓
Refresh Token Expiration 604,800 seconds
* Primary Audience https://84FBFB8DED3A4EE6B70672
Secondary Audiences Add

Secondary Audience	Protected
urn:opc:lbaas:logicalguid=[redacted]	Yes

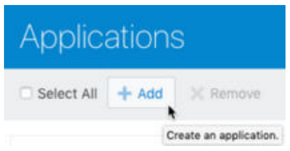
Scopes Add

Scope	Protected	Description	Requires Consent
/ic/api/	Yes	OIC REST endpoint scope	false
urn:opc:resource:consumer::all	Yes	1 PaaS Scope	false

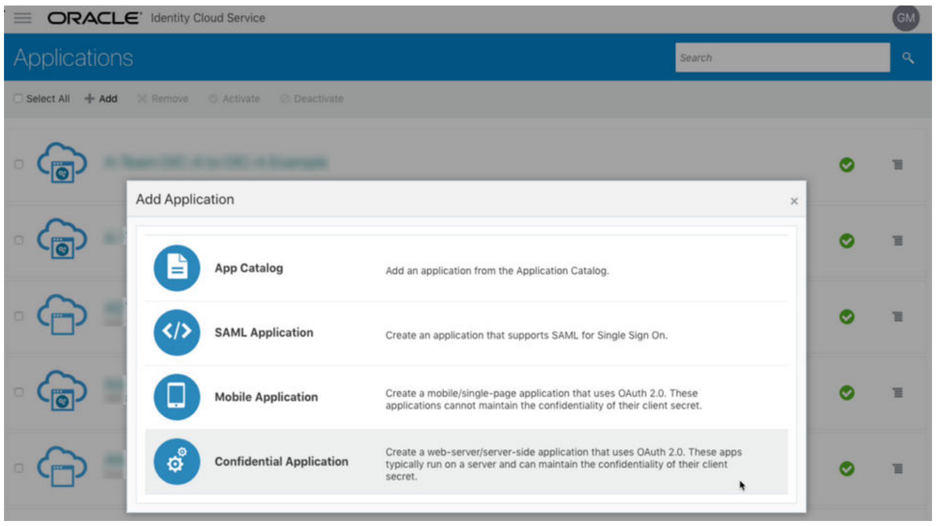
  3. Add the appropriate users to the various Oracle Integration roles. For standard/ production configurations, use the **ServiceUser** role. (See Oracle Integration Service Roles in *Provisioning and Administering Oracle Integration 3*.)
  4. To assign the user, go to the **Application Roles** section of the application.



- Configure the client application:
  To trigger the integration with OAuth, a client application is required.
  1. In the Oracle Identity Cloud Service Console, go to the **Applications** section to create a new application that allows you to trigger an integration with OAuth.

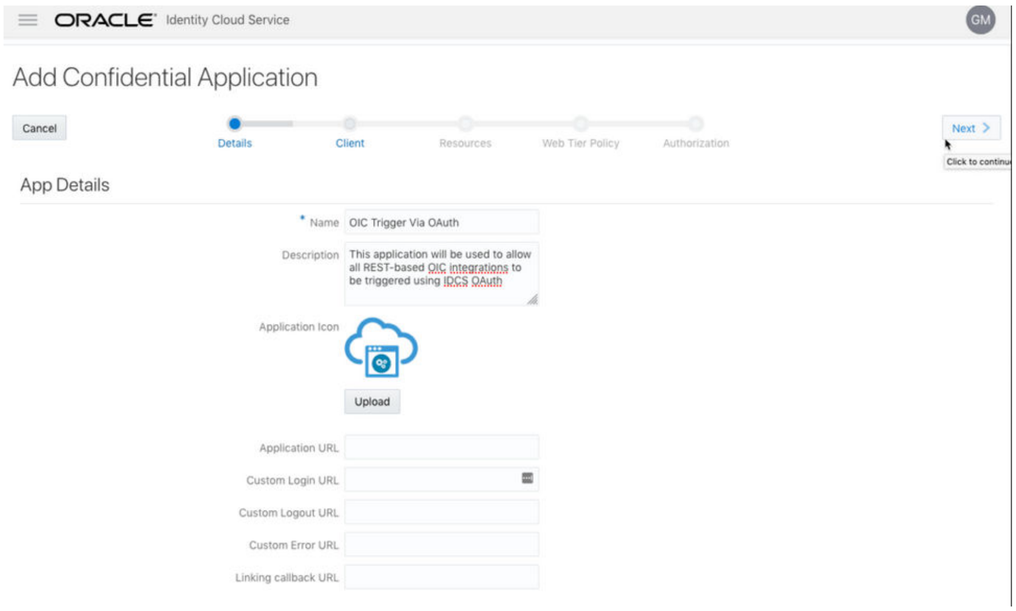
     

  2. Click **Add**.
  3. Select **Confidential Application**.

App Catalog	Add an application from the Application Catalog.
SAML Application	Create an application that supports SAML for Single Sign On.
Mobile Application	Create a mobile/single-page application that uses OAuth 2.0. These applications cannot maintain the confidentiality of their client secret.
Confidential Application	Create a web-server/server-side application that uses OAuth 2.0. These apps typically run on a server and can maintain the confidentiality of their client secret.

  4. Complete the **Details** page, and go to the **Client** page.



Add Confidential Application

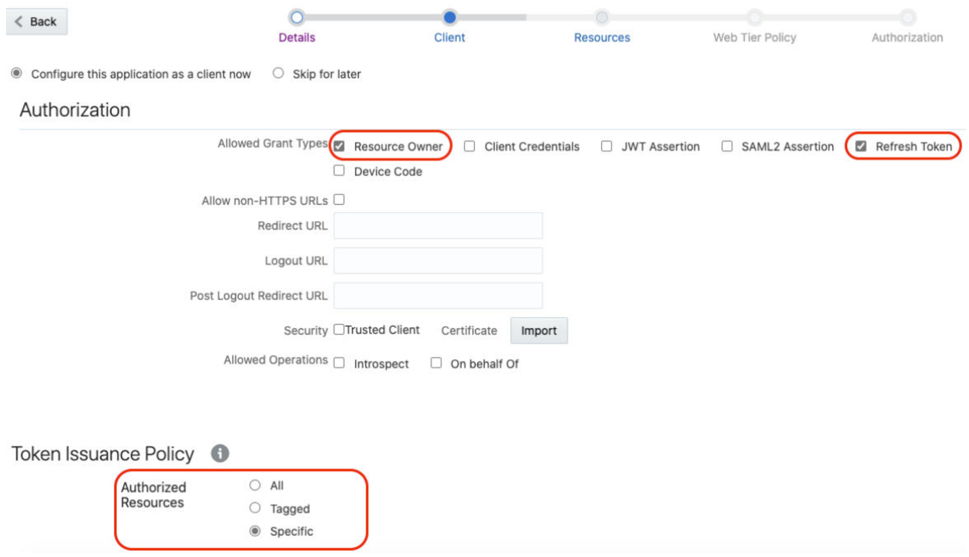
App Details

* Name: OIC Trigger Via OAuth

Description: This application will be used to allow all REST-based OIC integrations to be triggered using IDCS OAuth

Application URL
Custom Login URL
Custom Logout URL
Custom Error URL
Linking callback URL

5. On the **Client** page, select **Configure this application as a client now** and add the following.
   a. Select **Resource Owner** and **Refresh Token** for **Allowed Grant Types**.
   b. Select **Specific** in the **Authorized Resources** section.



[✓ Configure this application as a client now] Skip for later

Authorization

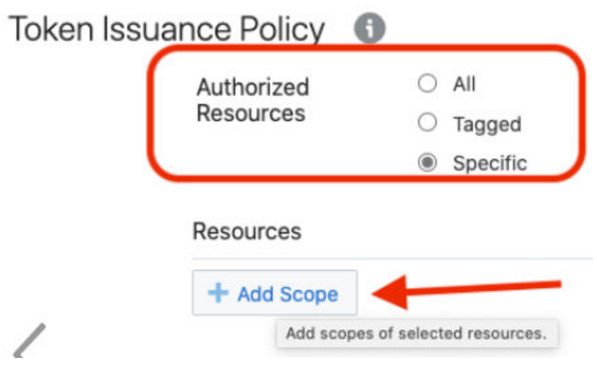
Allowed Grant Types: [✓ Resource Owner] Client Credentials JWT Assertion SAML2 Assertion [✓ Refresh Token] Device Code

Allow non-HTTPS URLs
Redirect URL
Logout URL
Post Logout Redirect URL
Security: Trusted Client Certificate
Allowed Operations: Introspect On behalf Of

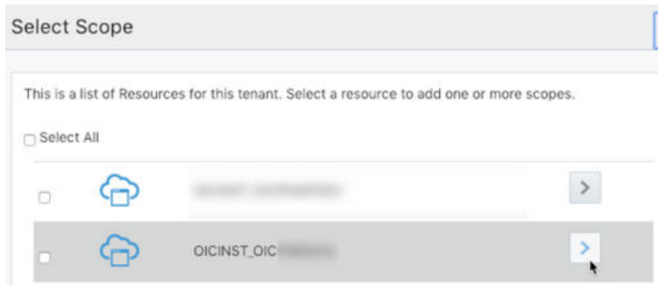
Token Issuance Policy

Authorized Resources: All Tagged [✓ Specific]

   c. Click **Add Scope** under the **Resources** section.



d. Find the Oracle Integration application.


Select Scope

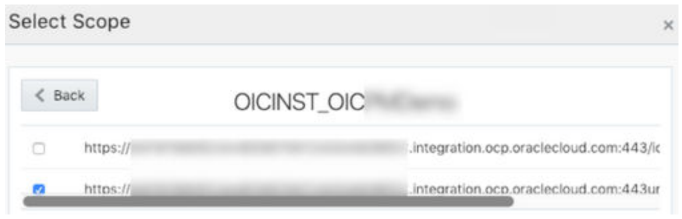
This is a list of Resources for this tenant. Select a resource to add one or more scopes.

Select All

[redacted]

OICINST_OIC[redacted]

e. Add the scope containing **urn:opc:resource:consumer::all**, and click **>**.


Select Scope

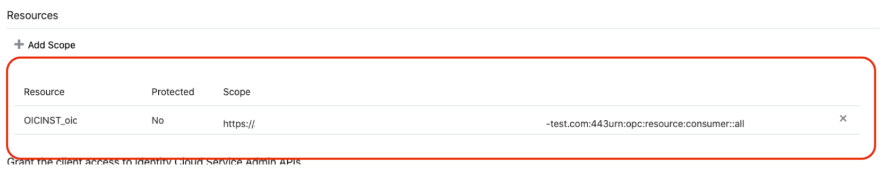
Back

OICINST_OIC[redacted]

https://[redacted].integration.ocp.oraclecloud.com:443/ic

https://[redacted].integration.ocp.oraclecloud.com:443ur

The scope containing **urn:opc:resource:consumer::all** is added.


Resources

Add Scope

Resource	Protected	Scope
OICINST_oic	No	https:// -test.com:443urn:opc:resource:consumer::all

Grant the client access to Identity Cloud Service Admin APIs

f. Save your changes.

6. Click through the remaining wizard pages without making changes and save the application.
7. Activate the application for use.

- Validate the client application:

1. To fetch the access client, make a request to Oracle Identity Cloud Service with the user name and password in the payload.

   ```
   ##Syntax
   curl -i -H 'Authorization: Basic <base64Encoded_clientid:secret>' -H
   'Content-Type: application/x-www-form-urlencoded;charset=UTF-8' --
   request POST https://<IDCS-Service-Instance>.identity.oraclecloud.com/
   oauth2/v1/token -d 'grant_type=password&username=<user-
   name>&password=<password>&scope=<App_Scope>%20offline_access'

   ###where
   #### <base64-clientid-secret> - Base 64 encode clientId:ClientSecret
   #### <username> - user for token needs to be issued (must be in
   serviceuser role).
   #### <password> - password for above user
   #### <app_scope> - Scope added while creating application in client
   configuration section (Ends with urn:opc:resource:consumer::all)
   ##Example
   curl -i -H 'Authorization: Basic OGQyM...ZDA0Mjcz' -H 'Content-Type:
   application/x-www-form-urlencoded;charset=UTF-8' --request POST
   https://<idcs_host>/oauth2/v1/token -d
   'grant_type=password&username=sampleUser&password=SamplePassword&scope
   =https://
   <Resource_APP_Audience>urn:opc:resource:consumer::all%20offline_access
   '
   ```

2. Capture the `access_token` and `refresh_token` from the response.

   ```
   {
       "access_token": "eyJ4NXQjG...dfsdfsFgets2ed",
       "token_type": "Bearer",
       "expires_in": 3600,
       "refresh_token": "AQAgY2MzNjVlOTVhOTRh...vM5S0MkrFSpzc="
   }
   ```

3. Use the `access_token` in the authorization header to invoke the Oracle Integration trigger endpoint.

   ```
   curl --location --request GET 'https://OIC host/OIC endpoint' \
   --header 'Authorization: Bearer eyJ4NXQjG...dfsdfsFgets2ed'
   ```

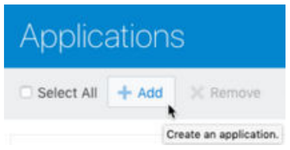
4. To update the access token, use the refresh token and make a request to Oracle Identity Cloud Service.
5. Capture the `access_token` and `refresh_token` from the response for further use.

   ```
   curl -i -H 'Authorization: Basic <base64-clientid-secret>' -H
   'Content-Type: application/x-www-form-urlencoded;charset=UTF-8' --
   request POST https://<IDCS-Service-Instance>.identity.oraclecloud.com/
   oauth2/v1/token  -d
   'grant_type=refresh_token&refresh_token=<refresh_token>'

   ##Example
   curl -i -H 'Authorization: Basic OGQyM...ZDA0Mjcz' -H 'Content-Type:
   application/x-www-form-urlencoded;charset=UTF-8' --request POST
   ```

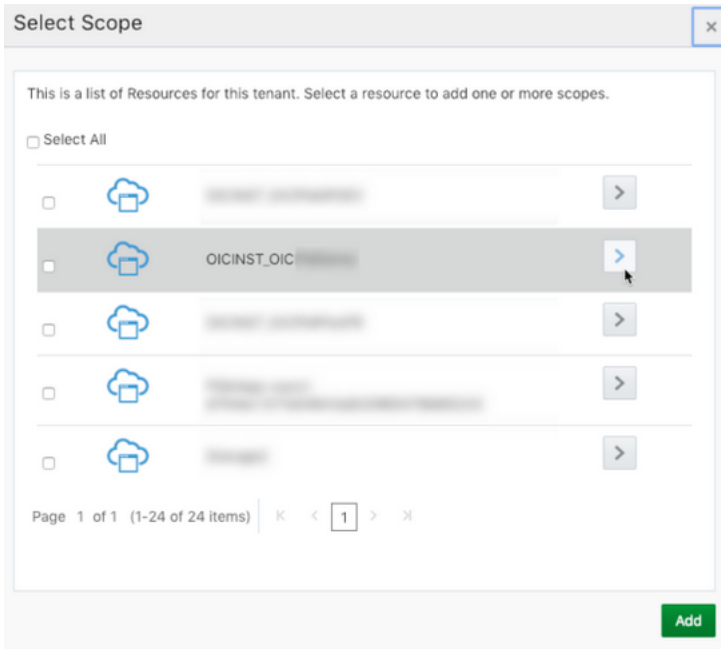
```
https://<IDCS-Service-Instance>.identity.oraclecloud.com/
oauth2/v1/token  -d
'grant_type=refresh_token&refresh_token=AQAgY2MzNjVlOTVhOTRh...vM
5S0MkrFSpzc='
```

Prerequisites for Client Credentials

- Configure the client application.
  1. In the Oracle Identity Cloud Service Console, go to the **Applications** section to create a new application that allows you to trigger an integration with OAuth.

  2. Click **Add**.
  3. Select **Confidential Application**.
  4. Complete the **Details** page, and click **Next**.
  5. On the **Client** page, select **Configure this application as a client now**, and complete the following:
     a. Select **Client Credentials** from the **Allowed Grant Types** list.
     b. Select **Specific** in the **Authorized Resources** area of the **Token Issuance Policy** section.
     c. Click **Add Scope** under the **Resources** section.
     d. Find the Oracle Integration application, and click **>**.

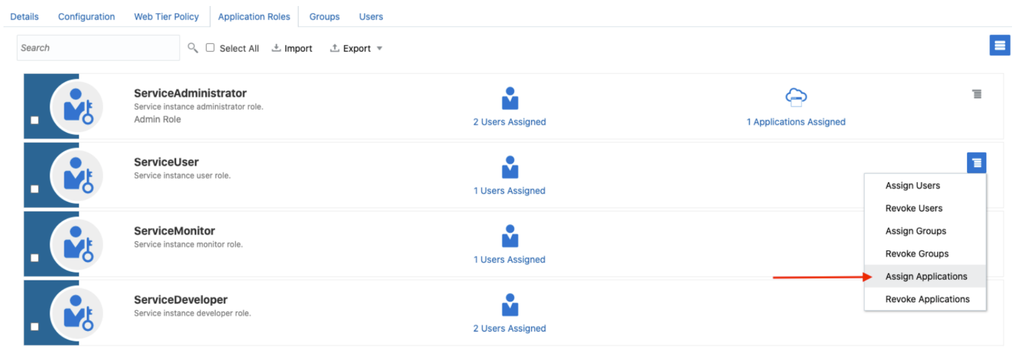
     
     Select Scope
     This is a list of Resources for this tenant. Select a resource to add one or more scopes.
     Select All
     OICINST_OIC
     Page 1 of 1 (1-24 of 24 items)
     1
     Add

     e. Add the scope containing **urn:opc:resource:consumer::all**.

   f. Save your changes.

6. Click through the remaining wizard pages without making changes and save the application.
7. Activate the application for use.

- Add roles to the client application.
  1. Go to the **Application Roles** tab of the Oracle Identity Cloud Service application.
  2. Select **Assign Applications** for the **ServiceUser** role.



- Validate the client application.
  1. Fetch the access client to make an access token request to Oracle Identity Cloud Service with the client credentials.

```
##Syntax
curl -i -H 'Authorization: Basic <base64Encoded clientid:secret>' -H
'Content-Type: application/x-www-form-urlencoded;charset=UTF-8' --
request POST https://<IDCS-Service-Instance>.identity.oraclecloud.com/
oauth2/v1/token -d 'grant_type=client_credentials&scope=<app scope>'
###where
#### <base64-clientid-secret> - Base 64 encode clientId:ClientSecret
#### <app scope> - Scope added while creating application in client
configuration section (Ends with urn:opc:resource:consumer::all)

##Example
curl -i -H 'Authorization: Basic OGQyM...ZDA0Mjcz' -H 'Content-Type:
application/x-www-form-urlencoded;charset=UTF-8' --request POST
https://<idcs_host>/oauth2/v1/token -d
'grant_type=client_credentials&scope=https://<Resource APP
Audience>urn:opc:resource:consumer::all'
```

  2. Capture the `access_token` from the response.

```
{
    "access_token": "eyJ4NXQjG...dfsdfsFgets2ed",
    "token_type": "Bearer",
    "expires_in": 3600
}
```

3. Use the `access_token` in the authorization header to invoke the trigger endpoint.

```
curl --location --request GET 'https://OIC host/OIC endpoint' \
--header 'Authorization: Bearer eyJ4NXQjG...dfsdfsFgets2ed'
```

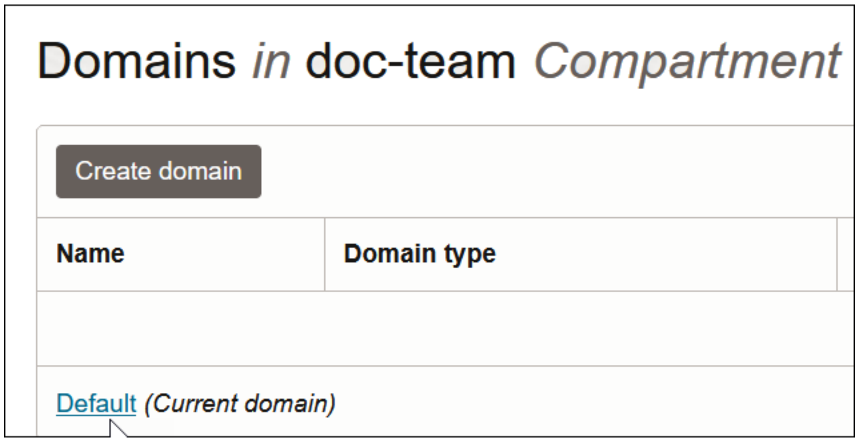
Use OAuth 2.0 Grants in Identity Domain Environments

To use an OAuth 2.0 grant type with this adapter in an identity domain environment of Oracle Integration, you must perform the following prerequisites.

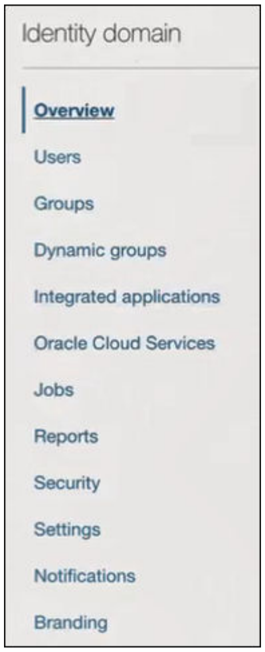
- Access the Identity Domain
- Prerequisites for Client Credentials and Resource Owner Password Credentials
- Prerequisites for JWT User Assertion
- Prerequisites for Authorization Code

Access the Identity Domain

- Log in to the Oracle Cloud Infrastructure Console with your identity domain administrator credentials.
  1. In the navigation pane, click **Identity & Security**.
  2. Click **Domains**.
  3. Select your compartment.
  4. Click the identity domain.


Domains *in* doc-team *Compartment*
Create domain
Name
Domain type
Default *(Current domain)*

  5. In the navigation pane, click **Integrated applications**.
     This is the location at which you create the client application for your grant type.



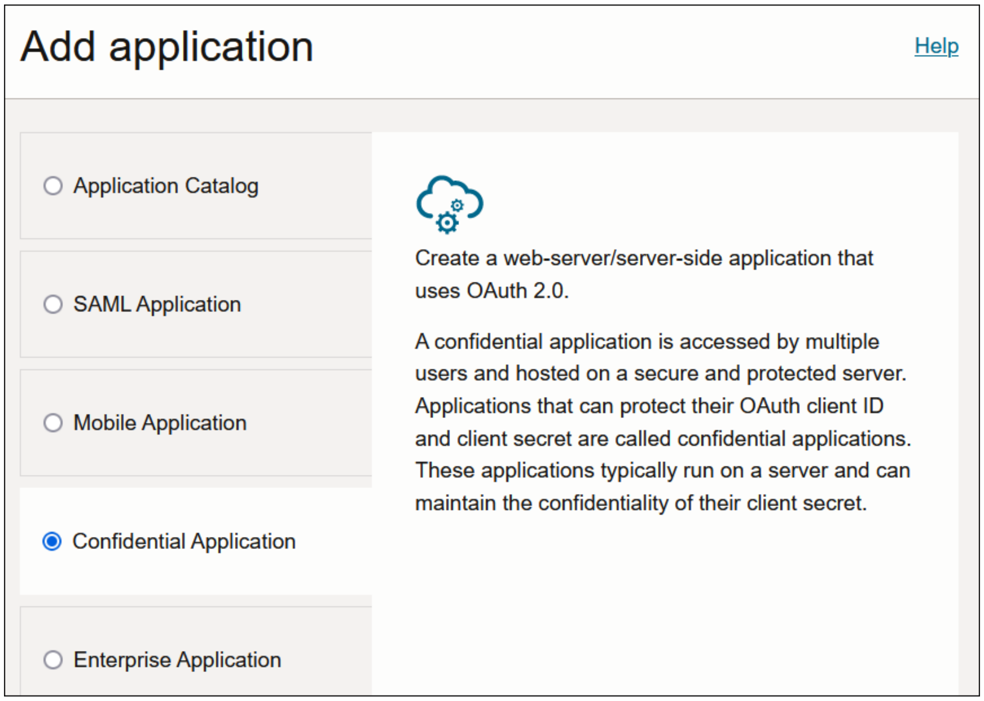
Prerequisites for Client Credentials and Resource Owner Password Credentials

To trigger the integration with OAuth, a client application is required. The prerequisites for the client credentials and resource owner password credentials grant types are very similar.

- Configure the client application
- Add roles to the client application

Configure the client application

1. Click **Add application**.
2. Select **Confidential Application**, then click **Launch workflow**.


Add application

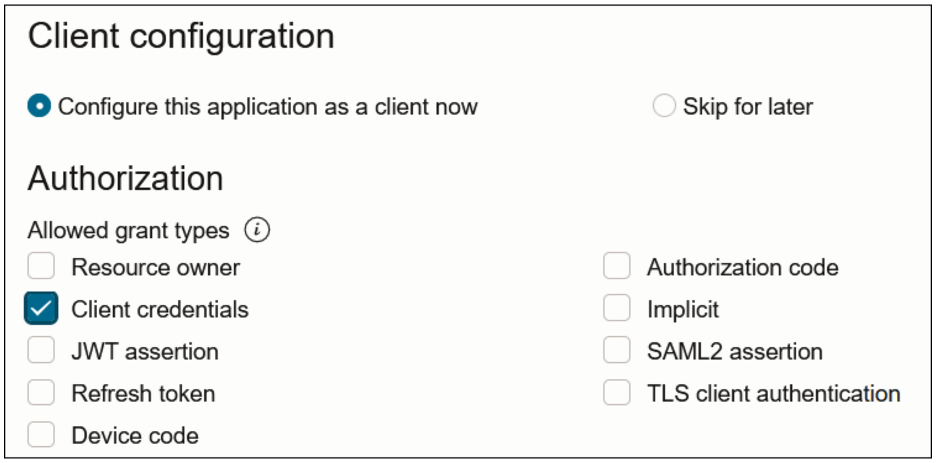
Help

- ○ Application Catalog
- ○ SAML Application
- ○ Mobile Application
- ● Confidential Application
- ○ Enterprise Application

Create a web-server/server-side application that uses OAuth 2.0.

A confidential application is accessed by multiple users and hosted on a secure and protected server. Applications that can protect their OAuth client ID and client secret are called confidential applications. These applications typically run on a server and can maintain the confidentiality of their client secret.

3. Enter a name. The remaining fields on this page are optional and can be ignored.
4. Click **Next**.
5. In the **Client configuration** box, select **Configure this application as a client now**.
6. Select the grant type to use:
   a. For client credentials, select **Client credentials** in the **Allowed grant types** section.


Client configuration

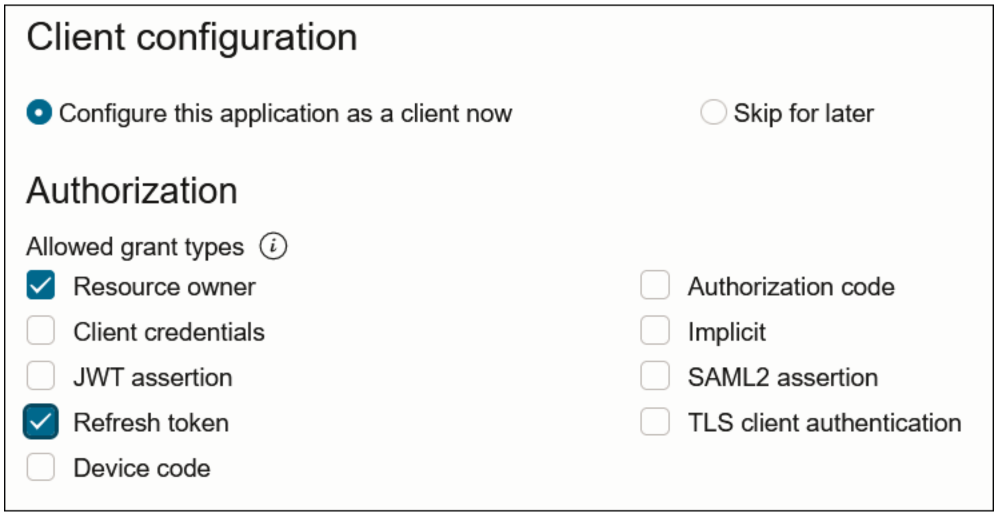
● Configure this application as a client now ○ Skip for later

Authorization

Allowed grant types

☐ Resource owner	☐ Authorization code
✓ Client credentials	☐ Implicit
☐ JWT assertion	☐ SAML2 assertion
☐ Refresh token	☐ TLS client authentication
☐ Device code	

   b. For resource owner password credentials, select **Resource owner** and **Refresh token** in the **Allowed grant types** section.


Client configuration

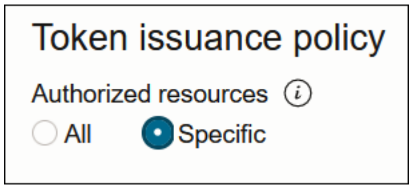
● Configure this application as a client now ○ Skip for later

Authorization

Allowed grant types

✓ Resource owner	☐ Authorization code
☐ Client credentials	☐ Implicit
☐ JWT assertion	☐ SAML2 assertion
✓ Refresh token	☐ TLS client authentication
☐ Device code	

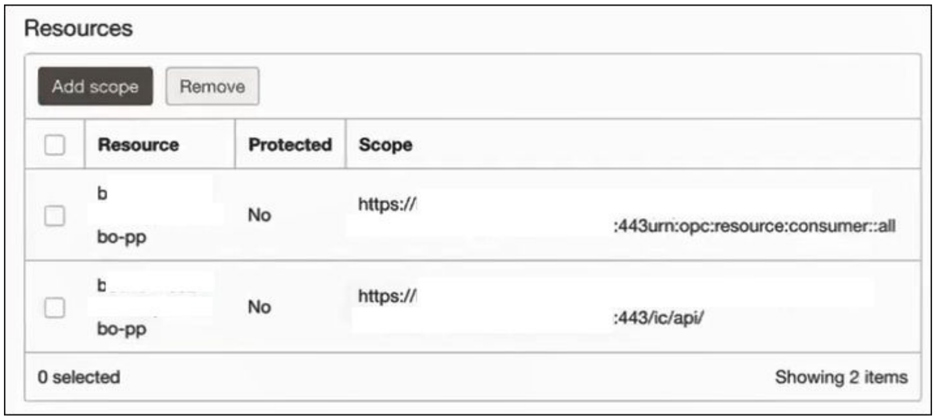
7. Complete the following steps for either grant type:
   a. Leave the **Redirect URL**, **Post-logout redirect URL**, and **Logout URL** fields blank.
   b. For **Client type**, ensure that **Confidential** is selected.
   c. Bypass several fields and scroll down to the **Token issuance policy** section.
   d. Select **Specific** in the **Authorized resources** section.


Token issuance policy

Authorized resources

○ All ● Specific

   e. Click the **Add Resources** check box.
   f. Click **Add scope**.
   g. Find the Oracle Integration application for your instance, and click ⌄.
   h. Select the two scopes appended with the following details:
      - **urn:opc:resource:consumer::all**
      - **ic/api/**
   i. Click **Add**.
      The scopes are displayed in the **Resources** section.



Resources

	Resource	Protected	Scope
☐	b bo-pp	No	https:// :443urn:opc:resource:consumer::all
☐	b bo-pp	No	https:// :443/ic/api/

0 selected Showing 2 items

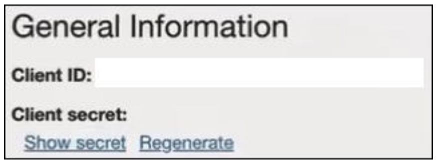
j. Ignore the **Add app roles** check box. This selection is not required.

k. Click **Next**, then click **Finish**.

The details page for the client application is displayed.

8. Click **Activate**, and then **Activate application** to activate the client application for use.

9. In the **General Information** section, note the client ID and client secret values. These values are required for the third-party application that is communicating with the identity domain.



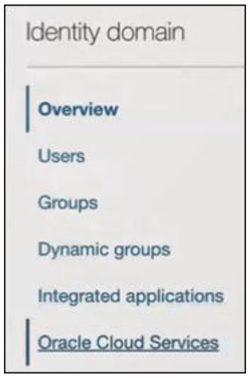
General Information

Client ID:

Client secret:

Add roles to the client application

1. In the navigation pane, click **Oracle Cloud Services**.

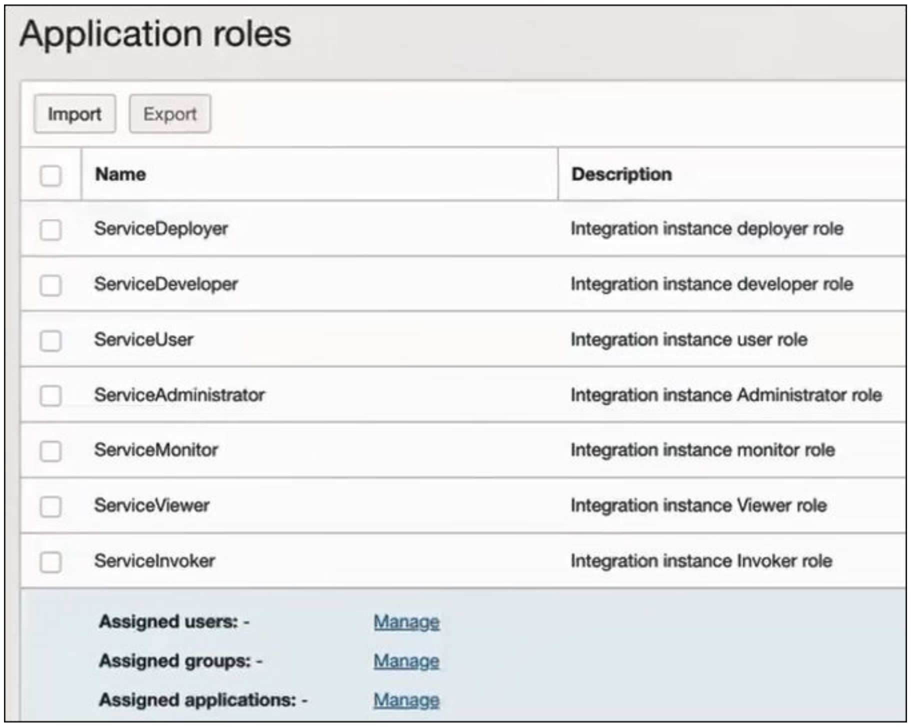


2. Select the specific application corresponding to the Oracle Integration instance.

3. In the navigation pane, click **Application roles**.

4. If configuring the client credentials grant type, select the following:

   a. Expand **ServiceInvoker**, then click **Manage** next to **Assigned applications**.



Name	Description
ServiceDeployer	Integration instance deployer role
ServiceDeveloper	Integration instance developer role
ServiceUser	Integration instance user role
ServiceAdministrator	Integration instance Administrator role
ServiceMonitor	Integration instance monitor role
ServiceViewer	Integration instance Viewer role
ServiceInvoker	Integration instance Invoker role

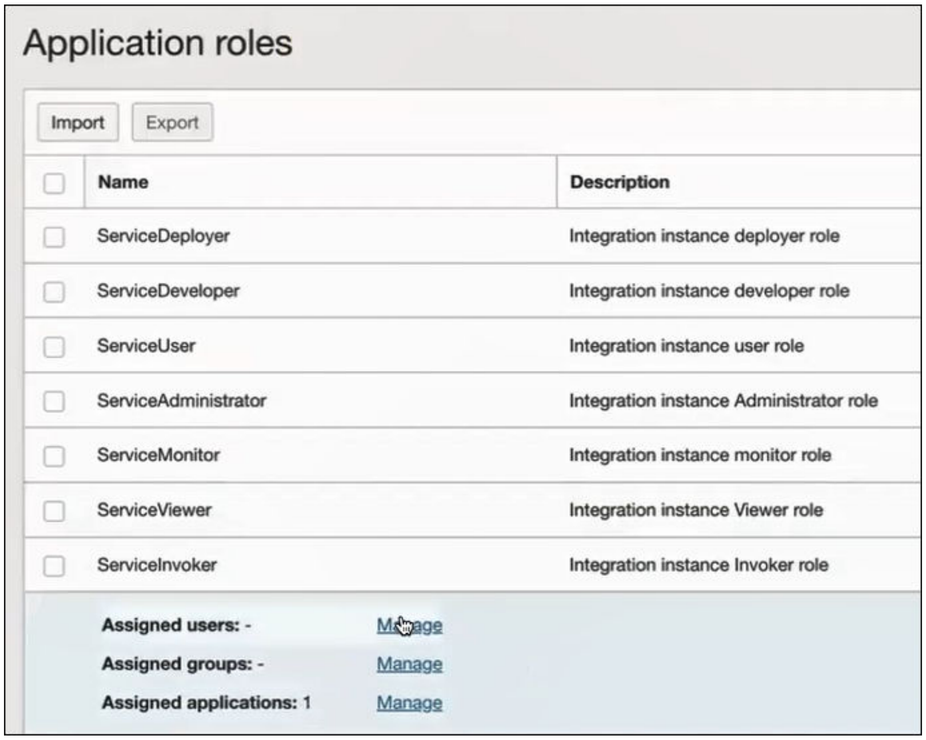
Assigned users: - Manage
Assigned groups: - Manage
Assigned applications: - Manage

b. Click **Show available applications**.

c. Select the application you just created and click **Assign**, then click **Close**.

5. If configuring the resource owner password credentials grant type, select the following:

   a. Expand **ServiceInvoker**, then click **Manage** next to either **Assigned users** or **Assigned groups**. For example, if you click **Assigned users**:



Application roles

Name	Description
ServiceDeployer	Integration instance deployer role
ServiceDeveloper	Integration instance developer role
ServiceUser	Integration instance user role
ServiceAdministrator	Integration instance Administrator role
ServiceMonitor	Integration instance monitor role
ServiceViewer	Integration instance Viewer role
ServiceInvoker	Integration instance Invoker role

Assigned users: -	Manage
Assigned groups: -	Manage
Assigned applications: 1	Manage

b. Click **Show available users**.

c. Select the user and click **Assign**, then click **Close**.

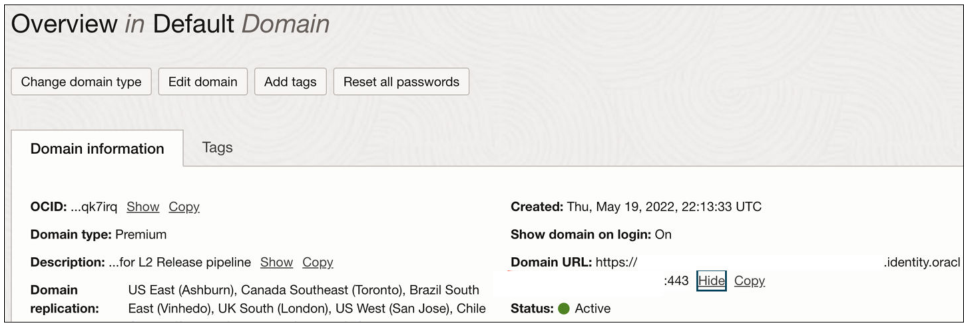
6. Validate the client application for the grant type you are using.

   a. For the client credentials grant type:

      i. Fetch the access client to make an access token request with the client credentials.

```
##Syntax
curl -i -H 'Authorization: Basic <base64Encoded
clientid:secret>' -H 'Content-Type: application/x-www-form-
urlencoded;charset=UTF-8' --request POST https://
<Identity_Domain_Service_Instance>.identity.oraclecloud.com/
oauth2/v1/token -d 'grant_type=client_credentials&scope=<app
scope>'
###where
#### <base64-clientid-secret> - Base 64 encode
clientId:ClientSecret
#### <app scope> - Scope added while creating application in
client configuration section (Ends with
urn:opc:resource:consumer::all)

##Example
curl -i -H 'Authorization: Basic OGQyM...ZDA0Mjcz' -H
'Content-Type: application/x-www-form-
urlencoded;charset=UTF-8' --request POST https://
<identity_domain_host>/oauth2/v1/token -d
'grant_type=client_credentials&scope=https://<Resource APP
Audience>urn:opc:resource:consumer::all'
```

Where `Identity_Domain_Service_Instance` is the value in the **Domain URL** field.



Overview *in* Default *Domain*

OCID: ...qk7irq Show Copy

Domain type: Premium

Description: ...for L2 Release pipeline Show Copy

Domain replication: US East (Ashburn), Canada Southeast (Toronto), Brazil South East (Vinhedo), UK South (London), US West (San Jose), Chile

Created: Thu, May 19, 2022, 22:13:33 UTC

Show domain on login: On

Domain URL: https:// .identity.oracl :443 Hide Copy

Status: ● Active

ii. Capture the `access_token` from the response.

```
{
    "access_token": "eyJ4NXQjG...dfsdfsFgets2ed",
    "token_type": "Bearer",
    "expires_in": 3600
}
```

iii. Use the `access_token` in the authorization header to invoke the trigger endpoint.

```
curl --location --request GET 'https://OIC host/OIC endpoint' \
--header 'Authorization: Bearer eyJ4NXQjG...dfsdfsFgets2ed'
```

b. For the resource owner password credentials grant type:

i. To fetch the access client, make a request with the user name and password in the payload.

```
##Syntax
curl -i -H 'Authorization: Basic <base64Encoded_clientid:secret>' -
H 'Content-Type: application/x-www-form-urlencoded;charset=UTF-8'
--request POST https://
<Identity_Domain_Service_Instance>.identity.oraclecloud.com/
oauth2/v1/token -d 'grant_type=password&username=<user-
name>&password=<password>&scope=<App_Scope>%20offline_access'

###where
#### <base64-clientid-secret> - Base 64 encode
clientId:ClientSecret
#### <username> - user for token needs to be issued (must be in
serviceinvoker role).
#### <password> - password for above user
#### <app_scope> - Scope added while creating application in
client configuration section (Ends with
urn:opc:resource:consumer::all)
##Example
curl -i -H 'Authorization: Basic OGQyM...ZDA0Mjcz' -H 'Content-
Type: application/x-www-form-urlencoded;charset=UTF-8' --request
POST https://<identity_domain_host>/oauth2/v1/token -d
'grant_type=password&username=sampleUser&password=SamplePassword&sc
```

```
ope=https://
<Resource_APP_Audience>urn:opc:resource:consumer::all%20offli
ne_access'
```

ii. Capture the `access_token` and `refresh_token` from the response.

```
{
    "access_token": "eyJ4NXQjG...dfsdfsFgets2ed",
    "token_type": "Bearer",
    "expires_in": 3600,
    "refresh_token": "AQAgY2MzNjVlOTVhOTRh...vM5S0MkrFSpzc="
}
```

iii. Use the `access_token` in the authorization header to invoke the Oracle Integration trigger endpoint.

```
curl --location --request GET 'https://OIC host/OIC
endpoint' \
--header 'Authorization: Bearer eyJ4NXQjG...dfsdfsFgets2ed'
```

iv. To update the access token, use the refresh token and make a request.

v. Capture the `access_token` and `refresh_token` from the response for further use.

```
curl -i -H 'Authorization: Basic <base64-clientid-secret>' -
H 'Content-Type: application/x-www-form-
urlencoded;charset=UTF-8' --request POST https://
<Identity_Domain_Service_Instance>.identity.oraclecloud.com/
oauth2/v1/token  -d
'grant_type=refresh_token&refresh_token=<refresh_token>'

##Example
curl -i -H 'Authorization: Basic OGQyM...ZDA0Mjcz' -H
'Content-Type: application/x-www-form-
urlencoded;charset=UTF-8' --request POST https://
<Identity_Domain_Service_Instance>.identity.oraclecloud.com/
oauth2/v1/token  -d
'grant_type=refresh_token&refresh_token=AQAgY2MzNjVlOTVhOTRh.
..vM5S0MkrFSpzc='
```

Prerequisites for JWT User Assertion

- Generate the key
- Configure the client application
- Add a certificate as a trusted partner
- Generate the JWT user assertion
- Validate the client application

Generate the key

You must first generate the key to import when you configure the client application for the JWT user assertion.

1. Generate the self-signed key pair.

```
keytool -genkey -keyalg RSA -alias <your_alias> -keystore <keystore_file>
-storepass <password> -validity 365 -keysize 2048

##example
keytool -genkey -keyalg RSA -alias assert -keystore sampleKeystore.jks -
storepass samplePasswd -validity 365 -keysize 2048
```

2. Export the public key for signing the JWT assertion.

```
keytool -exportcert -alias <your_alias> -file <filename> -keystore
<keystore_file> -storepass <password>

##example
keytool -exportcert -alias assert -file assert.cer -keystore
sampleKeystore.jks -storepass samplePasswd

## This should show a success message e.g. Certificate stored in file
<assert.cer>
```

3. Convert the keystore to P12 format.

```
keytool -importkeystore -srckeystore <filename> -srcstorepass <password> -
srckeypass <password> -srcalias <your_alias> -destalias <your_alias> -
destkeystore <destFileName> -deststoretype PKCS12 -deststorepass
<password> -destkeypass <password>

##example
keytool -importkeystore -srckeystore sampleKeystore.jks -srcstorepass
samplePasswd -srckeypass samplePasswd -srcalias assert -destalias assert -
destkeystore assert.p12 -deststoretype PKCS12 -deststorepass samplePasswd
-destkeypass samplePasswd

## This should show a success message e.g. Importing keystore
sampleKeystore.jks to assert.p12...
```

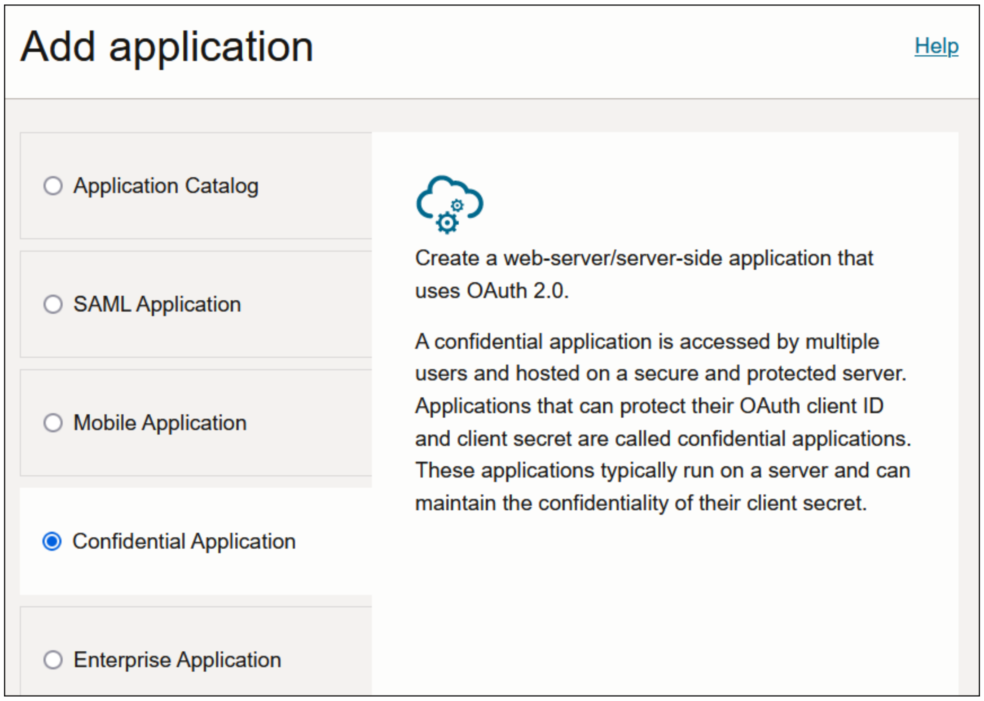
4. Export the private key from the P12 keystore.

```
openssl pkcs12 -in <destFileName> -nodes -nocerts -out <pem_file>

##example
openssl pkcs12 -in assert.p12 -nodes -nocerts -out private_key.pem

## This should show a success message: MAC verified OK
```

Configure the client application

To trigger the integration with OAuth, a client application is required.

1. Click **Add application**.
2. Select **Confidential Application**, and click **Launch workflow**.



Add application

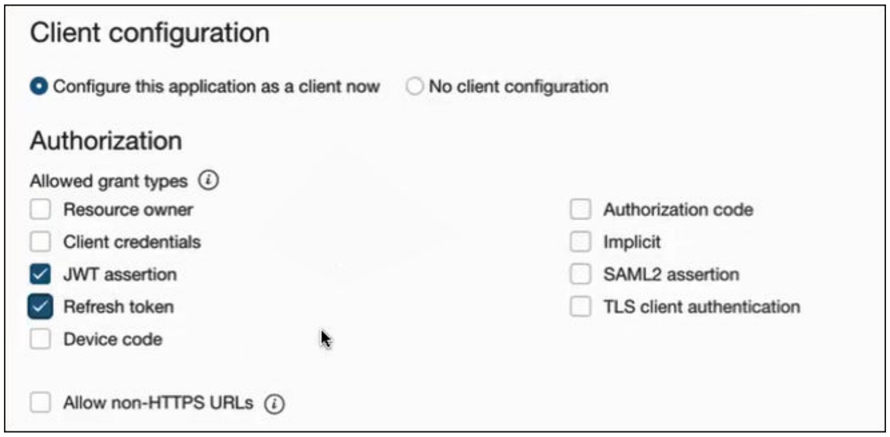
Help

- ○ Application Catalog
- ○ SAML Application
- ○ Mobile Application
- ● Confidential Application
- ○ Enterprise Application

Create a web-server/server-side application that uses OAuth 2.0.

A confidential application is accessed by multiple users and hosted on a secure and protected server. Applications that can protect their OAuth client ID and client secret are called confidential applications. These applications typically run on a server and can maintain the confidentiality of their client secret.

3. Enter a name. The remaining fields on this page are optional and can be ignored.
4. Click **Next**.
5. In the **Client configuration** box, select **Configure this application as a client now**.
6. For JWT user assertions, select **JWT assertion** and **Refresh token** in the **Allowed grant types** section.



Client configuration

● Configure this application as a client now ○ No client configuration

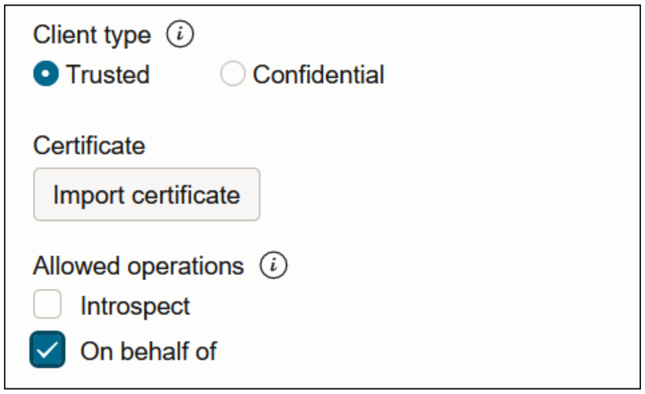
Authorization

Allowed grant types

☐ Resource owner	☐ Authorization code
☐ Client credentials	☐ Implicit
✓ JWT assertion	☐ SAML2 assertion
✓ Refresh token	☐ TLS client authentication
☐ Device code	

☐ Allow non-HTTPS URLs

7. Complete the following steps for the grant type:
   a. Leave the **Redirect URL**, **Post-logout redirect URL**, and **Logout URL** fields blank.
   b. In the **Client type** section, select **Trusted**.



Client type ⓘ
● Trusted ○ Confidential

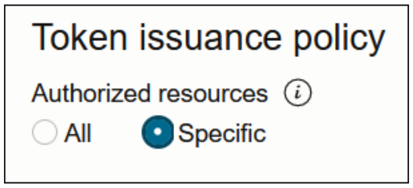
Certificate
Import certificate

Allowed operations ⓘ
☐ Introspect
☑ On behalf of

c. Upload the certificate created in section Generate the key. This action adds the certificate as a trusted partner.

d. Bypass several fields and scroll down to the **Token issuance policy** section.

e. Select **Specific** in the **Authorized resources** section.



Token issuance policy
Authorized resources ⓘ
○ All ● Specific

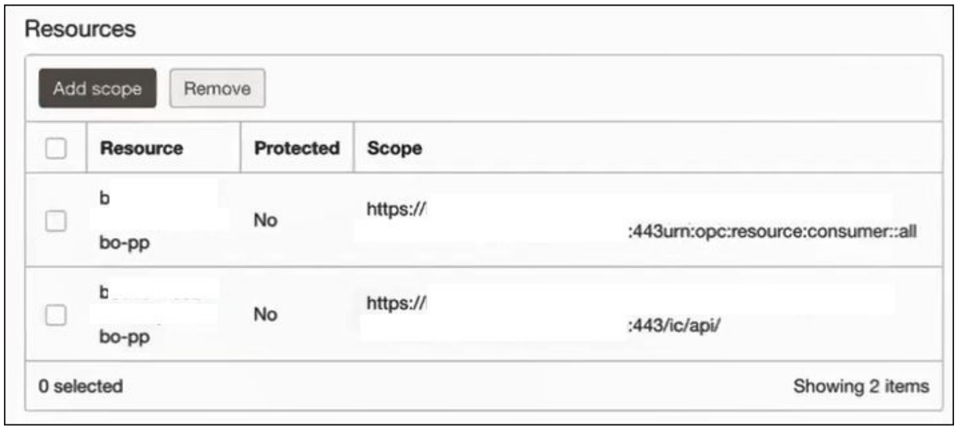
f. Click the **Add Resources** check box.

g. Click **Add scope**.

h. Find the Oracle Integration application for your instance, and click ⌄.

i. Select the two scopes appended with the following details:
   - **urn:opc:resource:consumer::all**
   - **ic/api/**

j. Click **Add**.
The scopes are displayed in the **Resources** section.



Resources

Add scope | Remove

	Resource	Protected	Scope
☐	b bo-pp	No	https:// :443urn:opc:resource:consumer::all
☐	b bo-pp	No	https:// :443/ic/api/

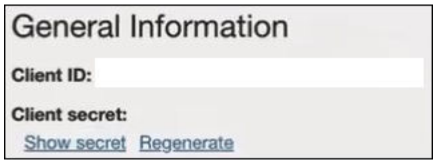
0 selected — Showing 2 items

k. Ignore the **Add app roles** check box. This selection is not required.

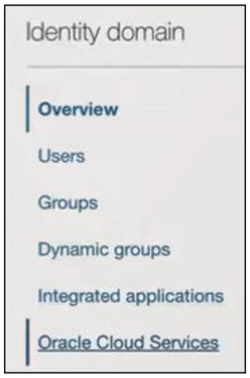
l. Click **Next**, then click **Finish**.
The details page for the client application is displayed.

m. Click **Activate**, and then **Activate application** to activate the client application for use.

n. In the **General Information** section, note the client ID and client secret values. These values are required for the third-party application that is communicating with the identity domain.



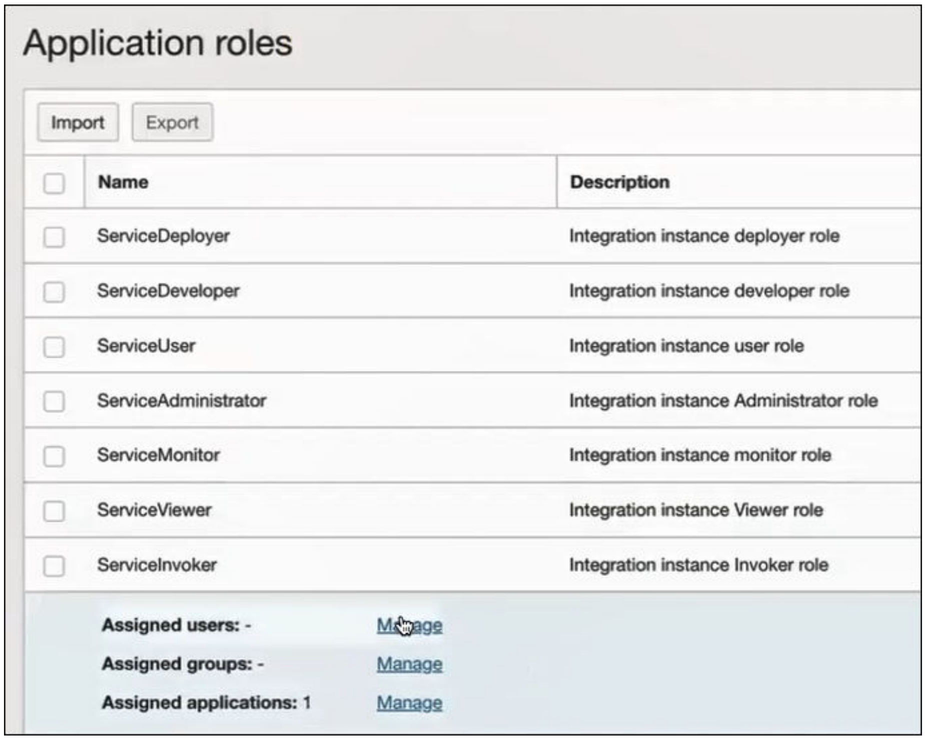
8. In the navigation pane, click **Oracle Cloud Services**.



9. Select the specific application corresponding to the Oracle Integration instance.

10. In the navigation pane, click **Application roles**.

11. Expand **ServiceInvoker**, then click **Manage** next to either **Assigned users** or **Assigned groups**. For example, if you click **Assigned users**:


Application roles

Name	Description
ServiceDeployer	Integration instance deployer role
ServiceDeveloper	Integration instance developer role
ServiceUser	Integration instance user role
ServiceAdministrator	Integration instance Administrator role
ServiceMonitor	Integration instance monitor role
ServiceViewer	Integration instance Viewer role
ServiceInvoker	Integration instance Invoker role

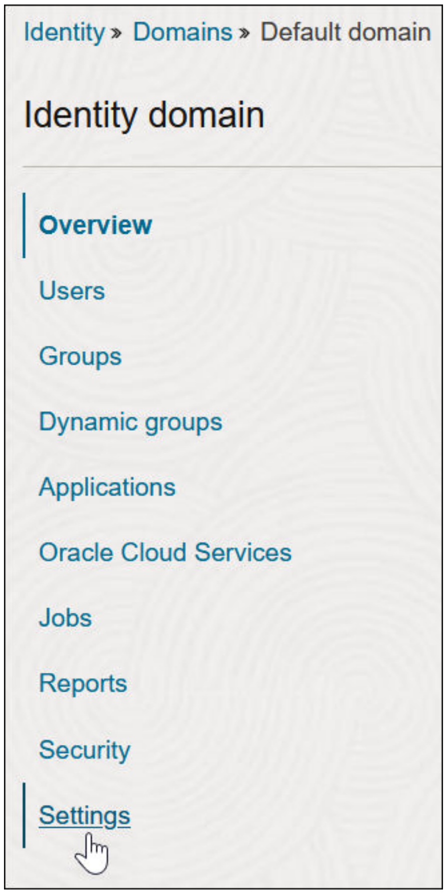
Assigned users: - Manage
Assigned groups: - Manage
Assigned applications: 1 Manage

12. Click **Show available users**.
13. Select the user and click **Assign**, then click **Close**.

Add a certificate as a trusted partner

In addition to importing the signing certificate into the client application, you are also required to include the certificate as a trusted partner certificate.

1. In the navigation pane, click **Settings**.



2. Click **Trusted partner certificates**.
3. Click **Import certificate** to upload the certificate created in section Generate the key.

Generate the JWT user assertion

1. Generate the JWT user assertion using the generated private key and simple Java code.

> **Note:**
>
> You can use the https://github.com/jwtk/jjwt library to generate the user assertion. There are many libraries listed at https://jwt.io/ for multiple technologies.

```
Sample:
header:
{
"alg": "RS256",
"typ": "JWT",
"kid": "assert"
}
```

```
payload:
{
"sub": "ssaInstanceAdmin",
"jti": "8c7df446-bfae-40be-be09-0ab55c655436",
"iat": 1589889699,
"exp": 1589909699,
"iss": "d702f5b31ee645ecbc49d05983aaee54",
"aud": "https://identity.oraclecloud.com/"
}
```

Where:

- `sub` specifies the user name for whom user assertion is generated.
- `jti` is a unique identifier
- `iat` is issued (epoch seconds).
- `exp` is the token expiry (epoch seconds).
- `iss` is the client ID.
- `aud` must include the identity domain audience `https://identity.oracle.com/`. The signing algorithm must be RS256.
- `kid` specifies the key to use to verify the signature. Therefore, it must match with the uploaded certificate alias.

Validate the client application

1. Once you generate the JWT user assertion, generate the access token as follows.

```
##Syntax
curl -i -H 'Authorization: Basic <base64Encoded clientid:secret>' -H
'Content-Type: application/x-www-form-urlencoded;charset=UTF-8' --request
POST https://<Identity_Domain_Service_Instance>.identity.oraclecloud.com/
oauth2/v1/token -d 'grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-
type%3Ajwt-bearer&assertion=<user assertion>&scope=<app_scope>'

###where
#### grant type - urn:ietf:params:oauth:grant-type:jwt-bearer
#### <base64-clientid-secret> - Base 64 encode clientId:ClientSecret
#### <user assertion> - User assertion generated
#### <app scope> - Scope added while creating application in client
configuration section (Ends with urn:opc:resource:consumer::all)
```

2. Capture the `access_token` from the response.

```
{
    "access_token": "eyJ4NXQjG...dfsdfsFgets2ed",
    "token_type": "Bearer",
    "expires_in": 3600
}
```

3. Use an `access_token` in the authorization header to invoke the Oracle Integration trigger endpoint.

```
curl --location --request GET 'https://OIC host/OIC endpoint' \
--header 'Authorization: Bearer eyJ4NXQjG...dfsdfsFgets2ed'
```

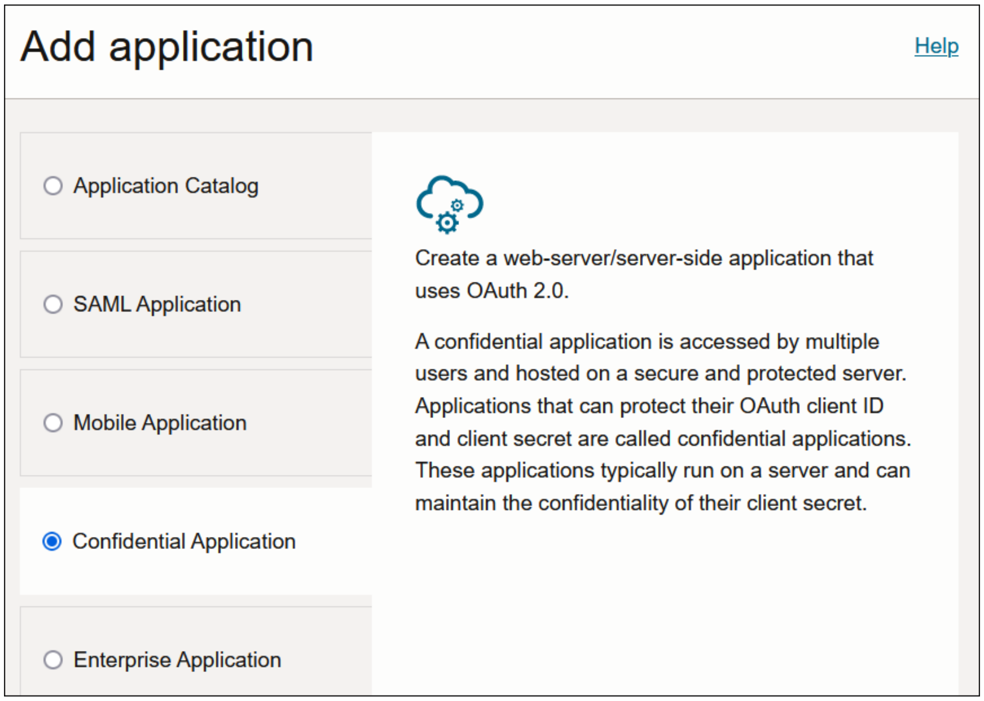
Prerequisites for Authorization Code

- Configure the client application
- Validate the Oracle Integration application and user roles
- Validate the client application

Configure the client application

To trigger the integration with OAuth, a client application is required.

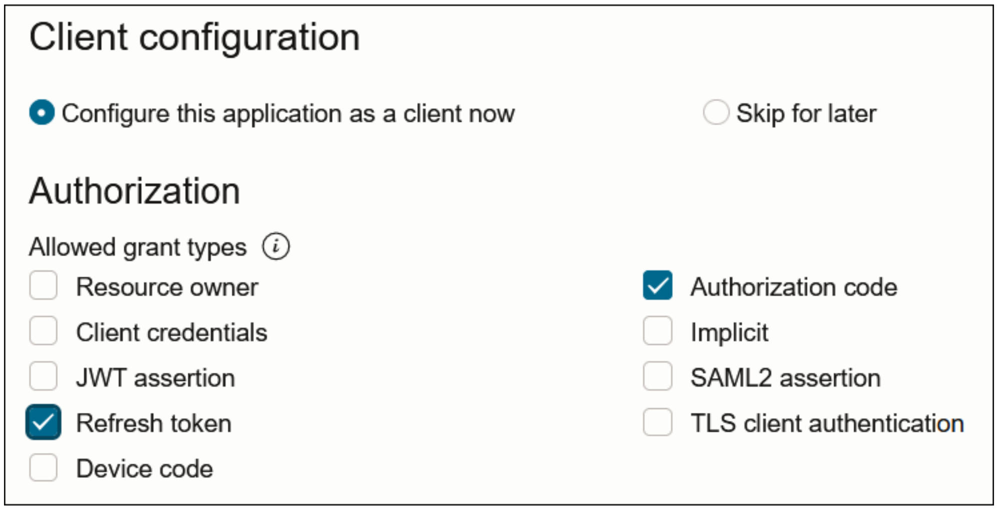
1. Click **Add application**.
2. Select **Confidential Application**. then click **Launch workflow**.



Create a web-server/server-side application that uses OAuth 2.0.

A confidential application is accessed by multiple users and hosted on a secure and protected server. Applications that can protect their OAuth client ID and client secret are called confidential applications. These applications typically run on a server and can maintain the confidentiality of their client secret.

3. Enter a name. The remaining fields on this page are optional and can be ignored.
4. Click **Next**.
5. In the **Client configuration** box, select **Configure this application as a client now**.
6. Select the grant type to use:
   a. For authorization code, select **Refresh token** and **Authorization code** in the **Allowed grant types** section.


Client configuration

[✓ Configure this application as a client now] Skip for later

Authorization

Allowed grant types

- ☐ Resource owner
- ☐ Client credentials
- ☐ JWT assertion
- ☑ Refresh token
- ☐ Device code
- ☑ Authorization code
- ☐ Implicit
- ☐ SAML2 assertion
- ☐ TLS client authentication

b. In the **Redirect URL** field, enter the redirect URL of the client application. After user login, this URL is redirected to with the authorization code. You can specify multiple redirect URLs. This is useful for development environments in which you have multiple instances, but only one client application due to licensing issues.

Note:

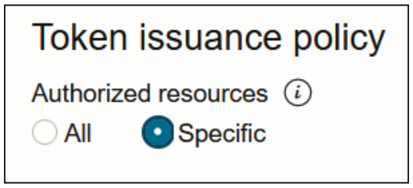
If you don't know the following information, check with your administrator:

- If your instance is new or upgraded from Oracle Integration Generation 2 to Oracle Integration 3.
- The complete instance URL with the region included (required for new instances).

For Connections...	Include the Region as Part of the Redirect URL?	Example of Redirect URL to Specify...
Created on new Oracle Integration 3 instances	Yes.	`https://`*`OIC_instance_URL.region`*`.ocp.oraclecloud.com/icsapis/agent/oauth/callback`
Created on instances upgraded from Oracle Integration Generation 2 to Oracle Integration 3	No. This applies to both: • New connections created after the upgrade • Existing connections that were part of the upgrade	`https://`*`OIC_instance_URL`*`.ocp.oraclecloud.com/icsapis/agent/oauth/callback`

c. In the **Client type** section, click **Confidential**.

d. Select **Specific** in the **Authorized resources** section.



e. Click the **Add Resources** check box.

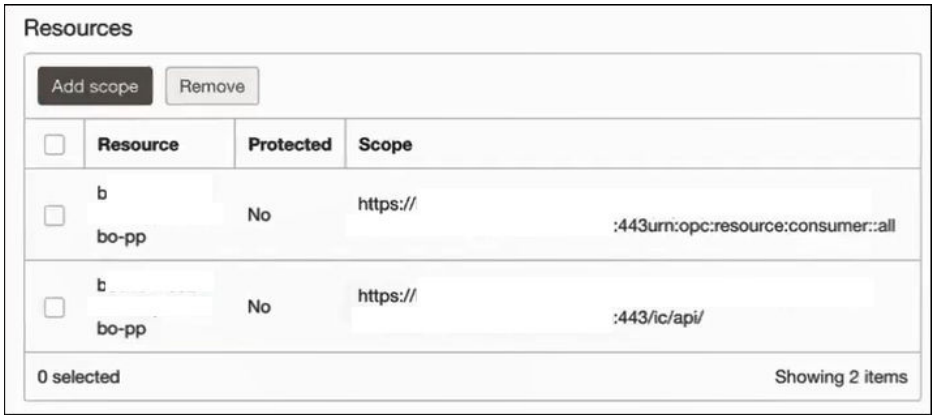
f. Click **Add scope**.

g. Find the Oracle Integration application for your instance, and click ⌄.

h. Select the two scopes appended with the following details:

- **urn:opc:resource:consumer::all**
- **ic/api/**

i. Click **Add**.
The scopes are displayed in the **Resources** section.



	Resource	Protected	Scope
	b bo-pp	No	https:// :443urn:opc:resource:consumer::all
	b bo-pp	No	https:// :443/ic/api/

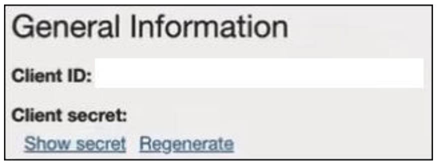
0 selected
Showing 2 items

j. Ignore the **Add app roles** check box. This selection is not required.

k. Click **Next**, then click **Finish**.
The details page for the client application is displayed.

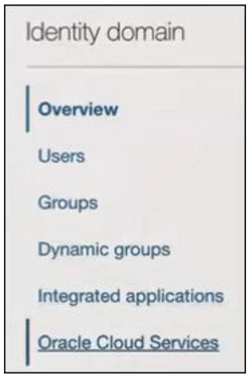
l. Click **Activate**, and then **Activate application** to activate the client application for use.

m. In the **General Information** section, note the client ID and client secret values. These values are required for the third-party application that is communicating with the identity domain.

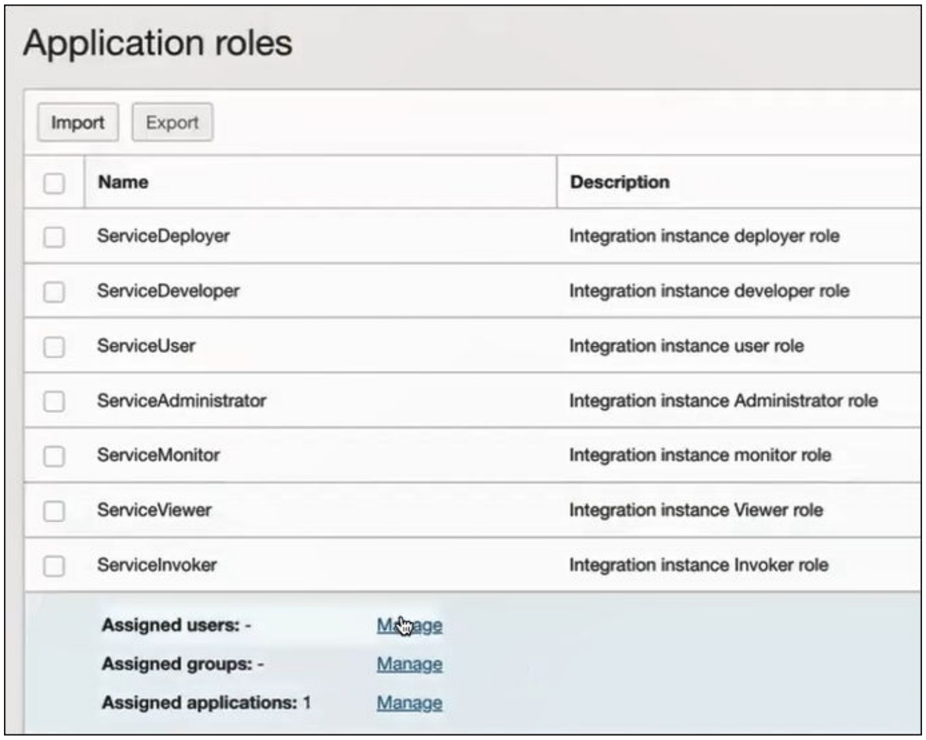


Validate the Oracle Integration application and user roles

1. In the navigation pane, click **Oracle Cloud Services**.



2. Select the specific application corresponding to the Oracle Integration instance.
3. In the navigation pane, click **Application roles**.
4. Expand **ServiceInvoker**, then click **Manage** next to either **Assigned users** or **Assigned groups**. For example, if you click **Assigned users**:



Name	Description
ServiceDeployer	Integration instance deployer role
ServiceDeveloper	Integration instance developer role
ServiceUser	Integration instance user role
ServiceAdministrator	Integration instance Administrator role
ServiceMonitor	Integration instance monitor role
ServiceViewer	Integration instance Viewer role
ServiceInvoker	Integration instance Invoker role

Assigned users: - Manage
Assigned groups: - Manage
Assigned applications: 1 Manage

5. Click **Show available users**.
6. Select the user and click **Assign**, then click **Close**.

Validate the client application

1. To fetch the authorization code, make the following request from the browser.

```
##Syntax
GET https://<Identity_Domain_Service_Instance>.identity.oraclecloud.com/
```

```
oauth2/v1/authorize?client_id=<client-
id>&response_type=code&redirect_uri=<client-redirect-
uri>&scope=<app_scope>%20offline_access&nonce=<nonce-
value>&state=<unique_value>

###where
#### <client-id> - ID of Client application generated.
#### <client-redirect-uri> - Redirect URI, in client application.
#### <app_scope> - scope added while creating application in client
configuration. (Ends with urn:opc:resource:consumer::all)
#### nonce - Optional, unique value to mitigate replay attacks
#### state - Recommended, Opaque to IDCS. Value used to maintain
state between the request and the callback
##Example
GET https://<identity_domain_host>/oauth2/v1/authorize?
client_id=<clientID>&response_type=code&redirect_uri=https://
app.getpostman.com/oauth2/callback&scope=https://
<Resource_APP_Audience>urn:opc:resource:consumer::all%20offline_acce
ss&nonce=121&state=12345544
```

2. If the user is not already logged in, you are challenged to authenticate your user credentials. (For authentication, the user assigned the **ServiceInvoker** role must be used.)
   After authentication is successful, the client URL is redirected to with the authorization code and state added to the URL.

```
##Response URL
https://<redirect_URL>?code=<code_value>=&state=<state_value>

###Client should validate state received is same as one sent in
request.
```

3. Capture the code value from the above response and make the following request to get the access token.

```
##Syntax
curl -i -H 'Authorization: Basic <base64-clientid-secret>' -H
'Content-Type: application/x-www-form-urlencoded;charset=UTF-8' --
request POST https://
<Identity_Domain_Service_Instance>.identity.oraclecloud.com/
oauth2/v1/token -d 'grant_type=authorization_code&code=<authz-
code>&redirect_uri=<client-redirect-uri>

###where
#### <base64-clientid-secret> - BAse 64 encode clientId:ClientSecret
#### <authz-code> - code value received as response on redirect.
#### <client-redirect-uri> - Redirect URI, in client application.

##Example
curl -i -H 'Authorization: Basic MDMx..NGY1' -H 'Content-Type:
application/x-www-form-urlencoded;charset=UTF-8' --request POST
https://<identity_domain_host>/oauth2/v1/token -d
'grant_type=authorization_code&code=AQAg...3jKM4Gc=&redirect_uri=htt
ps://app.getpostman.com/oauth2/callback
```

4. Capture the `access_token` and `refresh_token` from the response.

```
{
    "access_token": "eyJ4NXQjG...dfsdfsFgets2ed",
    "token_type": "Bearer",
    "expires_in": 3600,
    "refresh_token": "AQAgY2MzNjVlOTVhOTRh...vM5S0MkrFSpzc="
}
```

5. Use the `access_token` in the authorization header to invoke the Oracle Integration trigger endpoint.

```
curl --location --request GET 'https://OIC host/OIC endpoint' \
--header 'Authorization: Bearer eyJ4NXQjG...dfsdfsFgets2ed'
```

6. To update the access token, use the refresh token and make the request.
7. Capture the `access_token` and `refresh_token` from a response for further use.

```
curl -i -H 'Authorization: Basic <base64-clientid-secret>' -H 'Content-
Type: application/x-www-form-urlencoded;charset=UTF-8' --request POST
https://<Identity_Domain_Service_Instance>.identity.oraclecloud.com/
oauth2/v1/token  -d
'grant_type=refresh_token&refresh_token=<refresh_token>'


##Example
curl -i -H 'Authorization: Basic OGQyM...ZDA0Mjcz' -H 'Content-Type:
application/x-www-form-urlencoded;charset=UTF-8' --request POST https://
<Identity_Domain_Service_Instance>.identity.oraclecloud.com/oauth2/v1/
token  -d
'grant_type=refresh_token&refresh_token=AQAgY2MzNjVlOTVhOTRh...vM5S0MkrFSp
zc='
```

3 Create a SOAP Adapter Connection

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

Topics:

- Prerequisites for Creating a Connection
- Create a Connection
- Upload a Certificate to Connect with External Services

Prerequisites for Creating a Connection

You must satisfy the following prerequisites to create a connection with the SOAP Adapter.

Accessible WSDL

Ensure that the WSDL to use is reachable. There is no restriction on the type of WSDL to use.

OAuth Security Policies Use

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.

Create a Connection

Before you can build an integration, you must create the connections to the applications with which you want to share data.

To create a connection in Oracle Integration:

1. In the navigation pane, click **Design**, then **Connections**.
2. Click **Create**.

> **Note:**
>
> You can also create a connection in the integration canvas. See Define Inbound Triggers and Outbound Invokes.

3. In the Create connection panel, select the adapter to use for this connection. To find the adapter, scroll through the list, or enter a partial or full name in the **Search** field.
4. Enter the information that describes this connection.

Element	Description
Name	Enter a meaningful name to help others find your connection when they begin to create their own integrations.
Identifier	Automatically displays the name in capital letters that you entered in the **Name** field. If you modify the identifier name, don't include blank spaces (for example, `SALES OPPORTUNITY`).
Role	Select the role (direction) in which to use this connection (trigger, invoke, or both). Only the roles supported by the 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, you'll get an error when you try to drag the adapter into the section you didn't select. For example, assume you configure a connection for the Oracle Service Cloud (RightNow) Adapter as only an **invoke**. Dragging the adapter to a **trigger** section in the integration produces an error.
Keywords	Enter optional keywords (tags). You can search on the connection keywords on the Connections page.
Description	Enter an optional description of the connection.

Element	Description
Share with other projects	**Note**: This field only appears if you are creating a connection in a project. Select to make this connection publicly available in other projects. Connection sharing eliminates the need to create and maintain separate connections in different projects. When you configure an adapter connection in a different project, the **Use a shared connection** field is displayed at the top of the Connections page. If the connection you are configuring matches the same type and role as the publicly available connection, you can select that connection to reference (inherit) its resources. See Add and Share a Connection Across a Project.

5. Click **Create**.

   Your connection is created. You're now ready to configure the connection properties, security policies, and (for some connections) access type.

Configure Connection Properties

Enter connection information so your application can process requests.

1. Go to the **Properties** section.
2. Specify the following details.

Element	Description
WSDL URL	Specify the URL in either of two ways: a. Click **Upload** ⭱, then click **Browse** to select the WSDL to upload. If you upload a ZIP file, the file is validated and the page is refreshed to display the **Service WSDL** list. The relative paths of all WSDLs in the ZIP are displayed. Select the WSDL to use in the connection. b. Manually specify the WSDL to use.
Target Server's TLS Version (Optional) (Under **Optional properties**.)	If no value is selected, the default value used for outbound connections is Transport Layer Security (TLS) version 1.3. It's up to your discretion and the end application's requirements to select either TLS version 1.2 or 1.1 as the default. • **TLSv1.1** • **TLSv1.2** TLSv1 is no longer supported. If you previously configured a connection in a version prior to Oracle Integration 3 to use TLSv1.1, either update the connection by not selecting a value for this field or select TLSv1.2. The TLS protocol provides privacy and data integrity between two communicating computer applications. For trigger-only connections, you cannot select a TLS version. Oracle Integration accepts what it receives as long as it's TLSv1.1 or TLSv1.2.

Element	Description
Suppress insertion of timestamp into the request (Optional) (Under **Optional properties**.)	Optionally suppress the timestamp in the WS-Security header. Suppression applies to the Username Password Token security policy in the invoke (outbound) direction. In secure Web Services transactions, a WS-Utility (WSU) timestamp can be inserted into a WS-Security header to define the lifetime of the message in which it is placed. • **Yes**: No timestamp is added to the WS-Security header sent as part of the outbound request. For inbound requests with the basic authentication security policy, no timestamp is required to be sent by the client. • **No**: Clients are expected to send a timestamp in the WS-Security header with the request.
Ignore timestamp in the response message (Optional) (Under **Optional properties**.)	Specify if the timestamp is not required in the response message. • **Yes**: The timestamp is not required in the response message. If the timestamp is present in the SOAP security header when the response is received from the service , it is ignored. • **No**: The timestamp is received in the response from the service is not ignored.
Enable two way SSL for outbound connections (Optional) (Under **Optional properties**.)	If you are configuring the SOAP Adapter for use with a two-way SSL-enabled server, select **Yes**. .
Identity keystore alias name (Optional) (Under **Optional properties**.)	Enter the key alias name configured for two-way SSL communication. Both the client and server pass certificates to each other to establish an SSL link when two-way SSL is enabled. This value should match the alias that was provided to import identity to Oracle Integration. This is the name you entered in the **Key Alias Name** field when uploading the identity certificate in the Upload Certificate dialog. See Upload a Certificate to Connect with External Services. The alias name to provide must match the name provided for the private key entry in the JKS file.

Configure Connection Security

Configure security for your SOAP Adapter connection by selecting the security policy.

1. Go to the **Security** section.
2. Select the security policy.

   The page is refreshed to display the login credential fields.
3. Specify the login credentials. For trigger (inbound) connections, the security policy must be either username password token, basic authentication, SAML, or OAuth 2.0. This is because all Oracle Integration inbound endpoints are protected with any of these policies.

   When configuring the SOAP Adapter with the trigger-only role, you cannot select **No Security Policy** because all Oracle Integration endpoints are protected.

   - New connections do not show **No Security Policy** in the dropdown list.
   - Existing connections default to the **Username Password Token** security policy.

- Connections updated with the REST APIs are automatically changed to use the **Username Password Token** security policy by default even though the request payload used **No Security Policy**.

Security Policy	Fields
Basic Authentication (In the trigger (inbound) direction, supports HTTP basic authentication over SSL: oracle/ wss_http_token_over_ssl_service_policy). Note the following behavior: • If the invoking client is secured with Oracle Web Services Manager (OWSM) using an oracle/wss* policy, the client receives a failure. • In the inbound (trigger) direction, if the **Suppress insertion of timestamp into the request (Optional)** field is enabled, then oracle/ http_basic_auth_over_ssl_service_policy is supported. • In customer-managed environments, when configuring a trigger SOAP Adapter with Basic Authentication, the wss_http_token_service_policy policy is used regardless of whether the **Suppress insertion of timestamp into the request** option is set to **Yes** or **No** in the Connections page. Therefore, with or without the timestamp added in the SOAP header, as long as the username and password credentials are valid, the connection runs successfully at runtime. If Basic Authentication is required for both trigger and invoke connections, create one connection with the **Trigger and Invoke** role that uses the Basic Authentication security policy.	• **Username** — Enter the name of a user who has access to the destination web service. • **Password** — Enter the password. • **Confirm Password** — Reenter the password.
Username Password Token (In the trigger (inbound) direction, supports oracle/ wss_username_token_over_ssl_service_policy.)	• **Username** — Enter the name of a user • **Password** — Enter the password. • **Confirm Password** — Reenter the password.

Security Policy	Fields
OAuth Client Credentials (In the invoke (outbound) direction.)	• **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. • **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. • **Client authentication** — You can optionally configure OAuth flows with client authentication. This is similar to the Postman user interface feature for configuring client authentication. – **Send client credentials as basic auth header**: Pass the client ID and client secret in the header as basic authentication. – **Send client credentials in body**: Pass the client ID and client secret in the body as form fields. When you click **Test**, the access token is requested using the provided access token URL with client credentials. The access token is persisted in the credential store.
OAuth Authorization Code Credentials (In the invoke (outbound) direction.)	• **Client Id** — The client identifier issued to the client during the registration process. • **Client secret** — 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. • **Client Authentication** — You can optionally configure OAuth flows with client authentication. This is similar to the Postman user interface feature for configuring client authentication. – **Send client credentials as basic auth header**: Pass the client ID and client secret in the header as basic authentication. – **Send client credentials in body**: Pass the client ID and client secret in the body as form fields. When you click **Provide Consent**, the authorization code request is sent to the authorization URL along with the redirect URL. You are then redirected to the sign in page. After signing in with the authorized user name and password, the authorization code is sent back to the caller. The authorization code, client ID, and client secret are sent to the access token URL. To exchange the access token, expiry and refresh token are sent back to caller. When you test the connection, the refresh token flow is called to validate the refresh token.

Security Policy	Fields
No Security Policy	No fields are displayed.
Security Assertion Markup Language (SAML)	This policy is *only* available when configuring the SOAP Adapter as a trigger. If you attempt to add the SOAP Adapter with this security policy configuration as an invoke in an integration, you receive an error. • **Username** — Optionally enter the name of the SAML user.
OAuth 2.0	This policy is *only* available when configuring the SOAP Adapter as a trigger. If you attempt to add the SOAP Adapter with this security policy configuration as an invoke in an integration, you receive an error. No fields are displayed.

If you select a security policy, the following behavior occurs.

If the Inbound SOAP Connection is Configured with Security Policy...	Then...
Username Password Token	• The client should send the username/ password and timestamp as part of the WSEE header. • The response includes only the SOAP payload.
Basic Authentication	• The client should send the username/ password in the HTTP headers and timestamp as part of the WSEE header. • The response includes only the SOAP payload.
Basic Authentication and the **Suppress insertion of timestamp into the request (Optional)** field is enabled	• The client should send the username/ password in the HTTP headers. • The response includes only the SOAP payload.

Note:

If no timestamp is included as part of the header, configure the SOAP Adapter connection with the Basic Authentication security policy (oracle/ http_basic_auth_over_ssl_client_policy) and set **Suppress insertion of timestamp into the request (Optional)** to **Yes**.

Configure the Endpoint Access Type

Configure access to your endpoint. Depending on the capabilities of the adapter you are configuring, options may appear to configure access to the public internet, to a private endpoint, or to an on-premises service hosted behind a fire wall.

- Select the Endpoint Access Type
- Ensure Private Endpoint Configuration is Successful

Select the Endpoint Access Type

Select the option for accessing your endpoint.

Option	This Option Appears If Your Adapter Supports ...
Public gateway	Connections to endpoints using the public internet.
Private endpoint	Connections to endpoints using a private virtual cloud network (VCN). **Note**: To connect to private endpoints, you must complete prerequisite tasks in the Oracle Cloud Infrastructure Console. Failure to do so results in errors when testing the connection. See Connect to Private Resources in *Provisioning and Administering Oracle Integration 3* and Troubleshoot Private Endpoints in *Using Integrations in Oracle Integration 3*.
Connectivity agent	Connections to on-premises endpoints through the connectivity agent. 1. Click **Associate agent group**. The Associate agent group panel appears. 2. Select the agent group, and click **Use**. To configure an agent group, you must download and install the on-premises connectivity agent. See Download and Run the Connectivity Agent Installer and About Creating Hybrid Integrations Using Oracle Integration in *Using Integrations in Oracle Integration 3*.

Ensure Private Endpoint Configuration is Successful

- To connect to private endpoints, you must complete prerequisite tasks in the Oracle Cloud Infrastructure Console. Failure to do so results in errors when testing the connection. See Connect to Private Resources in *Provisioning and Administering Oracle Integration 3*.
- When configuring an adapter on the Connections page to connect to endpoints using a private network, specify the fully-qualified domain name (FQDN) and *not* the IP address. If you enter an IP address, validation fails when you click **Test**.
- IPSec tunneling and FastConnect are not supported for use with private endpoints.

Test the Connection

Test your connection to ensure that it's configured successfully.

1. In the page title bar, click **Test**. What happens next depends on whether your adapter connection uses a Web Services Description Language (WSDL) file. Only some adapter connections use WSDLs.

If Your Connection...	Then...
Doesn't use a WSDL	The test starts automatically and validates the inputs you provided for the connection.
Uses a WSDL	A dialog prompts you to select the type of connection testing to perform: • **Validate and Test**: Performs a full validation of the WSDL, including processing of the imported schemas and WSDLs. Complete validation can take several minutes depending on the number of imported schemas and WSDLs. No requests are sent to the operations exposed in the WSDL. • **Test**: Connects to the WSDL URL and performs a syntax check on the WSDL. No requests are sent to the operations exposed in the WSDL.

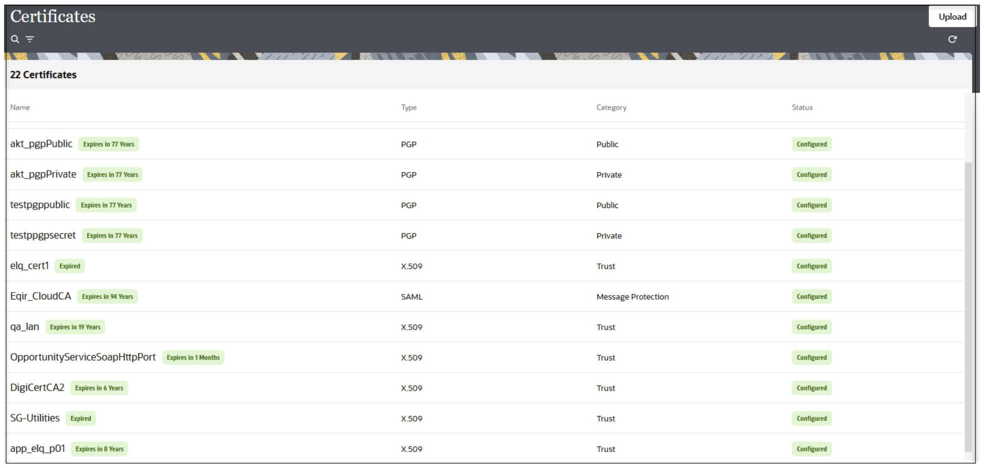
2. Wait for a message about the results of the connection test.
   - If the test was successful, then the connection is configured properly.
   - If the test failed, then edit the configuration details you entered. Check for typos and verify URLs and credentials. Continue to test until the connection is successful.
3. When complete, click **Save**.

Upload a Certificate to Connect with External Services

Certificates allow Oracle Integration to connect with external services. If the external service/ endpoint needs a specific certificate, request the certificate and then import it into Oracle Integration.

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

1. Sign in to Oracle Integration.
2. In the navigation pane, click **Settings**, then **Certificates**.
   All certificates currently uploaded to the trust store are displayed on the Certificates page.
3. Click **Filter** to filter by name, certificate expiration date, status, type, category, and installation method (user-installed or system-installed). Certificates installed by the system cannot be deleted.



22 Certificates

Name	Type	Category	Status
akt_pgpPublic Expires in 77 Years	PGP	Public	Configured
akt_pgpPrivate Expires in 77 Years	PGP	Private	Configured
testpgppublic Expires in 77 Years	PGP	Public	Configured
testppgpsecret Expires in 77 Years	PGP	Private	Configured
elq_cert1 Expired	X.509	Trust	Configured
Eqir_CloudCA Expires in 94 Years	SAML	Message Protection	Configured
qa_lan Expires in 19 Years	X.509	Trust	Configured
OpportunityServiceSoapHttpPort Expires in 1 Months	X.509	Trust	Configured
DigiCertCA2 Expires in 6 Years	X.509	Trust	Configured
SG-Utilities Expired	X.509	Trust	Configured
app_elq_p01 Expires in 8 Years	X.509	Trust	Configured

4. Click **Upload** at the top of the page.
   The Upload certificate panel is displayed.
5. Enter an alias name and optional description.
6. In the **Type** field, select the certificate type. Each certificate type enables Oracle Integration to connect with external services.
   - Digital Signature
   - X.509 (SSL transport)
   - SAML (Authentication & Authorization)
   - PGP (Encryption & Decryption)
   - Signing key

Digital Signature

The digital signature security type is typically used with adapters created with the Rapid Adapter Builder. See Learn About the Rapid Adapter Builder in Oracle Integration in *Using the Rapid Adapter Builder with Oracle Integration 3*.

1. Click **Browse** to select the digital certificate. The certificate must be an X509Certificate. This certificate provides inbound RSA signature validation. See Implement Digital Signature Validation (RSA) in *Using the Rapid Adapter Builder with Oracle Integration 3*.
2. Click **Upload**.

X.509 (SSL transport)

1. Select a certificate category.
   a. **Trust**: Use this option to upload a trust certificate.
      i. Click **Browse**, then select the trust file (for example, `.cer` or `.crt`) to upload.
   b. **Identity**: Use this option to upload a certificate for two-way SSL communication.
      i. Click **Browse**, then select the keystore file (`.jks`) to upload.

ii. Enter the comma-separated list of passwords corresponding to key aliases.

> **Note:**
>
> When an identity certificate file (`.jks`) contains more than one private key, all the private keys must have the same password. If the private keys are protected with different passwords, the private keys cannot be extracted from the keystore.

iii. Enter the password of the keystore being imported.

c. Click **Upload**.

SAML (Authentication & Authorization)

1. Note that **Message Protection** is automatically selected as the only available certificate category and cannot be deselected. Use this option to upload a keystore certificate with SAML token support. Create, read, update, and delete (CRUD) operations are supported with this type of certificate.
2. Click **Browse**, then select the certificate file (`.cer` or `.crt`) to upload.
3. Click **Upload**.

PGP (Encryption & Decryption)

1. Select a certificate category. Pretty Good Privacy (PGP) provides cryptographic privacy and authentication for communication. PGP is used for signing, encrypting, and decrypting files. You can select the private key to use for encryption or decryption when configuring the stage file action.
   a. **Private**: Uses a private key of the target location to decrypt the file.
      i. Click **Browse**, then select the PGP file to upload.
      ii. Enter the PGP private key password.
   b. **Public**: Uses a public key of the target location to encrypt the file.
      i. Click **Browse**, then select the PGP file to upload.
      ii. In the **ASCII-Armor Encryption Format** field, select **Yes** or **No**.
         - **Yes** shows the format of the encrypted message in ASCII armor. ASCII armor is a binary-to-textual encoding converter. ASCII armor formats encrypted messaging in ASCII. This enables messages to be sent in a standard messaging format. This selection impacts the visibility of message content.
         - **No** causes the message to be sent in binary format.
      iii. From the **Cipher Algorithm** list, select the algorithm to use. Symmetric-key algorithms for cryptography use the same cryptographic keys for both encryption of plain text and decryption of cipher text. The following supported cipher algorithms are FIPS-compliant:
         - AES128
         - AES192
         - AES256

        - TDES

    c. Click **Upload**.

Signing key

A signing key is a secret key used to establish trust between applications. Signing keys are used to sign ID tokens, access tokens, SAML assertions, and more. Using a private signing key, the token is digitally signed and the server verifies the authenticity of the token by using a public signing key. You must upload a signing key to use the OAuth Client Credentials using JWT Client Assertion and OAuth using JWT User Assertion security policies in REST Adapter invoke connections. Only PKCS1- and PKCS8-formatted files are supported.

1. Select **Public** or **Private**.
2. Click **Browse** to upload a key file.
   If you selected **Private**, and the private key is encrypted, a field for entering the private signing key password is displayed after key upload is complete.
3. Enter the private signing key password. If the private signing key is not encrypted, you are not required to enter a password.
4. Click **Upload**.

4

Add the SOAP Adapter Connection to an Integration

When you drag the SOAP Adapter into the trigger or invoke area of an integration, the Adapter Endpoint Configuration Wizard appears. This wizard guides you through the configuration of the SOAP Adapter endpoint properties.

These topics describe the wizard pages that guide you through configuration of the SOAP Adapter as a trigger or invoke in an integration.

Topics:

Basic Info Page

You can enter a name and description on the Basic Info page of each adapter in your integration.

Element	Description
What do you want to call your endpoint?	Provide a meaningful name so that others can understand the responsibilities of this connection. You can include English alphabetic characters, numbers, underscores, and hyphens in the name. You can't include the following characters: • No blank spaces (for example, `My Inbound Connection`) • No special characters (for example, `#;83&` or `righ(t)now4`) except underscores and hyphens • No multibyte characters
What does this endpoint do?	Enter an optional description of the connection’s responsibilities. For example: `This connection receives an inbound request to synchronize account information with the cloud application.`

Element	Description
Do you want to configure this as a callback invoke?	Select **Yes** to configure the SOAP Adapter as a callback invoke. This option is only available when configuring the SOAP Adapter as an invoke in an integration. See Asynchronous Trigger Support in Orchestrated Integrations.

Trigger Operation Page

Enter the port type and operation for the SOAP Adapter. If your WSDL includes only a single service, port type, and operation, they are automatically selected. If the WSDL includes multiple services and port types, then select the ones to use in your integration. Based on the selected values, other objects such as the request object, response object, and fault object may also be automatically displayed.

Element	Description
Selected Port Type	Displays the selected port type. If your WSDL includes multiple port types, select the port type.
Select the Operation	Displays the selected operation. If your WSDL includes multiple operations, select the operation.
Request Object	Displays the request object (if your WSDL includes request objects).
Response Object	Displays the response object (if your WSDL includes response objects).
Disable SoapAction validation	Select **Yes** to disable SOAP action validation for inbound requests. This is useful for environments in which your WSDL includes custom code and you want to bypass validation. When set to **No** (the default), Oracle Integration validates the SOAP action to ensure that it matches the WSDL.

Trigger Callback Operation Page

Enter the trigger callback response operation details for the SOAP Adapter. This page is only displayed when the SOAP Adapter is used in integrations or a one-way operation is selected on the Operations page.

Element	Description
No Response	Select if a one-way call without a response is expected.
Delayed Response	Select if a delayed callback response is expected.

Invoke Operation Page

Enter the service, port, and operation for the SOAP Adapter invoke connection or enter the port type and operation for the SOAP Adapter callback invoke connection. If the WSDL file you specified during SOAP Adapter connectivity configuration includes only a single service, port type, or operation, they are automatically selected for use. If the WSDL included multiple services, port types, or operations, then select the ones to use in this integration.

Element	Description
Selected Port Type	Displays the selected port type. If your WSDL includes multiple port types, select the port type.
Selected Operation	Displays the selected operation. If your WSDL includes multiple operations, select the operation.
Request Object	Displays the request object (if your WSDL includes request objects).
Response Object	Displays the response object (if your WSDL includes response objects).

Header Page

Enter the header details for the SOAP Adapter. The following table describes the key information on the Oracle SOAP Adapter Header page. The headers you specify are applied to the request and/or response object of the selected operation. The selected elements are included under respective wrapper elements in the integration WSDL and are displayed in the mapper as a request and/or response.

Element	Description
Configure MTOM Attachment Options	MTOM attachment options are shown when a base64Binary element is present in the WSDL messages for a given operation in both the request and response messages (for synchronous integrations). • **Send attachments in request**: Select to configure MTOM for the outbound request. • **Accept attachments in response**: Select to configure MTOM for the outbound response.
Do you want to configure headers for this Endpoint?	Select **Yes**, then select the headers to include. **Yes** is automatically selected for you in the following situations: • Your endpoint already contains SOAP headers. • An asynchronous trigger (trigger with a one-way operation selected on the Operations Page and **Delayed Response** selected on the Callback Operations page) or callback invoke (invoke with **Do you want to configure this as a callback invoke?** set to **Yes** on the Basic Info page) was configured. The selection is disabled in this case and cannot be modified.

Element	Description
SOAP Headers	This option is automatically selected in the following situations and cannot be modified: • Your endpoint already contains SOAP headers. • An asynchronous trigger (trigger with a one-way operation selected on the Operations Page and **Delayed Response** selected on the Callback Operations page) or callback invoke (invoke with **Do you want to configure this as a callback invoke?** set to **Yes** on the Basic Info page) was configured.
Standard HTTP Headers	Select this check box to add standard HTTP headers in the request and/or response.
Custom HTTP Headers	Select this check box to add custom HTTP headers in the request and/or response.
Custom SOAP Headers	Select this check box to add custom SOAP headers in the request and/or response.

Note:

Based on the selections made on this page, separate tabbed pages are shown in the Request Header page and/or Response Header page for configuring the selected headers.

Request Header Page

Enter the request header details for the SOAP Adapter. You can configure and view standard HTTP, custom HTTP, and custom SOAP request header parameters for the SOAP Adapter based on the selections made on the Header page.

Tabbed Page	Description
SOAP Headers	View the SOAP headers contained in the WSDL defined on the Connections page. These headers cannot be modified: If an asynchronous trigger (a trigger with a one-way operation selected on the Operations page and **Delayed Response** selected on the Callback Operations page) or callback invoke (**Do you want to configure this as a callback invoke?** set to **Yes** on the Basic Info Page) were configured, `ReplyTo`, `MessageID`, and `Action` WS-Addressing are displayed.
Standard HTTP Headers	Configure the standard HTTP headers. Click the **Add** icon to add headers from the prepopulated list. Some of the mandatory standard HTTP headers are disabled for selection because allowing them to change may provide unexpected results (for example, authorization).

Tabbed Page	Description
Custom HTTP Headers	Configure the custom HTTP headers. Click the **Add** icon to add custom HTTP header names and descriptions.
Custom SOAP Headers	Configure the custom SOAP headers. Click **Browse** to add a schema file from which to select custom SOAP headers to be configured in the integration WSDL. Select a header element from the uploaded schema and click **Add Header**. See Configure Custom SOAP Headers for the SOAP Adapter.

Response Header Page

Enter the response header details for the SOAP Adapter. You can configure and view standard HTTP, custom HTTP, and custom SOAP response header parameters for the SOAP Adapter based on the selections made in the Header page.

Tabbed Page	Description
SOAP Headers	View the SOAP headers contained in the WSDL defined on the Connections page. These headers cannot be modified.
Standard HTTP Headers	Configure the standard HTTP headers. Click the **Add** icon to add headers from the prepopulated list. Some of the mandatory standard HTTP headers are disabled for selection because allowing them to change may provide unexpected results (for example, authorization).
Custom HTTP Headers	Configure the custom HTTP headers. Click the **Add** icon to add custom HTTP header names and descriptions.
Custom SOAP Headers	Configure the custom SOAP headers. Click **Browse** to add a schema file from which to select custom SOAP headers to be configured in the integration WSDL. Select a header element from the uploaded schema and click **Add Header**. See Configure Custom SOAP Headers for the SOAP Adapter.

Invoke Callback Operation Page

Enter the callback response operation details for the SOAP Adapter. This page is also visible for the normal invokes **(Do you want to configure this as a callback invoke** is set to **No** on the Basic Info Page) case in which the one-way operation was selected on the Operations Page.

Element	Description
No Response	Select if a no callback response is expected.
Delayed Response	Select if a delayed callback response is expected.

Element	Description
Select the Port Type	Select the port type to use for the asynchronous callback response.
Selected Callback Operation	View the callback operation associated with the selected port type. In the case of multiple operations, Select the operation.
Flow Identifier	Specify the name of the callback integration. (that is, to be used in the request integration). The identifier value must be the same as the callback integration flow identifier seen in Oracle Integration integrations.
Flow Version	Specify the version number of the callback integration. The version value must be the same as the callback integration flow version seen in Oracle Integration integrations.

Summary Page

You can review the specified adapter configuration values 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 **Go back**. To cancel your configuration details, click **Cancel**.

5

Implement Common Patterns Using the SOAP Adapter

You can use the SOAP Adapter to implement the following common patterns.

Topics:

Note:

Oracle Integration offers a number of prebuilt integrations, known as *recipes*, that provide you with a head start in building your integrations. You can start with a recipe, and then customize it to fit your needs and requirements. Depending upon the solution provided, a variety of adapters are configured in the prebuilt integrations.
See the Recipes and Accelerators page on the Oracle Help Center.

Best Practices for Invoking SOAP Endpoints

Follow these best practices for invoking SOAP endpoints with the SOAP Adapter.

- If you receive errors (for example, `401`, `429`, or `50x` errors) while invoking SOAP endpoints with the SOAP Adapter, ensure that you employ instance retries.
- Client applications should cache the token while invoking OAuth-protected SOAP endpoints.

Configure MTOM Support in the SOAP Adapter

This use case describes how to configure Message Transmission Optimization Mechanism (MTOM) support in the SOAP Adapter.

See SOAP Adapter Capabilities for conceptual information.

SOAP Message Examples and Structure

The following example shows a SOAP message with inline binary content:

```
MIME-Version: 1.0
Content-Type: Multipart/Related; boundary=MIME_boundary;
type="application/soap+xml"; start="<claim@insurance.com>"

--MIME_boundary
Content-Type: application/soap+xml; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-ID: <claim@insurance.com>

<Envelope>
  <Body>
    <ReceiveImage>
     <filename>abc.jpg</filename>
      <image>.... JPEG image base64 .....</image>
    </ReceiveImage>
  </Body>
</Envelope>
```

The following example shows a SOAP message with MTOM/XOP:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:xop='http://www.w3.org/2004/08/xop/include'
xmlns:xop-mime='http://www.w3.org/2005/05/xmlmime'>
 <soap:Body>
   <Order>
     <orderNumber>ABC</orderNumber>
     <orderType>backorder</orderType>
     <image xop-mime:content-type='image/jpeg'>
       <image xop-mime:content-type='image/jpeg'>
     </image>
   </Order>
 </soap:Body>
</soap:Envelope>


--MIME_boundary


Content-Type: image/jpeg
Content-Transfer-Encoding: binary
Content-ID: <image@insurance.com> 4
```

```
...binary JPG image...


--MIME_boundary--
```

The MTOM message structure is as follows:

- The `start` parameter indicates which part of the MIME message is the root XOP document.
- The `Content-ID` value identifies a part of the MIME message. In this case, it is the root XOP document.
- The `<xop:Include>` element references the JPEG binary attachment.
- The `Content-ID` identifies the JPEG in the binary attachment.

application/octet-stream MIME attachment

A MIME attachment with the content type `application/octet-stream` is a binary file. It is typically an application or document that must be opened in an application, such as a spreadsheet or word processor. If the attachment has a file name extension associated with it, you may be able to identify the type of file. For example, an `.exe` extension indicates it is a Windows or DOS program (executable). A file ending in `.doc` can probably be opened in Microsoft Word. In addition to the generic `application/octet-stream` content type, you may also encounter attachments that have different subtypes (for example, `application/postscript`, `application/x-macbinary`, and `application-msword`). They are similar to `application/octet-stream`, but apply to specific types of files.

SOAP Message Transmission Optimization Mechanism/XML-binary Optimized Packaging (MTOM/XOP) describes a method for optimizing the transmission of XML data of type `xs:base64Binary` in SOAP messages. When the transport protocol is HTTP, MIME attachments carry that data while at the same time allowing both the sender and the receiver direct access to the XML data in the SOAP message without having to be aware that any MIME artifacts were used to marshal the `xs:base64Binary` data. The binary data optimization process involves the following:

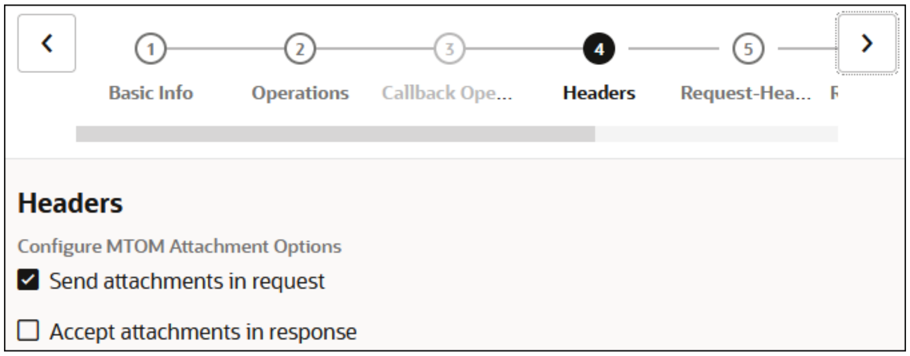
- Encoding the binary data
- Removing the binary data from the SOAP envelope
- Compressing and attaching the binary data to the MIME package
- Adding references to that package in the SOAP envelope

When MTOM is enabled, the MTOM specification does not require that the web service runtime use XOP binary optimization when transmitting base64binary data. Instead, the specification enables runtime to choose to do so. This is because in certain cases the runtime may decide that it is more efficient to send base64binary data directly in the SOAP message (for example, when transporting small amounts of data in which the overhead of conversion and transport consumes more resources than just inlining the data as is). However, the Oracle WebLogic Server web services implementation for the MTOM for JAX-RPC service always uses MTOM/XOP when MTOM is enabled.

Design Time

When you configure the SOAP Adapter as an invoke connection in an integration, MTOM attachment options are shown in the Adapter Endpoint Configuration Wizard when a base64Binary element is present in the WSDL messages for a given operation in both request and response messages (for synchronous).

1. Specify the base64Binary element-based WSDL in the Connections page when configuring the SOAP Adapter.
2. Enable the appropriate **Send attachments in request** (for outbound request) and **Accept attachments in response** (for outbound response) options to enable MTOM processing for that endpoint.



Headers

Configure MTOM Attachment Options

☑ Send attachments in request

☐ Accept attachments in response

MTOM support cannot be configured in a trigger connection.

If you select to enable MTOM for a request message, the XPath for that binary node is calculated and stored in the JCA file as an interaction spec property.

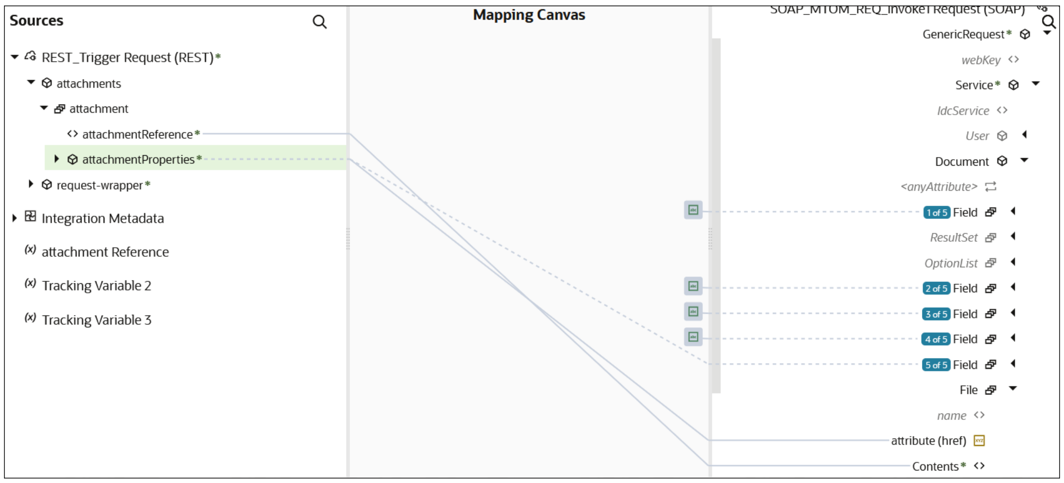
For example:

```
<property name="attachmentXpathInfo" value="/*[namespace-uri()='http://
www.oracle.com/UCM' and
local-name()='GenericRequest']/*[namespace-uri()='http://
www.oracle.com/UCM' and local-name()=
'Service']/*[namespace-uri()='http://www.oracle.com/UCM' and local-
name()='Document']/*[namespace-uri()
='http://www.oracle.com/UCM' and local-name()='File']/*[namespace-
uri()='http://www.oracle.com/UCM'
and local-name()='Contents']"/>
```

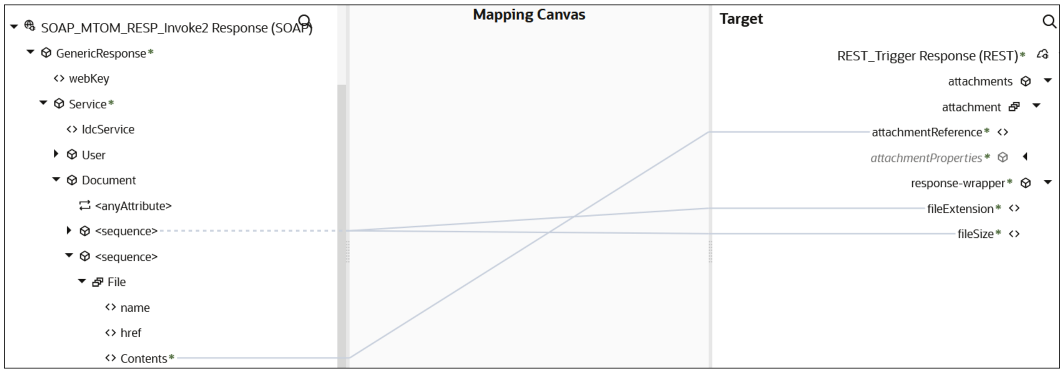
For a response message, the property `attachmentXpathInfoForResponse` is used in the JCA file to represent the XPath.

Mapping

1. For an outbound request, any attachment reference from the virtual file system (VFS) can be mapped to the base64Binary element of the outbound message. As shown below, **attachmentReference** from the REST source is mapped to the base64Binary element of the message.



2. For an outbound response, the attachment is saved to the VFS and the base64Binary element of the payload holds the VFS reference. The VFS reference can be further used to map it to a REST resource or an FTP Adapter:



Runtime

During runtime, MTOM processing is triggered based on the availability of the `attachmentXpathInfo` and `attachmentXpathInfoForResponse` properties in the JCA file. This information is persisted during design time.

In the outbound request, the XPath information given by `attachmentXpathInfo` in the JCA file creates an attachment in the cloud message and structures the SOAP message in the MTOM-specific format.

In the outbound response, the logic checks if there is any attachment received in the response of the cloud message, which is further saved in the VFS. The node represented by the property `attachmentXpathInfoForResponse` substitutes it with the VFS reference of the attachment in the cloud message.

Consume Taleo SOAP APIs

Taleo enterprise edition SOAP APIs are protected. To use these APIs with the SOAP Adapter, you must use the WSDL file upload feature in the Connection Properties dialog of the Connections page instead of specifying the WSDL with the HTTP URI.

See Configure Connection Properties.

Invoke a SOAP-Based Integration with a Timestamp

You can use an interface such as SoapUI to invoke a SOAP-based web service integration in Oracle Integration. If you attempt to invoke a SOAP-based web service through SoapUI, you can receive the following error regardless of the security policy you selected in the Connections page (Username Password Token, Basic Authentication, or No Security Policy).

```
OWSM ICS Service request handler failed: InvalidSecurity
```

If this error occurs, add a timestamp to your WSS security. For example:

```
<soap:Header>
  <wsse:Security  xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-wssecurity-secext-1.0.xsd">
<wsu:Timestamp wsu:Id="IS-918F6052E6918D5F2414550387589204">
  <wsu:Created>2017-04-17T17:20:50.9200</wsu:Created>
  <wsu:Expires>2017-04-17T17:21:50.9200</wsu:Expires>
</wsu:Timestamp>
<wsse:UsernameToken>
<wsse:Username>Joe</wsse:Username>
<wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-username-token-profile-1.0#PasswordText">My_password</
wsse:Password>
</wsse:UsernameToken>
</wsse:Security>
  </soap:Header>
```

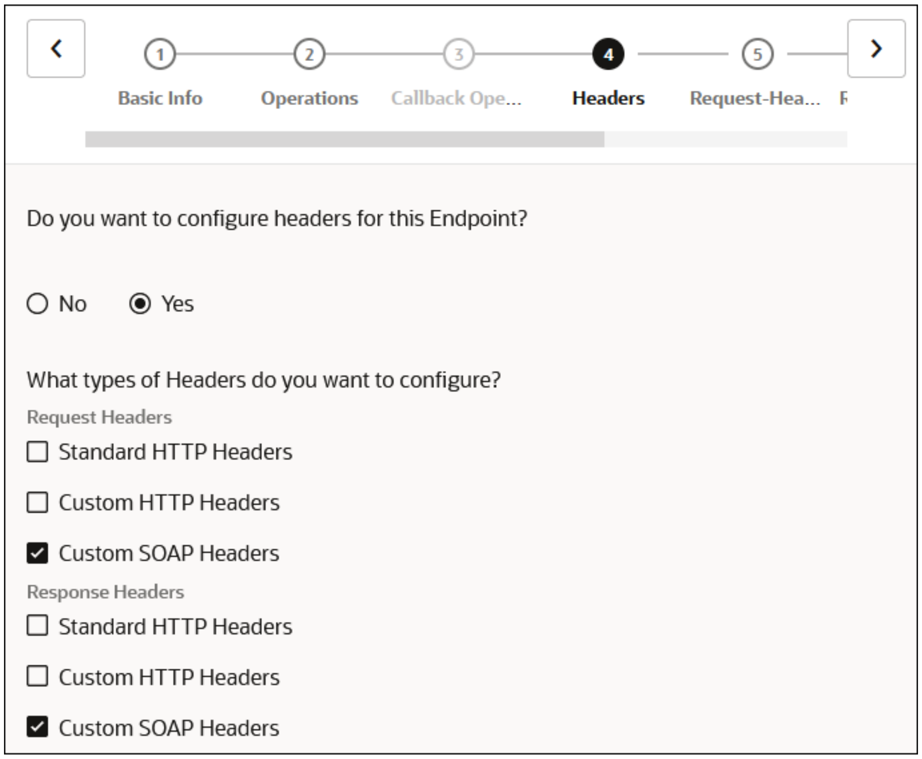
Configure Custom SOAP Headers for the SOAP Adapter

Oracle Integration supports adding custom SOAP headers that are not defined in the WSDL definition provided on the Connection page. You add custom headers by uploading a schema on the **Custom SOAP Headers** tab on the Request Headers page and/or Response Headers page, respectively, of the Adapter Endpoint Configuration Wizard.

Add Custom SOAP Headers

Configure custom SOAP headers and specify a valid schema from which an element can be selected as a custom SOAP header in the Adapter Endpoint Configuration Wizard:

1. On the Header page, select **Yes** for **Do you want to configure headers for this Endpoint?**.
2. For **What types of Headers do you want to configure**, select **Custom SOAP Headers** in the **Request Headers** and **Response Headers** sections, as applicable.



Do you want to configure headers for this Endpoint?

No [✓ Yes]

What types of Headers do you want to configure?

Request Headers

- [ ] Standard HTTP Headers
- [ ] Custom HTTP Headers
- [x] Custom SOAP Headers

Response Headers

- [ ] Standard HTTP Headers
- [ ] Custom HTTP Headers
- [x] Custom SOAP Headers

3. On the Request page and/or Response page, upload a valid schema from which an element can be selected as a custom SOAP header. Note the following restrictions on schemas to upload.
   - The schema to upload must have a `targetNamespace`.
   - Endpoints that expect custom SOAP headers without a namespace are not supported.
   - Ensure the schema is defined with `elementFormDefault` as `qualified` (for child elements with a namespace prefix) or `unqualified` (for child elements without a namespace prefix), as required.
   - A schema with dependencies (imports/includes/cross references) is not supported. See Example Schemas.
   - Only one element can be selected as a header from the uploaded schema. To select multiple elements:
     – For the same `targetNamespace`, a schema must be uploaded again for each element.
     – For a different `targetNamespace`, a different schema with required `targetNamespace` can be used.
4. Select the required element from the list of elements displayed from the uploaded schema.
5. Click **Add Header** to add the element to the configured headers table.

Delete Custom SOAP Headers

1. On the Request and/or Response page, select the check box next to the required header element in the configured headers table.

2. Click **Delete Header** to remove the element from the configured headers table.

Example Schemas

The following schema includes a `targetNamespace`. A prefix for a child is required.

Schema	XML

`<xsd:schema`
`elementFormDefault="`**`qualified`**`"`
`targetNamespace="http://`
`xmlns.oracle.com/`
`sample" xmlns:xsd="http://`
`www.w3.org/2001/XMLSchema">`
`  <xsd:element name="headerRoot">`
`    <xsd:complexType>`
`      <xsd:sequence>`
`        <xsd:element`
`name="headerChild"`
`type="xsd:string"/>`
`      </xsd:sequence>`
`    </xsd:complexType>`
`  </xsd:element>`
`</xsd:schema>`
	`<hdr:headerRoot xmlns:hdr=http://`
	`xmlns.oracle.com/sample>`
	`<hdr:headerChild>headerValue</`
	`hdr:headerChild>`
	`</hdr:headerRoot >`

The following schema includes a `targetNamespace`. A prefix for a child is not required.

Schema	XML

`<xsd:schema`
`elementFormDefault="`**`unqualifi`**
`ed``" targetNamespace="http://`
`xmlns.oracle.`
`com/sample" xmlns:xsd="http://`
`www.w3.org/2001/XMLSchema">`
`  <xsd:element name="headerRoot">`
`    <xsd:complexType>`
`      <xsd:sequence>`
`        <xsd:element`
`name="headerChild"`
`type="xsd:string"/>`
`      </xsd:sequence>`
`    </xsd:complexType>`
`  </xsd:element>`
`</xsd:schema>`
	`<hdr:headerRoot xmlns:hdr=http://`
	`xmlns.oracle.com/sample>`
	`<headerChild>headerValue</`
	`headerChild>`
	`</hdr:headerRoot >`

Call Oracle Fusion Applications Business Intelligence Publisher Report Services

Oracle Fusion Applications provides web services for Business Intelligence (BI) Publisher services to execute reports synchronously and get responses using the following services.

- Public Services: Expects credentials in the payload/body of the SOAP envelope.
  - ReportService: runReport method (See ReportService in *Developer's Guide for Oracle Business Intelligence Publisher*.)
    - `xmlpserver/services/v2/`**`ReportService`**`?wsdl`
  - ScheduleService: getDocumentData method (See ScheduleService in *Developer's Guide for Oracle Business Intelligence Publisher*.)
    - `xmlpserver/services/v2/`**`ScheduleService`**`?wsdl`
- Protected Services: Expects credentials as part of headers in the SOAP envelope.

  > **Note:**
  >
  > This service is recommended if you have to use synchronous calls for BI reports.

  - ExternalReportWSSService: runReport method
    - `xmlpserver/services/`**`ExternalReportWSSService`**`?wsdl`
  - ScheduleReportWSSService: getDocumentData method
    - `xmlpserver/services/`**`ScheduleReportWSSService`**`?wsdl`

When the SOAP Adapter calls an Oracle Fusion Applications business intelligence publisher (BIP) report synchronously, the report data is always returned as a base64-encoded string in the response. **These services do not support attachments in the response.** In this case, the feature to configure MTOM support in the SOAP Adapter does not work.

Handling the BI Response

The BI response can be parsed in the integration with the following approaches:

- Recommended: Use the decodeBase64ToReference mapper function to convert the base64 string to a file reference and use the file reference in a stage file action/FTP Adapter read file operation scenario for further processing.
- Alternative: Write the base64 string using an opaque schema to a file using a stage file action/FTP Adapter write file operation scenario, then read the file using a stage file action/FTP Adapter scenario for further processing.

Note:

Upon calling a BIP report synchronously, the request sent to the BIP web service initiates report generation and sends a response back after the report is generated. This can cause latency issues in an Oracle Integration environment.

For the above reason, calling BI report services synchronously with the SOAP Adapter is recommended for short-running and/or small size reports. Be aware of the payload size restrictions. See Service Limits in *Provisioning and Administering Oracle Integration 3*.

For anything above those limits (that is, long-running and/or large size reports), the recommended approaches are as follows:

Using Oracle Universal Content Management (UCM)

Using the Oracle ERP Cloud Adapter and SOAP Adapter:

1. Create a BIP report to schedule/execute and configure the report so that the report response is deposited to UCM.
2. Create a custom ESS job, then execute the BIP report data sent through step 3.
3. Create an Oracle ERP Cloud Adapter invoke connection using the ErpIntegrationService and choose the exportBulkData operation. Ensure that the jobName is set to the custom ESS job created in step 2. jobOptions are set with EnableEvent=Y. This enables an event when the job is completed.
4. Create an Oracle ERP Cloud Adapter trigger connection to subscribe to the ExportBulkDataEvent. This provides a documentId in the event payload that is deposited in UCM.
5. Create a SOAP Adapter invoke using the UCM web service (GenericSoapService) and GET_FILE command in the operation to get the BIP report response as an attachment.

Using FTP

1. Configure the BI report to place the report response in FTP.
2. Create a custom ESS job in Oracle Fusion Applications to execute the BIP report.
3. Submit the ESS job using the ESS web service. See Implement Oracle Enterprise Scheduler Web Service Calls.
4. Get the response from FTP as an attachment once the job completes and the callback is received.

Integrate PeopleSoft with Oracle Integration

You can integrate PeopleSoft with Oracle and non-Oracle Cloud applications using Oracle Integration. This use case describes how to synchronize contact information from Salesforce.com with PeopleSoft, which is a SOAP web service-based integration.

- PeopleSoft Integration using Oracle Integration – Part 1
- PeopleSoft Integration using Oracle Integration – Part 2

Create a Keystore File for a Two-Way, SSL-Based Integration

If you need to create an integration that communicates with a two-way, SSL-enabled server, you must create the keystore file required for establishing an Oracle Integration identity to facilitate a two-way, SSL-based integration.

You can obtain the client certificate in a variety of ways. Select the method that is best for your environment. For example, you can obtain the certificate directly from many certificate authorities. The steps in this section describe how to generate a certificate signing request (CSR) and have it signed by a well known certificate authority.

Note:

- This section describes how to configure Oracle Integration for use in outbound, two-way SSL integrations. To use Oracle Integration in inbound, two-way SSL integrations, you can use the Oracle Cloud Infrastructure API Gateway. The Oracle Cloud Infrastructure API Gateway is integrated with the Oracle Cloud Infrastructure certificates service. This approach enables you to deliver APIs implemented with Oracle Integration that enforce client mTLS.
- Two-way SSL is not supported for calls to external services through the connectivity agent. Two-way SSL requires direct connectivity from Oracle Integration without the connectivity agent.
- The alias name to provide must match the name provided for the private key entry in the JKS file.

See this blog and Adding mTLS support to API Deployments.

- Commonly Used Terms and Tools
- Commands to Create a Client Certificate with the keytool Utility
- Example: Create a Client Certificate with the keytool Utility
- Manage Certificates with openSSL
- Certificate Management - Two-Way SSL or mTLS

Commonly Used Terms and Tools

Term	Description
Secure socket layer (SSL) and Transport Layer Security (TLS)	SSL and TLS, its successor, are protocols for establishing authenticated and encrypted links between networked computers.
Digital certificate	A data file that holds the cryptographic key provided to an organization or entity by a trusted authority. A simple analogy is a driver's license. The license uniquely identifies the person to whom it is issued. The license is issued by the DMV, a trusted authority.

Term	Description
Certificate	A public key and private key form a pair used to encrypt and decrypt data. Public keys can be freely given to anyone who needs to securely exchange data. Private keys must never be shared and must be stored securely. If private keys are listed or compromised, the issuing certificate authority must be notified so they can be added to the certificate revocation list.
Certificate authority (or certification authority)	An entity that issues digital certificates. A digital certificate certifies the ownership of a public key by the named subject of the certificate.
Certificate encoding/formats	• Privacy Enhanced Mail (PEM): The full specification of PEM is in RFC 7468. PEM is the most commonly-used X509 certificate format. It's a base64-encoded string enclosed between:`-----BEGIN CERTIFICATE-----` and `-----END CERTIFICATE-----` • Distinguished Encoding Rules (DER): Binary Format. Cannot be viewed in an editor. • Public Key Cryptography Standards (PKCS): These are a group of public key cryptography standards devised and published by RSA Security LLC. See https://datatracker.ietf.org/wg/pkix/documents/.
TrustStore	A password-protected repository for trust or public certificates. The default location in Java is `$JAVA_HOME/jre/lib/security/cacerts`. All well known certificate authority root and intermediate certificates are available in the JDK truststore.
Keystore	A password-protected repository to hold client or private certificates. Since this store holds private keys, it is imperative that the store resides in a secure location.
Certificate chain	A certificate chain is an ordered list of certificates ending with the root certificate. For trust to be established, the entire certificate chain is traversed. Each certificate is validated by finding the public key of the next issuing certificate authority or intermediate certificate authority, until the root certificate is reached. Certificate chains are usually cached to validate the certificate locally.

The two most commonly used tools for SSL are the following:

Tool	Description
`keytool`	A JDK utility used to perform CRUD operations on a truststore and keystore and to administer certificates. All the commands require the password that was used to create the store. Consult your Java truststore documentation for the default password.

Tool	Description
`openssl`	This is a robust, commercial-grade, full-featured toolkit for the TLS and SSL protocols. It is also a general-purpose cryptography library.

Commands to Create a Client Certificate with the keytool Utility

Commonly used `keytool` commands are as follows.

Caution:

Replace the italicized variables in the commands below with values appropriate to your environment.

Description	Command
List the entire contents of the store	`keytool -list -keystore` *`path_to_the_keystore`*
List the contents in the store for a specific alias	`keytool -list -keystore` *`path_to_the_keystore`* `-alias` *`alias_name`*
View the contents of a certificate	`keytool -printcert -v -file` *`name_of_the_file`*
Export a certificate from the store	`keytool -export -alias` *`alias_name`* `-file` *`certificate_name`* `-keystore` *`path_to_the_store`*
Import a new certificate into the store	`keytool -import -trustcacerts -file` *`path_to_the_certificate`* `-alias` *`alias_name`* `-keystore` *`path_to_the_store`*

To create a client certificate:

Caution:

Italicized variables indicate placeholder variables for which you must supply particular values. If you copy the commands below, ensure that you replace the variables shown in italics with values appropriate to your environment.

1. Go to the Java `bin` directory.

   ```
   %JAVA_HOME%/jre/bin
   ```

2. Enter the following command to create a JKS keystore to hold the certificates.

   ```
   keytool -genkey -keyalg RSA -alias alias_name -keystore
   identityKeystore.jks -storepass password_for_the_keystore -validity 360 -
   keysize 2048
   ```

3. When prompted, change the values provided based on your company's security policy.

```
What is your first and last name?
  [Unknown]:  <FQDN>
What is the name of your organizational unit?
  [Unknown]:  Your_functional_org
What is the name of your organization?
  [Unknown]:  Company
What is the name of your City or Locality?
  [Unknown]:  City_name
What is the name of your State or Province?
  [Unknown]:  State_name
What is the two-letter country code for this unit?
  [Unknown]:  US
Is CN=<>, OU=<>, O=<>, L=Redwood Shores, ST=California, C=US
correct?
  [no]:  yes
Enter key password for <oicclient>
        (RETURN if same as keystore password):
```

4. Verify the existence of the JKS keystore file.

```
ls
```

5. Create a certificate that is ready to be signed.

```
keytool -certreq -alias alias_name -keystore name_of_keystore -
storepass password -storetype JKS -file name_of_csr_certificate.csr
```

6. List the JKS keystore and certificate files in the directory.

```
ls
```

7. Validate your CSR file at the following site.

```
https://ssltools.digicert.com/checker/views/csrCheck.jsp
```

8. Provide the `.csr` certificate file to a signing authority. A signed certificate and any root and intermediate certificates are signed and returned by the authority. A self-signed certificate can be used for testing, but is not allowed in a production environment.

9. If you have root and intermediate certificates, perform the following substeps. Otherwise, go to Step 10.

   a. If you have a root certificate, enter the following command to import the signed root certificate.

   ```
   keytool -import -keystore keystore_name -file
   path_to_root_certificate -alias root_alias_name
   ```

The following example is what you see when importing the DigiCert root certificate.

```
Enter keystore password:
Owner: CN=DigiCert Global Root CA, OU=www.digicert.com, O=DigiCert
Inc, C=US
Issuer: CN=DigiCert Global Root CA, OU=www.digicert.com, O=DigiCert
Inc, C=US
Serial number: 83be056904246b1a1756ac95991c74a
Valid from: Thu Nov 09 16:00:00 PST 2006 until: Sun Nov 09 16:00:00
PST 2031
Certificate fingerprints:
     MD5:  79:E4:A9:84:0D:7D:3A:96:D7:C0:4F:E2:43:4C:89:2E
     SHA1: A8:98:5D:3A:65:E5:E5:C4:B2:D7:D6:6D:40:C6:DD:2F:B1:9C:54:36
     SHA256:
43:48:A0:E9:44:4C:78:CB:26:5E:05:8D:5E:89:44:B4:D8:4F:96:62:BD:26:DB:2
5:7F:89:34:A4:43:C7:01:61
     Signature algorithm name: SHA1withRSA
     Version: 3Extensions:#1: ObjectId: 2.5.29.35 Criticality=false
AuthorityKeyIdentifier [
KeyIdentifier [
0000: 03 DE 50 35 56 D1 4C BB   66 F0 A3 E2 1B 1B C3
97  ..P5V.L.f.......
0010: B2 3D D1 55                                        .=.U
]
]#2: ObjectId: 2.5.29.19 Criticality=true
BasicConstraints:[
  CA:true
  PathLen:2147483647
]#3: ObjectId: 2.5.29.15 Criticality=true
KeyUsage [
  DigitalSignature
  Key_CertSign
  Crl_Sign
]#4: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 03 DE 50 35 56 D1 4C BB   66 F0 A3 E2 1B 1B C3
97  ..P5V.L.f.......
0010: B2 3D D1 55                                        .=.U
]
]Trust this certificate? [no]:  yes
Certificate was added to keystore
```

b. If you have an intermediate certificate, enter the following command to import the signed intermediate certificate.

```
keytool -import -keystore keystore_name -file
path_to_intermediate_certificate -alias intermediate_certificate_alias
```

```
Enter keystore password: replace_with_strong_password
Certificate was added to keystore
```

10. If you have only a single certificate, enter the following command to import the signed certificate.

    ```
    keytool -import -keystore keystore_name -file
    path_to_signed_certificate -alias
    the_same_alias_used_to_create_the_keystore



    Enter keystore password: replace_with_strong_password
    Certificate was added to keystore
    ```

11. Check if all the certificates are in the store.

    ```
    keytool -list -keystore
    ```

12. Export the public certifcate.

    ```
    keytool -export -alias certificate_alias -keystore
    identity_keystore -file your_public_certificate_filename


    Enter keystore password: replace_with_strong_password
    ```

13. Export the public certificate to provide to the server.

    ```
    keytool -export -alias certificate_alias -keystore
    identityKeystore.jks -file your_public_certificate_filename
    Enter keystore password:
    Certificate stored in file your_public_certificate_filename
    ```

14. Import the new keystore into Oracle Integration as an X509 identity certificate.
15. List the entire contents of the store.

    ```
    keytool -list -keystore path_to_the_keystore
    ```

Example: Create a Client Certificate with the keytool Utility

This section provides an example of how to create a client certificate. It uses actual file names. Replace those names with values appropriate to your environment.

1. Enter the following command to create a JKS keystore to hold the certificates.

    ```
    keytool -genkey -keyalg RSA -alias oicclient -keystore
    identityKeystore.jks -storepass replace_with_strong_password -
    validity 360 -keysize 2048
    ```

    Where the following values are entered for this example:

    - `-alias` is the `oicclient` keystore alias.
    - `-keystore` is the `identityKeystore.jks` keystore file.

2. When prompted, change the values provided based on your company's security policy.

   ```
   What is your first and last name?
     [Unknown]:  Joe Smith
   What is the name of your organizational unit?
     [Unknown]:  Development
   What is the name of your organization?
     [Unknown]:  GlobalChips
   What is the name of your City or Locality?
     [Unknown]:  Redwood Shores
   What is the name of your State or Province?
     [Unknown]:  California
   What is the two-letter country code for this unit?
     [Unknown]:  US
   Is CN=<>, OU=<>, O=<>, L=Redwood Shores, ST=California, C=US correct?
     [no]:  yes
   Enter key password for oicclient
           (RETURN if same as keystore password):
   ```

3. Verify the existence of the JKS keystore file.

   ```
   ls
   ```

4. Create a certificate that is ready to be signed.

   ```
   keytool -certreq -alias oicclient -keystore identityKeystore.jks -
   storepass replace_with_strong_password  -storetype JKS -file oicclient.csr
   ```

   Where the following values are entered for this example:

   - `-alias` is the `oicclient` keystore alias.
   - `-keystore` is the `identityKeystore.jks` keystore file.
   - `-file` is the `oicclient.csr` certificate file.

5. List the JKS keystore and certificate files in the directory.

   ```
   ls
   oicclient.csr  identityKeystore.jks
   ```

6. Validate your `.csr` certificate file at the following site.

   ```
   https://ssltools.digicert.com/checker/views/csrCheck.jsp
   ```

7. Provide the `.csr` certificate file to a signing authority. The certificate and any root and intermediate certificates are signed and returned by the authority. A self-signed certificate can be used for testing, but is not allowed in a production environment.
8. If you have root and intermediate certificates, perform the following substeps. Otherwise, go to Step 9.

a. If you have a root certificate, enter the following command to import the signed root certificate.

```
keytool -import -keystore identityKeystore.jks -file
DigiCertGlobalRootCA.crt -alias DigiCertCARoot
```

Where the following values are entered for this example:

- `-keystore` is the `identityKeystore.jks` keystore file.
- `-file` is the `DigiCertGlobalRootCA.crt` signed root certificate file.
- `-alias` is the `DigiCertCARoot` alias.

```
Enter keystore password: replace_with_strong_password
Owner: CN=DigiCert Global Root CA, OU=www.digicert.com,
O=DigiCert Inc, C=US
Issuer: CN=DigiCert Global Root CA, OU=www.digicert.com,
O=DigiCert Inc, C=US
Serial number: 83be056904246b1a1756ac95991c74a
Valid from: Thu Nov 09 16:00:00 PST 2006 until: Sun Nov 09
16:00:00 PST 2031
Certificate fingerprints:
     MD5:  79:E4:A9:84:0D:7D:3A:96:D7:C0:4F:E2:43:4C:89:2E
     SHA1:
A8:98:5D:3A:65:E5:E5:C4:B2:D7:D6:6D:40:C6:DD:2F:B1:9C:54:36
     SHA256:
43:48:A0:E9:44:4C:78:CB:26:5E:05:8D:5E:89:44:B4:D8:4F:96:62:BD:26
:DB:25:7F:89:34:A4:43:C7:01:61
     Signature algorithm name: SHA1withRSA
     Version: 3Extensions:#1: ObjectId: 2.5.29.35
Criticality=false
AuthorityKeyIdentifier [
KeyIdentifier [
0000: 03 DE 50 35 56 D1 4C BB   66 F0 A3 E2 1B 1B C3
97  ..P5V.L.f.......
0010: B2 3D D1 55                                       .=.U
]
]#2: ObjectId: 2.5.29.19 Criticality=true
BasicConstraints:[
  CA:true
  PathLen:2147483647
]#3: ObjectId: 2.5.29.15 Criticality=true
KeyUsage [
  DigitalSignature
  Key_CertSign
  Crl_Sign
]#4: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 03 DE 50 35 56 D1 4C BB   66 F0 A3 E2 1B 1B C3
97  ..P5V.L.f.......
0010: B2 3D D1 55                                       .=.U
]
]Trust this certificate? [no]:  yes
Certificate was added to keystore
```

b. If you have an intermediate certificate, enter the following command to import the signed intermediate certificate.

```
keytool -import -keystore identityKeystore.jks -file
DigiCertGlobalInterCA.crt -alias DigiCertCAInter
```

Where the following values are entered for this example:

- `-keystore` is the `identityKeystore.jks` keystore file.
- `-file` is the `DigiCertGlobalInterCA.crt` signed intermediate certificate.
- `-alias` is the `DigiCertCAInter` alias.

```
Enter keystore password: replace_with_strong_password
Certificate was added to keystore
```

9. If you have only a single certificate, enter the following command to import the signed certificate.

```
keytool -import -keystore identityKeystore.jks -file
my_company_signedcert.pem -alias oiclient
```

Where the following values are entered for this example:

- `-keystore` is the `identityKeystore.jks` keystore file.
- `-file` is the `my_company_signedcert.pem` signed certificate.
- `-alias` is the `oiclient` alias.

```
Enter keystore password: replace_with_strong_password
Certificate was added to keystore
```

10. Check if all the certificates are in the store.

```
keytool -list -keystore identityKeystore.jks
```

11. Export the public certificate corresponding to the private identity certificate.

```
keytool -export -alias oicclient -keystore identityKeystore.jks -file
my_company_signedcert.pem
```

Where the following values are entered for this example:

- `-alias` is the `oicclient` keystore alias.
- `-keystore` is the `identityKeystore.jks` keystore file.

- `-file` is the `my_company_signedcert.pem` public certificate file.

```
Enter keystore password: replace_with_strong_password
Certificate stored in file my_company_signedcert.pem
```

12. Import the new keystore (`.jks` file) into Oracle Integration as an X509 identity certificate.
13. List the entire contents of the store.

```
keytool -list -keystore identityKeystore.jks
```

Manage Certificates with openSSL

Commonly used `openssl` commands are as follows:

Description	Command
Check a certificate	`openssl x509 -in` *`certificate_name`* `-text -noout`
Get all certificates from a server	`openssl s_client -connect` *`host`*`:`*`ssl_port`* `-showcerts`
Convert a DER format certificate to PEM format	`openssl x509 -inform der -in` *`path_to_DER_certificate`* `-out` *`path_to_PEM_certificate`*
Convert a `.pfx` file to a JKS store	`keytool -importkeystore -srckeystore` *`path_to_.pfx_file`* `-srcstoretype pkcs12 -destkeystore` *`path_to_the_jks_file`* `-deststoretype JKS -srcstorepass` *`pfx_passwd`* `-deststorepass` *`pfx_passwd`*
Convert a `.jks` file to PKCS12 format	`keytool -importkeystore -srckeystore` *`path_to_.jks_file`* `-destkeystore` *`full_path_to_.p12_file`*`-srcstoretype JKS - deststoretype PKCS12 -deststorepass` *`pkcs12_store_password`*
Extract a private key from a `.pfx` file	`openssl pkcs12 -info -in` *`path_to_.pfx_file`* `-nodes -nocerts -out` *`private_key_file_name`*
Extract a public certificate from a `.pfx` file	`openssl pkcs12 -in` *`path_to_.pfx_file`* `-out` *`path_to_certificate_file`* `-nokeys`

Certificate Management - Two-Way SSL or mTLS

See Debugging SSL/TLS Connections.

To upload an identity certificate:

1. In the navigation pane, select **Home** > **Settings** > **Certificates**.
2. Click **Upload**.
3. Set the alias name to the alias listed in the keystore for the identity certificate. (Use `keytool -list` to see the contents of the keystore.)

4. Make sure the certificate category is set to **Identity**.
5. Upload the client certificate file in JKS format.
6. Enter the keystore and key passwords used to create the JKS store. If there is a mismatch in the passwords, Oracle Integration cannot access the identity certificates.
7. Create a new adapter connection (SOAP Adapter or REST Adapter connection) in Oracle Integration.
8. On the Connections page, select the two-way SSL checkbox and associate the alias required by the connection to use to complete the SSL connection. This alias must match the value that was entered in the Upload Certificate dialog.

To test Mutual TLS authentication (mTLS):

1. Test access to the endpoint from the browser first. Import the client certificate in `.p12` format into the browser of choice.
2. Enter the endpoint in the browser bar and press **Enter**. A message is displayed asking you to use the client certificate that was imported.
3. Follow the prompts in the message. If the certificate is valid, content is loaded in the browser.
4. If the browser test was successful, test the REST/SOAP adapter connection in Oracle Integration.

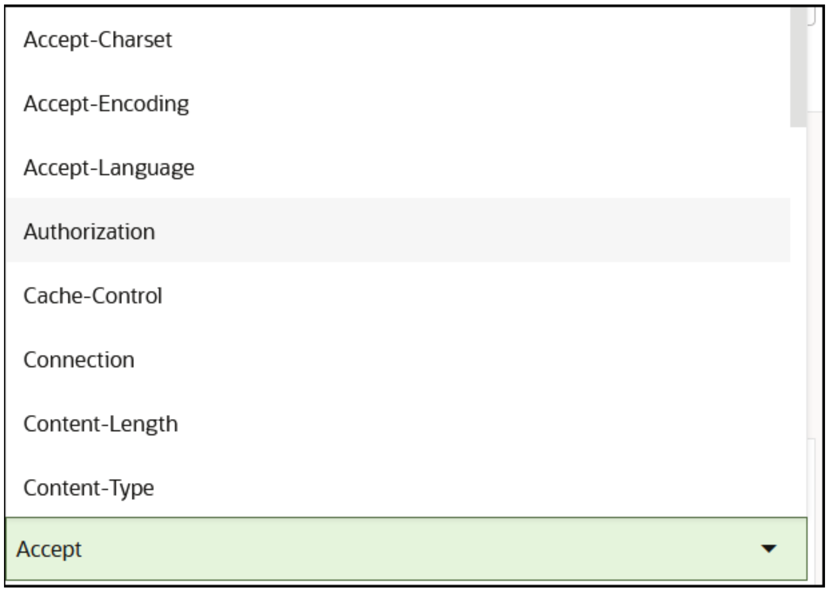
Invoke a SOAP Endpoint with an Explicit Authorization Header

For the Basic Authentication security policy, the runtime message includes an Authorization HTTP header with a BASIC mechanism only (for example, Authorization: Basic base64_content_of_credentials).

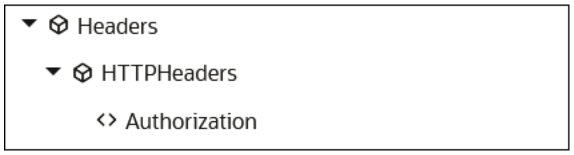
The authorization header can be overridden in the mapper under **Connectivity Properties** > **Security Properties** > **Authorization** in the outbound request mapping to pass dynamic credentials. However, this header also restricts use to the BASIC mechanism only.

To support a different mechanism such as the BEARER mechanism (for OAuth or JWT token) in the outbound direction, the **Authorization** HTTP header is enabled in the SOAP Adapter Request-Headers page. Perform the following steps to enable this support.

1. Create a SOAP Adapter connection.
2. Select **No Security Policy**.
3. Configure the SOAP Adapter as an invoke connection in your integration.
   a. Select **Yes** for **Configure Headers** on the Headers page.
   b. Select the **Standard HTTP Headers** checkbox in the **Request Headers** section.
   c. Click **Add** to enable the header list on the Request-Headers page.
   d. Double-click the newly-added row.
   e. Select the **Authorization** header from the down-down list.



4. Complete adapter configuration.
5. Open the mapper.
   The **Authorization** HTTP header is visible under the **Headers** > **HTTPHeaders** section of the request mapping. Use this to pass any value to the target endpoint.



Implement Oracle Enterprise Scheduler Web Service Calls

Oracle Fusion Applications provide a web service to submit and access Oracle Enterprise Scheduler jobs called the Oracle Enterprise Scheduler Web Service.

Oracle Enterprise Scheduler Web Service URL:

```
https://{FA_HOST}:{FA_PORT}/ess/esswebservice?WSDL
```

When this service URL is used directly within the SOAP Adapter, the following error occurs.

```
SOAPADAPTR-20027: Cause - Unable to parse definition as configured
wsdl is not supported, Action - Verify WSDL definition and please make
sure
that your schema documents are correct
```

This error occurs due to the peculiarity of its WSDL structure: the service URL has a concrete WSDL definition with abstract contents in a separate WSDL used as an import.

The concrete Oracle Enterprise Scheduler Web Service URL only has bindings and a service section in the definition. This imports another WSDL URL (shown below) that contains abstract information (types, messages, and portTypes sections) in the definition.

Oracle Enterprise Scheduler Web Service abstract URL:

```
https://{FA_HOST}:{FA_PORT}/ess/esswebservice?WSDL=ESSWebServiceAbstract.wsdl
```

Receive a Callback from Oracle Enterprise Scheduler Jobs

To receive a callback from Oracle Enterprise Scheduler jobs, an Oracle Integration SOAP endpoint must be exposed. For this, create an integration with a SOAP Adapter as a trigger connection configured with the following information.

1. Create a SOAP Adapter connection with the trigger role.
   a. Enter the Oracle Enterprise Scheduler Web Service abstract URL in the **WSDL URL** field.
   b. Select the **Security Assertion Markup Language (SAML)** security policy.
2. Use this SOAP Adapter connection as a trigger connection in an integration.
   a. On the Operations page of the Adapter Endpoint Configuration Wizard, select the ESSWebServiceCallback port type.
   b. Complete integration design and activate the integration.
3. Ensure that the callback integration endpoint is specified in the corresponding request integration: ReplyTo EndpointURI.

Submit Requests to Oracle Enterprise Scheduler Jobs

To submit requests to Oracle Enterprise Scheduler, the SOAP Adapter must invoke the Oracle Enterprise Scheduler web service. Since the structure of the Oracle Enterprise Scheduler web service is not supported by the SOAP Adapter and requests to the service require WS-Addressing headers to be passed, the following steps describe how to implement this service call.

- Configure the connection using an uploaded WSDL instead of a URL.
  1. Create a SOAP Adapter connection with the invoke role.
  2. Obtain the WSDL file to use from support note 2377662.1 at https://support.oracle.com.
  3. Change the `soap:address` location in the WSDL to the specific Oracle Fusion Applications environment.

     ```
     <soap:address location="https://{FA_HOST}:{FA_PORT}/ess/
     esswebservice"/>
     ```

  4. On the Connections page for the SOAP Adapter, click the **Upload WSDL** button and select the WSDL you updated.
  5. Select the **Username Password Token** security policy.
- Configure the invoke connection.
  The service expects WS-Addressing headers for most calls. Because WS-Addressing headers are not supported out-of-the-box for typical invoke connections, configure the

invoke connection to use custom SOAP headers. See Configure Custom SOAP Headers for the SOAP Adapter.

Most requests require `wsa:Action` and `wsa:MessageId` headers. The following schema can be used to configure the WS-Addressing headers for the invoke connection.

Sample WS-Addressing Schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema attributeFormDefault="unqualified"
    elementFormDefault="qualified"
    targetNamespace="http://www.w3.org/2005/08/addressing"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:wsa="http://
www.w3.org/2005/08/addressing">
    <xsd:element name="MessageID" type="xsd:string"/>
    <xsd:element name="Action" type="xsd:string"/>
</xsd:schema>
```

Troubleshoot

Error	Cause
Service invocation fails with the following error: `A required header representing a Message Addressing Property is not present.`	One or more of the WS-Addressing headers are missing in the request.

Service invocation fails with the following error:
 	
`Job definition with name {JOB_DEFINITION_NAME} not found.`
	The Oracle Enterprise Scheduler web service endpoint URL should be the following:
	
	`https://{FA_HOST}:{FA_PORT}/ess/esswebservice`
	
	and not this:
	
	`https://{FA_HOST}:{FA_PORT}/bi/ess/esswebservice`

Service invocation fails with the following error: `java.lang.NullPointerException`	The Oracle Enterprise Scheduler web service request expects a `requestParameters` element in the request sent. Ensure that you create a target node for this element in the request mapper so that the element is created in the runtime request with an empty/null value. You can also map `#NULL` for this element.

Error	Cause
Service invocation fails with the following error: `User $USER does not have sufficient privilege to perform operation submitRequest.`	See this blog for possible reasons.

6

Troubleshoot the SOAP Adapter

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

Topics:

Regenerate the SOAP Adapter Connection After WSDL Definition Updates

When a WSDL definition is updated in the connection, perform the following steps described in sequence for the SOAP Adapter to reflect the changes.

Common use cases for regenerating the SOAP Adapter after updating the connection are as follows:

- The updated WSDL definition in the connection is not reflected in the integration.
- Regeneration does not occur for the trigger/invoke connection.
- The **Regenerate Artifacts** option is not available for an integration that contains the trigger/invoke connection.

1. Update and save the SOAP Adapter connection.
2. From the menu at the far right for the updated SOAP Adapter connection, click **Refresh Metadata**.

3. Click the **Information** icon for the connection and look for the **Last Refresh Status** to indicate **Complete**.
4. Edit the integration and edit the SOAP Adapter trigger or invoke connection to start the Adapter Endpoint Configuration Wizard.
5. Re-navigate through the pages of the wizard until you reach the Summary page, then click **Done**.
6. Accept the major changes or minor changes warning message (as applicable based on the changes to the connection).

   The changes are reflected in the mapper and Expression Builder.

Use the Correct SoapUI Version to Load and Test SOAP Endpoints

When using the SoapUI to load and test Oracle Integration SOAP endpoints, note that older versions of the SoapUI do not use TLS 1.2 as the default communication protocol, while Oracle Integration SOAP endpoints support only TLS 1.2 for the inbound (trigger) direction. This mismatch results in the following error:

```
unable to load wsdl
```

To avoid this issue, perform one of the following tasks:

- Use a SoapUI version greater than 5.4.0.
- Set the following flag in `SoapUI.vmoptions`:

  ```
  -Dsoapui.https.protocols=TLSv1.2
  ```

Edit Adapter Connections in Active Integrations

You can edit adapter connections that are currently in use in active integrations. For the changes to take effect, you must then reactive the impacted integrations. This is the expected behavior.

For example, assume you import and activate an integration that includes a SOAP Adapter connection as the initial trigger. You then edit the in-use SOAP Adapter connection on the Connections page and modify the WSDL URL, test the connection, and save it. A message is displayed that warns you to reactivate any integrations using this connection for these changes to take effect.

SOAP Endpoint Invocation Fails with OSB-380001: mustUnderstand Error

If SOAP endpoint invocation fails in an integration with the following `OSB-380001 - mustUnderstand` fault, there are several possible solutions.

```
<fault xmlns="http://www.bea.com/wli/sb/context">
<errorCode>OSB-380001</errorCode><
```

```
<reason>Fault received on invocation of target : https://host:port/endpoint
<![CDATA[ Fault Code : codeNS:MustUnderstand Fault String : Unprocessed
'mustUnderstand' header element:
{http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd}
Security ]]> </reason>
<location>node>RouteNode1</node><path>response-pipeline</path>
</location>
</fault>
```

Solutions:

- If the SOAP Adapter connection was created with a static WSDL file or dynamic WSDL URL (valid URL), but the connection fails at runtime with the following error/fault message pointing to an unexpected host/port or endpoint URL, ensure that the WSDL has the correct service endpoint URL.

  ```
  <service name="PrecisionService">
     <port binding="tns:PrecisionServiceBinding" name="PrecisionPort">
      <soap:address
  location="http://correct_host:correct_port/correct_soap_endpoint>"/>
     </port>
  </service>
  ```

- If the WSDL has incorrect endpoint information as seen in the fault message, fix the WSDL service port to resolve the issue.

Specify Connection Property Values with the REST API

If using the REST API to specify SOAP Adapter connection properties, the property values to specify are different than those specified in the Connections page of Oracle Integration. If you incorrectly specify these values, the security policy is changed and you receive runtime errors.

To correctly configure these properties, specify the following values as JSON input:

Configuration Property	If Yes, Set the Property as Follows:	If No, Set the Property as Follows:
Suppress insertion of timestamp into the request (Optional)	`"propertyName" : "suppressTimestampForRequest", "propertyValue" : "true"`	`"propertyName" : "suppressTimestampForRequest", "propertyValue" : "false"`
Ignore timestamp in the response message (Optional)	`"propertyName" : "ignoreTimestampInResponse", "propertyValue" : "true"`	`"propertyName" : "ignoreTimestampInResponse", "propertyValue" : "false"`

Configuration Property	If Yes, Set the Property as Follows:	If No, Set the Property as Follows:
Enable two way SSL for outbound connections (Optional)	`"propertyName" : "enableTwoWaySSL", "propertyValue" : "true"`	`"propertyName" : "enableTwoWaySSL", "propertyValue" : "false"`

Specify the Transport Layer Security (TLS) version of the target server as follows.

For cases in which Oracle Integration calls Oracle Integration and you need to specify the TLS version, it should always be `TLSv1.2`.

Configuration Property	If the Property Value is TLSv1.1:	If the Property Value is TLSv1.2:
Target Server's TLS version (Optional)	`"propertyName" : "tlsVersion", "propertyValue" : "TLSv1.1"`	`"propertyName" : "tlsVersion", "propertyValue" : "TLSv1.2"`

Callback Integrations Fail with a Configured SOAP Action Mismatch Error

A callback integration fails with a configured SOAP action mismatch error when the trigger in the callback integration is configured with a connection using the **Upload File** checkbox to upload a WSDL that does not have a binding section.

As a workaround, change **Disable SOAP Action Validation** to **Yes** on the Operations page for the trigger configuration and reactivate the flow.

Integrations Fail with SAML Security Policy Selected in Inbound Direction

Verify if the target service issuer certificates (also called DemoCA or CloudCA) are imported into Oracle Integration using the **Message Protection Certificate** option. Verify if the SAML user is configured correctly.

Schemas Not Successfully Loaded in Mapper When Using Headers Configured with WSDLs Ending in asmx

If headers are configured in the SOAP Adapter to use Dot Net or Microsoft WCF-based services (a WSDL that generally ends with `asmx`), the schemas are not successfully loaded in the mapper because the prefixes for schema and prefix `s4` for the element type are not defined at the schema level.

The schemas are in line to the WSDL with prefix declarations at the WSDL level, but not at the schema level. The following code sample is from the WSDL:

```
<wsdl:definitions xmlns:s="http://www.w3.org/2001/XMLSchema" .........
xmlns:s4="http://webservices.com/1.0/Core/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
targetNamespace="http://tempuri.org/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
....
   <s:schema elementFormDefault="qualified"  targetNamespace="http://
webservices.com/1.0/Core/">
   <s:element name="WSHeader" type="s4:WSHeader" />
---
</wsdl:Definitions>
```

As a workaround add the prefix declarations manually in the schemas, re-import the IAR file and proceed. The following is an example of the schema in the WSDL after artifact generation:

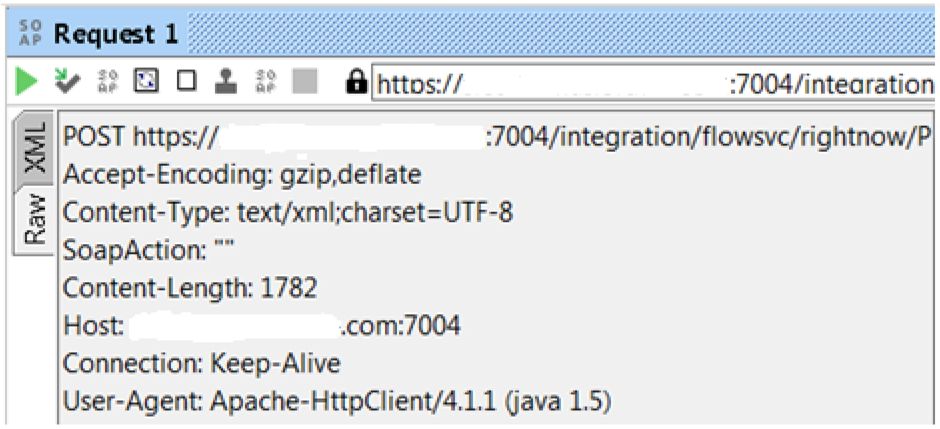
```
<s:schema elementFormDefault="qualified"
targetNamespace="http://webservices.com/1.0/Core/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:s4="http://webservices.com/1.0/Core/">
....
</s:schema>
```

Resolve SOAP Action Mismatch Errors in the SoapUI

If you have a SOAP action mismatch, a `500 Internal server` error occurs while invoking the integration from the SoapUI. For example, you may have a Salesforce Adapter configured with an empty SOAP action and expecting an empty SOAP action from the client. However, clients such as the SoapUI generate requests with a SOAP action identifier read/available from the WSDL file. To resolve this error, update the default SOAP action with SOAP headers that include empty values.

1. Start the SoapUI.
2. Import the WSDL.
3. Click the **Header** tab at the bottom of the page.
4. Click the **+** button to create a header.

   The Add HTTP Header dialog is displayed.
5. Specify the name of the header to add (for example, `SoapAction`)
6. Update the value for the SOAP action header. For this example, the **Value** field is left empty to match the Salesforce Adapter.
7. Verify the updated headers by clicking the **Raw** tab.



https://   :7004/integration

```
POST https://                    :7004/integration/flowsvc/rightnow/P
Accept-Encoding: gzip,deflate
Content-Type: text/xml;charset=UTF-8
SoapAction: ""
Content-Length: 1782
Host:                    .com:7004
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.1.1 (java 1.5)
```

Resolve Exceptions While Invoking Oracle Integration From External Clients

If you receive an exception error while invoking Oracle Integration from an external client (for example, SoapUI), Oracle Integration requires a WSS username token. The WSS username token must be passed from an external client such as SoapUI.

A sample header with a WSS username token looks as follows:

```
<wsse:Security soapenv:mustUnderstand="1" xmlns:wsse="http://
docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
   <wsu:Timestamp wsu:Id="TS-135B2A61F1A7AB1C3914919876769232">
      <wsu:Created>2017-04-12T09:01:16.922Z</wsu:Created>
      <wsu:Expires>2017-04-12T09:02:16.922Z</wsu:Expires>
   </wsu:Timestamp>
   <wsse:UsernameToken
wsu:Id="UsernameToken-135B2A61F1A7AB1C3914919876702101">
    <wsse:Username>weblogic</wsse:Username>
    <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-username-
token-profile-1.0#PasswordText">password</wsse:Password>
    <wsse:Nonce EncodingType="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-soap-
message-security-1.0#Base64Binary">Jkyg2D3NCGPBs5q6j/jhQg==</
wsse:Nonce>
    <wsu:Created>2017-04-12T09:01:10.208Z</wsu:Created>  </
wsse:UsernameToken>
 </wsse:Security>
```

To set the WSS username token:

1. Start SoapUI.
2. Click the **AUTH** tab.
3. Select **Basic** from the Add Authorization dialog, and click **OK**.

4. Complete the authorization credentials.
5. Right-click the payload and select the **Add WSS Username Token** and **Add WSS–Timestamp** options.
6. Specify the time in the Specify Time-to-Live value dialog, and click **OK**.

Connection Error When Using the Incorrect TLS Version

If you receive the following design time or runtime error and you have already imported your SSL certificate into Oracle Integration, ensure that you are using the correct TLS version.

Design Time Error:

```
javax.net.ssl.SSLHandshakeException: Remote host closed connection during
handshake
or
java.net.SocketException: Connection reset
or
Caused by: javax.net.ssl.SSLHandshakeException: Remote host closed connection
during handshake
at sun.security.ssl.SSLSocketImpl.readRecord(SSLSocketImpl.java:946)
at
sun.security.ssl.SSLSocketImpl.performInitialHandshake(SSLSocketImpl.java:131
2
)
at sun.security.ssl.SSLSocketImpl.startHandshake(SSLSocketImpl.java:1339)
at sun.security.ssl.SSLSocketImpl.startHandshake(SSLSocketImpl.java:1323)
at sun.net.www.protocol.https.HttpsClient.afterConnect(HttpsClient.java:563)
....
.....
```

Runtime Error:

```
"type" : "http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.5.1",
"title" : "Internal Server Error",
"detail" : "Internal server error. Please contact oracle support for
details.",
"o:errorCode" : "500",
"o:errorDetails" : [ {
"type" : "UnMappedFault:execute",
"instance" : "\n \n \n \n \n \n SYSTEM_ADMINISTRATOR\n \n \n sysadmin\n \n
\n
...
...
```

Extra Information is Included in the Response Headers Returned as Part of the Response Message

When standard HTTP headers are used in the response headers as part of the response message, extra information is included as part of the returned output data at runtime. This issue occurs only when headers are used in the SOAP Adapter. Without headers, the output is returned without extra information in the response message.

The extra information is not an extra namespace. It is a valid namespace matching the prefix of the element. Without headers, the namespace comes as an attribute in the root element. With headers, since elements are converted from a wrapper, the child elements are copied along with their namespaces.

For example, with a connection that uses the same web service, but one with a header (custom HTTP Header) and another one without a header, the body elements in the response message are different as shown below:

Without a header:

```
<nstrgmpr:result>
   <nsmpr6:PartyId>10</nsmpr6:PartyId>
   <nsmpr6:PartyName>Acme Corp</nsmpr6:PartyName>
</nstrgmpr:result>
```

With a header:

```
<nsmpr2:result>
   <nsmpr8:PartyId
xmlns:nsmpr8="http://xmlns.oracle.com/apps/cdm/foundation/parties/
organizationService/">10
   </nsmpr8:PartyId>
            <nsmpr8:PartyName
xmlns:nsmpr8="http://xmlns.oracle.com/apps/cdm/foundation/parties/
organizationService/">Acme Corp
            </nsmpr8:PartyName>
   </nsmpr2:result>
```

Unexpected Use of the Suppression Insertion of Timestamp into WS-Security Header Feature in the SOAP Adapter Causes an Unrelated Error Response

When creating a SOAP Adapter connection with **Suppress insertion of timestamp into WS-Security header** set to **Yes** on the Connections page and using a WSDL that requires a timestamp header, the connection fails during runtime with the following error instead of a message indicating that a valid timestamp is not present:

```
<errorCode>OSB-380001</errorCode>
<reason>InvalidSecurityToken : The security token is not valid.</
reason>
```