Infor API Gateway Administration Guide

Release 2025.x

Important Notices

The material contained in this publication (including any supplementary information) constitutes and contains confidential and proprietary information of Infor.

By gaining access to the attached, you acknowledge and agree that the material (including any modification, translation or adaptation of the material) and all copyright, trade secrets and all other right, title and interest therein, are the sole property of Infor and that you shall not gain right, title or interest in the material (including any modification, translation or adaptation of the material) by virtue of your review thereof other than the non-exclusive right to use the material solely in connection with and the furtherance of your license and use of software made available to your company from Infor pursuant to a separate agreement, the terms of which separate agreement shall govern your use of this material and all supplemental related materials ("Purpose").

In addition, by accessing the enclosed material, you acknowledge and agree that you are required to maintain such material in strict confidence and that your use of such material is limited to the Purpose described above. Although Infor has taken due care to ensure that the material included in this publication is accurate and complete, Infor cannot warrant that the information contained in this publication is complete, does not contain typographical or other errors, or will meet your specific requirements. As such, Infor does not assume and hereby disclaims all liability, consequential or otherwise, for any loss or damage to any person or entity which is caused by or relates to errors or omissions in this publication (including any supplementary information), whether such errors or omissions result from negligence, accident or any other cause.

Without limitation, U.S. export control laws and other applicable export and import laws govern your use of this material and you will neither export or re-export, directly or indirectly, this material nor any related materials or supplemental information in violation of such laws, or use such materials for any purpose prohibited by such laws.

Trademark Acknowledgements

Publication Information

Release: Infor ION API 2025.x
Publication Date: November 25, 2025
Document code: ionapi_2025.x_apigatewayag_cloud_en-us

Contents

About this guide

The *Infor API Gateway Administrator Library* provides information about creating, deploying, and administering application programming interfaces (APIs) in the Infor API Gateway platform. It includes instructions to help you implement and manage APIs within your organization to support integration and system-to-system communication.

This guide is intended for:

- API developers who create and deploy APIs through API Gateway. See the *Infor OS User and Administration Library* and select **ION API**.
- API client developers who consume APIs in their applications.
- Infor administrators who configure, monitor, and troubleshoot API Gateway and API interactions.
- API modelers who design API orchestrations in **ION Desk**.

Contacting Infor

If you have questions about Infor products or documentation, go to Infor Concierge at https://concierge.infor.com/ and select **Infor Customer Portal** to create a case.

For the latest documentation, go to Documentation Central at docs.infor.com. We recommend that you check this website periodically for updated documentation.

Chapter 1: Overview of API Gateway

API Gateway is a robust platform designed to manage and optimize API interactions within an organization. It provides a common base URL path and a unified authentication mechanism, simplifying access and enhancing security for API consumers.

Serving as a reverse proxy, API Gateway efficiently brokers requests between API consumers and providers, including Infor enterprise and third-party services.

By centralizing API management, API Gateway facilitates seamless integration across systems. It enables administrators to enhance functionality and ensure consistent communication among diverse applications.

Benefits of using API Gateway

Infor API Gateway offers numerous advantages for both API consumers and providers, establishing itself as an essential tool for efficient API management.

Below are some of the key benefits of using API Gateway:

Common base URL path

By offering a common base URL path, Infor API Gateway simplifies the process of accessing various APIs. This feature allows API consumers to easily locate and interact with APIs without having to remember different URLs for each service.

Common authentication mechanism

Infor API Gateway integrates with Infor OS Security to provide a common authentication mechanism. The unified authentication mechanism ensures that only authorized users can access APIs, protecting sensitive data and maintaining system integrity.

Efficient request brokering

As a reverse proxy, Infor API Gateway efficiently brokers requests between API consumers, such as web, mobile, or backend applications, and API providers, including Infor enterprise and third-party services. This intermediary role allows API Gateway to optimize request handling and response times.

Enhanced integration and functionality

The Infor API Gateway plays a pivotal role in facilitating seamless integration across various systems and applications, each of which may have different target authentication servers. By offering a centralized platform for API management, it enables administrators to effectively coordinate and oversee API interactions. This ensures that diverse systems, regardless of their underlying technologies, can communicate and function harmoniously.

Unified monitoring and troubleshooting

Administrators can use the Infor API Gateway to monitor API traffic and troubleshoot issues in real time. API Gateway's comprehensive monitoring capabilities allow for quick identification and resolution of problems, minimizing downtime and ensuring smooth operation.

API orchestration

The API Flows feature within Infor ION is powered by the Infor API Gateway, which allows API administrators to design and manage complex workflows involving multiple APIs. This enables the automation of processes that require interaction between various services. By orchestrating these flows, organizations can achieve higher efficiency and ensure that tasks are performed in a precise and orderly manner. This feature also supports conditional logic, enabling dynamic decision-making based on the responses received from interconnected APIs.

Enterprise Connector

The Enterprise Connector enables secure communication between API Gateway and on-premises APIs, even when there is no direct network connection. It works by running the API Gateway hybrid service as an independent application within the Enterprise Connector.

Backend as a Service (BaaS)

Infor BaaS enables you to build services locally and deploy them in a cloud environment. This approach allows developers to focus on creating and testing back-end functionalities within their own infrastructure before transitioning to a scalable, managed cloud platform.

WIth BaaS, organizations can use prebuilt back-end components, minimize the need for extensive custom coding, and accelerate the deployment process. Additionally, BaaS provides consistent security, efficient data management, and enhanced reliability.

Note: BaaS requires an add-on license in addition to your Infor OS core license. To learn more about Infor BaaS, please visit the Infor documentation site.

Summary

By using the Infor API Gateway, organizations can enhance the efficiency, security, and functionality of their API interactions, ultimately driving better integration and performance across their technological landscape.

Components

Each component is described in the sub-sections that follow.

Client applications

These are the mobile and web applications built by Infor and by Infor customers.

These applications are the ones that are calling the APIs. Rather than calling the many different API servers directly as they did in the past they are now calling via API Gateway. A client application cannot use API Gateway unless it is registered as an authorized client application.

API Gateway engine

This is the gateway engine that is receiving requests from client applications.

The gateway:

- Sets a context for the request; this is like a blackboard where we can keep track of the details of each of many possible in-process requests
- Verifies “inbound” (client application-to-gateway) security using the authentication server
- Obtains the execution plan for the request from the backend service
- Passes the (possibly adjusted) request to the target API server
- Receives the response from the target API server
- Passes the (possibly adjusted) response back to the original calling client application

At any point during processing, we may need to abort the request and return an error response. Finally, we clean up and dispose of the context for the completed request.

Target API servers

These are the real API servers for which API Gateway is acting as a proxy.

Rather than clients directly talking to these servers, client applications talk to the gateway, and the gateway talks to these target servers and adds value while doing so.

Note: Target API servers should be reachable by API Gateway. If these APIs are running on a local network, you can connect to them via Enterprise Connector. See Enterprise Connector on page 69 for details.

API backend service

The backend service was designed to support API Gateway.

It provides services to:

- Obtain the configuration of the gateway instance, for example, what port number to use
- Obtain the execution plan for a request
- Obtain the list of domains a given tenant is allowed to access to support Cross Origin Resource Sharing (CORS)

Authentication server

Infor has built an OAuth 2.0 authentication server that API Gateway uses to validate the OAuth 2.0 bearer token that is passed as part of the request.

If the token is valid, then the Tenant and Identity2 GUID (unique user identifier) are saved into the gateway's request context. If the token is missing or not valid, the gateway immediately returns a 401 unauthorized error to end the request.

Backend security

The following sections discuss backend security, which refers to the security scheme that is implemented as part of your target API server in addition to the OAuth 2.0 inbound security that is built into API Gateway.

Backend protocol HTTPS vs HTTP

Your target API server should allow access only via SSL (HTTPS vs. HTTP) so that all traffic in and out is encrypted.

For this, you need to obtain a certificate and private key from a Certificate Authority (CA) such as Comodo. This key and certificate need to be installed and configured into the engine (IIS, Tomcat, and so on) that is hosting your API. The instructions for doing this are beyond the scope of this guide.

Backend authentication

API Gateway offers a first line of security by requiring that a valid OAuth 2.0 bearer token be passed in the authorization header of the request and that "belongs to" the tenant called out in the request URL.

We still want you to be sure to have a second layer of security at your target API server.

This second layer of security is needed for several reasons:

- Because your target API server is accessible via the public internet as required for API Gateway to be able to reach it.

- Because you may still have legacy applications that access your target API server directly and have not yet been modified to access the API via API Gateway.

Backend authentication choices

API Gateway currently works with three common authentication schemes, which are described in sections below.

Basic authentication

Basic authentication is the simplest and least secure authentication scheme.

Basic authentication can only be considered because the traffic is encrypted using SSL. It is a username + colon + password that is encoded in Base64 and passed in the authorization header of the request. The gateway at runtime builds and adds the proper basic authentication header value to the request before it passes it on to the target API server. The target server verifies that the header is present and decodes the username and password from the header and verifies the values against a database. The target server could use the username to decide what data and actions to which a user has permissions.

Different endpoints can be configured to use different username/password combinations.

OAuth 1.0a Zero-Legged Authentication

OAuth 1.0a Authentication is a signature that is computed using a ConsumerKey and Secret and the details of the request.

If your target API server is using OAuth 1.0a then you likely already have a ConsumerKey and Secret value.

To use OAuth 1.0a Authentication, you will have to supply the ConsumerKey and Secret value at the time of configuration.

At runtime, the gateway applies the ConsumerKey and Secret to the various parts of the request as called on the OAuth 1.0a algorithm to generate a signature string that looks like this:

```
OAuth oauth_consumer_key="YourConsumerKey",oauth_signature_method="HMAC-SHA1",oauth_times
tamp="1444671481",oauth_nonce="19z7xA",oauth_version="1.0",oauth_signature="NHuhgYNoWFAigB
wQidd00Fypjo4%3D”
```

The gateway adds or replaces this signature as the valid authorization header that is passed on to your target API server. Your target API, knowing the ConsumerKey and secret and having access to the same request variables, timestamp, and nonce, should be able to generate the same value as passed in the oauth_signature. If the signatures match, you know the call is valid and can proceed. If they do not match, something is wrong and your server can reject the request as unauthorized.

MutualSSL Authentication

Mutual SSL Authentication or certificate-based mutual authentication refers to two parties authenticating each other through verifying the provided digital certificate so that both parties are assured of the others' identity.

The process of authenticating and establishing an encrypted channel using certificate-based mutual authentication involves these steps:

1. A client (API Gateway) requests access to a protected resource (your target server).
2. The target server presents its certificate to the client.
3. API Gateway verifies the server's certificate.
4. If successful, API Gateway sends its copy of your target's certificate to the server.
5. The target server verifies the supplied certificate.
6. If successful, the target server grants access to the API resource requested by the gateway.

To use MutualSSL on your target server, you must supply a PKCS#12.pfx file that contains your server's certificate, the private key used to generate the certificate, and all the intermediate certificates up to and including the CA root certificate. It is best security practice that your .pfx file is protected by a passphrase. You must supply this passphrase as well. These items will be required at the time of configuration.

At runtime API Gateway reads the BASE64 string from the certificate and converts it back into a binary buffer, which it attaches to the request along with the passphrase. Thus, your target server receives its own certificate as part of the request, which API Gateway forwards to it. If the certificate is not correct, your server would reject the call, but that should never happen unless there has been a misconfiguration or the supplied .pfx was incorrect in some way.

Chapter 2: API client development

This section of the guide is for developers who will be calling APIs via API Gateway to implement applications (API consumers).

The application might be mobile for an Android or iPhone, it might be a web application, or it might be a system-to-system integration.

For more information, see https://github.com/infor-cloud/ion-api-sdk

Application workflow

The following information explains the steps that a developer must perform to build an application that consumes APIs offered via API Gateway.

Preparing to call APIs

After you have chosen the APIs that you will use for your application, review the available documentation for the API and for each operation you will call.

The documentation explains details such as:

- What data/object models the API uses/supports.
- The latter part of the URL path to invoke a specific operation on a specific piece of data.
- Any query-string parameters that are required or optional.
- What HTTP methods (GET, POST) to use when.

You can view any available API documentation via the API Gateway application:

1. Go to **Available APIs** in API Gateway.
2. Double-click an API suite.
3. Click **Endpoints**.
4. To see endpoint documentation, click **Documentation**.

If you prefer to view documentation programmatically, you can use the these URLs:

- REST APIs – {ION API BASE URL}/{API CONTEXT}/{ENDPOINT}/ionapi-doc
- SOAP APIs - {ION API BASE URL}/{API CONTEXT}/{ENDPOINT}?WSDL

These documentation URLs are secured in the same method as the proxy endpoint.

Registering the client application

When creating a new application, you must self-register it in the API Gateway application within the Infor Ming.le Portal.

Registering your application generates an OAuth 2.0 ClientID and Client Secret associated with the application. Your application uses the ClientID/Secret to obtain valid OAuth bearer tokens that allow your application to make calls into API Gateway to access your chosen set of APIs.

Within the API Gateway Application, the user must have the IONAPI-Administrator security role:

1. Select the **Authorized Apps** tab.
2. Select **Add New App** option.
3. Specify a **Name** and select the **Type** of application.
4. Additional fields are shown depending on the Type. Complete the fields required.
5. Click **Save**.

As you save the detail of your application, the system generates a ClientID and associated Secret for the application. As a convenience, there is a button to download these values as a file so you can reference them within the code of your application. The downloaded file contains the following:

Property	Description
ti	Tenant identifier
cn	Application name
ci	ClientID that must be passed to the Authorization Server
cs	Client Secret to pass to the Authorization Server
iu	Base URL for calling API Gateway for this tenant/environment
pu	Base URL for calling the authorization server for this tenant/environment
oa	Path to append to "pu" to create the Authorization URL
ot	Path to append to "pu" to create the Access Token URL
or	Path to append to "pu" to revoke a previously obtained token
SAAK	Service Account Access Key
SASK	Service Account Secret Key

Best practice for OAuth 2.0 client management

When you implement OAuth 2.0 across multiple client applications, assign a unique client ID to each one instead of using a single ID for all.

It is important for these reasons:

1. Security isolation

   Separate client IDs help contain the impact if one application gets compromised. Sharing credentials across apps increases the potential damage.

2. Granular access control

   Different applications often require different scopes or access levels. Unique client IDs let you configure access precisely for each app.

3. Better auditing and monitoring

   With individual client IDs, you can track token usage more effectively, spot irregularities, and troubleshoot problems for each app.

4. Simplified revocation

   You can disable or revoke a single app without disrupting others.

5. Improved lifecycle management

   Each app can be updated, retired, or deployed on its own, using its specific credentials.

6. Compliance and policy enforcement

   You can apply policies, such as token lifetime or redirect URI checks, only when each app is registered individually.

Developing the application

This section includes information for developing your application.

Infor API Gateway SDK

API Gateway is a powerful API management tool. For additional information go to infor.com.

You can use the SDK to:

- Use a previously configured ClientID and Secret to handshake with the Infor Authorization Server to obtain a valid OAuth 2.0 Bearer token. The token is expected by the gateway unless the API endpoint is configured to use the AnonymousInboundSecurity policy, which should be used sparingly.
- Send and receive responses for HTTPS requests using various methods such as GET, PUT, POST, DELETE and provide the appropriate headers and payload (body).
- Handle errors indicated by HTTP status codes other than 200.

Note: Information on developing mobile client applications that authenticate through and access API Gateway services can be found separately in the Infor Mobile SDK.

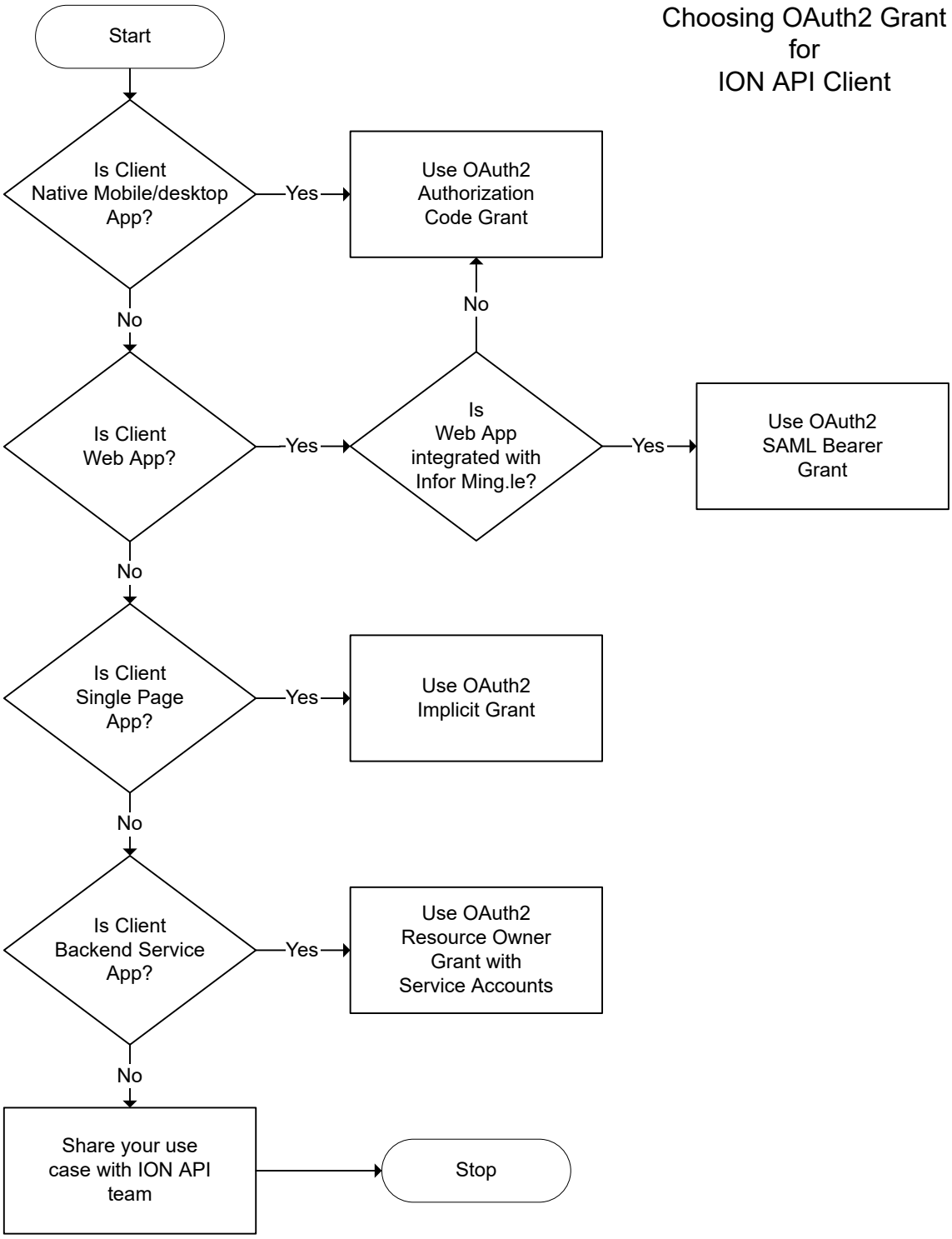
Choosing a grant type

OAuth2 supports different flows to securely consume APIs for different access patterns.

API Gateway supports these grants:

- Authorization code grant - suitable for native mobile/desktop apps and web apps
- Implicit grant - suitable for single page/user agent based applications
- Resource owner grant - suitable for server to server access, for example, backend service client. In these cases user/resource owner is not present for authorization so service accounts are used for back channel authentication and authorization.
- SAML bearer grant - suitable for applications plugged in with Infor Ming.le, for example, apps that have SSO with the Infor Ming.le federation hub.

Based on your client's access pattern, you must implement the appropriate OAuth2 Grant. Here is a decision flow to help you choose an OAuth2 grant:


Choosing OAuth2 Grant
for
ION API Client
Start
Is Client
Native Mobile/desktop
App?
Yes
Use OAuth2
Authorization
Code Grant
No
No
Is Client
Web App?
Yes
Is
Web App
integrated with
Infor Ming.le?
Yes
Use OAuth2
SAML Bearer
Grant
No
Is Client
Single Page
App?
Yes
Use OAuth2
Implicit Grant
No
Is Client
Backend Service
App?
Yes
Use OAuth2
Resource Owner
Grant with
Service Accounts
No
Share your use
case with ION API
team
Stop

Java web applications

API Gateway inbound security requires the client application to use OAuth 2.0 tokens to access API Gateway CE resources.

Web applications must implement an OAuth 2.0 authorization code grant flow to obtain tokens from IFS CE and use the tokens to consume API Gateway CE.

These are the steps required for Java web applications to consume API Gateway CE resources. Additionally, this document provides a sample implementation.

1. Acquire the OAuth Client
2. Obtain the OAuth Token
3. Use the OAuth Token to consume API Gateway

Acquire the OAuth client for Java web applications

To obtain and use OAuth tokens to consume API Gateway, you must acquire an OAuth client specific to your application.

The OAuth client, specific to your app, is created while integrating your app with API Gateway. Your application should keep OAuth 2.0 client details, along with IFS authorization server endpoints.

Obtain the OAuth token for Java web applications

After your app has the OAuth client and IFS authorization server details, use these steps to obtain the OAuth tokens:

1. Send an Authorization Code Request to the IFS authorization server.

   To initiate obtaining the OAuth token, send an authorization code request to the IFS authorization server. This is an HTTP GET or POST request to the authorization endpoint with these parameters:

   client_id
   Specify the OAuth client ID specific to your app.

   redirect_uri
   Specify the URL where the IFS authorization server sends the code upon user consent. This must be the same URL as registered in IFS during integration.

   response_type=code
   Indicate the IFS authorization server to send the authorization code upon user consent parameters.

2. Resource Owner (User) Authentication and Consent (IFS functionality).

   The IFS authorization server works with the IFS Federation Hub to authenticate the user/resource owner and get user consent to release the claims to your app. If the user approves sharing claims with your application, then the IFS authorization server releases the authorization code to your application.

3. Exchange the authorization code for an access token and refresh token.

Using the token endpoint of the IFS authorization server, exchange the authorization code for an OAuth access token and refresh token. Send these parameters as Content-Type "application/x-www-form-urlencoded"

client_id
Specify the OAuth Client ID specific to your app.

client_secret
Specify the OAuth client secret received while acquiring OAuth client details.

grant_type=authorization_code
Specify the hint authorization server about the grant type being used.

redirect_ur
Specify the URL where the authorization server sends the access token. This URL must match the URL registered in ION API CE/IFS CE during integration.

code
Specify the authorization code sent by authorization server in the previous step.

In exchange, the authorization server provides the token_type, for example, Bearer.

Use the OAuth token to consume API Gateway for Java web applications

Use the access token to consume API Gateway endpoints.

You must send the access token in the authorization (HTTP) header.

See the API Gateway inbound security documentation for details.

Refresh the access token for Java web applications

The access token is valid for two hours by default.

However, the access token lifetime can be customized for each authorized app. If the access token is expired, a new access token can be obtained by using refresh token.

These parameters are used to renew the access token using the IFS token endpoint:

- **grant_type=refresh_token**
- **refresh_token**
- **client id**: Use as the username for HTTP basic authentication
- **client secret**: Use as the password for HTTP basic authentication

The OAuth client library automatically handles refreshing expired tokens.

Revoke the token for Java web applications

Revoking tokens prevents orphan grants; therefore, it is crucial to revoke the tokens.

When you revoke the tokens depends on how your application is handling the refresh and access token. Tokens should be revoked before they are discarded by your application or when you want the user/resource owner to reconfirm the grant. After the refresh token is revoked, the corresponding access token and grant are revoked as well.

Use these parameters to revoke the token using the HTTP POST operation for the IFS token endpoint:

- **token**: Refresh token
- **token_type_hint=refresh_token**
- **client id**: Use as the username for HTTP basic authentication
- **client secret**: Use as the password for HTTP basic authentication

Example implementation for Java web applications

You can use an OAuth client library to ease OAuth 2.0 adoption for your application.

The OAuth 2.0 client library handles OAuth-related low-level functionality and provides a simple interface to implement steps documented in the above sections.

See http://oauth.net/2/ for lists of popular OAuth 2.0 client libraries for Java. A sample implementation based on the Apache Oltu OAuth 2.0 Client is provided here. This implementation is a simple web application that integrates with API Gateway and Infor IFS. These are code snippets to implement OAuth:

Request authorization code

```
OAuthClientRequest request = OAuthClientRequest
                .authorizationProvider("https://mingledev01-sso.mingledev.in
for.com:443/ACME_PRD/as/authorization.oauth2")
                .setClientId("ACME_PRD~QxG91-i82CO4P7L5R1YR4YwdOyWw5caGh0UqkvqYrUY")
               .setRedirectURI("http://sample-oauth2-client.infor.com:8080/SampleAppOAuth2/redi
rect"
                .setResponseType("code")
                .buildQueryMessage();
servletResponse.sendRedirect(request.getLocationUri());
```

Exchange authorization code for access token

```
OAuthClientRequest request = OAuthClientRequest
       .tokenLocation("https://mingledev01-sso.mingledev.infor.com:443/ACME_PRD/as/token.oauth2")

        .setGrantType(GrantType.AUTHORIZATION_CODE)
        .setClientId("ACME_PRD~QxG91-i82CO4P7L5R1YR4YwdOyWw5caGh0UqkvqYrUY")
        .setClientSecret("G1-DsyjDTlC6uzaelRKMZMDkfUU-3SUbs2zNdq-Rf9e0xE2G_mJhjqPCZXUPYHTqXQdMP
KEqCwEO94rzmYleBg")
        .setRedirectURI("http://sample-oauth2-client.infor.com:8080/SampleAppOAuth2/redirect")
        .setCode(code)
        .buildQueryMessage();
OAuthClient oAuthClient = new OAuthClient(new URLConnectionClient());
OAuthAccessTokenResponse oauthResponse = oAuthClient.accessToken(request);
String accessToken = oAuthResponse.getAccessToken();
String expiresIn = oAuthResponse.getExpiresIn();
```

Use access token

```
OAuthClientRequest bearerClientRequest = new OAuthBearerClientRequest("https://mingledev01-ion
api.mingledev.infor.com/ACME_PRD/weather/geolookup/q/FL/32266.json")'+
```

```
        .setAccessToken(accessToken)'+
        .buildQueryMessage();'+
OAuthResourceResponse resourceResponse = oAuthClient.resource(bearerClientRequest, OAuth.Http
Method.GET, OAuthResourceResponse.class);
```

Refresh token

```
String reqParam = "refresh_token="+varRefreshToken+"&grant_type=refresh_token";
OAuthClientRequest oauthrequest = OAuthClientRequest.tokenLocation(https://mingledev01-sso.min
gledev.infor.com:443/ACME_PRD/as/token.oauth2+"?"+reqParam)
    .buildBodyMessage();
oauthrequest.addHeader("Authorization", "Basic "+authStringEnc);//use client_id as username,
client_secret as password
OAuthClient oAuthClient = new OAuthClient(new URLConnectionClient());
OAuthResourceResponse resourceResponse = oAuthClient.resource(oauthrequest, OAuth.HttpMethod.POST,
 OAuthResourceResponse.class);';
```

Revoke token

```
String reqParam = "token="+varRefreshToken+"&token_type_hint=refresh_token";
OAuthClientRequest oauthrequest = OAuthClientRequest.tokenLocation(https://mingledev01-sso.min
gledev.infor.com:443/ACME_PRD/as/revoke_token.oauth2+"?"+reqParam)
    .buildBodyMessage();
oauthrequest.addHeader("Authorization", "Basic "+authStringEnc);//use client_id as username,
client_secret as password
OAuthClient oAuthClient = new OAuthClient(new URLConnectionClient());
OAuthResourceResponse resourceResponse = oAuthClient.resource(oauthrequest, OAuth.HttpMethod.POST,
 OAuthResourceResponse.class);
```

Sample application for Java web applications

A sample Java web application is included in this SDK. The application uses the **userdetails** endpoint of the Infor Ming.le API by default.

To change the endpoint, modify the URL in **bearerClientRequest** of `com/infor/ionapi/sample/client/web/OAuth2Servlet.java`. To run the source, extract the source and run `mvn jetty:run` to use an embedded jetty or to deploy to your container.

1. Extract the source and run the `mvn` package (maven 2 required).
2. Deploy the war file to the `j2ee` container. The redirect URL for your client application changes depending on your context root. The preregistered client has the redirect URL configured with `redirect_url=http://sample-oauth2.infor.com:8080/RedirectServlet`. This assumes the sample application runs at the root context and port 8080.
3. Add `sample-oauth2.infor.com` to the hosts file to point to the `j2ee` container IP address (windows hosts or /etc/hosts).
4. Open this URL in the browser: `http://sample-oauth2.infor.com:8080`

Java thick clients

API Gateway inbound security requires client application to use OAuth 2.0 tokens to access API Gateway resources.

Thick client applications must implement an OAuth 2.0 authorization code grant flow to obtain tokens from Infor IFS and use the tokens to consume API Gateway.

This section describes the steps required for Java-based thick-client applications to consume API Gateway resources. Additionally, this section includes a sample implementation. The process requires these steps:

- Acquire the OAuth Client
- Obtain the OAuth Token
- Use the OAuth Token to consume API Gateway

Acquire the OAuth client for Java thick clients

To obtain and use OAuth tokens to consume API Gateway CE, you must acquire an OAuth client that is specific to your application.

The OAuth client that is specific to your application is created while integrating your app with API Gateway CE. Your application should keep OAuth 2.0 client details, along with IFS CE authorization server endpoints.

Obtain the OAuth token for Java thick clients

After your application has OAuth client and IFS CE authorization server details, you can obtain the OAuth tokens.

1 Send an Authorization Code Request to IFS CE authorization server.

Initiate the process of obtaining the OAuth token by sending authorization code request to the IFS CE authorization server. This is an HTTP GET or POST request to the authorization endpoint with these parameters:

client_id
Specify the OAuth client ID specific to your application.

redirect_uri
Specify the URL where the IFS authorization server sends the code upon user consent. This must be the same URL as registered in IFS during integration.

response_type=code
Specify the IFS authorization server to send the authorization code upon user consent.

2 Resource Owner (User) Authentication and Consent (IFS functionality).

The IFS authorization server works with the IFS Federation Hub to authenticate the user/resource owner and get user consent to release the claims to your app. If the user approves sharing claims with your application, then the IFS authorization server releases the authorization code to your application.

3 Exchange the authorization code for an access token and refresh token.

By using the token endpoint of the IFS authorization server, exchange the authorization code for an OAuth access token and refresh token. Send these parameters as `Content-Type "application/x-www-form-urlencoded"`

client_id
Specify the OAuth client ID specific to your application.

client_secret
Specify the OAuth client secret received while acquiring the OAuth client details.

grant_type=authorization_code
Specify the hint authorization server about the grant type being used.

redirect_uri
Specify the URL where the authorization server sends the access token. This URL must match the URL registered in API Gateway CE/IFS CE during integration.

code
Specify the authorization code sent by the authorization server in the previous step.

4 In exchange, the authorization server provides these parameters:
- **token_type**- This is the type of token issued, for example, Bearer.
- **expires_in** - This is the validity period of the access token.
- **refresh_token** - This is the refresh token to be used to renew the expired access token.
- **access_token** - This is the token to be used for accessing protected resources.

Use the OAuth token to consume API Gateway for Java thick clients

Use the access token to consume API Gateway endpoints.

You must send the access token in the Authorization (HTTP) header.

See the API Gateway inbound security for details.

Refresh the access token for Java thick client

The refresh access token is valid for two hours by default.

If the access token is expired, a new access token can be obtained by using refresh token. These are the parameters used to renew the access token using IFS CE token endpoint.

- **grant_type=refresh_token**
- **refresh_token**
- **client id**: Use as the username for HTTP basic authentication
- **client secret**: Use as the password for HTTP Bbsic authentication

The OAuth client library automatically handles refreshing expired tokens.

Revoke the token for Java thick clients

Revoking tokens prevents orphan grants; therefore it is crucial to revoke the tokens.

When you revoke the tokens depends on how your application is handling the refresh and access token. Tokens should be revoked before they are discarded by your application or when you want the user/resource

owner to reconfirm the grant. After the refresh token is revoked, the corresponding access token and grant are revoked as well.

Use these parameters to revoke the token using the HTTP POST operation for the IFS CE token endpoint:

- **token**: Refresh token
- **token_type_hint=refresh_token**
- **client id**: Use as the username for HTTP basic authentication
- **client secret**: Use as the password for HTTP basic authentication

Example implementation for Java thick clients

You can use an OAuth client library to ease OAuth 2.0 adoption for your application.

The OAuth 2.0 client library handles OAuth-related low-level functionality and provides a simple interface to implement the steps in the previous sections.

See http://oauth.net/2/ lists of popular OAuth 2.0 client libraries for Java. A sample implementation based on the Apache Oltu OAuth 2.0 Client is provided here. This implementation is a simple thick-client application that integrates with API Gateway and IFS. These are code snippets to implement OAuth:

Request authorization code

```
OAuthClientRequest request = OAuthClientRequest
                .authorizationProvider("https://mingledev01-sso.mingledev.in
for.com:443/ACME_PRD/as/authorization.oauth2")
                .setClientId("ACME_PRD~QxG91-i82CO4P7L5R1YR4YwdOyWw5caGh0UqkvqYrUY")
               .setRedirectURI("http://sample-oauth2-client.infor.com:8080/SampleAppOAuth2/redi
rect"
                .setResponseType("code")
                .buildQueryMessage();
servletResponse.sendRedirect(request.getLocationUri());
```

Exchange code for token

```
OAuthClientRequest request = OAuthClientRequest
       .tokenLocation("https://mingledev01-sso.mingledev.infor.com:443/ACME_PRD/as/token.oauth2")

        .setGrantType(GrantType.AUTHORIZATION_CODE)
        .setClientId("ACME_PRD~QxG91-i82CO4P7L5R1YR4YwdOyWw5caGh0UqkvqYrUY")
        .setClientSecret("G1-DsyjDTlC6uzaelRKMZMDkfUU-3SUbs2zNdq-Rf9e0xE2G_mJhjqPCZXUPYHTqXQdMP
KEqCwEO94rzmYleBg")
        .setRedirectURI("http://sample-oauth2-client.infor.com:8080/SampleAppOAuth2/redirect")
        .setCode(code)
        .buildQueryMessage();
OAuthClient oAuthClient = new OAuthClient(new URLConnectionClient());
OAuthAccessTokenResponse oauthResponse = oAuthClient.accessToken(request);
String accessToken = oAuthResponse.getAccessToken();
String expiresIn = oAuthResponse.getExpiresIn();
```

Use access token

```
OAuthClientRequest bearerClientRequest = new OAuthBearerClientRequest("https://mingledev01-ion
api.mingledev.infor.com/ACME_PRD/weather/geolookup/q/FL/32266.json")'+
        .setAccessToken(accessToken)'+
        .buildQueryMessage();'+
```

```
OAuthResourceResponse resourceResponse = oAuthClient.resource(bearerClientRequest, OAuth.Http
Method.GET, OAuthResourceResponse.class);
```

Refresh token

```
String reqParam = "refresh_token="+varRefreshToken+"&grant_type=refresh_token";
OAuthClientRequest oauthrequest = OAuthClientRequest.tokenLocation(https://mingledev01-sso.min
gledev.infor.com:443/ACME_PRD/as/revoke_token.oauth2+"?"+reqParam)
    .buildBodyMessage();
oauthrequest.addHeader("Authorization", "Basic "+authStringEnc);//use client_id as username,
client_secret as password
OAuthClient oAuthClient = new OAuthClient(new URLConnectionClient());
OAuthResourceResponse resourceResponse = oAuthClient.resource(oauthrequest, OAuth.HttpMethod.POST,
 OAuthResourceResponse.class);';
```

Revoke token

```
String reqParam = "token="+varRefreshToken+"&token_type_hint=refresh_token";
OAuthClientRequest oauthrequest = OAuthClientRequest.tokenLocation(https://mingledev01-sso.min
gledev.infor.com:443/ACME_PRD/as/revoke_token.oauth2+"?"+reqParam)
    .buildBodyMessage();
oauthrequest.addHeader("Authorization", "Basic "+authStringEnc);//use client_id as username,
client_secret as password
OAuthClient oAuthClient = new OAuthClient(new URLConnectionClient());
OAuthResourceResponse resourceResponse = oAuthClient.resource(oauthrequest, OAuth.HttpMethod.POST,
 OAuthResourceResponse.class);
```

Sample application for Java-thick clients

A sample rich client java application is included in this SDK. To run the source:

1. Extract the source and run `dist/SampleThickClientOAuth2.jar`.
2. Alternatively, compile the source and run the resulting jar.

.Net web applications

API Gateway inbound security requires the client application to use OAuth 2.0 tokens to access API Gateway resources.

Web applications must implement an OAuth 2.0 authorization code grant flow to obtain tokens from IFS CE and use the tokens to consume API Gateway.

This section describes the steps required for .Net-based web applications to consume API Gateway resources. A sample implementation is provided.

- Acquire the OAuth Client
- Obtain the OAuth Token
- Use the OAuth Token to consume API Gateway

Acquire the OAuth client for .Net web applications

To obtain and use OAuth tokens to consume API Gateway CE, you must acquire an OAuth client that is specific to your application.

The OAuth client that is specific to your application is created while integrating your app with API Gateway CE. Your application should keep OAuth 2.0 client details, along with IFS CE authorization server endpoints.

Obtain the OAuth token for .Net web applications

After your application has OAuth client and IFS CE authorization server details, you can obtain the OAuth tokens.

1 Send an Authorization Code Request to the IFS authorization server.

Initiate the process of obtaining the OAuth token by sending an authorization code request to the IFS CE authorization server. This is an HTTP GET or POST request to the authorization endpoint with these parameters:

client_id
Specify the OAuth client ID specific to your application.

redirect_uri
Specify the URL where the IFS CE authorization server sends the code upon user consent. This must be the same URL as registered in IFS CE during integration.

response_type=code
Specify the IFS authorization server to send the authorization code upon user consent.

2 Resource Owner (User) Authentication and Consent (IFS CE functionality).

The IFS CE authorization server works with the IFS CE Federation Hub to authenticate the user/resource owner and get user consent to release the claims to your app. If the user approves sharing claims with your application, then the IFS CE authorization server releases the authorization code to your application.

3 Exchange the authorization code for an access token and refresh token.

By using the token endpoint of IFS CE authorization server, exchange the authorization code for an OAuth access token and refresh token. Send these parameters as `Content-Type "application/x-www-form-urlencoded"`

client_id
Specify the OAuth client ID specific to your application.

client_secret
Specify the OAuth client secret received while acquiring OAuth client details.

grant_type=authorization_code
Specify the hint authorization server about the grant type being used.

redirect_uri
Specify the URL where the authorization server sends the access token. This URL must match the URL registered in API Gateway CE/IFS CE during integration.

code
Specify the authorization code sent by the authorization server in the previous step.

4 In exchange, the authorization server provides these parameters:
- **token_type**: This is the type of token issued, for example, Bearer.
- **expires_in**: This is the validity period of the access token.
- **refresh_token**: This is the refresh token to be used to renew the expired access token.
- **access_token**: This is the token to be used for accessing protected resources.

Use the OAuth token to consume API Gateway for .NET web applications

Use the access token to consume API Gateway endpoints.

You must send the access token in the authorization (HTTP) header.

See the API Gateway inbound security for details.

Refresh the access token for .Net web applications

The refresh access token is valid for two hours by default.

If the access token is expired, a new access token can be obtained by using refresh token. These are the parameters used to renew the access token using IFS CE token endpoint.

- **grant_type=refresh_token**
- **refresh_token**
- **client id**: Use as the username for HTTP basic authentication
- **client secret**: Use as the password for HTTP basic authentication

The OAuth client library automatically handles refreshing expired tokens.

Revoke the token for .Net web applications

Revoking tokens prevents orphan grants; therefore it is crucial to revoke the tokens.

When you revoke the tokens depends on how your application is handling the refresh and access token. Tokens should be revoked before they are discarded by your application or when you want the user/resource owner to reconfirm the grant. After the refresh token is revoked, the corresponding access token and grant are revoked as well.

Use these parameters to revoke the token using the HTTP POST operation for the IFS CE token endpoint:

- **token**: Refresh token
- **token_type_hint=refresh_token**
- **client id**: Use as the username for HTTP basic authentication
- **client secret**: Use as the password for HTTP basic authentication

Example implementation for .Net web applications

You can use an OAuth client library to ease OAuth 2.0 adoption for your application.

The OAuth 2.0 client library handles OAuth-related low-level functionality and provides a simple interface to implement the steps in the previous sections.

See http://oauth.net/2/ lists of popular OAuth 2.0 client libraries for .Net. A sample implementation based on the ThinkTecture IdentityServer3 Sample is provided here. This implementation is a simple thick-client application that integrates with API Gateway CE and IFS CE. These are code snippets to implement OAuth:

Request authorization code

```
var state = Guid.NewGuid().ToString("N");
var nonce = Guid.NewGuid().ToString("N");
SetTempState(state, nonce);
var client = new OAuth2Client(new Uri(Constants.AuthorizeEndpoint));
var url = client.CreateCodeFlowUrl(
clientId: Constants.ClientId,
scope: scopes,
redirectUri: Constants.RedirectUrl,
state: state,
nonce: nonce);
return Redirect(url);
```

Exchange code for token

```
var client = new OAuth2Client(
new Uri(Constants.TokenEndpoint),
Constants.ClientId,
Constants.ClientSecret);
var code = Request.QueryString["code"];
var tempState = await GetTempStateAsync();
Request.GetOwinContext().Authentication.SignOut("TempState");
var response = await client.RequestAuthorizationCodeAsync(
code, Constants.RedirectUrl);
await ValidateResponseAndSignInAsync(response, tempState.Item2);
return View("Token", response);
```

Use access token

```
var principal = User as ClaimsPrincipal;
var client = new HttpClient();
client.SetBearerToken(principal.FindFirst("access_token").Value);
var result = await client.GetStringAsync(Constants.AspNetWebApiSampleApi + Constants.AspNetWe
bApiSampleApiEndpoint);
//"ACME_PRD/M3/m3api-rest/execute/CRS610MI/ChgFinancial?CUNO=Y30000&BLCD=0");
return View((object)result);
```

Refresh token

```
var client = new OAuth2Client(
new Uri(Constants.TokenEndpoint),
Constants.ClientId,
Constants.ClientSecret);
var principal = User as ClaimsPrincipal;
var refreshToken = principal.FindFirst("refresh_token").Value;
• Revoke Token
var refreshToken = (User as ClaimsPrincipal).FindFirst("refresh_token").Value;
var client = new HttpClient();
```

```
client.SetBasicAuthentication(Constants.ClientId, Constants.ClientSecret);
var postBody = new Dictionary<string, string>
{
{ "token", refreshToken },
{ "token_type_hint", "refresh_token" }
};
var result = await client.PostAsync(Constants.TokenRevocationEndpoint, new FormUrlEncodedCon
tent(Source
```

Sample application for .Net web applications

A sample rich client java application is included in this SDK. To run the source:

1. Build and deploy the solution. Use port 443 and context root `/SampleAppOAuth2`.
2. Add `sample-oauth2-client.infor.com` to the host files to point to the IIS host ip address (windows hosts or /etc/hosts).
3. Open this URL in a browser: `https://sample-oauth2-client.infor.com/SampleAppOAuth2`

.Net based thick clients

The suggested grant for thick clients is the authorization code.

There are multiple OAuth2.0 libraries available, for example: http://www.nuget.org/packages/Thinktecture.IdentityModel.Client/. This URL provides a library with utility functions to implement the OAuth2.0 protocol. The client application can leverage the library to construct the correct URL query parameters and the form post required as part of the interaction with the authorization service.

To facilitate the adoption of the `Thinktecture.IdentityModel.Client` application library, a sample application has been created to showcase the different interactions with the authorization service in the OAuth2.0 protocol. The sample application is based on the samples from the Thinktecture team located at: https://github.com/IdentityServer/IdentityServer3.Samples/tree/master/source/Clients

The application provides the functionality to obtain an authorization code **Get Code** button. Then the code can be exchanged by an access_token with the **Get Access Token with code** button. After a token is obtained and the client is configured to receive a refresh_token, you can obtain a new access_token with the **Refresh Access Token** button. You can call the API Gateway with the **Call Service** button.

When the application does not need the access_token or the refresh_token, they can be revoked by using either **Revoke Access Token** or **Revoke Refresh Token**. The sample application showcases the interaction of the client with the authorization service. This sample app does not treat the access_token or refresh_token securely. Maintaining the access_token and refresh_token secure is the responsibility of the final application and should be secured as any other existing secret.

Request the authorization code

This code creates a new instance of the OAuth2Client passing the authorization end point.

Additionally, it leverages the OAuth2Client to construct the correct URL to obtain the authorization code. The constructed URL must be presented to a user through a user-agent, for example, an embedded browser in the client.

The user must authenticate and authorize by providing the tokens to the thick client. **LoginWebView** is a window that has a WebBrowser. This navigates the user to the specified URI, checks the different URLs that the user is redirected to, and detects when the user navigates to the RedirectUri.

```
var state = Guid.NewGuid().ToString("N");
var nonce = Guid.NewGuid().ToString("N");
var client = new OAuth2Client(new Uri(OAuth2AuthorizationEndpoint));
var startUrl = client.CreateCodeFlowUrl(
clientId: ClientId,
redirectUri: RedirectUri,
state: state,
nonce: nonce);
LoginWebView webView = new LoginWebView();
webView.Owner = this;
webView.Done += _login_Done;
webView.Show();
webView.Start(new Uri(startUrl), new Uri(RedirectUri));
```

Obtain the authorization code

After the user finishes the interaction with the authorization server, the user is navigated to the RedirectUri.

The user can inspect every redirect request handling the navigating event on the WebBrowser, for example:

```
private void webView_Navigating(object sender, NavigatingCancelEventArgs e)
{
if (e.Uri.ToString().StartsWith(_callbackUri.AbsoluteUri))
{
AuthorizeResponse = new AuthorizeResponse(e.Uri.AbsoluteUri);
e.Cancel = true;
this.Visibility = System.Windows.Visibility.Hidden;
if (Done != null)
{
Done.Invoke(this, AuthorizeResponse);
}
}
if (e.Uri.ToString().Equals("javascript:void(0)"))
{
e.Cancel = true;
}
}
```

Note: There is a special case for `javascript:void(0)`. While testing, it was detected that such navigation makes the browser unresponsive. Be aware that this functionality works when using WPF WebView as in the sample provided. Using the Winforms browser may not provide all the events as expected. In this scenario, the Thinktecture library was leveraged to parse the response URL: `AuthorizeResponse = new AuthorizeResponse (e.Uri.AbsoluteUri)`; the AuthorizeResponse includes an error if there is a problem while obtaining the authorization code.

Obtain the access_token and refresh_token

After you obtain the authorization code, you can obtain an access token

```
var client = new OAuth2Client(new Uri(OAuth2TokenEndpoint), ClientId, ClientSecret);
var response = client.RequestAuthorizationCodeAsync(this.CodeTextBox.Text, RedirectUri).Result;
```

In the sample application, this is done in two steps to showcase the difference. Applications can obtain an access token directly out of the authorization code without user interaction. The response is TokenResponse. Additionally, it indicates if there is an error. If there is no error, it includes the access_token and refresh token.

Calling the service

You can call the web service with the access token by passing the access token as a bearer token.

```
var client = new HttpClient
{
BaseAddress = new Uri(IONAPIBaseUrl)
};
client.SetBearerToken(this.AccessTokenTextBox.Text);
var response = client.GetAsync(this.WebServiceEndpoint.Text).Result;
```

Revoke the access token

When the token is not required anymore, the access_token should be revoked.

Currently, the Thinktecture library does not provide a method to revoke the token.

This is an example:

```
private void _revokeToken(string token, string tokenType)
{
var client = new HttpClient();
client.SetBasicAuthentication(ClientId, ClientSecret);
var postBody = new Dictionary<string, string>
{
{ "token", token },
{ "token_type_hint", tokenType }
};
var result = client.PostAsync(OAuth2TokenRevocationEndpoint, new FormUrlEncodedContent(postBody)).}
```

To revoke the access token, use this call:

```
_revokeToken(this.AccessTokenTextBox.Text, OAuth2Constants.AccessToken);
```

Refresh the token for .Net-based thick clients

You can obtain a refresh token as part of the access token.

The refresh token is a token with a longer expiration time that allows clients to obtain a new set of access_token and refresh_token without requiring the user to authenticate again.

Use this code to refresh the access token:

```
var client = new OAuth2Client(new Uri(OAuth2TokenEndpoint), ClientId, ClientSecret);
TokenResponse response = client.RequestRefreshTokenAsync(this.RefreshTokenTextBox.Text).Result;
```

Revoke the refresh token for .Net-based thick clients

If a refresh token is provided when the authorization from the user is no longer required, then the refresh token should be revoked.

Use this call to revoke the refresh token:

```
_revokeToken(this.RefreshTokenTextBox.Text, OAuth2Constants.RefreshToken);
```

Backend applications (Java or .Net)

Backend applications are applications that do not have a user available to authenticate.

For these applications, the recommended grant to use is the resource owner.

Because of the disparity for the location of the Infor Ming.le identities, a new set of credentials is used for the resource owner grant. Through the resource owner grant, only service accounts can be authenticated. A service account can be associated with a user by making the call to the backend application on behalf of the user. The administrator of the API Gateway creates the service account in IFS and registers your application.

Register your backend application to obtain an OAuth ClientID and secret

You must register the service account that acts as the user for your backend application.

When the applications is registered, an OAuth ClientID and Client-Secret are generated for the application.

You must have this information to contact the Infor authorization server to obtain a OAuth 2.0 bearer token that your backend application can use to make API requests via API Gateway:

- Application ClientID
- Application Client-Secret
- Service Account AccessKey
- Service Account SecretKey

Example HTTP request for the OAuth2 resource owner grant

The OAuth2 resource owner grant facilitates obtaining an access token for backend services using a back-channel HTTP POST request to the authorization server token endpoint, for example `as/token` or `connect/token`.

Use these parameters:

- **grant_type** = password (fixed)
- **username** = service account accesskey
- **password** = service account secretkey
- **client_id**= authorized app
- **client id client_secret** = authorized app
- **client secret scope** = oauth2 scope (Optional)

.Net applications

There are multiple OAuth2.0 libraries available, including http://www.nuget.org/packages/Thinktecture.IdentityModel.Client/.

It is a library with utility functions to implement the OAuth2.0 protocol. The client application can leverage the library to construct the correct URL query parameters and the form post required as part of the interaction with the authorization service.

Sample application

A .NET sample application is provided in this SDK and leverages the Thinktecture library to obtain, refresh, and revoke tokens and call a webservice client with the token.

The sample client showcases the functionality available by the library. The sample application is based on samples from the Thinktecture team located at: https://github.com/IdentityServer/IdentityServer3.Samples/tree/master/source/Clients

The sample application showcases the interaction of the client with the authorization service. This sample application does not treat the access_token or refresh_token securely. Maintaining the access_token and refresh_token secure is the responsibility of the final application and should be secured as any other existing secret.

Create client

With the provided token_endpoint, ClientId, and ClientSecret, you can construct a client to use in further interactions. You can make a request authentication using the ClientId and ClientSecret.

```
_oauth2 = new OAuth2Client(new Uri(OAuth2TokenEndpoint), ResourceOwnerClientId, ResourceOwner
ClientSecret);
```

Obtain access_token

With the service account accessKey and secretKey, you can request an access token. The response type is TokenResponse. This indicates if there is an error obtaining the token. If successful, then it includes the access_token and, if available for the client, the refresh_token.

```
_oauth2.RequestResourceOwnerPasswordAsync(ServiceAccountAccessKey, ServiceAccountSecretKey).Result;
```

Calling service

With the token from the TokenResponse, you can call the service passing the access token as a bearer token.

```
var client = new HttpClient
            {
                BaseAddress = new Uri(IONAPIBaseUrl)
            };
            client.SetBearerToken(token);
            var response = client.GetAsync("M3/m3api-rest/execute/CRS610MI/ChgFinan
cial?CUNO=Y30000&BLCD=0").Result;
```

Revoke access token

When the token is not needed anymore, we recommend that you revoke the access_token. Currently, the Thinktecture library does not provide a method to revoke the token. You may use this method:

```
private static void RevokeToken(string token, string tokenType)
{
    var client = new HttpClient();
    client.SetBasicAuthentication(ResourceOwnerClientId, ResourceOwnerClientSecret);
    var postBody = new Dictionary<string, string>
            {
                { "token", token },
                { "token_type_hint", tokenType }
            };
    var result = client.PostAsync(OAuth2TokenRevocationEndpoint, new FormUrlEncodedContent(post
Body)).Result;

}
```

To revoke an access_token it should be called with these parameters:

```
RevokeToken(token.AccessToken, OAuth2Constants.AccessToken);
```

Refresh token

If a refresh token is available as part of the response, you can obtain a new access_token and refresh_token without requiring the service account credentials.

```
_oauth2.RequestRefreshTokenAsync(refreshToken).Result;
```

Revoke refresh token

If a refresh token is provided and there is no longer the need to make calls to the webservice without providing the service account credentials, then the refresh token should be revoked. Use the same method as the one provided to revoke access tokens to revoke refresh tokens.

```
RevokeToken(token.RefreshToken, OAuth2Constants.RefreshToken);
```

Go applications

Go programming language (Golang) provides a package `golang.org/x/oauth2` to implement the OAuth2.0 protocol.

Make OAuth 2.0 configuration

The first step is to define a configuration. A reference to downloaded credentials properties is used in all code examples.

```
conf := &oauth2.Config{
       ClientID:     <Application ClientID>,
       ClientSecret: <Application Client-Secret>,
       Scopes: []string{
              "openid profile",
       },
       Endpoint: oauth2.Endpoint{
```

```
                AuthURL:  <pu> + <oa>,
                TokenURL: <pu> + <ot>,
        },
}
```

Obtain tokens

Now you are ready to obtain tokens. Token struct in Go contains both access and refresh tokens.

```
tok, err := conf.PasswordCredentialsToken(oauth2.NoContext, <Service Account AccessKey>, <Service
 Account SecretKey>)
if err != nil {
        // handle error
}
```

Create HTTP client and make a request

The OAuth 2.0 configuration struct also has a method to create the HTTP client.

```
client := conf.Client(oauth2.NoContext, tok)

resp, err := client.Get(<Request URL>)
if err != nil {
        // handle error
}
```

Note: You do not need to refresh the token manually; the client does it automatically.

Revoke tokens

The package does not provide methods to revoke any token. You can call the revoke service directly.

```
resp, err := http.Get(<pu> + <or> + "?token=" + tok.AccessToken)
if err != nil {
        // handle error
}
```

OAuth 2.0 Token Management

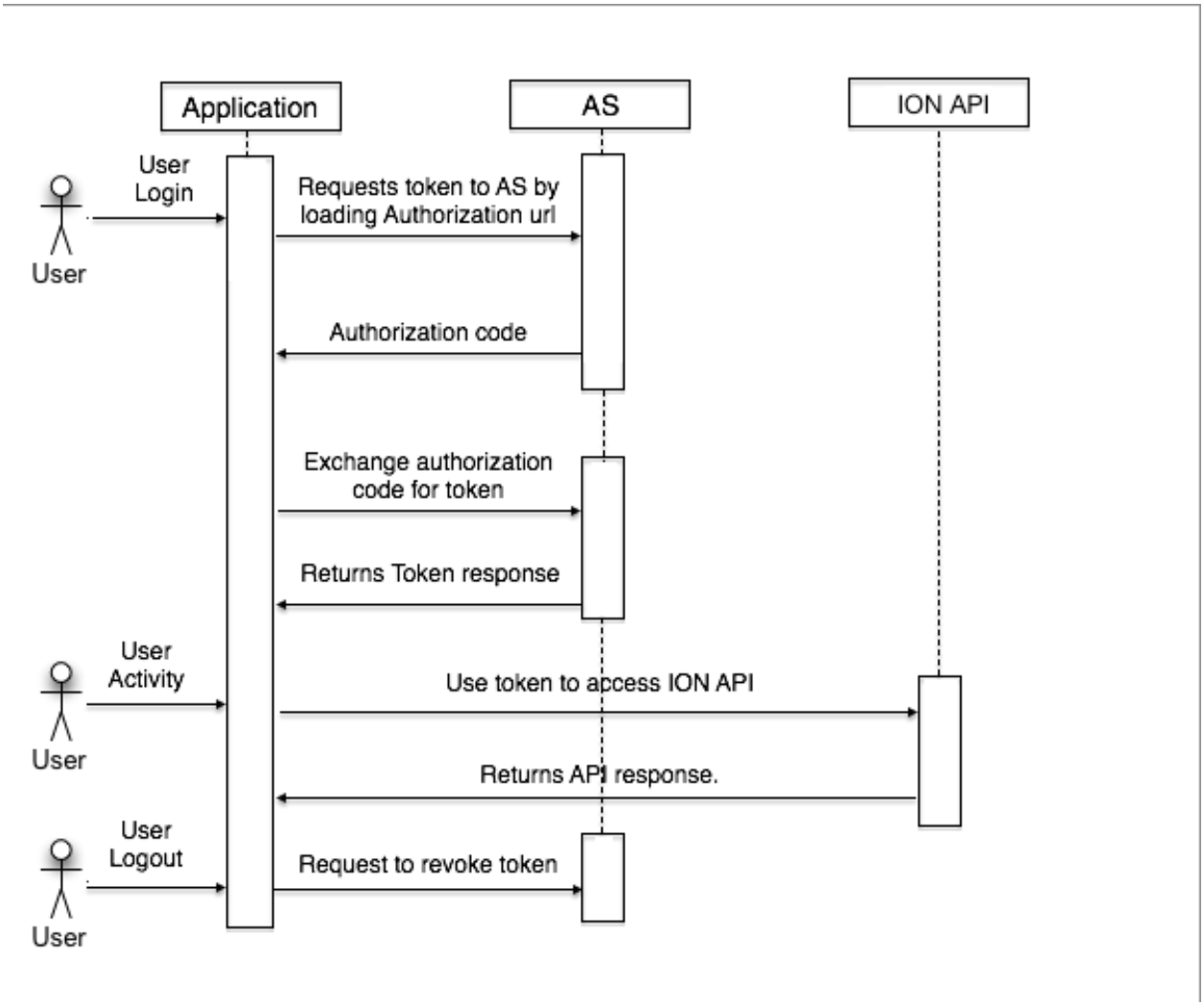
Beyond implementing the screens and business logic of your application, you must include code to interact with the Infor Authorization Server (AS) to obtain a valid token that allows you make calls to your selected APIs via API Gateway.

The diagram below shows the sequence of calls that happens back and forth between the authorization server, the mobile application, and API Gateway.

The authorization sequence begins when the application launches the sign-in process. The application loads an authorization page in the browser or within the application (based on your preference); the URL includes query parameters that indicate the type of access being requested. The result is an authorization code, which the application can exchange for an access token and a refresh token.

By default, access tokens have limited lifetimes (currently about two hours). If your application needs access to an API Gateway beyond the lifetime of a single access token, it can obtain a refresh token. A refresh token allows your application to obtain new access tokens. The application should store the refresh token for future

use and use the access token to access an API Gateway. Once the access token expires, the application uses the refresh token to obtain a new one.



Application development - handling API errors

In general, an HTTP 200 indicates success, and any other status such as 4xx or 5xx indicates a problem.

Some APIs return a response payload, for example, a bit of JSON and contains its own status and/or error information. In this case, a 200 may indicate only that a response was delivered correctly and you still need to examine the status/errors properties of the response to decide if the call was truly successful or not. This is where having read and understood your APIs documentation is critically important.

Common error statuses

401 Unauthorized

OAuth 2.0 inbound bearer token is missing, invalid, or expired, or it could be that the target API server rejected the backend security (not as likely).

404 Not Found

The resource/path request was not found. This should normally not happen and indicates perhaps a mistake in the URL you are calling.

405 Method Not Allowed

You are calling an API by using a method that is not supported. For example, you are trying to POST to an API that allows only a GET.

429 Too Many Requests

You are calling an API with a Quota policy applied to it, and you have exceeded the allowed number of requests in a given time period.

504 Gateway Timeout

The target API server did not respond with the gateway's configured timeout period of 60 seconds.

Releasing the application

After thoroughly testing the application, you can release it following the guidelines for its specific platform (mobile) or go live (web).

Time-outs

There are multiple time-outs in the communication process between an authorized app and the target product API:

	Stage	Time-out
1	ION API Authorized App/Client (t1)	HTTP Timeout – Controlled by the Authorized App. 5+ minutes recommended.
2	ION API ALB (t2)	Idle Timeout – 5 minutes.

	Stage	Time-out
3	ION API Gateway (t3)	Target Timeout – 1 minute. Can be extended up to 5 minutes by using a policy.
4	Product ALB/ELB (t4)	Idle Timeout – Controlled by the product. 5 minutes, 30 seconds or more recommended.
5	Product API	Processing Time – Controlled by the product. 1 minute or less recommended.

Chapter 3: Infor API flow

The Infor API Flows suite is automatically created when a tenant-administrator creates an API flow in ION. This suite shows all the active API flows as endpoints.

These API flows can be tried out by clicking the documentation icon. A swagger is shown with query parameters and body generated from the configuration from ION.

By default, API flows are synchronous.

API flows can be invoked in an asynchronous mode by setting an additional parameter on the API call: `ion apiRespStyle=[sync|async]`

- When an API flow is invoked in an asynchronous mode, API Gateway returns a session ID immediately and executes the actual flow in the background: `{ "started": 1634291722075, "status": "RUNNING", "flowId": "008bba7d-c8cc-479c-9eb2-0b631cecbb9f“ }`
- The client must return to retrieve the final response based on the session ID within 10 minutes: `Get end point/ionapi/apiflow/v1/TENANT/result/ { Unknown macro: \{apiflowid}}`
- If the data is not retrieved within 10 minutes, the API Gateway discards the results.

Chapter 4: Administration

This section is for server administrators who need to diagnose and troubleshoot problems with API Gateway.

Limitations

Payload limits

IONAPI Gateway has no payload limit; however, for buffered policies the payload limit is 10 MB.

Timeout limits

For IONAPI Gateway, the default timeout is 1 minute, but it can be extended up to 5 minutes.

For IONAPI Gateway with buffered policies the timeout is maximum is 5 minutes.

Available APIs

The **Available APIs** page shows all available API suites that the tenant is authorized to use.

On this page, you can:

- Search all available API suites with the use of the search bar
- Filter the API suites by type:
  - All suites
  - Infor Provisioned
  - Infor Non-Provisioned
  - Non-Infor
- Delete an API suite
- Add an API suite
- Export Non-Infor suites, up to 10 suites at a time
- Import Non-Infor suites, up to 10 suites at a time

Adding a new API suite

To add a new API suite:

1. Select **Available APIs** in API Gateway.
2. Click **Add New API Suite**.
3. Select the appropriate option:
   - For an Infor Non-Provisioned suite, select an **API Suite template**.
   - For a Custom or Non-Infor suite, click **Create New**.

Caution: API Gateway acts as a proxy for target APIs and can help mitigate some vulnerabilities in target APIs with policies (for example: threat protection, quota, rate limiting, transformation) and high standards for communication between clients and API Gateway (for example: strong TLS, inbound security, access control). Infor APIs follow secure development practices to address such concerns. Before adding non-Infor third-party APIs, review the target API vulnerabilities and make sure proper mitigation and monitoring is in place. The customer is responsible for either mitigating or accepting risks with customer-provisioned APIs.

Infor Non-Provisioned

1. Select a **Suite Name**
2. Specify the **API Context**. The **API Context** must be unique, and changing the **API Context** will break any Infor mobile applications or Homepages widgets that are expecting to use this API suite.
3. Select a **Suite Icon**. For Infor Non-Provisioned applications, the icon is preselected. Click **Choose Icon** to change the color of the icon.
4. To add deployment information, click **Add Deployment** and enter a **Deployment Name**. For more information, see Adding a deployment on page 50.
5. If the application does not use HTTPS, click the **Use HTTPS** slider to disable the use of HTTPS.
6. If the application ignores certificate errors, click the **Ignore Certificate Errors** slider to ignore the certificate errors.
7. Specify **Host Name**.
8. Specify **Port**.
9. Specify **Context**.
10. Specify **Default Tenant ID**.
11. If the application will use mutual SSL, click the **Use Mutual SSL** slider to enable mutual SSL. Enabling this setting adds the **Key Password** field and the **Load Certificate** button.
12. Specify an **Authentication Type**. The details displayed depend on your selection:

    OAuth 1.0a
    Specify the **Algorithm**, **Target Endpoint Access Key**, and the **Target Endpoint Secret Key**.

    Anonymous
    No additional fields are displayed.

    Basic
    Specify the **User ID** and **Password**.

OAuth 2.0

a Specify the **Token Endpoint** and the **Revoke Endpoint**.

b Specify the **Grant Type**. The details displayed depend on your selection:

- If you select **Resource Owner**, specify the **Username**, **Password**, **Client ID**, and **Client Secret**.
- If you select **Client Credentials**, specify the **Client ID** and the **Client Secret**.

Sales Force

a Specify the **Token Endpoint**. This is prepopulated with `https://login.salesforce.com/services/oauth2/token` but can be edited.

b Specify the **Revoke Endpoint**. This is prepopulated with `https://login.salesforce.com/services/oauth2/revoke` but can be edited. **`Resource Owner`** is the only selection available for **Grant Type**.

c Specify the **Username**, **Password**, **Client ID**, and **Client Secret**.

WS-Security Username Token

Specify the **User Name** and **Password**.

13 Click **Save**.

Endpoints and deployments can be configured if needed.

Custom or Non-Infor

1 Specify **Application Name**.

2 Specify **Suite Name**.

3 Specify **API Context**.

4 Specify **Description**.

5 To select a suite icon, click **Choose Icon**.

a Select an icon.

b Select an icon color.

c Click **Save**.

6 To add a target endpoint, click **Add Endpoint**. For more information, see Adding an endpoint on page 47.

7 Specify the **Target Endpoint URL**.

8 If the application ignores certificate errors, click the **Ignore Certificate Errors** slider to ignore the certificate errors.

9 Specify the **Target Endpoint Description**.

10 Specify the **Proxy Endpoint URL**.

The Public Facing Proxy Endpoint is prepopulated and cannot be edited.

11 Select the **Proxy Security** from the list. The options are:

- OAuth 2.0
- Anonymous

12 If the application will use mutual SSL, click the **Use Mutual SSL** slider to enable mutual SSL. Enabling this setting adds the **Key Password** field and the **Load Certificate** button.

13 Specify an **Authentication Type**. The details displayed depend on your selection:

OAuth 1.0a
Specify the **Algorithm**, **Target Endpoint Access Key**, and the **Target Endpoint Secret Key**.

Anonymous
No additional fields are displayed.

Basic
Specify the **User ID** and **Password**.

OAuth 2.0
- **a** Specify the **Token Endpoint** and the **Revoke Endpoint**.
- **b** Specify the **Grant Type**. The details displayed depend on your selection:
  - If you select **Resource Owner**, specify the **Username**, **Password**, **Client ID**, and **Client Secret**.
  - If you select **Client Credentials**, specify the **Client ID** and the **Client Secret**.

Sales Force
- **a** Specify the **Token Endpoint**. This is prepopulated with `https://login.salesforce.com/services/oauth2/token` but can be edited.
- **b** Specify the **Revoke Endpoint**. This is prepopulated with `https://login.salesforce.com/services/oauth2/revoke` but can be edited.
- **c** `Resource Owner` is the only selection available for **Grant Type**.
- **d** Specify the **Username**, **Password**, **Client ID**, and **Client Secret**.

WS-Security Username Token
Specify the **User Name** and **Password**.

14 Click **Save**.
Endpoints can be configured if needed.

Editing the API suite name and description

This is available only for Non-Infor API suites.

1 Go to **Available APIs** in API Gateway.
2 Click a non-Infor API suite.
3 Make your changes.
4 Click **Save**.

Adding policies

Suite policies are applied to every endpoint within the suite and are applied before any endpoint-specific policies.

If the same policy is set at the suite and endpoint levels, the endpoint-specific policies overwrite the suite policy.

See Policies on page 89 for more information on the policies available.

This is available only for Non-Infor API suites.

Adding suite policies

1. Go to **Available APIs** in API Gateway.
2. Click a non-Infor API suite.
3. Click the **Suite Policies** tab.
4. Configure the request policies or response policies as needed.

Adding endpoint policies

1. Go to **Available APIs** in API Gateway.
2. Click a non-Infor API suite.
3. Select **Details**.
4. Select the **Endpoint Policies** tab.
5. Configure the request policies or response policies as needed.

Editing and deleting policies

1. Go to **Available APIs** in API Gateway.
2. Click a non-Infor API suite.
3. For suite policies, select the **Suite Policies** tab, or for endpoint policies, select **Details** and then select the **Endpoint Policies** tab.
4. Select the request or response policy.
5. Click **Edit** or **Delete** to edit or delete the policy, or use the up and down arrows to change the order of the policy execution.

Deleting an API suite

1. Go to **Available APIs** in API Gateway.
2. Hover over the bottom of an API suite.
3. Click **Delete API Suite**.
4. Click **Yes**.

API endpoints

This section contains information on API endpoints.

Adding an endpoint

This is available only for non-Infor API suites.

1. Go to **Available APIs** in API Gateway.
2. Click an API suite.
3. Click **Endpoints**.
4. Click **Add Endpoint**.
5. Specify the Target Endpoint URL.
6. If the application ignores certificate errors, click the **Ignore Certificate Errors** slider to ignore the certificate errors.
7. Specify the **Target Endpoint Description**.
8. If the application will use mutual SSL, click the **Use Mutual SSL** slider to enable mutual SSL. Enabling this setting adds the **Key Password** field and the **Load Certificate** button.
9. Specify an **Authentication Type**. The details displayed depend on your selection:

   OAuth 1.0a
   Specify the **Algorithm**, **Target Endpoint Access Key**, and the **Target Endpoint Secret Key**.

   Anonymous
   No additional fields are displayed.

   API Key
   Provide the applicable **Key Name** and **Key Value**.

   AWS Signature
   a. Specify the **AWS Access Key ID**.
   b. Specify the **AWS Secret Access Key**.
   c. Specify the **AWS Region**.
   d. Specify the **AWS Service**.

   IONAPI Bridge
   For information on how to set up ION API bridge, see API Gateway bridge solution on page 72.

   JWT Target Authentication
   a. Specify the JWT Header.
   b. Specify the JWT Payload.
   c. Optionally, specify the token expiration in seconds.
   d. Use/Generate the new Key ID.
   e. Specify the algorithm from RS256, RS384, RS512.
   f. Optionally, specify any custom parameters.

Basic
Specify the **User ID** and **Password**.

OAuth 2.0
a Specify the **Token Endpoint** and the **Revoke Endpoint**.
b Specify the **Grant Type**. The details displayed depend on your selection:
- If you select **Resource Owner**, specify the **Username**, **Password**, **Client ID**, and **Client Secret**.
- If you select **Client Credentials**, specify the **Client ID** and the **Client Secret**.

Sales Force
a Specify the **Token Endpoint**. This is prepopulated with `https://login.salesforce.com/services/oauth2/token` but can be edited.
b Specify the **Revoke Endpoint**. This is prepopulated with `https://login.salesforce.com/services/oauth2/revoke` but can be edited.
c `Resource Owner` is the only selection available for **Grant Type**.
d Specify the **Username**, **Password**, **Client ID**, and **Client Secret**.

WS-Security Username Token
Specify the **User Name** and **Password**.

10 Specify the **Proxy Endpoint URL**.
11 Specify the **Proxy Security**.
12 Click **Save**.

Editing endpoint details

This is available only for non-Infor API suites.

1 Go to **Available APIs** in API Gateway.
2 Click an API suite.
3 Click **Endpoints**.
4 Click **Details**.
5 Make your changes.
6 Click **Save**.

Deleting an endpoint

This is available only for non-Infor API suites.

1 Go to **Available APIs** in API Gateway.
2 Click an API suite.
3 Click **Endpoints**.
4 Select the endpoint to delete.

5 Click **Delete**.
6 Click **Yes**.

Viewing API endpoint resources

1 Go to **Available APIs** in API Gateway.
2 Click an API suite.
3 Click **Endpoints**.
4 Click **Details**.
5 To see endpoint resources, click **Resources**.

Viewing API endpoint documentation

1 Go to **Available APIs** in API Gateway.
2 Click an API suite.
3 Click **Endpoints**.
4 To see endpoint documentation, click **Documentation**.

If you prefer to view documentation programmatically, you can use the these URLs:
- REST APIs – {ION API BASE URL}/{API CONTEXT}/{ENDPOINT}/ionapi-doc
- SOAP APIs - {ION API BASE URL}/{API CONTEXT}/{ENDPOINT}?WSDL

These documentation URLs are secured in the same method as the proxy endpoint.

Adding API endpoint documentation

This is available only for non-Infor API suites.

1 Go to **Available APIs** in API Gateway.
2 Click an API suite.
3 Click **Endpoints**.
4 Click **Documentation**.
5 Click **Add Documentation**.
6 Specify the **Name**.
7 Specify the **URL**.
8 Specify the **Document Type**. PDF, WSDL, and Swagger are supported. Swagger is recommended. The Swagger document size limit is 4 MB.
9 Click **Save**.

Note: For OpenAPI 3.x Swagger, API Gateway requires the Swagger to have a servers array containing an object with a URL property. If missing, the Gateway cannot replace the URL with the proper Gateway-facing URL, and the Swagger try-it-out feature does not work.

API deployments

Deployment information is available only for Infor non-provisioned API suites.

Adding a deployment

1. Go to **Available APIs** in API Gateway.
2. Click an API suite.
3. Click **Add Deployment**.
4. Select and specify **Deployment Name**.
5. If the application does not use HTTPS, click the **Use HTTPS** slider to disable the use of HTTPS.
6. If the application ignores certificate errors, click the **Ignore Certificate Errors** slider to ignore the certificate errors.
7. Specify the **Host Name**.
8. Specify the **Port**.
9. Specify the **Context**.
10. Specify the **Default Tenant ID**.
11. If the application will use mutual SSL, click the **Use Mutual SSL** slider to enable mutual SSL. Enabling this setting adds the **Key Password** field and the **Load Certificate** button.
12. Specify an **Authentication Type**. The details displayed depend on your selection:

    OAuth 1.0a
    Specify the **Algorithm**, **Target Endpoint Access Key**, and the **Target Endpoint Secret Key**.

    Anonymous
    No additional fields are displayed.

    Basic
    Specify the **User ID** and **Password**.

    OAuth 2.0
    a. Specify the **Token Endpoint** and the **Revoke Endpoint**.
    b. Specify the **Grant Type**. The details displayed depend on your selection:
    - If you select **Resource Owner**, specify the **Username**, **Password**, **Client ID**, and **Client Secret**.
    - If you select **Client Credentials**, specify the **Client ID** and the **Client Secret**.

Sales Force

a Specify the **Token Endpoint**. This is prepopulated with `https://login.salesforce.com/services/oauth2/token` but can be edited.

b Specify the **Revoke Endpoint**. This is prepopulated with `https://login.salesforce.com/services/oauth2/revoke` but can be edited.

c **`Resource Owner`** is the only selection available for **Grant Type**.

d Specify the **Username**, **Password**, **Client ID**, and **Client Secret**.

WS-Security Username Token

Specify the **User Name** and **Password**.

13 Click **Save**.

Editing a deployment

1 Go to **Available APIs** in API Gateway.

2 Click an API suite.

3 Click **Deployment Information**.

4 Click **Edit**.

5 Make your changes.

6 Click **Save**.

Deleting a deployment

1 Go to **Available APIs** in API Gateway.

2 Click an API suite.

3 Click **Deployment Information**.

4 Click **Delete**.

5 Click **Yes**.

Viewing deployed endpoints

1 Go to **Available APIs** in API Gateway.

2 Click an API suite.

3 Click **Deployment Information**.

4 Click the number in the Deployed Endpoints column.

Associating endpoints

1 Go to **Available APIs** in API Gateway.
2 Click an API suite.
3 Click **Deployment Information**.
4 Click **Associate Endpoints**
5 Select the endpoints to be associated with this deployment.
6 Click **Save**.

Authorized applications

This page shows all of the authorized applications.

Each application listed displays the application name, type, source, and status. You can delete the app, download credentials, or view the QR code (for mobile apps). You can use the search bar to search through all the authorized apps, and you can add new applications.

Adding a non-Infor authorized application

1 Go to **Authorized Apps** in API Gateway.
2 Click **Add New Application**.
3 Specify **Name**.
4 Select the **Type** of application. The details displayed depend on your selection:
   - Mobile: Android
     a Specify the **Description**.
     b Specify the **Redirect URL**.
     c Specify the **Download URL**.
     d Specify the **Package Name**.
     e Select a **Signing Certificate Fingerprint (SHA1)**: Click **Load Certificate**, select the certificate file, and click **Open**.
     f Select the length of time for **OAuth 2.0 Access Token**.
     g If the application does not request refresh tokens, click the slider to disable **Issue Refresh Tokens**.
       - If **Issue Refresh Tokens** is disabled, **Refresh Token Grant Lifetime** is also disabled. •
       - If **Issue Refresh** is enabled, specify the desired length of time and specify whether the value is in hours or days.

     Note: The **Refresh Token Grant Lifetime** value must be greater than or equal to the **OAuth 2.0 Access Token** value.
   - Mobile: IOS
     a Specify the **Description**.

b Specify the **Redirect URL**.
c Specify the **Bundle ID**.
d Specify the **App Store ID**.
e Select the length of time for **OAuth 2.0 Access Token**.
f If the application does not request refresh tokens, click the slider to disable **Issue Refresh Tokens**.
- If **Issue Refresh Tokens** is disabled, **Refresh Token Grant Lifetime** is also disabled.
- If **Issue Refresh Tokens** is enabled, specify the desired length of time and specify whether the value is in hours or days.

Note: The **Refresh Token Grant Lifetime** value must be greater than or equal to the **OAuth 2.0 Access Token** value.

- Mobile: Windows
  a Specify the **Description**.
  b Specify the **Redirect URL**.
  c Select the length of time for **OAuth 2.0 Access Token**.
  d If the application does not request refresh tokens, click the slider to disable **Issue Refresh Tokens**.
  - If **Issue Refresh Tokens** is disabled, **Refresh Token Grant Lifetime** is also disabled.
  - If **Issue Refresh Tokens** is enabled, specify the desired length of time and specify whether the value is in hours or days.

  Note: The **Refresh Token Grant Lifetime** value must be greater than or equal to the **OAuth 2.0 Access Token** value.
- Mobile: Others
  a Specify the **Description**.
  b Specify the **Redirect URL**.
  c Specify the **Download URL**.
  d Select the length of time for **OAuth 2.0 Access Token**.
  e If the application does not request refresh tokens, click the slider to disable **Issue Refresh Tokens**.
  - If **Issue Refresh Tokens** is disabled, **Refresh Token Grant Lifetime** is also disabled.
  - If **Issue Refresh Tokens** is enabled, specify the desired length of time and specify whether the value is in hours or days.

  Note: The **Refresh Token Grant Lifetime** value must be greater than or equal to the **OAuth 2.0 Access Token** value.
- Web
  a Specify the **Description**.
  b Specify the **Redirect URL**.
  c Specify the **Authorized JavaScript Origins**.
  d Specify the **Logout URL**.
  e Select a **Signing Certificate Fingerprint (SHA1)**: Click **Load Certificate**, select the certificate file, and click **Open**.
  f Select the length of time for **OAuth 2.0 Access Token**.
  g If the application does not request refresh tokens, click the slider to disable **Issue Refresh Tokens**.
  - If **Issue Refresh Tokens** is disabled, **Refresh Token Grant Lifetime** is also disabled.

- If is enabled, specify the desired length of time and specify whether the value is in hours or days.

Note: The **Refresh Token Grant Lifetime** value must be greater than or equal to the **OAuth 2.0 Access Token** value.

- Backend Service
  Note: There are options for **User Impersonation** and **ID Translation** that are described in API Gateway bridge solution on page 72. If you are not setting up a hybrid system, leave disabled.
  a Specify the **Description**.
  b Select the length of time for **OAuth 2.0 Access Token**.
  c If the application does not request refresh tokens, click the slider to disable **Issue Refresh Tokens**.
  - If **Issue Refresh Tokens** is disabled, **Refresh Token Grant Lifetime** is also disabled.
  - If **Issue Refresh Tokens** is enabled, specify the desired length of time and specify whether the value is in hours or days.

  Note: The **Refresh Token Grant Lifetime** value must be greater than or equal to the **OAuth 2.0 Access Token** value.

5 Click **Save**.

Editing an authorized application

1 Go to **Authorized Apps** in API Gateway.
2 Select an application.
3 Make your changes.
4 Click **Save**.

Deleting an authorized application

1 Go to **Authorized Apps** in API Gateway.
2 Delete the application by clicking **Delete**, or select an application.
3 At the bottom of the details page of the selected application, click **Delete**.
4 Click **Yes**.
5 Click **OK**.

Locking an authorized application

You can lock an authorized application to make its configuration read-only and to prevent accidental or unauthorized edits or deletions. When the lock is enabled, updates and deletions are not permitted. The lock status is shown on the Authorized Apps page and the Authorized App Detail page. System events record

changes in lock status, such as **Lock enabled** and **Lock disabled**. This helps ensure compliance, reduce the risk of unauthorized changes, and maintain an audit trail.

1 Select **OS > API Gateway > Authorized Apps**.
2 Search for the non-Infor authorized application.

   Search
   Specify the application name or client ID, then press **Enter**.

3 Click the **Lock** icon.
4 Confirm the action if prompted.
5 Verify that the lock status shows as **Lock enabled** on the list and detail pages.
6 Verify that a system event logs that the application is locked.
7 Review additional lock options.

   To unlock an authorized application, repeat the navigation to the Authorized Apps page, search for the application, and click the **Lock** icon again to disable the lock. To view lock status for any application, use the Authorized Apps and Authorized App Detail pages and review the lock status column or field.

Downloading credentials for an authorized application

1 Go to **Authorized Apps** in API Gateway.
2 Select an application.
3 Click **Download Credentials**.

Resetting the secret of an authorized application

Resetting the secret key helps protect sensitive data and maintain the integrity of the application's authentication process. If a secret key is compromised or suspected to be at risk, resetting it immediately invalidates the old key and prevents unauthorized access. This action is especially useful when application credentials are exposed, when rotating secrets as part of routine security maintenance, or when transferring access between teams or systems.

1 Navigate to **OS** > **API Gateway** > **Authorized Apps** in API Gateway.
2 Select a non-Infor authorized application.
3 Click **Reset Secret**.

Note: Resetting the secret for an authorized application immediately invalidates the previous secret. All integrations and systems that use the old secret lose access until they are updated with the new credentials. To minimize service disruptions, use the **Smooth Reset** feature. This functionality allows both the old and new secrets to remain valid for a limited time, providing a transition period to update all affected systems and integrations.

Using the Smooth Reset Secret feature for authorized applications

Resetting credentials can cause disruptions if the old credentials are immediately revoked. The **Smooth Reset** feature reduces this risk by allowing the old credentials to remain active for a specified period.

Example scenario: A company's application must update its credentials regularly for security reasons. Without a smooth reset, the application might experience downtime while new credentials propagate. With Smooth Reset enabled, the old credentials remain active for up to seven days, allowing the application to continue running while new credentials are configured.

1. Navigate to **OS > API Gateway > Authorized Apps**.
2. Select the authorized app and click **Reset Secret**.
3. In the **Reset Secret** dialog box, turn on the **Smooth Rollover** toggle.
   When the Smooth Rollover toggle is enabled, a text box appears for setting the old secret retention period.
4. Specify a value between 1 and 7 days.
   This period defines how long the previous credentials remain valid after new credentials are issued.
5. If the old secret is no longer in use, click **Revoke Old Secret** to invalidate the old credentials before the retention period ends.

Smooth Reset includes validation checks and permission requirements. Use these troubleshooting tips to resolve common issues:

Invalid retention period
The retention period field accepts only values between 1 and 7 days. Enter a whole number within this range. The system prompts for correction if an invalid value is entered.

Smooth Reset toggle not visible
The Smooth Reset feature is available only to IONAPI-Admin users. Ensure that you have the required permissions. Contact your system administrator if the toggle remains hidden.

Old credentials not revoked
If previous credentials remain active beyond the retention period, check for system delays or errors. Manually revoke the old credentials if needed.

UI validation errors
Validation errors can occur when entering values in the retention period field. Enter numeric values within the valid range and follow on-screen prompts to resolve the issue.

Issuing a refresh token for an authorized application

Authorized applications can be configured to issue refresh tokens to extend access token validity without requiring users to reauthenticate.

By default, authorized applications do not issue refresh tokens.

If the application does not request refresh tokens, keep the **Issue Refresh Token** toggle off to disable issuing refresh tokens.

When the refresh token option is enabled, the application receives a refresh token together with an access token. The refresh token is used to obtain a new access token after the current one expires.

By default, the refresh token rotates periodically, for example, every eight hours. Application teams must use the latest rotated refresh token until it rotates again.

If the application team prefers not to rotate the refresh token, they can disable the **Roll Refresh** toggle. Disabling this setting also makes the **Refresh Token Grant Lifetime** infinite.

Application teams can modify the **Refresh Token Grant Lifetime** to define how long the refresh token remains valid. If the value is set to zero, the refresh token remains valid indefinitely. However, if a refresh token remains idle for more than 30 days, it is invalidated automatically.

After a refresh token expires or is revoked, the authorized application must reinitiate authorization using the original grant.

Emailing the QR code for an authorized application

1. Go to **Authorized Apps** in API Gateway.
2. Select an application.
3. Click **Email QR Code to Users**.

Disabling an authorized application

1. Go to **Authorized Apps** in API Gateway.
2. Select an Infor application.
3. Click **Disable**.
4. Click **Yes**.

Enabling an authorized application

1. Go to **Authorized Apps** in API Gateway.
2. Select an Infor application.
3. Click **Enable**.
4. Click **Yes**.

Cloning an authorized application

Note: Specific cases in which an application can or cannot be cloned are:

- The **Clone** button is available only when the application is disabled.

- An application cannot be enabled if a cloned version of the application is enabled.
- The **Clone** button is disabled if a clone of the application already exists.
- Cloned applications cannot be cloned.

1. Go to **Authorized Apps** in API Gateway.
2. Select a disabled application.
3. Click **Clone**.
4. Make your changes.
5. Click **Save**.

API metadata

API metadata provides a way for users to get information about suites, products, and operations within these suites and products through the use of Swagger UI.

API Gateway metadata is an index of suites, products, and operations as found in each endpoints' Swagger documentation. The API Gateway indexing service runs every five days or whenever an API suite is added or deleted.

While you can search within API Gateway metadata by using the provided Swagger UI, API Gateway metadata is used by other applications to query what suites, products, and operations are available in your environment. To be sure your API is included in API Gateway metadata, add swagger documentation to each of your endpoints as specified in Adding API endpoint documentation on page 49.

Troubleshooting API metadata issues

Scenario 1: If the operations are not showing up in the metadata, manually trigger metadata reindexing. This should fix the issue, and this should also fix missing operations in ION workflow, APIFlow, and so on.

Scenario 2: Usually metadata reindexing takes from 15 minutes to 1 hour to complete; however, if the metadata indexing status is stuck at triggered status for a long time, create a dummy non-Infor suite. That should re-trigger metadata, and the status should stabilize after the new refresh is completed.

Configuration

TLS version

For all incoming traffic between authorized apps and API Gateway, both TLS version 1.2 and TLS version 1.3 are supported; however, for outbound communication between the Gateway and target APIs, API Gateway is more accommodating. As tenant administrator, you can set a minimum TLS version for target endpoints from the **General Settings** tab.

The versions available are:

- 1.0
- 1.1
- 1.2
- 1.3

The default value supported is the lowest version: 1.0.

Once the minimum TLS version is set, all target APIs using a lower TLS version will fail.

Additionally, when the minimum TLS version is updated from the **General Settings** page, endpoints that were recently called still use the previous TLS version until the cache is cleared for that endpoint.

Changing the minimum TLS version

To change the minimum TLS version in theAPI Gateway user interface:

1. In the API Gateway application, click the **Configuration** tab.
2. Select **General Settings**.
3. Enforce a minimum TLS version from the list.

   Note: The available TLS versions are applicable for communication between API Gateway and target APIs. This does not apply to communication between authorized apps and API Gateway.

   Communication with any target APIs using a TLS version lower than TLSv1.3 will not work.

General Settings

General Settings contains settings that can be applied.

Export

By default, the setting to enable the export of target endpoint credentials while exporting API suites is disabled.

If enabled, target endpoint credentials are exported in plain text.

You are responsible for keeping the export file safe as the credentials allow access to the target endpoint.

JWK management

When setting up JWT target authentication for an endpoint in the API Gateway user interface, a JWK key is generated. The JWK Management table, located on the **General Settings** tab, is used to keep track of generated JWK keys.

The JWK Management table can also be used for setting up API Gateway bridge credentials for a backend service application. To set up API Gateway bridge credentials:

1. Navigate to the API Gateway application.
2. Click **Configuration**.
3. Click **General Settings**.
4. Download a generated key from the API Gateway JWK Management table. The key generates into a .json file.
5. Click **Authorized Apps** on the side-panel.
6. While creating a backend service application, engage the **Use Bridge Credentials** slider.
7. Click the **User Impersonation** and/or the**ID Translation** slider.
8. Click **Upload Public Key**.
9. Select the `ionapi-bridge-public-jwt.json` file downloaded previously.

OAuth 2.0

If you need to authorize API Gateway in your third-party system, you can use the **Authentication Code Grant Redirect URL** as the redirect URL or the callback URL.

Copy the link in this field and paste it in the appropriate field in the third-party system.

Custom OAuth 2.0 scopes

This document provides IT administrators with step-by-step instructions for managing Custom OAuth 2.0 Scopes in Infor OS. It covers scope creation, association with API suites and Authorized Apps, deletion procedures, configuration prerequisites, system limitations, and troubleshooting tips to ensure secure and efficient API access control within the Infor OS environment.

What are OAuth 2.0 scopes?

OAuth 2.0 scopes define permissions that control the level of access an application has to a user's data. They ensure that a client application performs only authorized actions. For example, an access token may allow READ and WRITE access or just READ access. A token with READ scope lets the client retrieve information, but not modify it. A WRITE scope allows both reading and updating data.

Scopes improve security by limiting access to sensitive data and functionality. They also offer flexibility in managing access by letting administrators assign scopes based on the application's and user's needs. Each application receives only the permissions it requires, minimizing the risk of unauthorized access.

Custom scopes offer more granular control over API access. You can define permissions precisely and enable applications to interact with APIs securely. Custom scopes help API Gateway administrators tailor permissions to specific needs, enhancing security and access management.

Business benefits of Scopes:

- Enhanced security – Scopes allow precise control over API access. You can restrict access to specific API suites using tenant-level scopes, so only authorized applications make API calls.
- Flexibility – Create and manage scopes that suit your needs. You can associate them with different API suites and Authorized Applications.

What are custom OAuth2.0 scopes

Custom scopes in API Gateway are user-defined OAuth 2.0 scopes. They specify access levels for non-Infor API suites. As an administrator, you can use them to control and limit API calls through API Gateway.

Benefits of custom scopes in API Gateway:

In addition to general benefits, custom scopes offer:

1. Globalization support. You can include descriptions in global languages like Chinese, Arabic, French, and Spanish.
2. Improved user experience. The API Gateway UI simplifies creating, managing, and associating custom scopes. Features like search, sorting, and tile-based displays enhance usability.
3. Audit and monitoring. You can track scope-related activities using audit events, which supports compliance and security efforts.

Examples:

- ReadOnlyScope. Grants read-only access to specific endpoints.
- AdminScope. Allows management of user roles, settings, and configurations.
- HRScope. Gives access to employee data, payroll, and HR systems based on the user's HR role.

Required role to access custom scopes

To manage custom scopes, you need the IONAPI-Admin role in the User Management system. This role grants permissions to create, edit, and configure scopes. It also lets you associate scopes with API suites and applications.

Creating a custom scope

Steps:

1. Verify Role. Make sure you have the IONAPI-Admin role.
2. Open IONAPI UI. Go to the interface and find OAuth2.0 Custom Scopes under OAuth2.0 Settings.
3. Add Scope. Click '+ Add' and provide details:
   - Name. Enter a valid name (only English characters allowed).
   - Description. Optionally, add a description using global characters.
4. Save. Click Save to create the scope.

Associating a custom OAuth 2.0 scope with an API suite

1. Open the custom scopes page

   In the IONAPI interface, go to OAuth2.0 Settings and select OAuth2.0 Custom Scopes.

   Click the name of the custom scope you want to associate with an API suite.
2. Link the custom scope to API suites

a Scroll to the API Suites tab at the bottom of the custom scope page.
b Click the '+' icon to open the selection UI for non-Infor API suites.
c In the search bar, type the name of the API suite. Matching results appear in a dropdown.
d Select the desired suite(s) and click Associate. The system automatically saves your changes.
e Associated suites are listed under the API Suites tab, and the same custom scope appears on each suite's details page for clear cross-reference.

Associating a Custom OAuth 2.0 Scope with an Authorized App

1 Enable Scope toggle
   - In the IONAPI Admin UI, open Configuration > OAuth2.0 Settings.
   - Turn on 'Enable scopes per authorized app' to activate the scope dropdown for all apps.
2 Assign a custom scope to an App
   - Open the Authorized Apps page and find the app that needs the custom scope.
   - In the Scope dropdown, type the scope name. Matching scopes appear as you type.
   - Select the scope to complete the association.
   - To verify, download the credentials and check the scopes list. The custom scope should be included.
   - The association also appears under the Authorized App tab of the scope, helping you track and manage linked apps.

This process mirrors how standard scopes are associated, ensuring a consistent and familiar experience. Following these steps helps you manage custom scopes efficiently and securely across your system.

Deleting a custom scope

1 Check Associations. Ensure the scope is not linked to any suite or app.
2 Open Custom Scopes List. Locate the tile for the scope.
3 Delete. Hover to reveal the delete option and confirm the action.

Limitations of custom OAuth 2.0 scopes

Scope limit per tenant

You can create up to 100 custom scopes per tenant. This limit helps keep the system manageable and avoids clutter that could complicate authorization and permission management.

Scope name validations

Custom scopes must follow these naming rules to ensure consistency and prevent conflicts:

- The name must not start with the prefix infor-. This prefix is reserved and using it causes conflicts and integration issues.
- The name can be up to 50 characters long to maintain clarity and ease of management.
- The description cannot exceed 500 characters. Keep descriptions concise to clearly explain each scope's purpose.
- Each scope name must be unique. Duplicate names are not allowed to avoid confusion in authorization management.
- The name cannot be openid since this is reserved for OpenID Connect and using it can interfere with authentication processes.

- Custom scope names do not support globalized characters.

Troubleshooting custom scopes issues

When working with custom OAuth 2.0 scopes, you might encounter various issues that affect scope creation, association, or API access. This table outlines common problems and provides practical solutions to help you quickly identify and resolve these issues. Following these guidelines ensures your authorization system remains efficient, secure, and well-integrated within your environment.

Issue	Description	Solution
Scope name conflicts	Custom scope name conflicts with reserved prefixes or existing scopes.	Avoid names starting with "infor-" or "openid". Make sure each scope name is unique and does not duplicate existing ones.
Exceeding character limits	Scope name or description exceeds allowed length.	Keep scope names less than or equal to 50 characters and descriptions less than or equal to 500 characters. Shorten if necessary.
Globalization issues	Scope names contain unsupported globalized characters.	Use only universally supported characters; avoid special or locale-specific characters.
Authorization errors	Authorization fails due to misconfigured scopes.	Verify scope names and descriptions to meet guidelines. Confirm scopes are correctly defined and integrated.
Exceeding scope limits	Tenant exceeds the maximum allowed number of custom scopes.	Review scopes, consolidate or delete unnecessary ones to stay within the 100-scope tenant limit.
Integration problems	Custom scopes cause integration issues with other systems.	Confirm names and descriptions comply with integration requirements. Avoid reserved prefixes or conflicting names.
Scopes not visible in Authorized App page	Scope dropdown does not appear in Authorized App details.	Ensure the Scopes toggle is enabled under Configuration > OAuth2.0 Settings.
API Gateway returns 403 after scope association	API calls return 403 Forbidden errors after associating custom scopes.	Check token requests include all required custom scopes. Confirm custom scope matches API suite. Review target server policies and authentication settings.

Monitoring

Monitoring allows you to view all transactions that have passed through your gateway.

Monitoring data is purged after 30 days.

This feature is available only to users with one of these security roles:

- IONAPI-Administrator

- Infor-SystemAdministrator
- IONAPI-Tracing

Monitoring is based on searching for a transaction. Monitoring also has some general gateway information dashboard tiles:

- API Gateway Health
- API Gateway Monitoring
- API Gateway Info

API Gateway Health

The API Gateway Health tile shows the general health of your environment.

If there are any concerning issues, a warning icon is shown.

Clicking API Gateway Health gives you a high-level health check of your environment. The status of any errors, policy execution, and token validation are displayed.

There is an option to download the API Gateway Health as a JSON file if, for example, you need to send it to technical support to view the errors.

API Gateway Monitoring

The API Gateway Monitoring tile shows a green icon when application-level monitoring or system-level monitoring is enabled.

Application-level monitoring is the general monitoring of every request that has passed through the gateway. This information is displayed in the API Gateway Monitoring user interface.

System-level monitoring is system log entries that can be used by a developer to delve deeper into how a particular request was processed. This information is contained only in the downloadable JSON file from the transaction details page.

Clicking **API Gateway Monitoring** enables you to change the settings of API Gateway Monitoring.

API Gateway Monitoring settings

By default, **Capture Transaction Details** is enabled.

This enables or disables application level monitoring.

API Suite Tracing

By default, **API Suite Tracing** is disabled for each API suite.

This setting applies to system-level monitoring and overrides the general application-level monitoring. If **API Suite Tracing** is enabled for an API suite, system- and application-level monitoring is applied even if application-level monitoring is disabled for the environment.

API Suite Tracing is resource intensive and affects performance of the gateway. It is recommended that**API Suite Tracing** be used only for deep troubleshooting.

You can filter the list of API suites for API suites that have **API Suite Tracing** enabled or disabled. You can disable all or search for a particular API suite. If you make a change, click **Refresh** within the screen to see your change.

If your environment has many API suites, use the paging feature to view ten suites at a time.

API Gateway Info

The API Gateway Info tile shows the version and build of API Gateway.

Search

API Gateway Monitoring is based on searching for a particular transaction.

You can filter your search by:

- All
- API Suite: the suite that contains the requested transaction
- Path: the last part of the path on which the request was made, starting with `/{Tenant}/{API Context}/{End point}/{Resource}`
- Request ID: each request is assigned an ID in GUID format
- User name: the name of the user who made the request, if available

You can limit your search by time:

- Last 5 minutes: the **From** time and Service account – the ID of the service account that made the request, if available**To** time display search results from the last five minutes.
- Last 1 Hour: the **From**Service account – the ID of the service account that made the request, if time and **To** time display search results from the last hour.
- Last 24 Hours: the **From** time and **To** time display search results from the last 24 hours.
- Custom: select your own **From** time and **To** time.

  Monitoring data is purged after 30 days.

Click **GO** to execute your search.

Most Recent

Most Recent shows the three most recent transactions that have passed through your gateway.

An option to refresh the transaction list is available.

Search Results

Search Results display:

- API suite: the suite of the requested transaction
- Path: the last part of the path in which the request was made, starting with `/{Tenant}/{API Context}/{End point}/{Resource}`
- Request ID: each request is assigned an ID in GUID format
- User name: the name of the user who made the request, if available
- Service account: the ID of the service account that made the request, if available
- Response time: the total round trip time the request was handled by the gateway (in milliseconds)
- Request time stamp: the gateway time stamp at the time the request was received

You can filter your search results:

- Success: transactions that successfully completed
- Failure: transactions that have a failure somewhere in the request and response round trip

You can sort your search results:

- Time stamp ascending, oldest results first
- Time stamp descending (default), newest results first
- Response time ascending, shortest response time first
- Response time descending, longest response time first

Clicking a search result displays the transaction details of that request.

Transaction Details

Transaction Details display:

- API suite: the suite for the requested transaction
- Path: the last part of the path on which the request was made, starting with `/{Tenant}/{API Context}/{End point}/{Resource}`
- Request ID: each request is assigned an ID in GUID format
- User name: the name of the user who made the request, if available
- Service account: the ID of the service account that made the request, if available.
- Response time: the total round trip time the request was handled by the gateway (in milliseconds)
- Request time stamp: the gateway time stamp at the time the request was received.

Also displayed is each step the gateway took to process this transaction. Any failure is highlighted. Clicking each step shows details about that step.

For transactions that used API flows, this display first lists each API that was used. Click an API to shows details about the steps that were taken, as defined above.

An option to download the transaction details is available. The transaction detail JSON file contains application-level monitoring and system-level monitoring (if enabled).

Request Response Logging

Detailed Request Response logging enables users to download the total request and response payload for these transactions:

- Requests that API Gateway receives
- Requests that API Gateway sends to target
- Responses that API Gateway receives
- Responses that API Gateway sends to the client

This enables API Gateway administrators to monitor and debug custom/non-Infor suite payloads.

Turning on Detailed Request Response logging in API Gateway Monitoring

Prerequisites

Detailed Request Response Logging is generally available for Infor OS for AWS Commercial version 2024.10.

The ability to use Detailed Request Response logging is available only to users assigned to the IONAPI-Admin role.

Turning on Detailed Request Response logging

1. In Infor OS, navigate to API Gateway and click **Monitoring**.
2. Turn on the **Enable Detailed Logging** toggle and click **`Yes`** on the **Enable Detailed Logging** the dialog box.
3. Select the API suites that you want to monitor. Only user-created suites can be tracked:
   - Infor non-provisioned suites
   - Third Party suites
   - Non-Infor suites

   You can enable suites only after selecting the **Transaction Detail** check box.
4. Save the selections. API Gateway makes a call to the selected API suites and searches for the transactions on the monitoring page.

   You can filter the transactions between **`Basic`** and **`Detailed`**.

   With Detailed Logging enabled, you can use the download icon to download the transactions.

Limitations

- Detailed logging is available for 10 minutes. A quota of 10 requests per minute is enforced. The first 10 requests in a minute have detailed logging.
- The maximum log file size supported is 40 MB.
- For security purposes, the file expires in 24 hours. After 24 hours, the file is no longer available for download.

Authorizations

The **Authorizations** page enables you to authorize API Gateway to access a third-party API on the users' behalf.

The items displayed on the **Authorizations** page correspond to each endpoint that uses OAuth 2.0 authentication or deployment.

Authorizing and revoking authorization

To authorize an endpoint, the app suite must be added to ION API, and API Gateway must be authorized in the third-party system. See "API endpoints" and ""General Settings" for more information.

1. Go to **API Gateway > Authorizations**.
2. To authorize the endpoint, click the tab for the desired API and endpoint combination.
3. Provide credentials to the third-party application. API Gateway can now access a third-party API on the user's behalf.
4. To revoke authorization, hover on the desired API and endpoint combination.
5. Click the **Revoke** icon. API Gateway can no longer access a third-party API on the user's behalf.

Chapter 5: Hybrid Deployment

Enterprise Connector

Use the Enterprise Connector to establish communication between API Gateway and an on-premises target API without a direct network connection. The API Gateway hybrid service runs as an independent application/service in the Enterprise Connector. This feature is available only on the Cloud.

To access the option, the desired enterprise location must first be provisioned in ION by a user with the IONDeskAdmin role. See the *Infor ION User Guide* for details.

This feature can be accessed in non-Infor target endpoint details, Infor non-provisioned deployment details, or third-party deployment details.

Prerequisites for Enterprise Connector with API Gateway

For prerequisites for the Enterprise Connector service, see the “Enterprise Connector prerequisites” section of the *Infor ION User Guide-Cloud Edition*.

Additionally, note these prerequisites for using the Enterprise Connector service with API Gateway:

- Only SQL Server is supported.
- Only Windows is supported.
- The Enterprise Connection version must be at least 2020-08.

Service	Operating System	DB	Remarks
API Gateway	Windows/Linux	SQL Server/Postgres	

Limitations

Payload limit

As of now, only a payload size up to 5 MB is supported by Enterprise Connector (EC).

Limited supported target authentication policies

Currently, these target authentication policies are supported by EC:

- Anonymous
- API Key
- Basic
- JWT Target Authentication
- OAuth 1.0a
- WS-Security Username Token

Manual import of the public key/certificate

For HTTPS-based target APIs, importing of the public key/certificate must be performed manually in the EC grid administration. See the knowledge base article 2155379 on the Infor Support Portal for instructions: https://support.infor.com/espublic/EN/AnswerLinkDotNet/SoHo/Solutions/SoHoViewSolution.aspx?SolutionID=2155379&kb_accessed_from=KBViews

Compression support note

Content-Encoding: Deflate is not supported in Hybrid Service. Deflate is rarely used and has been superseded by more efficient algorithms.

Timeout limitations

These are the default timeouts for EC service, which you can configure:

- `connectTimeout`: 1 minute
- `socketTimeout`: 1 minute
- `connectionRequestTimeout`: 5 minutes

You can configure these timeout values through these EC GRID variables:

```
'''json
"ionapi.connect.timeout"   :  connect Timeout
"ionapi.socket.timeout"    :  socket Timeout
"ionapi.request.timeout"   :  connection Request Timeout
'''
```

Configuring Enterprise Connector for API Gateway

The Enterprise Connector service is automatically installed at a specified location the first time a suite is deployed. Likewise, the Enterprise Connector service will be uninstalled when the last API suite referring to that location is removed. If the service is running and if the Enterprise Connector is upgraded using the newer version of installer, then API Gateway service will also be upgraded automatically.

To configure Enterprise Connector for API Gateway:

1. Add or edit one of the above endpoint/deployment type details as described in the API Endpoints or API Deployments section.

2 Click the toggle to enable Enterprise Connector.
3 From the list, select an enterprise location. Enterprise locations that have been provisioned in ION are displayed in the dropdown.
4 Click **Save**. The target endpoint/deployment details are saved with the selected location. The Enterprise Connector policy is added for the target endpoint.

   Creation and deletion of the web API service for the Enterprise Connector location are managed automatically.

Enterprise Connector performance metrics

By default, an Enterprise Connector uses a single Hybrid Service node. Expect an average throughput of 30 requests per second and an average latency of ten seconds. The Hybrid Service is horizontally scalable, and performance scales relatively linearly.

With three Hybrid Service nodes, expect an average throughput of 80 requests per second and an average latency of three to four seconds. Note that the latency value listed is observed only when reaching the maximum throughput. Low throughput results in lower latency.

Default installation with a single hybrid service node

Average requests per second	30
Maximum requests per second	33
Average latency (in seconds) at maximum throughput	10.3

Horizontally scaling up to three service nodes

Average requests per second	80
Maximum requests per second	102
Average latency (in seconds) at maximum throughput	3.7

Interpreting the Enterprise Connector status

For API Gateway service and the Enterprise Connector errors and warnings, messages are color coded to indicate the status:

- Green: Modeling can continue.
- Yellow: Modeling can continue, but one of the dependent components has an issue that can be fixed offline and the API will work.
- Red: Prerequisites are not met so modeling cannot continue:

This information provides a more detailed explanation of the error or warning status:

EC OK

HB: OK -> Icon: Green -> Tooltip: EC running, EC version supports hybrid services, ION service running

HB: Pending -> Icon: Green -> Tooltip: Hybrid Service installation is pending

HB: Unknown -> Icon: Green -> Tooltip: Hybrid Service not available

HB: Error/Not Ok -> Icon: Red -> Tooltip: Hybrid Service not running, EC version supports hybrid service

EC ERROR*

Icon: Yellow -> Tooltip: EC not running

EC PENDING/EC NOTEXISTED

Icon: Red -> Tooltip: Please install EC first -> Extra functionality: Disable save button, disable suite

EC UNSUPPORTED/ UNKNOWN

Icon: Red -> Tooltip: Please upgrade EC to latest version - > Extra functionality: Disable save button, disable suite

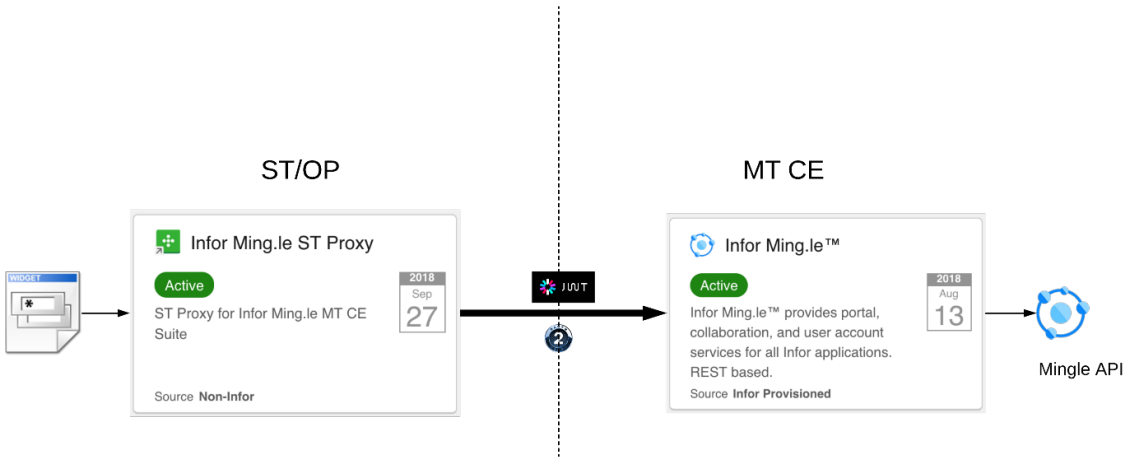
API Gateway bridge solution

The information in this chapter applies to Infor OS 12.0.29 and later upgrades.

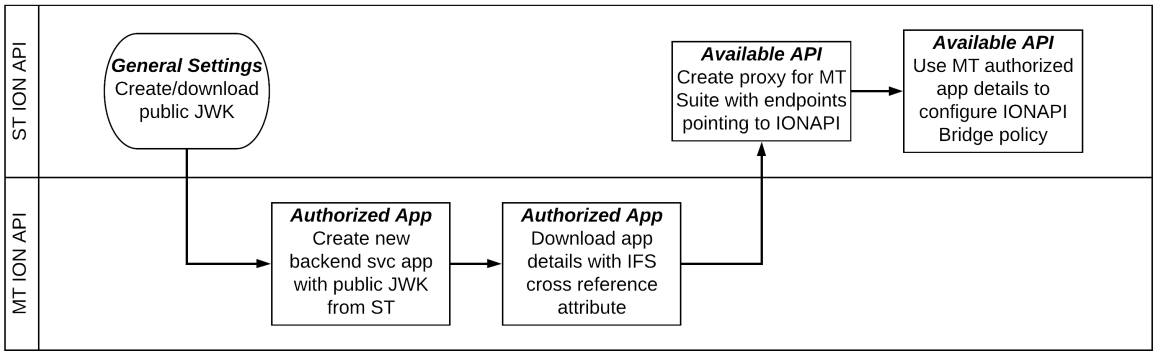
API Gateway bridge solution refers to deployment topologies where API Gateway and its authorized app reside in different network boundaries. For example, API Gateway running in the multi-tenant cloud and the authorized app running on premises.

These scenarios require API Gateway functionality to handle identity translation and/or user impersonation.

This image shows an authorized app accessing a multi-tenant API suite via an API Gateway proxy that has been secured using the API Gateway Bridge:



This image shows the steps required to set up this bridge solution:



Prerequisites

These prerequisites are required to set up an API Gateway bridge solution:

- An on-premises or single-tenant API Gateway instance
- A multi-tenant API Gateway instance
- An API suite to be used as a proxy in the bridge solution
- Users that exist in both a multi-tenant and single-tenant/on-premises instance with at least one user management (IFS) property that matches, for example: the multi-tenant EmailAddress matches the single-tenant/on-premises Identity2 for each user
- API Gateway Administrator security role

Common terms

Identity translation
Localization of the user identity claim (Identity2) while crossing network boundaries.

User Impersonation
Allowing an authorized app to make API calls on behalf of a different user.

Multi-Tenant (MT)
Infor cloud-based architecture in which a single instance of a software application serves multiple customers. Each customer is called a tenant.

Single-Tenant (ST)
In single-tenant architecture, the tenant purchases a copy of the software. Infor Single Tenant usually refers to an on-premises installation hosted on the Infor Cloud.

On-Premises (OP)
A single-tenant architecture, where the tenant purchases a copy of the software and the software is installed at the customer's premises.

API Gateway Bridge
An API Gateway Target security method that allows for a bridge solution.

API Gateway Bridge Subject
A user property from the authorized app side of the bridge, to be matched with the cross reference on the target endpoint side of the bridge.

API Gateway Bridge Cross reference (XRC)

A user property in the target endpoint side of the bridge, to be matched with the subject on the authorized app side of the bridge.

Configuration

This section describes the steps that must be performed to set up the bridge solution. The steps must be performed in the order listed in this section.

In your single-tenant or on-premises API Gateway (Initial steps)

1. Select **General Settings** from the API Gateway menu.
2. Within **ION API Bridge Credentials**:
   a. Click **Add**. A key ID is generated.
   b. Select **Download** for the key ID that is generated and save the `IONAPI-Bridge-Public-JWT.json` file to a secure location.

In your multi-tenant API Gateway

1 Select **Authorized Apps** from the API Gateway menu.
2 Click **Add**.
3 Specify a name for your authorized app. For example: API Gateway Bridge Backend Service.
4 Select **Type**: **`Backend Service`**.
5 Specify a description.
6 To allow for user impersonation, select **User Impersonation**.
7 To allow for ID translation, select **ID Translation**.
8 Select **Upload Public Key**.

   Navigate to the `IONAPI-Bridge-Public-JWT.json` file downloaded from your single-tenant/on-premises API Gateway in the previous section. **Key ID**, **Key Value**, and **Algorithm** are populated from the `IONAPI-Bridge-Public-JWT.json` upload.
9 Click **Save**. The client ID and secret are generated.
10 Select **Download Credentials**.
11 Select **Create Service Account**.
12 Select a **User Management Property for ID Translation**.

   Note: This is the cross reference (XRC) and is what the API Gateway Bridge subject will be matched against.
13 Click **Download** and save your API Gateway Bridge `Backend Service.ionapi` file to a secure location.
14 Select **Available APIs** from the API Gateway menu.
   a Choose an API suite.
   b Within the suite details, copy the endpoint to be bridged.
15 Return to your single-tenant/on-premises API Gateway to complete the remaining configuration steps.

In your single-tenant/on-premises API Gateway (Return steps)

1 Select **Available APIs** from the API Gateway menu.
2 Click **Add** (+).
3 Click **Create new** (+).
4 Complete these fields:
   - **Application Name**
   - **Suite Name**
   - **Description**
   - **API Context**
   - **Choose an icon**
5 Click **Add endpoint** (+).
   a Paste the endpoint copied previously as **`Target Endpoint URL`**.
   b Complete these fields:
      - **Target Endpoint Description**
      - **Proxy Endpoint URL**

- **Choose Proxy Security**

6 Under Target Endpoint Security:

a Select **ION API Bridge Authentication Type**.

b Select **Load File** and select the API Gateway Bridge `Backend Service.ionapi` file previously downloaded.

These fields are populated:

- **Tenant ID**
- **Application Name**
- **Client ID**
- **Client Secret**
- **OAuth2 Authorization Server URL**
- **OAuth2 Authorization Endpoint**
- **OAuth2 Token Endpoint**
- **OAuth2 Revoke Endpoint**
- **Scope**
- **Environment**
- **Version**
- **Service Account Access Key**
- **Service Account Secret Key**
- **Key ID**
- **Cross Reference**

7 Select **Subject Type**.

Note: This is the API Gateway Bridge subject and will be matched against the cross reference (XRC).

8 Click **Save**. Your API Gateway Bridge solution is now configured successfully.

Chapter 6: Gateway support for WebSocket

WebSocket is a protocol that provides full-duplex communication channels over a single TCP connection, enabling real-time data exchange between clients and servers. API Gateway supports endpoints that use the WebSocket protocol, providing a robust framework for real-time communication.

Overview

Unlike traditional HTTP, which follows a request–response model, WebSocket allows a persistent connection that reduces latency and improves application performance. This makes it suitable for applications that require continuous data streaming, such as live chat, online gaming, and financial trading platforms.

Benefits of using the WebSocket protocol

- Lower latency: The persistent connection reduces the need for constant reconnection, leading to faster data transmission.
- Efficient message format: WebSocket frames are smaller than HTTP headers, resulting in more efficient data exchange.
- Real-time communication: Enables instant updates and interactions, making it ideal for dynamic applications.
- Reduced overhead: Because the connection remains open, there is less overhead compared to opening multiple HTTP connections.
- Bi-directional communication: Both client and server can send messages independently, enabling more interactive features.

WebSocket support in API Gateway

API Gateway supports adding endpoints that use the WebSocket protocol, providing a reliable structure for real-time communication. The following sections describe how to set up, test, and monitor WebSocket endpoints within API Gateway. This ensures seamless integration and optimal performance of WebSocket-based services.

Adding Infor non-provisioned WebSocket endpoints in API Gateway

Note: Because API templates for non-provisioned suites are registered in the API Suite Registry in API Gateway, the Infor application team must register the endpoint in their application suite as a **`WebSocket`** type before adding a WebSocket deployment.

1. Navigate to the **API Suites** page.

   To create a new non-provisioned API suite, click the **+ Add** tile and select an available API template. You can then configure the suite and add its WebSocket endpoint. If the suite already exists, open it to continue with the deployment steps.

2. Add a deployment.

   In the selected API suite, click **Add Deployment** to begin creating a new deployment.

3. Select the WebSocket protocol.

   In the deployment setup, choose **`ws`** (WebSocket) from the list of protocols, then click **Submit**. This designates the endpoint as a WebSocket deployment.

4. Specify the WebSocket server URL.

   Enter the WebSocket server URL in the **Target Endpoint URL** field. The URL must begin with **`ws://`** for non-secure connections or **`wss://`** for secure connections.

5. Verify the proxy URL.

   After submitting the deployment, verify that the endpoint is configured correctly by reviewing the proxy URL on the **Endpoint Detail** page. This page displays the details of the newly created WebSocket deployment.

Adding a WebSocket type non-Infor endpoint

Use these steps to add a WebSocket type non-Infor endpoint in API Gateway. You can add the endpoint to an existing API suite or create one specifically for non-Infor endpoints.

1. Navigate to the API suite.

   Access the API Gateway interface and locate the API suite where you want to add the WebSocket endpoint. This can be an existing suite or a newly created one for non-Infor endpoints.

2. Add a new endpoint.

   Within the API suite, click the **+** button on the **Endpoints** tab.

3. Enter the WebSocket target endpoint URL.

   Provide the WebSocket server URL, starting with **`ws://`** for non-secure connections or **`wss://`** for secure connections. Enter this URL in the **Target Endpoint URL** field.

4. Verify the proxy URL.

   After submitting the deployment configuration, check the proxy URL on the **Endpoint Detail** page to confirm that the endpoint has been set up successfully.

The WebSocket type non-Infor endpoint is added in API Gateway and is ready for testing and monitoring.

Testing the endpoints

After configuring the WebSocket endpoint, test its functionality to ensure that it operates correctly. You can use web clients or API testing tools such as Postman to perform these tests.

To test using a web client:

1. Open a web client that supports WebSocket connections, such as Chrome Developer Tools.
2. Navigate to the console and enter the WebSocket server URL, for example **`ws://example.com/socket`** or **`wss://secure.example.com/socket`**.
3. Initiate the connection and send test messages to verify communication.
4. Observe the responses and check for errors or unexpected behaviour.

To test using an API testing tool:

1. Open Postman or another API testing tool that supports WebSocket connections.
2. Select the WebSocket request type and enter the WebSocket server URL.
3. Connect to the server and send test messages to verify functionality.
4. Review the responses for success indicators or error messages.

These tests help confirm that the WebSocket endpoint is operational and ready for monitoring and further use.

Monitoring

The **Monitoring** page in API Gateway displays both WebSocket and HTTP requests. You can filter the results to view only WebSocket requests and track their status and activity.

Overview

The API Gateway **Monitoring** page shows WebSocket requests in addition to existing HTTP requests. You can filter the results to display only WebSocket requests by selecting **Protocol > Filter** from the toolbar.

Connection status

The status of the connection is indicated as follows:

- **`Active`**: Green
- **`Completed`**: Gray
- **`Error`**: Red

Testing WebSocket endpoints

When working with WebSocket endpoints, verify their functionality by sending various test messages to the server and reviewing the responses. Successful indicators or error messages help determine whether the WebSocket endpoint is fully operational.

Limitations and best practices

WebSocket is a powerful communication protocol but has certain limitations and considerations that affect its configuration and usage in API Gateway.

Limitations

While WebSocket provides real-time, bidirectional communication, it has these limitations:

- WebSocket does not support Swagger documentation.
- There is no metadata indexing for WebSocket endpoints.
- By default, WebSocket calls time out when idle. Idle is defined as fewer than or equal to six bytes of data incoming or outgoing on the connection within a 30-minute period.
- Some policies must be blocked for WebSocket endpoints because they can interfere with functionality:
  - CacheResponse
  - FaultHandling
  - JsonThreatProtection
  - regExThreatProtection
  - targetTimeout
  - jsonTransform
  - user-security-claims
  - xmlThreatProtection
  - xmlToJson
  - Quota
  - Throttling

Best practices when using WebSocket endpoints

- Ensure that the client can re-establish the connection if a timeout occurs.
- Verify the validity of the token during reconnection attempts.
- Implement a retry mechanism with a fallback step in case of repeated reconnection failures.
- Always use secure WebSocket (**`wss://`**) for enhanced security.

Chapter 7: Backend as a Service

Infor BaaS (Backend as a Service) enables you to build services locally and deploy them in a cloud environment. **Note:** BaaS requires an add-on license in addition to your Infor OS core license. The BaaS functionality is only accessible with the appropriate license and user security roles specifically for IONAPI-BAAS.

You create APIs to send data between Infor products or between an Infor product and a third-party app.

Using BaaS, you manage the complete API lifecycle:

- Build serverless, event driven functions (FaaS) and expose them as APIs (BaaS)
- Build serverless APIs
- Customize Infor APIs
- Add third-party integrations

Services on page 81 is for the administrators managing customer-specific services developed with Backend as a Service.

Subsequent sections in this chapter are for developers who are responsible for building APIs used for sending data between Infor products or between an Infor product and a third-party app.

Prerequisite knowledge

To fully understand the information presented in this chapter, you should first have expertise and knowledge of programming in Typescript and/or Java.

Services

Use the options in the **Services** view to import, export, deploy, and maintain the BaaS services.

Viewing deployed BaaS services

1. In Infor OS, open API Gateway from the App Menu, and select **Backend As A Service > Services**.
   The BaaS services are presented as a list and can also be displayed in a tile view.
2. Review the selection of deployed services.
   Services are arranged alphabetically.

3 Type in the **Search** field to filter the display of the services.
4 Click **Details** to review the service-specific information, configurations and logs.

Viewing BaaS service details

1 In Infor OS, open API Gateway from the App Menu, and select **Backend As A Service > Services**.
2 Click the service-specific details.
3 Review the content.
4 Click **Back** to return to the previous view.

Updating the runtime configuration for a deployed service

1 In Infor OS, open API Gateway from the App Menu, and select **Backend As A Service > Services**.
2 Click the service-specific details.
3 On the **Variables** tab, review the current runtime configuration, and update the values as applicable.
4 Either save the values individually by clicking **Save**, or in bulk by clicking **Save all**.
The values are applied in the runtime immediately when the updated configuration is saved.

Updating the deployment configuration for a deployed service

The initial deployment configuration of a Backend as a Service (BaaS) service is defined during development. After deployment, you can update the configuration values in the service manifest to adapt them to the current environment. To apply the changes, redeploy the service with the updated configuration.

1 From the App Menu, open API Gateway and select **Backend As A Service > Services**.
2 In the service list, click **Details** for the service.
3 On the **Configurations** tab, review the current values and update them as required.

Configuration values
Replace the default settings with values that meet the requirements of the current environment.

4 If you update more than one value, click **Save all**.
5 Click **Redeploy** to apply the updated configuration.

Redeploying a BaaS service

You can redeploy a BaaS service with the current configuration, with the latest framework version, or with an updated deployment configuration.

Redeployment ensures that a Backend as a Service (BaaS) service runs with the required framework and configuration. These options are available when you redeploy a service:

- Redeploy with the current framework and configuration: Use this option to redeploy the service as is, without changes.
- Redeploy with the latest framework version: Use this option to redeploy the service with the version that is currently available from the server. This option is recommended in test environments to verify compatibility with the latest operating system version from the cloud provider.
- Redeploy with an updated deployment configuration: Use this option to redeploy the service with updated deployment properties. The set of properties is the same for all services and includes policies and other configuration values.
- To redeploy a service after updating its deployment configuration, follow the procedure described in Updating the deployment configuration for a deployed service on page 82.

Redeploying with the current framework and deployment configuration

Use this procedure to redeploy a Backend as a Service (BaaS) service with the current framework and deployment configuration. This option redeploys the service as is, without changes.

1. From the App Menu in Infor OS, open API Gateway and select **Backend As A Service > Services**.
2. In the service list, click **Actions**.
3. Select **Redeploy** from the menu.
4. In the confirmation dialog, click **Yes**.
5. Wait until the redeployment is complete. During redeployment, the menu option is inactive.

Redeploying with the latest framework

Use this procedure to redeploy a Backend as a Service (BaaS) service with the latest framework version. This option is recommended in test environments to verify compatibility with the most recent operating system version from the cloud provider.

1. In Infor OS, open API Gateway from the App Menu, and select **Backend As A Service > Services**.
2. In the service list, click **Actions**.
3. Select **Redeploy with Framework** from the menu.
4. In the confirmation dialog, click **Yes**.
5. Wait until the redeployment is complete. During redeployment, the menu option is inactive.

Logs

BaaS services generate and store log events based on the current log level setting.

Each Backend as a Service (BaaS) service generates and stores log events for each handler according to the configured log level. The log level determines which categories of events are included.

These are the available log levels:

- ERROR: Logs error events.
- WARN: Logs warning and error events.
- INFO: Logs informational, warning, and error events. This is the default level.
- DEBUG: Logs debug, informational, warning, and error events.
- TRACE: Logs all events, including trace, debug, informational, warning, and error events.

Each level includes events from the more severe categories. For example, the WARN level includes both warning and error events, and the TRACE level includes events for all categories.

Changing the log level setting for a deployed service

1. In Infor OS, open API Gateway from the App Menu, and select **Backend As A Service > Services**.
2. In the service list, click **Details** for the service you want to update.
3. Click **Expand**.
4. Specify this information:

   Setting for log level
   Select the desired log level. The system displays 'Log level successfully set' when the change is applied.

   The higher the log level, the higher the logging cost. Use DEBUG or TRACE only during troubleshooting sessions.

   The log level resets to INFO when the service is redeployed.

Display of service log events per handler

Overview of viewing and filtering service log events for BaaS handlers in Infor OS.

The **Logs** tab presents events corresponding to the activities of a given handler based on log level settings. By default, the 1000 latest events for the REST handler are shown. If the service has multiple handlers, select a different handler from the selection list to view its events.

This concept covers three main areas:

- Basic filtering of service log events: Filter events by specifying a search term in **Filter Logs**.
- Advanced filtering of service log events: Use the advanced filter to specify a date and time interval for the events and retrieve updated events.
- Viewing execution log events: Review runtime activities of service handlers in the **Execution Logs** tab, including resource consumption.

Basic filtering of service log events:

1. In Infor OS, open API Gateway from the App Menu, and select **Backend As A Service > Services**.
2. Click **Details** for the applicable service.
3. Click the **Logs** tab.
4. Enter a search term in **Filter Logs** to filter events. Only matching events are displayed.
5. Clear the **Filter Logs** field and click **Refresh** to reset the log view.

Advanced filtering of service log events:

1. Click the **Ellipsis** button and select **Show Advanced Filter**.
2. Specify a date and time interval in the **From** and **To** fields, then click **Search**.

Viewing execution log events:

1. Click the **Execution Logs** tab to view the latest 100 execution entries.
2. Select a handler from the **ALL** selection list to filter execution events for that handler.
3. Optionally, enter a search term in **Search Execution Logs** to further filter the events.

The execution log shows runtime activities of each service handler and resource consumption per trigger event.

Exporting a BaaS service

To perform this task you must have the IONAPI-Administrator role combined with either the IONAPI-BaaS-Administrator role or the IONAPI-BaaS-Developer role.

Note: You can only export services that contain promoted builds.

1. In Infor OS, open API Gateway from the App Menu, and select **Backend As A Service > Services**.
2. To export a service, locate the service, click **Actions** and select **Export**.
   The service files are exported to the default download location on your computer.

Importing a BaaS service

To perform this task you must have the IONAPI-Administrator role combined with either the IONAPI-BaaS-Administrator role or the IONAPI-BaaS-Developer role.

1. In Infor OS, open API Gateway from the App Menu, and select **Backend As A Service > Services**.
2. Click **Import**.
3. Locate the service file folder on your computer, select it, and click **Open** to import the service.
   The message 'Starting Service Import' is displayed. When the import is finished, the display is updated.

Deleting a BaaS service

To perform this task you must have the IONAPI-Administrator role combined with either the IONAPI-BaaS-Administrator role or the IONAPI-BaaS-Developer role.

All data associated with the service is removed.

1. In Infor OS, open API Gateway from the App Menu, and select **Backend As A Service > Services**.
2. To delete a deployed service, locate the service, click **Actions** and select **Delete**.

3 Confirm the deletion as requested and click **Delete**.
The message 'Starting Service Delete' is displayed. When the delete is finished, the display is updated.

Documents

Use the **Documents** menu to access code for building your APIs.

The information in this section is intended for developers.

Viewing BaaS documentation

1 Select **Backend as a service > Documentation**.
2 Click the folder icon to the left of a documentation type to display options:
   - Administration
     See the instructions in the "Backend As A Service" chapter of the *Infor API Gateway Administration Guide* in the Infor OS User and Administration Documentation Library (Cloud) on Infor Documentation Central.
   - Code
     Review all code examples. The examples assume that the "context" variable is available. This variable is included in all the service-generated functions.
   - Extension
     Learn of the various aspects of the BaaS developer tool extension in Visual Studio Code
   - Manual
     Read about the different areas of BaaS, from installation to service creation, and how different areas within the service work.
   - Support
     Here you can learn how to resolve issues experienced in BaaS.
   - Tools
     BaaS tools consist of a browser-based Administrator tool and a Developer tool as extensions in the free development environment of Visual Studio Code, build on Open Source and running on Windows, Linux, and MacOS.
3 Continue to use the folder icons to drill down to more levels of information. Use the resulting breadcrumb trail at the top of the page to return to higher levels of information.

Downloads

Use the **Downloads** menu options to access tools for building your APIs.

Downloading the SDK

To perform this task you must have the IONAPI-BaaS-Developer role.

1. Select **Backend as a service > Downloads**.
2. Under **Download the BaaS Visual Studio Code plugin, and start developing today**, click **Download**. The SDK is downloaded to the default download location on your computer.

Downloading the SDK from Visual Studio Code Marketplace

To perform this task you must have the IONAPI-BaaS-Developer role.

1. In Infor OS, open API Gateway from the App Menu, select **Backend as a service > Downloads**.
2. Click the **Visual Studio Code Marketplace** link.

Monitoring

Use the **Monitoring** menu option to view requests per API in deployed BaaS services.

Viewing requests per API in a deployed BaaS service

1. In Infor OS, open API Gateway from the App Menu, and select **Monitoring**.
2. In the **Search** fields, select **`Path`** and specify the name of an endpoint for a REST resource of a specific deployed BaaS service as the search criteria.
3. Click **Go**.

Configuration Management

Expand **Configuration Management**.

The **Tags** menu option is displayed if at least one tag has been created for the current tenant in the CloudSuite Self-Service Portal.

Adding resources to tags

Note: Only services that contain promoted builds are available for selection

1. In Infor OS, open API Gateway from the App Menu, and select **Configuration Management > Tags**.
2. Locate the tag and click **Active**.
3. Click the add (+) icon.
4. On the **Associate Resources** dialog box, select the required resource and click **Add**.
5. Select one or more resources and click **Associate**.
6. Add more resources as required.
7. Click the **Save associated resources** icon to save your changes.

Available APIs

All BaaS services belong to a suite. The suites are represented as tiles within **Avaliable APIs**.

Accessing REST API documentation for BaaS services

1. In Infor OS, open API Gateway from the App Menu, and select **Available APIs**.
2. Set the filter to display **`Infor BaaS Suites`**.
3. Click a tile to list the endpoints available for this suite. Each row corresponds to a BaaS service.
4. Select the relevant service and click the **Documentation** icon.
5. Click the **Documentation** tab to access the documentation of the applicable resources.

Appendix A: Policies

Policies are available at API suite level and proxy endpoint level.

Policy type	Level	Flow
FaultHandling	Suite and Proxy	Request and Response
Header	Suite and Proxy	Request and Response
Quota	Suite and Proxy	Request
CacheResponse	Proxy	Request
JsonThreatProtection	Proxy	Request
JsonTransform	Proxy	Request and Response
QueryParam	Proxy	Request
RegExThreatProtection	Proxy	Request
XmlThreatProtection	Proxy	Request
XmlToJson	Proxy	Request and Response
CookieRewrite	Proxy	Request
Transformation	Proxy	Request and Response
UserSecurityClaims		

FaultHandling

Use the FaultHandling policy to modify the code or message returned by the server in case of an error.

Example

In this example, the status code and message are replaced by the specified status code and message.

```
<faultHandling
     xmlns="http://www.infor.com/ion/api"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    name="faultHandling-example" displayName="faultHandling-example" enabled="true" version="1.0">
```

```
    <rules>
         <rule>
               <statusCode>500</statusCode>
               <set>
                    <statusCode>502</statusCode>
                    <message>Please contact the system administrator.</message>
               <set>
         </rule>
    </rules>
</faultHandling>
```

Configuration

Element name	Default	Presence	Type	Multiplicity
rules	n/a	Required	n/a	1
rules.rule	n/a	Required	n/a	1..*
rules.rule.status-Code	n/a	Required	string	1
rules.rule.set	n/a	Optional	n/a	0..1
rules.rule.set.Sta-tusCode	n/a	Optional	string	0..1
rules.rule.set.mes-sage	n/a	Optional	string	0..1

<faultHandling> attributes

<faultHandling name="faultHandling-example" displayName="faultHandling-example" enabled="true" version="1.0">

Field name	Description	Default	Presence
name	Name of this policy in-stance.	N/A	Required
displayName			Optional
enabled	Indicates if a policy is enforced or not. If set to false, a policy is turned off, and not enforced.	true	Optional
version	policy version	N/A	Required

<rules> element

List of rules to be enforced by this policy.

<rules.rule> element

Rule to be enforced by this policy.

<rules.rule.statusCode> element

Http code that in the response that would trigger the execution of this rule.

<rules.rule.set> element

Elements in the response that are set if this rule is executed.

<rules.rule.set.statusCode> element

Http code to set if the enclosing rule is executed.

<rules.rule.set.message> element

Message to return if the enclosing rule is executed.

Header

This policy allows you to set or delete headers to either the response or the request.
Note: Setting a header creates a new header unless it already exists, in which case it updates the existing header.

Example 1

In this example, a header is set for a request.

```
<header
    xmlns="http://www.infor.com/ion/api"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    name="header-example" displayName="header-example" enabled="true" version="1.0" >
    <action>set<action>
    <headerName>x-cache</headerName>
    <headerValue>true</headerValue>
</header>
```

Example 2

In this example, the header value is extracted from the context.

```
<header name="header-example" displayName="header-example" enabled="true" >
    <action>set<action>
    <headerName>x-cache</headerName>
    <headerValue ref='context.url.tenant'/>
</header>
```

In the example, reference is made to a variable in the context object. The context object is a shared dictionary of information that can be accessed from the policies.

Configuration

Element name	Default	Presence	Type	Multiplicity
action	n/a	Required	string	1
headerName	n/a	Optional	string	0..1
headerValue	n/a	Optional	string	0..1

<header> attributes

```
<header name="header-example" displayName="header-example" enabled="true" version="1.0">
```

Field Name	Description	Default	Presence
name	Name of this policy instance.	N/A	Required
displayName			Optional
enabled	Indicates if a policy is enforced or not. If set to false, a policy is turned off, and not enforced.	true	Optional
version	Policy version	N/A	Required

<action> element

The action indicates the intention of this policy:

- set: set the header value. Insert if the header does not already exist. Otherwise, update.
- delete: remove the header. If the header name is '*', the policy deletes all the headers.

```
<action>set<action>
```

<headerName> element

Name of the affected header.

```
<headerName>x-cache</headerName>
```

<headerValue> element

Value of the header being set. This element is required only if you are setting a header.

```
<headerValue>true</headerValue>
```

The header value can also refer to a variable:

```
<headerValue ref="context.url.tenant"/>
```

Quota

Use the Quota policy to configure the number of request messages that an app is allowed to submit to an API over the course of a second, minute, hour, or day.

Example

Use this sample code to enforce a quota of 1,000 calls. The policy starts and stops the counter based on the interval and unit of time of the time stamp for the first request message received by the API proxy.

For example, the first message is received at 2011-01-07 08:31:15. The quota counter starts at 2011-01-07 08:31:15 and the counter stops and resets to 0 at 2011-01-07 09:31:15 (1 hour from the start time). The start time is the clock or calendar start time of the defined TimeUnit value, such as second, minute, hour, or day. The end time is based on the elapsing of the Interval value in the defined TimeUnit.

If the counter reaches the 1,000-call quota before the end of the hour, calls beyond 1,000 are rejected.

Example:

```
<quota
    xmlns="http://www.infor.com/ion/api"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    name="quota-example" displayName="quota-example" enabled="true" version="1.0" >
        <userLevel>true</userLevel>
        <interval>1</interval>
        <timeUnit>hour</timeUnit>
        <allow>1000</allow>
</quota>
```

Configuration

Element name	Default	Presence	Type	Multiplicity
userLevel	false	Optional	boolean	1
interval	n/a	Required	integer	1
timeUnit	n/a	Required	day, hour, minute, second	1
allow	n/a	Required	integer	1

<quota> attributes

```
<quota name="quota-example" displayName="quota-example" enabled="true" version="1.0">
```

Field name	Description	Default	Presence
name	Name of this policy instance.	N/A	Required
displayName			Optional
enabled	Indicates if a policy is enforced or not. If set to false, a policy is turned off, and not enforced.	true	Optional
version	Policy version.	N/A	Required

<interval> element

Use to specify the interval of time (in seconds, minutes, hours, or days as defined by TimeUnit) applicable to the quota.

For example, an Interval of 10 with a TimeUnit of hours means that the quota is calculated over period of 10 hours.

```
<interval>1</interval>
```

<timeUnit> element

Use to specify the unit of time applicable to the quota.

For example, an Interval of 10 with a TimeUnit of hours means that the quota is calculated over period of 10 hours.

The valid time units are: second, minute, hour, and day.

```
<timeUnit>hour</timeUnit>
```

<allow> element

Specifies a message count for the quota.

```
<allow>1000</allow>
```

<userLevel> element

This flag indicates if the quota should be set at the user level.

```
<userLevel>true</userLevel>
```

CacheResponse

This policy uses a cache to store and retrieve a response from a back-end resource, reducing the number of requests to the resource.

Examples

In this example, a cache response policy is configured to cache responses for an hour based on the tenant, product name, and the request query parameters name and last name.

Although the tenant and product name are not mentioned as part of the key, these values are implied and automatically added by the system.

For example:

```
<cacheResponse
        xmlns="http://www.infor.com/ion/api"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        name="cache-response-example" displayName="cache-response-example" enabled="true" ver
sion="1.0" >
        <userLevel>false</userLevel>
        <cacheKey>
                <prefix>acme</prefix>
                <keyFragment ref="request.queryparam.name"><keyFragment>
                <keyFragment ref="request.queryparam.lastName"><keyFragment>
                <keyFragment ref="context.my_variable"><keyFragment>
                <keyFragment>Infor<keyFragment>
        </cacheKey>
        <expireSettings>
                <timeoutInSeconds>3600</timeoutInSeconds>
        </expireSettings>
</cacheResponse>
```

In this example, reference is made to a variable in the context object. The context object is a shared dictionary of information that can be accessed from the policies.

Configuration

Element name	Default	Presence	Type	Multiplicity
userLevel	false	Optional	boolean	0..1
cacheKey	n/a	Required	string	1
cacheKey.prefix	n/a	Required	string	1
cacheKey.keyFragment	n/a	Optional	string	1..*
expireSettings	n/a	Required	string	1
expireSettings.timeoutInSeconds	n/a	Required	string	1

<cacheResponse>

```
<cacheResponse name="cache-response-example" displayName="cache-response-example" enabled="true"
 version="1.0">
```

Attributes:

Field name	Description	Default	Presence
name	Name of this policy instance.	N/A	Required
displayName			Optional
enabled	Indicates if a policy is enforced or not. If set to false, a policy is turned off, and not enforced.	true	Optional
version	Policy version.	N/A	Required

<userLevel> element

This flag indicates if the cache should be set at the user level.

```
<userLevel>true</userLevel>
```

<cacheKey> element

`<CacheKey>` constructs the name of the key for the response stored in the cache. The key is often set using a value from context variables or query parameters.

The prefix as well as at least one keyFragment is required.

```
<cacheKey>
     <prefix>acme</prefix>
     <keyFragment ref="request.queryparam.name"><keyFragment>
     <keyFragment ref="request.queryparam.lastName"><keyFragment>
     <keyFragment ref="context.my_variable"><keyFragment>
     <keyFragment>Infor<keyFragment>
</cacheKey>
```

<cacheKey.prefix> element

Sets a prefix for the cache key.

<cacheKey.keyFragment> element

Sets a key fragment that is concatenated as part of the cache key. The keyFragment can either be a literal string or a value retrieved from the context. Use the attribute ref to retrieve a value from the context.

Field name	Description	Default	Presence
ref	Reference to a value in the context.	N/A	Optional

<expireSettings> element

Configuration of cache expiration for the response.

```
<expireSettings>
        <timeoutInSeconds>3600</timeoutInSeconds>
</expireSettings>
```

<expireSettings.timeoutInSeconds> element

Contains the number of seconds a response should be cached.

JsonThreatProtection

This policy enables you to reduce the risk of content-level attack by specifying limits on various JSON structures, such as arrays and strings.

This policy executes only if the content type header is set to **`json`**.

Example

In this example, a header is set for a request.

```
<jsonThreat xmlns="http://www.infor.com/ion/api"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       name="jsonThreat-example" displayName="jsonThreat-example" enabled="true" version="1.0">

    <arrayElementCount>255</arrayElementCount>
    <containerDepth>5</containerDepth>
    <objectEntryCount>100</objectEntryCount>
    <objectEntryNameLength>25</objectEntryNameLength>
    <stringValueLength>25</stringValueLength>
</jsonThreat>
```

Configuration

Element name	Default	Presence	Type	Multiplicity
arrayElement-Count	n/a	Optional	integer	1
containerDepth	n/a	Optional	integer	1
objectEntryCount	n/a	Optional	integer	1
objectEntryName-Length	n/a	Optional	integer	1

Element name	Default	Presence	Type	Multiplicity
stringValueLength	n/a	Optional	integer	1

<jsonThreat> attributes

```
<header name="jsonThreat-example" displayName="jsonThreat-example" enabled="true" version="1.0">
```

Field name	Description	Default	Presence
name	Name of this policy instance.	N/A	Required
displayName			Optional
enabled	Indicates if a policy is enforced or not. If set to false, a policy is turned off, and not enforced.	true	Optional
version	Policy version.	N/A	Required

<arrayElementCount> element

Optional element that indicates the maximum number of elements allowed in an array.

```
<arrayElementCount>255</arrayElementCount>
```

<containerDepth> element

Optional element that indicates the maximum allowed nested depth.

```
<objectEntryCount>100</objectEntryCount>
```

<objectEntryCount> element

Optional element that indicates the maximum number of entries allowed in an object.

```
<objectEntryNameLength>25</objectEntryNameLength>
```

<objectEntryNameLength> element

Optional element that indicates the maximum string length allowed for an object's entry name.

```
<objectEntryNameLength>25</objectEntryNameLength>
```

<stringValueLength> element

Optional element that indicates the maximum length allowed for a string value.

```
<stringValueLength>25</stringValueLength>
```

JsonTransform

Use the JsonTransform policy to adjust the JSON data returned from the target server.

The JSON can be adjusted in these ways:

- Selected properties can be deleted.
- Properties can be added with default values.
- An entirely new JSON response can be created by using selected parts of the original response.
- Any/All of the above can be applied during the same invocation of the policy.

Note that any/all deletions are done before any/all default values additions and before any/all transformations. In the case of deletions if your JSON paths refer to properties/objects/array-elements that do not exist, the delete request is ignored. In the case of deletions if your JSON paths refer to properties/objects/array-elements that do not exist, the resulting property will have a value of **`undefined`**.

Resources

The JSON transform policy relies on a special language called JSONPath that is used to select objects, sub-objects, properties, and array elements with the JSON document.

Examples

For these examples, assume that the target server normally returns the JSON response shown below:

Sample JSON Response from Target Server:

```
{
  "store": {
    "book": [
      {
        "category": "reference",
        "author": "Nigel Rees",
        "title": "Sayings of the Century",
        "price": 8.95
      },
      {
        "category": "fiction",
        "author": "Evelyn Waugh",
        "title": "Sword of Honour",
        "price": 12.99
      },
      {
        "category": "fiction",
        "author": "Herman Melville",
        "title": "Moby Dick",
        "isbn": "0-553-21311-3",
        "price": 8.99
      },
```

```
      {
        "category": "fiction",
        "author": "J. R. R. Tolkien",
        "title": "The Lord of the Rings",
        "isbn": "0-395-19395-8",
        "price": 22.99
      }
    ],
    "bicycle": {
      "color": "red",
      "price": 199.95,
      "size": "24-inch",
      "safetyRated": true,
      "features": {
        "style": "mountain",
        "brakes": "disc"
      }
    }
  }
}
```

The example below shows how parts of a response can be deleted. This can be used to reduce the payload size if it contains items that are never used by the calling client.

Policy Options to Delete Selected Properties:

```
 <jsonTransform continueOnError="false" displayName="jsonTransform_policy_transform"
               enabled="true" name="JSONTranform" version="1"
              xmlns="http://www.infor.com/ion/api" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
               xsi:schemaLocation="http://www.infor.com/ion/api jsonTransform.xsd">
    <deletions>
        <delete path="$.store.book[1..2].price"/> <!-- delete 2nd and 3rd book -->
        <delete path="$.store.bicycle.color"/> <!-- delete color from bicycle -->
    </deletions>
</jsonTransform>
```

Adjusted Response after Deletions:

```
 {
  "store": {
    "book": [
      {
        "category": "reference",
        "author": "Nigel Rees",
        "title": "Sayings of the Century",
        "price": 8.95
      },
      {
        "category": "fiction",
        "author": "J. R. R. Tolkien",
        "title": "The Lord of the Rings",
        "isbn": "0-395-19395-8",
        "price": 22.99
      }
    ],
    "bicycle": {
      "price": 199.95,
      "size": "24-inch",
      "safetyRated": true,
      "features": {
        "style": "mountain",
        "brakes": "disc"
      }
    }
  }
}
```

The next example shows how sub-objects and properties can be added if missing from the document. The policy uses the JSONPaths to check if the objects/properties already exist in the document. If they already exist, those existing values are returned in the document untouched. If they do not exist, they are added to the document with the values you specify.

Policy Options to Add Default Values:

```
<jsonTransform continueOnError="false" displayName="jsonTransform_policy_transform"
               enabled="true" name="JSONTranform" version="1"
              xmlns="http://www.infor.com/ion/api" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
               xsi:schemaLocation="http://www.infor.com/ion/api jsonTransform.xsd">
    <deletions/> <!-- no deletions (but if there were any they would all be done before the any
 default values were added -->
    <defaultValues>
        <default path="$.store.bicycle.terms.warranty">1 year parts and labor</default>  <!--
add a simple property and value -->
        <default path="$.store.bicycle.contactInfo">
          <![CDATA[JSON{"email": "sales@infor.com", "phone": "678-319-8000"}]]> <!-- Note CDATA-
wrapper and JSON{" syntax used to add and entire sub-object -->
        </default>
    </defaultValues>
</jsonTransform>
```

Adjust Response after Default Value Additions:

```
{
  "store": {
    "book": [
     ...  (book array is unchanged and omitted for the sake of brevity)
    ],
    "bicycle": {
      "color": "red",
      "price": 199.95,
      "size": "24-inch",
      "safetyRated": true,
      "features": {
        "style": "mountain",
        "brakes": "disc"
      },
      "terms": {
         "warranty": "1 year parts and labor"
      },
      "contactInfo": {
         "email": "sales@infor.com",
         "phone": "678-319-8000"
      }
    }
  }
}
```

This example shows how deletions, defaults, and full transformations can be combined. Note that all deletes are done before all default value additions and before any transformations.

Policy Options for Combined Delete, DefaultValue, and Transformation:

```
<jsonTransform continueOnError="false" displayName="jsonTransform_policy_transform"
               enabled="true" name="JSONTranform" version="1"
              xmlns="http://www.infor.com/ion/api" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
               xsi:schemaLocation="http://www.infor.com/ion/api jsonTransform.xsd">
    <deletions>
        <delete path="$.store.book[1].price"/> <!-- delete price from 2nd book -->
    </deletions>
    <defaultValues>
       <default path="$.store.bicycle.contactInfo">  <!-- add contactInfo sub-object if it does
```

```
 not already exist -->
            <![CDATA[JSON{"email": "sales@infor.com", "phone": "678-319-8000"}]]>
        </default>
    </defaultValues>
    <transformations>
        <transform><![CDATA[{   <!-- create an entirely new response document using selected
bits and pieces of the original target response -->
            "allPrices": ["$.store..price"],            <!-- Build new array of just prices of
everything in the store -->
           "contact": "$.store.bicycle.contactInfo",   <!-- Grab contactInfo object, but rename
 as contact -->
            "book3": "$.store.book[2]"                  <!-- Grab just 3rd book and rename it -
->
      }]]></transform>
    </transformations>
</jsonTransform>
```

Adjusted Response after Delete, Default, and Transform:

```
{
  "allPrices": [8.95, 8.99, 22.99, 199.95],  (Only 4 prices since 2nd book price was deleted)
  "contact": {                               (contactInfo did not even exist in document until
we used defaultValues to add it)
    "email": "sales@infor.com",
    "phone": "678-319-8000"
  },
  "book3": {
    "category": "fiction",
    "author": "Herman Melville",
    "title": "Moby Dick",
    "isbn": "0-553-21311-3",
    "price": 8.99
  }
}
```

Configuration

<jsonTransform> Attributes

```
<jsonTransform name="JSON transform example" displayName="jsonTransform" enabled="true" ver
sion="1.0">
```

Name	Default	Required	Description
name	n/a	yes	Name of this policy instance.
displayName			Display name of this policy instance.
enabled	true	yes	Indicates if the policy is enabled or not. If not enabled, the policy is ignored by API Gateway.
version	1.0	yes	Version of the policy.

Elements

Element	Default	Required	Type	Multiplicity
deletions	n/a	no	container of 0..* <delete> child elements	0..1
defaultValues	n/a	no	container of 0..* <default> child elements	0..1
transformation	n/a	no	container of 0..* <transform> child elements	0..1

<delete> Element

Zero or more `<delete>` elements can appear under the `<deletions>` element.

Name	Type	Required	Description
path	attribute, string	yes	JSONPath of the object/property/array-item in the document that is to be deleted.

<default> Element

Zero or more `<default>` elements can appear under the `<defaultValues>` element.

Name	Type	Required	Description
path	attribute, string	yes	JSONPath of the object/property/array-item in the document that should be checked for existence and created if it does not exist.
(element value)	string	yes	number, true|false, string, or JSON{} sub-object enclosed in a <<[CDATA[...]]> wrapper. This is the value that is added at the place indicated by the path if it does not already exist.

<transform> Element

Zero or more `<transform>` elements can appear under the `<transformations>` element.

Name	Type	Required	Description
kind	attribute, string	no	Kind of transformation to use. Choices are "handlebars" or "jsonpathObjectTransform". If omitted, the default is jsonpathObjectTransform.
output	attribute, string	no	Value to use for response content-type header. For example. you can say "application/xml" if you are transforming JSON from the target into XML. If omitted, the content-type provided by the target server is used unchanged.
n/a	CDATA string	yes	JSON template using JSONPath expressions describing how to build a new document from the original document.

QueryParam

Use the QueryParam policy to set, update, or delete a query parameter in the request.

Examples

Set using reference variable:

In this example, the value of the query parameter tenant is set to the tenant ID found in the API Gateway request context.

```
<queryParam
    xmlns="http://www.infor.com/ion/api"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    name="queryParam-example" displayName="queryParam-example" enabled="true" version="1.0">
    <action>set<action>
    <paramName>tenant</paramName>
    <paramValue ref="context.tenant"></paramValue>
</queryParam>
```

In the example, reference is made to a variable in the context object. The context object is a shared dictionary of information that can be accessed from the policies.

Set hard-coded value:

In this example, the query parameter **sort** is set to the value **true**.

```
<queryParam name="queryParam-example" displayName="queryParam-example" enabled="true" ver
sion="1.0">
    <action>set<action>
    <paramName>sort</paramName>
    <value>true</value>
</queryParam>
```

delete all:

In this example, all query-string parameters are being deleted. One reason you might want to do this is that the query-string values are used to create to create headers (using the Header policy) and you do not want the query-string values passed to the target server.

```
<queryParam name="queryParam-example" displayName="queryParam-example" enabled="true" ver
sion="1.0">
    <action>delete<action>
    <paramName>*</paramName>
    <value>true</value>
</queryParam>
```

Configuration

Element name	Default	Presence	Type	Multiplicity
action	n/a	Required	string (**set** or **delete**)	1
paramName	n/a	Optional	string (for delete can be special value *)	1
paramValue	n/a	Optional	string	1

<queryParam> attributes

```
<queryParam name="queryParam-example" displayName="queryParam-example" enabled="true" ver
sion="1.0">
```

Field name	Description	Default	Presence
name	Name of this policy instance.	N/A	Required
displayName			Optional

Field name	Description	Default	Presence
enabled	Indicates if a policy is enforced or not. If set to false, a policy is turned off, and not enforced.	true	Optional
version	Policy version.	N/A	Required

<action> element

The **set** action indicates the intention of this policy, which updates the query parameter value. Insert if the parameter does not already exist.

```
<action>set<action>
```

<name> element

Name of the affected query parameter.

```
<paramName>sort</paramName>
```

<value> element

Value of the query parameter being set.

```
<paramValue>true</paramValue>
```

The query parameter value can also make reference to a variable:

```
<paramValue ref="context.auth.tenant"/>
```

RegExThreatProtection

This policy enables you to reduce the risk of content-level attack by evaluating the request content against predefined regular expressions.

In case that specified regular expressions evaluate to true, the message is considered a threat and rejected.

No regular expression can eliminate all content-based attacks, and multiple mechanisms should be combined to enable defense-in-depth. With this in mind, these are recommended patterns for blacklisting content.

Blacklisted Patterns

Name	Regular xpression
SQL Injection	[\s]*((delete)|(exec)|(drop\s*table)|(insert)|(shut-down)|(update)|(or))
Server-Side Include Injection	<!--\s*<!--(include|exec|echo|config|printenv)\s+.*
XPath Abbreviated Syntax Injection	(/(@?[\w_?\w:*]+(\[[^]]+\])*)?)+
XPath Expanded Syntax Injection	/?(ancestor(-or-self)?|descendant(-or-self)?|following(-sibling))
JavaScript Injection	<\s*script\b[^>]*>[^<]+<\s*/\s*script\s*>
Java Exception Injection	.*Exception in thread.*

Examples

Example 1:

In this example, the uriPath, query parameters, headers, and jsonPayload are checked for threats:

```
<regExThreatProtection xmlns="http://www.infor.com/ion/api"
     nxmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     name="regExThreatProtection-example"
     displayName="regExThreatProtection-example" enabled="true" version="1.0">
      <uriPath>
            <pattern>\\d{3}[-.]?\\d{3}[-.]?\\d{4}</pattern>
      </uriPath>
      <jsonPayload>
           <jsonPath>
                <expression>^.*</expression>
                 <pattern>.*uglyThreat.*</pattern>
           </jsonPath>
      </jsonPayload>
      <queryParam name="name">
         <pattern>[tT]rue</pattern>
         <pattern>.*true.*</pattern>
      </queryParam>
      <header name="greetings">
         <pattern>[tT]rue</pattern>
         <pattern>.*true.*</pattern>
      </header>
      <header name="greeting">
          <pattern>[tT]rue</pattern>
          <pattern>.*true.*</pattern>
      </header>
</regExThreatProtection>
```

Example 2:

In this example, the xmlPayload is checked for threats:

```
<regExThreatProtection xmlns="http://www.infor.com/ion/api"
     nxmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     name="regExThreatProtection-example"
     displayName="regExThreatProtection-example" enabled="true">
      <xmlPayload>
           <namespaces>
               <namespace prefix="infor">http://www.infor.com</namespace>
               <namespace prefix="acme">http://www.acme.com</namespace>
           </namespaces>
```

```
            <xPath>
                 <expression>//infor:title/text()</expression>
                 <pattern>.*ThreatTitle.*</pattern>
            </xPath>
       </xmlPayload>
</regExThreatProtection>
```

Configuration

Element name	Default	Presence	Type	Multiplicity
uriPath	n/a	Optional	n/a	0..1
uriPath.pattern	n/a	Required	string	1..*
jsonPayload	n/a	Optional	n/a	0..1
jsonPayload.jsonPath	n/a	Required	n/a	1..*
jsonPayload.jsonPath.expression	n/a	Required	string	1..1
jsonPayload.jsonPath.pattern	n/a	Required	string	1..*
xmlPayload	n/a	Optional	n/a	0..1
xmlPayload.namespaces	n/a	Optional	n/a	0..1
xmlPayload.namespaces.namespace	n/a	Optional	string	0..*
xmlPayload.xPath	n/a	Required	n/a	1
xmlPayload.xPath.expression	n/a	Required	string	1
xmlPayload.xPath.pattern	n/a	Required	string	1..*
queryParam	n/a	Optional	n/a	0..1
queryParam.pattern	n/a	Required	string	1..*
header	n/a	Optional	n/a	0..1
header.pattern	n/a	Required	string	0..1

<regExThreatProtection> attributes

```
<regExThreatProtection name="regExThreatProtection-example" displayName="regExThreatProtection-
example" enabled="true" version="1.0">
```

Field name	Description	Default	Presence
name	Name of this policy instance.	N/A	Required
displayName			Optional
enabled	Indicates if a policy is enforced or not. If set to false, a policy is turned off, and not enforced.	true	Optional
version	Policy version.	N/A	Required

<uriPath> element

This is an optional element that configures the regular expressions that must be evaluated against the URIPath.

This element is used to match the path of the URI.

The path starts with a forward slash and is the string that comes after the host and port. The path does not include the protocol, host, and port of the URI. For example, the path of the URI is the `mywebsite/customers` section in the following URI:

```
https://myserver:8443/mywebsite/customers
```

Only one uriPath element is allowed. The uriPath can contain multiple patterns.

```
<uriPath>
     <pattern>\\d{3}[-.]?\\d{3}[-.]?\\d{4}</pattern>
</uriPath>
```

<jsonPayload> element

This is an optional element that configures the string to be extracted from a JSON payload and evaluated against the regular expressions provided.

Only one jsonPayload element is allowed. The jsonPayload can contain multiple jsonPath elements, which in turn can contain multiple patterns.

This rule executes only if the content type header is set to **`json`**.

```
<jsonPayload>
     <jsonPath>
          <expression>^.*</expression>
          <pattern>.*uglyThreat.*</pattern>
     </jsonPath>
</jsonPayload>
```

<queryParam> element

This is an optional element that indicates the regular expressions that must be evaluated against a given query parameter.

Multiple queryParam elements are allowed. The queryParam element can contain multiple patterns.

```
<queryParam name="name">
    <pattern>[tT]rue</pattern>
    <pattern>.*true.*</pattern>
</queryParam>
```

<xmlPayload> element

This is an optional element that configures the string to be extracted from an XML payload and evaluated against the regular expressions provided.

Only one xmlPayload element is allowed. The xmlPayload can contain multiple xPath elements, which in turn can contain multiple patterns.

This rule executes only if the content type header is set to **`xml`**.

```
<xmlPayload>
    <namespaces>
       <namespace prefix="infor">http://www.infor.com</namespace>
       <namespace prefix="acme">http://www.acme.com</namespace>
    </namespaces>
    <xPath>
       <expression>//infor:title/text()</expression>
       <pattern>.*ThreatTitle.*</pattern>
   </xPath>
</xmlPayload>
```

<nameSpaces> element

This is an optional element that indicates the allowed name spaces to be used in an xmlPayload element.

<header> element

This is an optional element that indicates the regular expressions that must be evaluated against the a given header.

Multiple header elements are allowed. The header element can contain multiple patterns.

```
<header name="greeting">
     <pattern>[tT]rue</pattern>
     <pattern>.*true.*</pattern>
</header>
```

XmlThreatProtection

This policy enables you to reduce the risk of content-level attack by specifying limits on various XML structures.

This rule executes only if the content type header is set to **`xml`**.

Example

In this example, a header is set for a request.

```
<xmlThreatProtection xmlns="http://www.infor.com/ion/api"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      name="xmlThreatProtection-example" displayName="xmlThreatProtection-example" enabled="true"
 version="1.0">

    <nameLimits>
        <element>20</element>
        <attribute>20</attribute>
    </nameLimits>
    <valueLimits>
        <text>500</text>
        <attribute>100</attribute>
        <comment>200</comment>
    </valueLimits>

</xmlThreatProtection>
```

Configuration

Element name	Default	Presence	Type	Multiplicity
nameLimits	n/a	Optional	n/a	0..1
nameLimits.element	n/a	Optional	integer	0..1
nameLimits.attribute	n/a	Optional	integer	0..1
valueLimits	n/a	Optional	n/a	0..1
valueLimits.text	n/a	Optional	integer	0..1
valueLimits.attribute	n/a	Optional	integer	0..1
valueLimits.comment	n/a	Optional	integer	0..1

<xmlThreatProtection> attributes

```
<header name="xmlThreatProtection-example" displayName="xmlThreatProtection-example" enabled="true"
 version="1.0">
```

Field name	Description	Default	Presence
name	Name of this policy instance.	N/A	Required
displayName			Optional

Field name	Description	Default	Presence
enabled	Indicates if a policy is enforced or not. If set to false, a policy is turned off, and not enforced.	true	Optional
version	Policy version.	N/A	Required

<nameLimits> element

This is an optional element that indicates the maximum number of characters allowed for element and attribute names in an xml document. All the elements inside the nameLimits element are also optional.

```
<nameLimits>
        <element>20</element>
        <attribute>20</attribute>
</nameLimits>
```

In the example above, the name limits are set so that an xmlThreatProtection event is raised if either an element name or attribute name exceeds 20 characters.

<valueLimits> element

This is an optional element that indicates the maximum number of characters allowed for the values of attributes, text, and comments. All the elements inside the valueLimits element are also optional.

```
<valueLimits>
    <text>500</text>
    <attribute>100</attribute>
    <comment>200</comment>
</valueLimits></objectEntryCount>
```

In the example above, the value limits are set so that an xmlThreatProtection event is raised in one of these cases:

- The text section of an xml element exceeds 500 characters.
- An attribute value exceeds 100 characters.
- A comment exceeds 200 characters.

XmlToJson

Use the XmlToJson policy to automatically convert an XML response from the target server to a JSON response returned to the calling client.

Basic XML to JSON

```
<xmlToJson xmlns="http://www.infor.com/ion/api"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
    name="queryParam-example" displayName="queryParam-example" enabled="true" version="1.0">
</queryParam>
```

Configuration

There are currently no configurable options for this policy.

CookieRewrite

Use the CookieRewrite policy to modify the path and/or domain string in a cookie set on the response.

If the policy is placed in the response flow, the cookie is flagged as secure.

Examples

Example 1:

In this example, the path of the cookie is replaced by a path built using the tenant ID and product name.

```
<cookieRewrite
        xmlns="http://www.infor.com/ion/api"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        name="cookieRewrite-example" displayName="cookieRewrite-example" enabled="true" ver
sion="1.0" >
        <cookieName>sessionId</cookieName>
        <path>/{context.mcc.Tenant.Id}/{context.mcc.Context}</path>
</cookieRewrite>
```

In the example above, reference is made to a variable in the context object. The context object is a shared dictionary of information that can be accessed from the policies.

Example 2:

In this example, the domain of the cookie is replaced by a string built using the tenant ID.

```
<cookieRewrite name="cookieRewrite-example" displayName="cookieRewrite-example" enabled="true"
>
        <cookieName>sessionId</cookieName>
        <domain>/{context.mcc.Context}</domain>
</cookieRewrite>
```

In the previous two examples, the path and domain are overwritten with a string literal. A smarter way of modifying a cookie is achieved through a set of rules as shown in the next example.

Example 3:

This example shows the use of two rules:

- Replace the beginning of the root up to the version (v1.0) with /ACME_PRD/BI/
- Add /extra_path/ to the end of the path

For example:

```
/mycompany/mobile/v1.0/Best_Practices_Templates -> /ACME_PRD/BI/api/mobile/Best_Practices_Templates/
extra_path
```

```
<cookieRewrite name="cookieRewrite-example" displayName="cookieRewrite-example" enabled="true"
>
    <cookieName>sessionId</cookieName>
    <rewriteRules on="path">
        <rule>
           <pattern>\/.*\/v1.0\/</pattern>                    <!--Matches the character strings
 up to  "v1.0"-->
          <replacement>/ACME_PRD/BI/api/mobile/</replacement>  <!--Replaces the found characters
 with /ACME_PRD/BI/ -->
        </rule>
        <rule>
           <pattern>$</pattern>                                <!--Matches the end of the path-
->
           <replacement>/extra_path</replacement>              <!--Replaces (actually appends)
 with /extra_path -->
        </rule>
    </rewriteRules>
</cookieRewrite>
```

Configuration

Element name	Deault	Presence	Type	Multiplicity
cookieName	n/a	Required	string	1
domain	n/a	Optional	string	0..1
path	n/a	Optional	string	0..1

<cookieRewrite> attributes

```
<cookieRewrite name="cookieRewrite-example" displayName="cookieRewrite-example" enabled="true"
version="1.0">
```

File name	Description	Default	Presence
name	Name of this policy instance.	N/A	Required
displayName			Optional
enabled	Indicates if a policy is enforced or not. If set to false, a policy is turned off, and not enforced.	true	Optional
version	Policy version.	N/A	Required

<cookieName> element

Use to specify the name of the cookie affected by this policy. The cookie name can be either a static string or a regular expression. A regular expression is denoted by forward slashes.

```
<cookieName>sessionId</cookieName>
```

Example using a regular expression:

```
<cookieName>/^sessionId.*/</cookieName>
```

<unsecureHttpTarget> element

If this element is placed in the request flow and the target uses http instead of https, this configuration element removes the secure flag from the given cookie.

```
<unsecureHttpTarget/>
```

<domain> element

This element is used to specify the desired string value for the cookie domain.

```
<domain>/myCompany</domain>
```

<path> element

This element is used to specify the desired string value for the cookie path.

```
<path>/myCompany/</path>
```

<rewriteRules> element

This element is used to specify the list of rules to apply to either the path of domain.

```
<rewriteRules on="path">
    <rule>
        <pattern>\/.*\/v1.0\/</pattern>                      <!--Matches the character strings
up to  "v1.0"-->
        <replacement>/ACME_PRD/BI/api/mobile/</replacement>  <!--Replaces the found characters
with /ACME_PRD/BI/ -->
    </rule>
    <rule>
        <pattern>$</pattern>                                 <!--Matches the end of the path-->
        <replacement>/extra_path</replacement>               <!--Replaces (actually appends)
with /extra_path -->
    </rule>
</rewriteRules>
```

Field name	Description	Default	Presence
on	Element of the cookie to which the rules apply - either path or domain.	N/a	Required

<rule> element

This element configures a rule to overwrite a cookie element.

<pattern> element

This element determines the regex pattern to match. Keep in mind that the regex expressions are evaluated in Javascript.

```
<pattern>\/.*\/v1.0\/</pattern> <!--Matches the character strings up to "v1.0"-->
```

Throttling

Use the Throttling policy to smooth the rate of requests or to arrest any spikes in the number of requests that may occur.

Example

In this example, Rate Smoothing is used to delay requests by one second after 5 requests in a minute have arrived. Also the Spike Arrest is set to reject, with a 429 status code, the 21st and greater requests in the same minute.

```
<throttling
        name="throttling-example" displayName="throttling-example" enabled="true" version="1.0"
       xmlns="http://www.infor.com/ion/api"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.infor.com/ion/api throttling.xsd">
 <timePeriodInMilliseconds>60000</timePeriodInMilliseconds> <!-- one minute -->
 <rateSmoothing>
  <delayAfterCount>5</delayAfterCount>
  <!-- start delaying requests after more then 5 have arrived during the same minute
  -->
  <delayFactorInMilliseconds>1000</delayFactorInMilliseconds> <!-- delay by a factor of 1 second
 -->
 </rateSmoothing>
 <spikeArrest>
  <maxRequestsPerPeriod>20</maxRequestsPerPeriod>
  <!-- Start rejecting with a 429 status requests 21, 22, etc. arriving during the same minute
→
 </spikeArrest>
</throttling>
```

Configuration

Element name	Deault	Presence	Type	Multiplicity
timePeriodInMilliseconds	n/a	Required	Positive integer	1
rateSmoothing	n/a	Optional	Complex	0..1
rateSmoothing.delayAfterCount	n/a	Required	Positive integer	1
rateSmoothing.delayFactorInMilliseconds	n/a	Required	Positive integer	1
spikeArrest	n/a	Optional	Complex	0..1

Element name	Deault	Presence	Type	Multiplicity
spikeArrest.maxRequestsPerPeriod	n/a	Required	Positive integer	1

<throttling> attributes

```
<throttling name="throttling-example" displayName="throttling-example" enabled="true" ver
sion="1.0”>
```

File name	Description	Default	Presence
Name	Name of this policy instance.	N/A	Required
displayName			Optional
enabled	Indicates if a policy is enforced or not. If set to false, a policy is turned off, and not enforced.	true	Optional
Version	Policy version.	N/A	Required

<timePeriodInMilliseconds> element

This is the time period within which to smooth the rate or arrest the spike.

```
<timePeriodInMilliseconds>60000</timePeriodInMilliseconds> <!-- one minute -->
```

<rateSmoothing> element

Specify this element to perform rate smoothing.

```
<rateSmoothing>
 <delayAfterCount>5</delayAfterCount>
 <!-- start delaying requests after 5 have arrived during the same minute -->
 <delayFactorInMilliseconds>1000</delayFactorInMilliseconds> <!-- delay by a factor of 1 second
 -->
</rateSmoothing>
```

<rateSmoothing.delayAfterCount> element

Sets the number of requests to accept in the time period before delaying any additional requests.

```
<delayAfterCount>5</delayAfterCount>
```

<rateSmoothing.delayFactorInMilliseconds> element

Sets the time (in ms) to delay any additional requests.

```
<delayFactorInMilliseconds>1000</delayFactorInMilliseconds>
```

<spikeArrest> element

Specify this element to perform Spike Arrest.

```
<spikeArrest>
 <maxRequestsPerPeriod>20</maxRequestsPerPeriod>
 <!-- Start rejecting with a 429 status requests 21, 22, etc. arriving during the same minute -
->
</spikeArrest>
```

<spikeArrest.maxRequestsPerPeriod> element

Sets the number of allowed requests in the time period before rejecting the next request with a 429 status code.

```
<maxRequestsPerPeriod>20</maxRequestsPerPeriod>
```

Transformation

The transformation policy is used to transform query and/or header parameter values to be received in the correct format. The transformation policy can be used at the endpoint level only, as a request or response.

Setting query parameters and/or headers using a transformation

Example:

```
<transformations>

        <transform kind="handlebars" outputType="application/xml"/>
```

Setting query-string parameters for a target API call

Assume your API Gateway endpoint usually is going to call this target API: `https://myserver/path?p1=v1&p2=v2`

The query-string parameters can be adjusted using a special transformation helper called `{SetReqQuery}`. For example, if you want to add an additional parameter of `p3=v3` to the target API call, you would add the following to your transformation: `SetReqQuery 'p3' 'v3'`

This causes the gateway to call the target using the following URL: `https://myserver/path?p1=v1&p2=v2&p3=v3`

Adding a query parameter with a constant value can also be done using the `QueryParam` policy. Adding a query parameter using a transformation comes from accessing the value from anywhere in the `POST` body or from any of several useful "request context" variables.

You can take a value from a `POST` payload and turn it into a query-parameter. For example, if our endpoint takes a `POST` JSON payload of the form:

Example

```
{   "customer": "Acme",   "creditLimit": 50000 }
```

You can use the following helper to add the `creditLimit` to the query-string:

Example

```
SetReqQuery 'cl' (jsonPathValue '$.creditLimit')
```

You can also take the value of a request header and pass it to the target as a query-parameter if your request includes a header called 'some-data' with a value of **`abc123`**.

We can send this to the target as a query-param called 'data' using this helper:

Example

```
SetReqQuery 'data' (requestVars 'request.header.some-data')
```

The gateway would call the target server using this URL: `https://myserver/path?p1=v1&p2=v2&data=abc123`

Setting headers for a target API call

There are two helpers for setting headers from a transformation. `SetReqHeader` is used for a request-side transformation to set a header passed to the target API server.

The other helper is called `SetResHeader`, is used on a response-side transformation to set a response header after the target server API has been called. This affects what headers the calling client will see as part of the API-call response.

SetReqHeader

For example, taking the value of the request header 'fred' and passing it as as 'wilma' to the target API:

Example

```
SetReqHeader 'wilma' (requestVars 'request.header.fred')
```

Suppose you do not want the 'fred' header to be passed to the target. In that case, you can apply a Header policy after the transformation policy to remove the 'fred' header (for example, remove it after the header was used by the transformation policy to set the 'wilma' header).

You could also transform a query-parameter into a header:

Example

```
SetReqHeader 'wilma' (requestVars 'request.queryparam.p1')
```

Headers from `POST` payloads also follow this process:

Example

```
SetReqHeader 'credit-limit' (jsonPathValue '$.creditLimit')
```

Note:

In addition to using these helpers to manipulate query-parameters and headers, you must make sure your transformation leaves the `POST` request or response payload.

A transformation policy that has only `SetReqQuery`, `SetReqHeader`, or `SetResHeader` helpers in it would leave the payload empty.

UserSecurityClaims

Use the UserSecurityClaims policy to set user claims in the header before the request is sent to the target.

Scopes

This policy can be attached to these Request Flow scopes:

- Proxy end point
- Proxy end point resource
- Target end point
- Target end point resource

Examples

In the following example, two headers are set with claim values:

- The roles header contains the security role values.
- The accounting-entity header contains the AccountingEntity claim value.

```
<user-security-claims continueOnError="false" displayName="Claim-type example Policy"
    enabled="true" name="Claim-type example policy" version="1"
    xmlns="http://www.infor.com/ion/api"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <claimTypes>
        <claimType name="http://schemas.infor.com/claims/SecurityRole" headerName="roles"/>
       <claimType name="http://schemas.infor.com/claims/AccountingEntity" headerName="accounting-
entity"/>
    </claimTypes>
</user-security-claims>
```

In the following example:

- The security roles are filtered by criteria.
- The email-address is added to the claim.

Filter roles and email address claim

```
<user-security-claims continueOnError="false" displayName="Claim-type example Policy" en
abled="true" name="Claim-type example policy" version="1.0" xmlns="http://www.infor.com/ion/api"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.in
for.com/ion/api userSecurityClaims.xsd">
<claimTypes>
<claimType name="http://schemas.infor.com/claims/SecurityRole" headerName="securityrole-claim">
<filter action="include">
<!-- action is one of include/exclude -->
<value>Infor-SuiteUser</value>
</filter>
<!-- match exact role -->
</claimType>
<claimType name="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress" headerName="x-
infor-identity2"/>
</claimTypes>
</user-security-claims>
```

Headers created

The value of the header is a comma-delimited string of values. For example, the roles header might contain a string with all the roles:user,administrator,superUser

Configuration

Element name	Default	Presence	Type	Multiplicity
claimTypes	N/A	Required	N/A	1
claimTypes.claim-Type	N/A	Required	N/A	1..*

<user-security-claims> attributes

<header name="Claim-type example Policy" displayName="Claim-type example Policy" enabled="true" version="1.0">

Field name	Description	Default	Presence
name	Name of this policy in-stance.	N/A	Required
displayName			Optional
enabled	Indicates if a policy is enforced or not. If set to false, a policy is turned off, and not enforced.	true	Optional
version	policy version	N/A	Required

<claimTypes> element

This is a required element that indicates the list of claims that must be set to the target.

```
<claimTypes>
    <claimType name="http://schemas.infor.com/claims/SecurityRole" headerName="roles"/>
    <claimType name="http://schemas.infor.com/claims/AccountingEntity" headerName="accounting-
entity"/>
</claimTypes>
```

<claimType> element

This is a required element for each requested claim.

Possible claim values:

- http://schemas.infor.com/claims/SecurityRole
- http://schemas.infor.com/claims/AccountingEntity
- http://schemas.infor.com/claims/ErpPersonId
- http://schemas.infor.com/claims/Location

```
<claimType name="http://schemas.infor.com/claims/SecurityRole" headerName="roles"/>
```

Target timeout

The target timeout defines how long API Gateway waits for a response from the target server before returning a timeout error. You can configure this value to accommodate longer-running requests when needed.

By default, API Gateway waits up to 60 seconds for a response from the target server. If the target server does not respond within this time frame, Gateway returns a `504 Timeout` status, indicating that the request cannot be completed due to a timeout.

You can modify this behavior by applying a timeout policy. This policy allows you to configure the Gateway to wait longer for a response—up to five minutes, if required. Adjusting the timeout value is useful in scenarios where the target server needs additional time to process and respond to requests.

Increasing the timeout ensures that longer-running operations are given enough time to complete without prematurely returning a timeout error.

To increase the timeout, update the policy settings for your API endpoints.

Target timeout policy scope

The target timeout policy can be applied at different scopes within the ION API Gateway to control how it affects request processing.

You can apply the target timeout policy to various scopes within the API Gateway to manage how requests are processed and how long the Gateway waits for a response from the target server.

The policy can be applied in these scopes:

- Request flow: Apply the policy to the request as the Gateway sends calls to the target server. This setting controls how long the Gateway waits for a response during request processing.
- Proxy endpoint: Apply the policy at the proxy endpoint level to set a target timeout for a specific endpoint in an API suite.

Target timeout policy example

This example shows how to configure a target timeout policy for an API endpoint in the ION API Gateway.

The example demonstrates how to configure a target timeout policy for an API endpoint in the API Gateway. This policy defines the duration that the Gateway waits for a response from the target server before timing out. In this example, the timeout is set to `120000` milliseconds, which equals two minutes. If the target server does not respond within that time, the Gateway stops waiting and returns a timeout error.

Example policy schema:

```
      <targetTimeout xmlns="http://www.infor.com/ion/api"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.infor.com/ion/api targetTimeout.xsd"
      version="1.0" name="targetTimeout-example" displayName="targetTimeout-example" en
abled="true">
      <timeout>120000</timeout> <!-- Milliseconds, Min 60000, Max 300000 -->
      </targetTimeout>
```

Target timeout elements

The target timeout policy includes several elements and attributes that define how the ION API Gateway handles response time from a target server.

The target timeout policy contains these key elements and attributes:

- `targetTimeout`: The root element of the policy. It includes attributes such as `version`, `name`, `displayName`, and `enabled` that describe and activate the policy.
- `timeout`: Specifies the timeout value in milliseconds that the API Gateway uses. In the example, it is set to `120000` milliseconds (two minutes). You can set a value between `60000` milliseconds (one minute) and `300000` milliseconds (five minutes), depending on how long you want the Gateway to wait for a response.
- `xmlns`, `xmlns:xsi`, and `xsi:schemaLocation`: These attributes define XML namespaces and specify the location of the schema file. They ensure that the policy configuration is valid and correctly interpreted by the Gateway.

Adjusting the timeout value in this policy allows you to control how long the API Gateway waits for a response from the target server. A longer timeout is useful when requests require more time to process, but setting it too high can lead to inefficient resource usage or delayed error handling.

Appendix B: Maintenance window

During planned or unplanned maintenance windows, the APIs return a standard response indicating the application is under maintenance.

The response will help API consumers identify maintenance apart from errors.

When applications are in maintenance mode, API Gateway displays the following response:

- HTTP Status - 503
- HTTP Header retry-after : <endTime in UTC datetime format > (https://www.ietf.org/rfc/rfc2616.txt)
- Response body:

```
{
"error":"This product is currently under maintenance. Normal operations are expected to resume
 on Friday, May 8 2020 8PM.",
"startTime" : "UTC datetime stamp",
"endTime" : "UTC datetime stamp"
}
```

Appendix C: OAuth2 scopes

OAuth2 scopes are an industry standard that provides a layer of additional authorization. They are designed to govern an authorized application's API access based on a preconfigured list of scopes permitted for the authorized application/client, as well as resource/owner consent during authorization.

Oauth2 scopes adoption by API Gateway (Infor suites and Infor/non-Infor authorized apps)

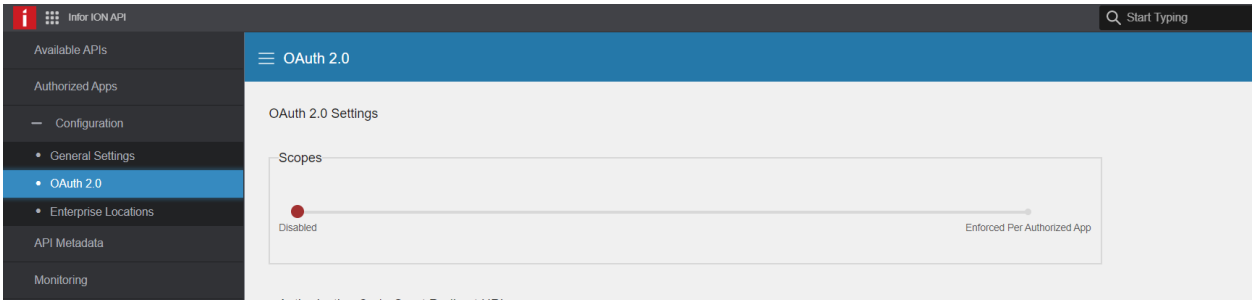
With the 2020-06 release of Infor OS CE, all API suites and authorized apps of the Infor OS platform are scopes compatible. Configuration settings for OAuth2.0 scopes are visible, but this configuration applies to the API suites of Infor OS and authorized apps using Infor OS APIs. There is no impact on authorized apps belonging to other Infor cloud suites or customers.

The scopes feature is kept **OFF** by default to maintain backward compatibility. A tenant administrator must opt in to use scopes.

Note: For custom application/backend service apps, when the tenant enables scopes, all custom apps created by the tenant (and the ION backend service app) do not participate in scopes. Using scopes is due to precautions such as assigning scopes to service accounts in IFS or modifying the web-mobile application code. Tenants can enable scopes for these authorized apps at the app level after the necessary precautions are taken.

Configuring OAuth2 settings in API Gateway

As the tenant administrator, you can use the **Scopes** setting in the **Configuration** section of the API Gateway administration user interface.



This setting has two levels:

- **`Disabled`**: This is the default state. This means that no OAuth2 scopes are enforced for any authorized app. The API from all clients to Infor OS API suites continues to work as before. Also, should anything go wrong with enabling scopes, the customer can always switch back to OFF.
- **`Enforced`**: All calls to Infor OS API suites, regardless of the caller, will be enforced for scopes check. Since not all suites and apps of a given tenant are scope-enabled, this option is kept disabled. This option will be enabled when all suites and apps are capable of working with scopes.

Adding scopes for authorized apps or service accounts

This section is applicable for the ION Backend Service authorized app and any authorized app created by the customer that uses an API from the Infor OS API suite.

These authorized apps are exempted from any scope check by default to maintain backward compatibility; however, if the global **Scopes** setting is set to **`Enforced`**, you have the option to opt in and use scopes for additional security. You can opt in using the process described in Using a backend service to opt into using scopes on page 126.

For webapp, mobile, and desktop clients:

- The authorized app must have the required scopes associated to request that scope
- The scope must be explicitly included in the client app's authorization request.
- The user must consent to granting that scope.

For Backend clients:

- The Service Account generated on behalf of the user must have the scopes associated with them
- The scope must be explicitly included in the client app's token request.

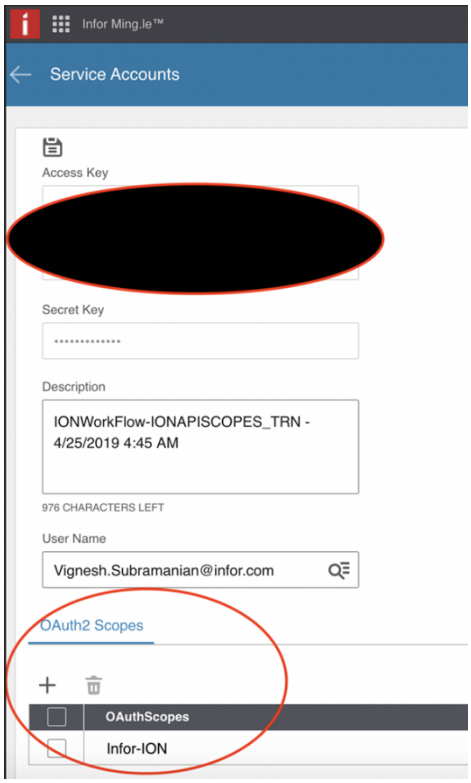
For a detailed explanation on OAuth Scopes management with examples, see KB https://inforsaas.service-now.com/kb?id=kb_article_view&sysparm_article=KB3537918.

Using a backend service to opt into using scopes

To use a backend service to opt into using scopes when the global **Scopes** setting is **`Enforced`**:

1. Prepare your authorized app for scopes. This process is dependent on the type of app being used.
2. Navigate to IFS.
3. Select the service accounts used by your backend service.
4. Attach the scopes required for your Backend Service to the service account.

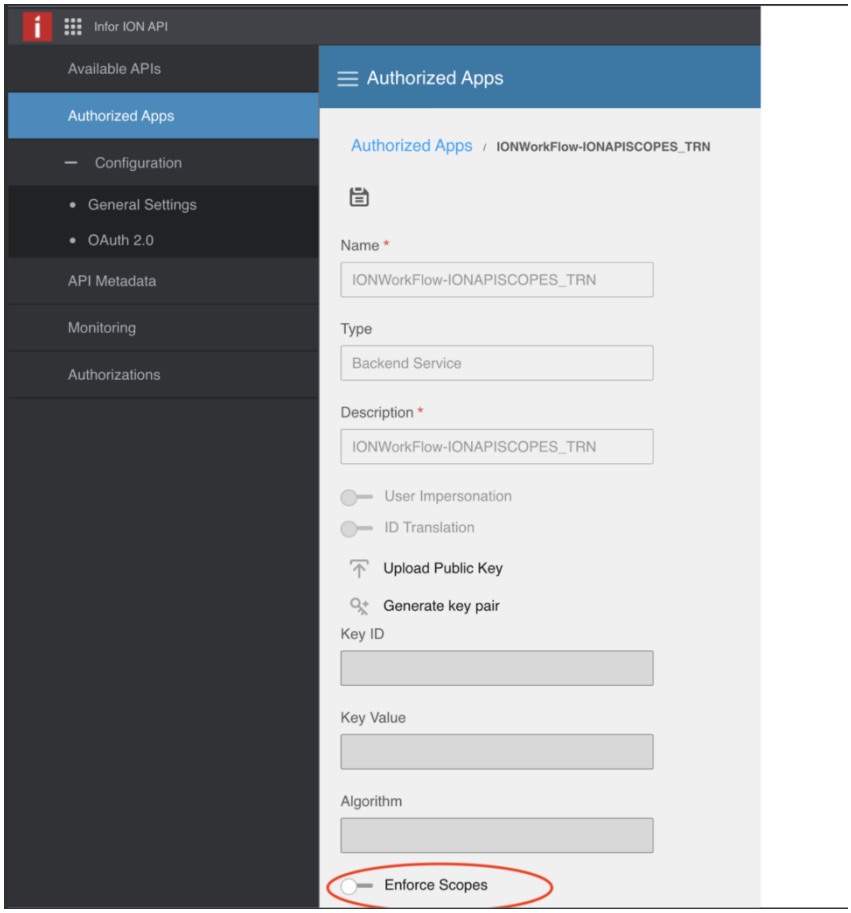
   Note: For ION, all service accounts used from this app must have scopes enabled. In the case of ION, this means that all service accounts used in all documentation and workflows must be modified to include ION in their scopes.



Using a mobile, web, or native application to opt into using scopes

To use a mobile, web, or native application to opt into using scopes when the global **Scopes** setting is **`Enforced`**:

1. Prepare your authorized app for scopes. This process is dependent on the type of app being used.
2. Modify the code and include the appropriate scopes in the request.
3. Enable **Enforce Scopes** at the app level:
   a. Navigate to the Infor API Gateway user interface.
   b. Click **Authorized Apps**.
   c. Select your authorized app.
   d. Turn on **Enforce Scopes**.



Additional scope-related items to consider while developing authorized apps

Depending on the type of grant used in the application, these scenarios must be considered:

For applications using OAuth2 SAML bearer grant

- The SAML assertion includes allowed API Gateway scopes for the client based on the Infor Registry configuration for the client application type.

  For example, the Infor Homepages application type has allowed the `Infor-homepages-all` scope associated in the Infor Registry. So, SAML assertion issued to homepages includes the `Infor-homepages-all` scope.
- If scopes associated with the access token do not match the scope required by the suite, API Gateway returns an HTTP 403 error to the client.

For applications using OAuth2 Authorization code grant

- When the client issues an authorization request, the client sends a list of space-delimited scopes. For example:
  - Request Infor-Mingle and Infor-IDM scopes
  - Infor Ming.le mobile app requesting openid and Infor-Mingle scopes
- The authorization server will ask for consent from the user to access the API depending on the requested scopes and grants approval for those appropriate scopes.
- If scopes associated with the access token do not match the suite's scope, API Gateway returns an HTTP 403 error to the client.

For applications using OAuth2 implicit grant

- Associate required scope/s with the authorized app in Infor registry or the API Gateway user interface.
- Scopes associated with authorized apps are provided in the QR code/.ionapi file. The authorized app must support the new .ionapi file format.
- While making an authorization request, send the list of scopes corresponding to suites that the authorized app must access.

For backend applications using resource owner grant

- You can associate required scopes with the IFS Service Account configured for the backend service.

For headless applications using device authorization grant

- When the headless app initiates device authorization, it can optionally send scopes allowed for the app.

Best practices

These are the best practices for managing the OAuth 2.0 token:

- Obtain an access token when you access the API for first time or when the existing access token expires.
- When the access token is issued, its “expires_in” time is provided as well. Use this as a guideline for keeping the access token.

  For example, if the access token expires in 2 hours, continue to use it for 1h55m. Then, obtain a new token when the current token expires or is revoked.
- Obtaining an access token for each API call is an anti-pattern and must be avoided.
- If you receive a refresh token along with an access token, use the refresh token to refresh the access token when the access token expires.
- For a clustered application, there should be a common OAuth2 token management layer that obtains and securely stores the tokens.

Appendix D: Client credentials grant

The client credentials grant type is an OAuth 2.0 authorization flow used for machine-to-machine authentication between applications.

Overview

The client credentials grant type is an OAuth 2.0 authorization flow used for machine-to-machine (M2M) authentication. In this flow, an application, referred to as the client, authenticates directly with an authorization server by using its own credentials, typically a client ID and client secret, to obtain an access token. Unlike other OAuth flows, it does not involve users or their service accounts. It is suited for backend services or applications that access resources or APIs on their own behalf rather than on behalf of a user.

Business scenarios and use cases

The client credentials grant type supports a variety of business scenarios that require secure system-to-system communication without user interaction. Common examples include:

- Automated data processing, such as backend services retrieving third-party API data without user involvement
- Microservices communication within an organization, enabling secure interservice calls
- Enterprise integration, connecting systems such as CRM and marketing platforms through server-to-server authentication
- Batch jobs and scheduled tasks, including data synchronization or report generation without user input
- DevOps and automation tools, allowing CI/CD pipelines or monitoring utilities to securely access APIs

Using API Gateway to obtain tokens with the client credentials grant

Use the client credentials grant flow in API Gateway to authenticate backend services and obtain access tokens for secure, automated API communication.

API Gateway supports the client credentials grant flow, an OAuth 2.0 authorization method for system-to-system communication.

To use this flow, create a backend service authorized application, download its credentials, and use those credentials in an API client to request an access token. The token provides access to protected resources through API Gateway.

Creating a backend service authorized app

Create a backend service authorized application to enable secure machine-to-machine authentication using the client credentials grant flow. This app provides the client ID and client secret required to obtain access tokens from API Gateway.

1 Navigate to **OS > API Gateway > Authorized Apps**.
2 Click **Add** to create a new authorized app.
3 Specify this information:

Name
Enter a unique name for the authorized app. This field is required.

Type
Select **`Backend Service`**. This field is required.

Description
Provide a brief description of the app's purpose. This field is required.

Grant Type
Expand the list and select **`Client Credentials`**. This field is required.

4 Click **Save**.

Downloading the backend service authorized app credentials

Download the backend service authorized app credential file to obtain the client ID and client secret used for token generation. Store this credential securely because it includes sensitive authentication details for accessing protected APIs through the client credentials grant flow.

1 Navigate to **OS > API Gateway > Authorized Apps**.
2 Select the backend service authorized app from the list.
3 Click **Download Credentials**.
4 Save the `.ionapi` credential file in a secure location.

Using the .ionapi credential to generate a Gateway token

Use the `.ionapi` credential file to authenticate and obtain an access token from API Gateway using the client credentials grant flow. The token enables secure communication between backend services and the API Gateway and requires a valid client ID, client secret, and token URL.

1 Open Postman or another API client.
2 Open the `.ionapi` file and view these attributes:

Token URL
Combine the **pu** and **ot** values.

Client ID
Value of **ci**.

Client Secret
Value of **cs**.

3 In the API request, specify the **Token URL**.
4 In the request body, include the **client_id**, **client_secret**, and **grant_type** values. Set **grant_type** to `client_credentials`.
5 Send the request and retrieve the access token from the response.
6 Optionally, validate the token using a JWT evaluation tool.

Best practices for applications using the client credentials grant type

Follow these best practices to ensure secure, efficient, and compliant use of the client credentials grant type in API Gateway and related integrations.

Implement these recommendations when configuring or managing applications that use the client credentials grant type:

- Use the client credentials grant only for machine-to-machine communication where user context is not required.
- Store and manage client IDs and secrets securely and rotate credentials on a regular basis.
- Limit the scope and permissions granted to each client to follow the principle of least privilege.
- Monitor and log all token requests and access events for auditing and anomaly detection.
- Validate the grant type and client credentials strictly on both authorization and resource servers.
- Ensure that APIs do not rely on user-specific information when using this grant type.
- Review and update access controls regularly to align with evolving security requirements.

Troubleshooting issues with the client credentials grant type

This section describes common issues that may be encountered when using the client credentials grant type in API Gateway and provides troubleshooting guidance for resolving them.

Policies not working as expected

The client credentials grant type does not have an associated service account. Therefore, certain API Gateway policies that depend on user association may return empty values or behave differently.

- **Header Policy**: When **identity2** information is requested, the response is empty.
- **Quota Policy**: When the **userlevel** attribute is set to `true`, the policy does not apply correctly, and the quota is enforced at the tenant level.
- **UserSecurityClaims Policy**: This policy does not return results because no user is associated with the token.

Unsupported features

Some features are disabled for backend service authorized applications that use the client credentials grant type.

- **IONAPI Bridge** functionality is not supported. The following controls are disabled:
  - User Impersonation
  - ID Translation
  - Upload Public Key
  - Generate Key Pair
- Refresh tokens are not supported. The **Issue Refresh Tokens** toggle is disabled.

Invalid grant type error

Token generation may fail if the grant type of the authorized application does not match the grant type specified in the token request. Verify that both configurations use the same grant type value.

Target API authentication failure

Some target API servers enforce additional user-based authentication. These APIs may reject requests made with the client credentials grant type, even when the same request passes authentication in the API Gateway. Ensure that the correct grant type is selected based on the target server's authentication requirements.

Appendix E: Auditing and monitoring support for API Gateway

Use the auditing and monitoring tools in Infor OS Security to track user activity, detect unauthorized access, and maintain records of system events.

The auditing and monitoring tools are essential for maintaining the security, integrity, and compliance of system operations. The audit and monitoring event search enables security administrators to review user activity, detect unauthorized access, and track critical events within the application.

When auditing and monitoring are enabled, API Gateway publishes all audit events to the Auditing and Monitoring Service. Administrators can then search for audit events related to API Gateway to identify potential issues, respond to incidents, and meet regulatory or internal compliance requirements.

For details about the audit event search in Infor OS Security, see the *Infor OS Portal User and Administration Library* and select **Security > Audit Event Search**.

Viewing audit events for API Gateway

Use the Auditing and Monitoring functionality in Infor OS Security to search for and view audit transactions for API Gateway. Reviewing these events helps verify configuration changes, monitor administrative actions, and maintain compliance with security and audit requirements.

To view audit events, your user must have one of these roles: `FS-AuditReportUser`, `IFS-AuditAdministrator`, or `Infor-SystemAdministrator`.

1. Select **OS > Security**.
2. Expand the side menu.
3. Select **Auditing and Monitoring > Audit Event Search**.
4. In the **Date and Time Range** field, select **`Last 60 minutes`**.
5. In the dropdown next to **Logical ID**, select `lid://infor.ionapi.ionapi`.
6. Click **Search**.
7. View the list of audit events for API Gateway that occurred within the last hour.
8. Select an audit event to view its details.

The selected audit event displays detailed information, including:

- Tenant ID: The tenant where the change occurred.

- Log time: The date and time when the change occurred.
- Entity type: The resource that was changed, such as Authorized App or Configuration.
- Session ID: A unique identifier for the event.
- Outcome: Indicates whether the event was successful or failed.
- Entity ID: The name and description of the entity that changed.
- Event creator: The email address of the user who performed the action, if available.
- Audit event type: The type of audit event, usually Application.
- Creator permissions: The roles assigned to the user.
- Severity: The severity level of the update, usually High.
- Transaction name: The specific name of the audit event transaction.
- Logical ID: The logical ID of the application, typically `lid://infor.ionapi.ionapi`.
- Before update and After update: A side-by-side comparison showing what changed.

Audited API Gateway events

API Gateway audits all actions that modify its configuration or operational data. This topic lists the API Gateway events that are tracked and recorded as audit entries.

Audit events are generated whenever an action changes the current state of API Gateway data. Examples include updating an API suite name, adding a policy to an endpoint, downloading the credentials of an authorized application, or changing TLS configuration settings.

The table lists all event types that are audited in API Gateway.

Entity type	Audit transaction	Description
API Suite	Activate or Deactivate API Suite	Audits when an API Suite is activated or deactivated.
API Suite	Create Infor Non-Provisioned API Suite	Audits creation of an Infor Non-Provisioned API Suite.
API Suite	Create Non-Infor Suite	Audits creation of a Non-Infor API Suite.
API Suite	Delete API Suite	Audits deletion of an API Suite, including Non-Infor, Non-Provisioned, or Third-Party Suites.
API Suite	Export Non-Infor Suite	Audits when a Non-Infor Suite is exported.
API Suite	Import Non-Infor Suite	Audits when a Non-Infor Suite is imported.
API Suite	Register IRN With Namespace	Audits when a namespace is added to an API Suite.
API Suite	Update API Suite	Audits when an API Suite is updated, such as when a description or policy is modified.
API Suite	Update Suite Tracing	Audits when tracing is enabled for a Suite in Monitoring.

Entity type	Audit transaction	Description
API Suite Deployment	Update Deployment Profile	Audits when a deployment profile is added or updated for a Third-Party or Infor Non-Provisioned API Suite.
API Suite Proxy Endpoint	Create Non-Infor Endpoint	Audits creation of a proxy endpoint.
API Suite Proxy Endpoint	Create, Update, or Delete API Policy	Audits creation, modification, or deletion of an API policy.
API Suite Proxy Endpoint	Enable or Disable Endpoint Reindexing	Audits when endpoint reindexing is started or stopped.
API Suite Proxy Endpoint	Update Non-Infor Endpoint	Audits updates to the endpoint description, proxy URL, or target authentication settings.
API Suite Proxy Endpoint	Create, Delete, or Update Documentation	Audits upload, update, or deletion of endpoint documentation.
Authorized Application	Create, Delete, or Update Authorized Application	Audits creation, modification, or deletion of an authorized application.
Authorized Application	Download Authorized Application Credentials	Audits when a user downloads a credential file.
Authorized Application	Enable or Disable Authorized Application	Audits when a user enables or disables an authorized application.
Authorized Application	Lock or Unlock Authorized Application	Audits when a user locks or unlocks an authorized application.
Authorized Application	Register IRN with Namespace	Audits when a namespace is added to an authorized application.
Authorized Application	Reset Secret Key	Audits when a user resets the secret key.
Authorized Application	Reset Secret Key with Smooth Rollover	Audits when a user resets the secret while keeping the previous key valid during rollover.
Authorized Application	Revoke Old Secret	Audits when a user revokes the old secret key.
Configuration	Create Associations to Namespace	Audits when a user associates configurations with a global package namespace.
Configuration	Create Associations to Tag	Audits when a user associates configurations with a local package tag.
Configuration	Create or Delete JWK Key Pair	Audits when a user creates or deletes a JWK key pair for ION API Bridge authentication.
Configuration	Update Include Target Endpoint Credentials Flag	Audits when a user enables or disables the option to export target security credentials in API Suite exports.

Entity type	Audit transaction	Description
Configuration	Update OAuth2 Scopes Flag	Audits when a user enables or disables the OAuth2 scopes option for authorized applications.
Configuration	Update TLS Version	Audits when a user changes the minimum TLS version setting.
Custom Scopes	Create Custom Scope	Audits when a custom scope is created.
Custom Scopes	Create, Update, or Delete Scope Mapping	Audits when custom scopes are associated or updated with scope mappings.
Metadata	Reindexing API Metadata	Audits when a user starts an API metadata refresh.
Monitoring	Update Detailed Logging	Audits when a user enables detailed logging for a Non-Infor API Suite.
Monitoring	Download Detailed Logging	Audits when a user downloads a detailed transaction log.
BaaS Service	Export or Redeploy Service	Audits when a BaaS service is exported or redeployed.
Authorizations	Authenticate or Revoke Authorization Code	Audits when a user authorizes or revokes API authorizations.