



Informatica® API Center
October 2025

API Policies

© Copyright Informatica LLC 2022, 2025

This software and documentation are provided only under a separate license agreement containing restrictions on use and disclosure. No part of this document may be reproduced or transmitted in any form, by any means (electronic, photocopying, recording or otherwise) without prior consent of Informatica LLC.

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation is subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License.

Informatica, Informatica Cloud, Informatica Intelligent Cloud Services, PowerCenter, PowerExchange, and the Informatica logo are trademarks or registered trademarks of Informatica LLC in the United States and many jurisdictions throughout the world. A current list of Informatica trademarks is available on the web at <https://www.informatica.com/trademarks.html>. Other company and product names may be trade names or trademarks of their respective owners.

Portions of this software and/or documentation are subject to copyright held by third parties. Required third party notices are included with the product.

The information in this documentation is subject to change without notice. If you find any problems in this documentation, report them to us at infa_documentation@informatica.com.

Informatica products are warranted according to the terms and conditions of the agreements under which they are provided. INFORMATICA PROVIDES THE INFORMATION IN THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND ANY WARRANTY OR CONDITION OF NON-INFRINGEMENT.

Publication Date: 2025-10-03

Table of Contents

Preface	5
Chapter 1: API policies.....	6
Chapter 2: Security policies.....	7
Basic authentication.	7
OAuth 2.0 authentication.	7
Creating an OAuth 2.0 client.	8
Managing OAuth 2.0 clients.	9
Downloading an OAuth 2.0 clients list.	9
Enabling and disabling OAuth 2.0 clients.	10
Accessing an API with OAuth 2.0 authentication.	10
Regenerating an OAuth 2.0 client secret.	11
Rules and guidelines for OAuth 2.0 authentication.	12
JSON web token authentication.	12
Generating a JSON web token.	12
API-level authentication policy.	13
Invoking an API with JSON web token authentication.	13
Session ID authentication.	13
Creating a security policy.	14
Chapter 3: CORS policies.....	15
CORS policy configuration.	16
Creating a CORS policy.	16
Creating a CORS group.	16
Enabling a CORS policy.	17
Editing a CORS policy.	18
Disabling a CORS policy.	18
Deleting a CORS policy.	18
Editing a CORS group.	18
Deleting a CORS group.	19
Rules and guidelines for CORS policies.	19
Chapter 4: Rate limit policies.....	20
Creating a rate limit policy.	20
Organization level rate limit policy.	21
User-level rate limit policy.	21
Creating a user-level rate limit policy.	22

Chapter 5: Response caching policies.....	24
Creating a response caching policy.	24
Chapter 6: Privacy policies.....	25
Creating a Personally Identifiable Information (PII) policy.	25
Chapter 7: IP filtering policies.....	27
API-specific IP filtering policy.	27
Creating an IP filtering policy.	28
Domain-level IP filtering.	28
Creating an IP filtering policy at the domain level.	28
Chapter 8: Third-party authentication and authorization.....	30
Federated or third-party authorization server support with OAuth 2.0 server.	30
Register an authorization server.	31
Edit an authorization server.	32
Delete an authorization server.	32
Define user permission.	32
Index.	33

Preface

Read *API Policies* to learn how you can use policies to enforce API security and control access to APIs.

CHAPTER 1

API policies

API policies are rules that you can create to enforce API security and control access to APIs.

You can use API Center to define and assign the following types of policies:

- Security. A security policy defines authentication methods that can be used to access an operation.
- CORS. A Cross-Origin Resource Sharing (CORS) policy to ensure that your APIs can be securely accessed by client applications from different domains.
- Operational. Operational policies include rate limit policies and response caching policies for an operation. A rate limit policy defines the number of times API consumers can invoke an operation during a designated time frame. A response caching policy defines how long API Center stores API responses for an operation in the cache.
- Privacy. A privacy policy defines which Personally Identifiable Information (PII) is sensitive data that API Center protects for an API or operation.
- IP filtering. An IP filtering policy defines access rules for a managed API.

For example, you can assign a basic authentication policy and a rate limit policy of three calls per minute to a specific operation in order to control API consumer access to the operation.

You can assign IP filtering, security, and privacy policies at the API level. You can assign security, operational, and privacy policies at the API operation level. Operation policies take precedence over API policies.

To create, edit, enable, delete, and disable policies, you must be assigned an API Policy Manager or Administrator role. To assign security, operational, and privacy policies and view policy details, you must be assigned the Deployer or Designer role. To assign an IP filtering policy, you must be assigned the Deployer role.

You can't create a policy in disabled state. You can disable a policy that is assigned to an API. You can't delete a policy that is assigned to an API.

CHAPTER 2

Security policies

Security policies are rules that define the authentication methods that API consumers must use when they access an operation. When you create a security policy, the default authentication method is basic. You can change the authentication method of the policy or add more authentication methods.

You can select one or more of the following authentication methods for a security policy:

- **Anonymous.** API consumers access the operation without having to enter a user name or password. You can't use anonymous authentication with any other authentication method.
- **Basic.** API consumers access the operation with an Informatica Intelligent Cloud Services user name and password.
- **OAuth 2.0.** You create an OAuth 2.0 client. API consumers use the client credentials to generate an OAuth 2.0 authorization token that they use to access the operation. API Center uses the client credentials grant type for OAuth 2.0 authentication.
- **JSON web token (JWT).** You generate a JSON web token that API consumers use to access the operation.
- **Session ID.** API consumers access their logged-in session by using a unique session ID to securely authenticate and authorize API requests.

You define and assign security policies when you create APIs and operations. You can use an existing security policy or create a new security policy. When you deploy an API, you can override the policies that are assigned to it.

The **API References** area of an authentication policy displays all the designed APIs that use the policy.

Basic authentication

You can select the basic authentication method as a security policy to assign to APIs or operations.

To access an API or an operation that requires basic authentication, API consumers authenticate to the API with an Informatica Intelligent Cloud Services user name and password.

OAuth 2.0 authentication

OAuth 2.0 is a protocol for authorization that provides specific authorization flows for web applications and helps in the secure transmission of information between API consumers and web services. You can create an

OAuth 2.0 authentication policy to assign to APIs or operations that invoke a process that uses basic authentication.

API Center uses the client credentials grant type for OAuth 2.0 authentication. To enable OAuth 2.0 authentication, you must create an OAuth 2.0 client. You specify the credentials of an organization user with access to run managed APIs and managed API groups that can use the client for authentication, and generate client credentials.

Client access tokens that you use for OAuth 2.0 authentication time out after a defined timeout period. After a token times out, you can't use it. You must regenerate the token. You set the timeout when you create the OAuth 2.0 client, and you can change it later.

You can now assign multiple managed APIs and managed API groups to a single OAuth 2.0 client. You can create a single OAuth 2.0 client to grant access to specific API sets, simplifying authorization control. Disabling or deleting an OAuth 2.0 client invalidates its authorization token, preventing unauthorized API access.

Creating an OAuth 2.0 client

Create an OAuth 2.0 client that enables API consumers to access APIs with a security policy that uses OAuth 2.0 authentication.

1. On the **Configuration** page, click the **Authorization** tab.
2. On the **Informatica OAuth 2.0 Server** tab, click **Add OAuth 2.0 Client**.
The **Add OAuth 2.0 Client** wizard appears.
3. Enter the user name and password of an organization user and click **Next**.
The user information that you enter here must be an **Allowed User** of the **Application Integration** process.
The **Details** step appears.
4. Enter a name for the client. The name is case sensitive and must be unique in the organization.
The name can contain up to 32 characters, including ASCII letters, digits, Japanese characters, and the following special characters: \$ () [] . ? `
5. Optionally, enter a description of the client.
The description can contain up to 255 characters.
6. Enter a timeout value in minutes for the access token.
The minimum value is 5 minutes and the maximum value is 1440 minutes or 24 hours.
After a token times out, you can't use it. You must regenerate the token.
7. Click **Next**.
The **Resources** step appears.
8. Select what managed APIs or managed API groups the OAuth 2.0 client applies to:
 - All Resources. The OAuth 2.0 client applies to all the managed APIs and managed API groups in the organization.
 - Selected Resources. Select managed APIs and/or managed API groups that the OAuth 2.0 client applies to.The managed APIs and managed API groups are organized into distinct tabs, making it easier to locate and interact with the specific resources you need.
9. Click **Next**.
API Center creates the client. The **Generated Credentials** step appears.

10. Copy the client credentials and use one of the following methods to send them to API consumers:
 - Click **Copy** next to **OAuth 2.0 Client ID** and **OAuth 2.0 Client Secret** to copy the credentials as plain text. API consumers use the client credentials in applications and software packages where you enter each detail separately.
Note: You can't copy the client secret after you exit the wizard.
 - Click **Copy** next to **Authorization Header Value** to copy the credentials as an authorization header value. API consumers use the value in applications and software packages where you enter the client credentials as a value in an authorization header.
Note: You can't use the `DOCTYPE` header in XML attachments.
11. Click **Finish**.

You can view the enabled OAuth 2.0 client on the **Informatica OAuth 2.0 Server** table on the **Authorization** tab of the **Configuration** page.
12. Click the number in the **APIs** or **API Groups** columns to view the details of the managed APIs or managed API groups associated with the respective OAuth 2.0 client.
All APIs or **All API Groups** in these columns denote that the specific OAuth 2.0 client is associated with all managed APIs or managed API groups.
You can filter the managed APIs and managed API groups based on the API name.
When you create an OAuth 2.0 client, you can view all the managed APIs and managed API groups that are currently active, shared, and deprecated. When you edit an OAuth 2.0 client, you can view the list of active, shared, and deprecated managed APIs and managed API groups that are associated with the OAuth 2.0 client.

Managing OAuth 2.0 clients

After you create an OAuth 2.0 client, you can edit or delete it.

1. On the **Informatica OAuth 2.0 Server** table on the **Authorization** tab of the **Configuration** page, click to open the **Actions** menu of the OAuth 2.0 client.
2. Perform one of the following tasks:
 - To edit an OAuth 2.0 client, select **Edit**. In the **Edit OAuth 2.0 Client** window, make the required changes and click **Finish**.
 - To delete an OAuth 2.0 client, select **Delete**. In the **Delete Client** window, click **OK**.

Downloading an OAuth 2.0 clients list

You can download a list of OAuth 2.0 clients.

API Center downloads an Excel file that shows details of the OAuth 2.0 clients that were created since the time the organization was created. The file shows the client name, client description, status, the APIs that the client applies to, and the last updated time for all the OAuth 2.0 clients.

To download the list, on the **Informatica OAuth 2.0 Server** table on the **Authorization** tab of the **Configuration** page, click **Download**.

Enabling and disabling OAuth 2.0 clients

When you create an OAuth 2.0 client, it is enabled by default. You can disable the client if needed. API consumers can't use disabled OAuth 2.0 clients for authentication.

1. On the **Informatica OAuth 2.0 Server** table on the **Authorization** tab of the **Configuration** page, click to open the **Actions** menu of the OAuth 2.0 client.
2. Select **Disable** or **Enable**.

The OAuth 2.0 client is disabled or enabled.

Accessing an API with OAuth 2.0 authentication

To access an API, API consumers generate an OAuth 2.0 authorization token and send the token to the API.

The following sections describe the stages of accessing an API that requires OAuth 2.0 authentication:

Generating an OAuth 2.0 authorization token

To generate the token, API consumers authenticate to the IDMC OAuth 2.0 server using the server URL and the OAuth 2.0 client credentials that you send to the API Portal administrator.

API consumers need to submit the following credential information to the OAuth 2.0 server:

- `client_id=<client_name>`
- `client_secret=<client_secret>`
- `grant_type=<client_credentials>`

You can submit this information using any of the following methods. Choose the method that works best based on the application or software package that you use to invoke the API.

- Method 1. Enter the **client_id** and **client_secret** in a Basic Authorization header. For the **grant_type**, add these to the request body as URL-encoded values. To do this, select **client_credentials** and add this to the request body as URL-encoded data, and enter the URL in the **Access Token URL** field.
- Method 2. Enter the **client_id** and **client_secret** in a Basic Authorization header. For the **grant_type**, add these to the URL parameter.
- Method 3. Enter the **client_id**, **client_secret**, and **grant_type** all within the request body as URL-encoded values.

For example, in Postman enter the details as follows and add them to the request body as URL-encoded data:

Key	Value
client_id	<client_name>
client_secret	<client_secret>
grant_type	<client_credentials>

You can find the access token URLs on the **Authorization** tab in the **Policies** page.

An access token POST URL uses the following format:

```
{protocol}://{host_URL}/authz-service/oauth/token
```

The following image shows a sample POST URL and other details:

POST ▼ https://[redacted]/authz-service/oauth/token

Params ● Authorization ● Headers (10) ● Body ● Scripts ● Settings

☐ none ☐ form-data ☒ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

Key	Value	Description
<input checked="" type="checkbox"/> client_id	[redacted]	
<input checked="" type="checkbox"/> client_secret	[redacted]	
<input checked="" type="checkbox"/> grant_type	client_credentials	

- Method 4. Enter the **client_id** and **client_secret** to the request body as URL-encoded data. For the **grant_type**, add these to the URL parameter.

Other combinations such as passing all three parameters in URL are supported but not recommended for security reasons.

Note: If the parameters are duplicated and passed both in the URL and request body, the request body parameters override the URL parameters.

The following image shows an API invocation through Postman with a Basic Authorization header:

KEY	VALUE
<input checked="" type="checkbox"/> Authorization	Basic 4879857439857349857

Sending the token to the managed API

API consumers pass the token that they receive from the OAuth 2.0 server to the managed API as an Authorization header with the prefix `Bearer` followed by the token.

The following image shows an API invoked through Postman with a Bearer Token authorization type and the token that the API consumer entered:

GET ▼ https://[redacted].informaticacloud.com/t/myorg.com/GetEmployee Send Save

Params ● Authorization ● Headers (7) ● Body ● Pre-request Script ● Tests ● Settings ● Cookies ● Code

TYPE
Bearer Token

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Token
eyJraWQ6OjlmNzliYzQ1OC1mNjUxLTQ1ZWVhOWQ6MCDyOWNjNzQyMjAyZTYlCjhbGd...

Regenerating an OAuth 2.0 client secret

You can regenerate an OAuth 2.0 client secret. When you regenerate the client secret, API Center disables the current client secret.

1. On the **Informatica OAuth 2.0 Server** table on the **Authorization** tab of the **Configuration** page, click to open the **Actions** menu of the OAuth 2.0 client.
2. Select **Regenerate Secret**.
3. Enter the password and click **Next**.

API Center regenerates the client secret and authorization header value. You can copy the new values and send them to the API consumers.

4. Click **Finish**.

Rules and guidelines for OAuth 2.0 authentication

Controlling API access is an integral part of development. Authorization is checked dynamically, allowing immediate API access without the need for a new token. Consider the following guidelines for OAuth 2.0 authentication:

- Authorization checks occur dynamically. If you remove a managed API from the OAuth2.0 client, any existing token for that API will no longer function. Conversely, if you add a managed API to the OAuth 2.0 client, any previously generated token will work for invoking the newly added API.
- When you disable or delete an OAuth 2.0 client, the existing tokens for that client remain valid, but the client authorization will fail.
- Authorization fails for all revoked access tokens, even if they were issued before the API access revocation.
- Inactive managed APIs or managed API groups do not appear for selection when you edit an OAuth 2.0 client.
- Managed APIs or managed API groups in inactive state are de-associated from all associated OAuth 2.0 clients.

JSON web token authentication

JSON Web Token (JWT) is an open standard that helps in the secure transmission of information between API consumers and REST web services. You can create a JSON web token policy to assign to operations.

You can enable JWT authentication for APIs that meet the following criteria:

- The associated process uses HTTP/SOAP binding.
- The associated process uses basic authentication and defines the user groups and users who can access the process service URL at run time.
- The associated process is published and exposed as a service.

Generating a JSON web token

Generate a JSON web token to send to API consumers. You can generate a JSON web token when you edit a managed API.

1. On the **API Console** page, click **Managed APIs**.
The **Managed APIs** table appears.
2. Click to open the **Actions** menu of the API and select **Edit Managed API**.
The API details window appears.
3. In the **Operations** area, select an operation.
The operation details panel appears.
4. In the **Operations** area, expand the **Authentication** section and select **JWT - JSON Web Token**.

5. Select an expiration date for the token and click **Generate New Token**.

API Center creates a token for the operation that appears on the page.

Note: You can click **Generate New Token** to generate a new token if your earlier token has expired. After you generate a token, you can't revoke the token.

6. Click **Copy** to copy the token and send the token to API consumers.

API-level authentication policy

You can use an API-level JSON web token (JWT) authentication to securely access API endpoints and safeguard all your data and API interactions.

When you create or edit a managed API, you can enable JSON web token authentication at the API level. You can use the JSON web token to authenticate the API and all its operations. Thus, you can avoid creating individual tokens for each API operation and can invoke the API and all its operations using a single token.

When you enable the API-level JSON web token authentication, the API-level authentication overrides the operational level authentication based on the confirmation that you provide in the confirmation dialog box. You can generate an API-level JSON web token in one of the following ways:

- **API Console > Managed APIs > Actions menu > Create Managed API > API Policies > Security > Authentication > JWT - JSON Web Token.**
- **API Console > Managed APIs > Actions menu > Edit Managed API > API Policies > Security > Authentication > JWT - JSON Web Token.**
- **API Console > Managed APIs > Actions menu > Test Managed API > API Policies > Security > Authentication > Generate New Token.**

To invoke the API, you must enter the API-level JSON web token that you generated for the API.

You can generate the API-level JSON web tokens for managed APIs in the active, shared, and deprecated states. However, if you reactivate a deactivated managed API, you can't use the same token that was generated when the managed API was active. You must generate a new token.

Invoking an API with JSON web token authentication

To invoke a managed API where JSON web token authentication is enabled, API consumers pass the token as a bearer token in the HTTP Authorization header.

Session ID authentication

Session ID based authentication manages and maintains your logged-in state on API Center. You can assign the new session ID authentication method to APIs and operations.

You can now invoke managed APIs in API Center using session IDs obtained from your IDMC login call. After you authenticate by sending your credentials to the IDMC login endpoint, you receive a session ID that represents your authenticated session. You can then use this session ID to make subsequent API requests without needing to manage separate tokens.

Session ID authentication lets you simplify session management by maintaining your authentication state through the session ID. You can include the session ID in your request headers or cookies to access managed APIs securely and efficiently. The server verifies the validity of the session ID and confirms your

authenticated status before processing each request. Sessions expire after a set period or are destroyed when you log out to prevent misuse.

If you create an API by importing a Data Quality asset, use only the session ID authentication method.

To invoke a managed API using session ID, pass the session ID header value as `IDS-SESSION-ID` and then run the API.

Creating a security policy

Create a security policy to assign to APIs or operations. You can select one or more authentication methods to define the policy.

1. On the **Policies** page, click **Create Policy**.
2. In the **Name** field, enter a name for the policy.
The policy name can contain up to 128 characters, including ASCII letters, digits, Japanese characters, and the following special characters: \$ () [] . ? `
3. From the **Type** list, select **Authentication** and then click **Create**.
4. Optionally, in the **Description** field, enter a description.
The description can contain up to 512 characters.
5. In the **Authentication** area, select one or more authentication methods.
6. Click **Create**.
The security policy appears on the **Policies** page, under **Security Policies**.

CHAPTER 3

CORS policies

You can create Cross-Origin Resource Sharing (CORS) policies to ensure that your APIs can be securely accessed by client applications from different domains. CORS is a security feature implemented by web browsers to prevent web pages from making requests to a different domain than the one that served the web page.

When your API is hosted on a different domain than your web application, you can configure CORS to allow requests from the web application's domain. CORS acts as a security layer that controls and regulates how external applications or domains access your API, ensuring secure data exchange while preventing unauthorized access.

When a web page makes an HTTP request to your API to load an asset such as a font, image, or JSON file, that request can come from many different places across the internet. If these HTTP requests go unchecked, the security of your browser might be at risk. CORS enables you to allow requests to be made on your behalf while simultaneously blocking any potential malicious request. It gives you control over which websites or applications can access your resources, ensuring that trusted sites can interact with your data, while untrusted or malicious sites are prevented from making unauthorized requests. CORS helps keep your information secure and ensures that only approved interactions take place across different domains.

The **API References** area of a CORS policy displays all the managed REST and SOAP APIs, managed custom APIs, and managed API groups that use the policy as shown in the following image:

The screenshot shows the Informatica API Center interface. On the left is a sidebar with navigation options: New, Explore, Policies, API Console, API Groups, API Monitor, and Configuration. The 'Policies' section is selected, and the 'CORS' sub-section is active. The main area displays the configuration for a CORS policy named 'CFS'. The policy is of type 'CORS' and is enabled. Below the configuration fields, there is a section for 'CORS Groups' with a 'Create CORS Group' button. A table lists the allowed origins, access control max age, allowed methods, allowed headers, and exposed headers. The 'API References' section is highlighted with a red box and contains a table of managed APIs.

Allowed Origins	Access Control Max Age	Allowed Methods	Allowed Headers	Exposed Headers
http://abc.com	123	GET	Authorization	

Name	Version	Type	Status	Context Name	Last Updated
AnnaProcess_pp	v1	SOAP	Created	100%	Sep 4, 2024, 03:33 PM

CORS policy configuration

After you create a CORS policy and associate the policy with a CORS group, you can apply the CORS policy at the API level.

Applying a CORS policy at the API level empowers you to enforce specific rules for a particular API. You can apply this policy to all operations or endpoints within that API unless overridden at the operation level. For example, you can allow cross-origin requests from certain origins for one API but restrict the same cross-origin requests for another.

When you configure a CORS policy, the policy is enabled by default. After configuring the policy, the policy manager can choose to enable or disable the policy.

Creating a CORS policy

Create a CORS policy to assign to all the managed REST and SOAP APIs, managed custom APIs, and managed API groups.

If you are assigned an API Policy Manager role, you can create and manage the CORS policies in your organization. If you are assigned a Deployer role, you can apply the CORS policy to the managed APIs and managed API groups.

1. On the **Policies** page, click **Create Policy**.
2. In the **Name** field, enter a name for the policy.
The name can contain up to 128 characters, including ASCII letters, digits, Japanese characters, and the following special characters: \$ () [] . ? `.
3. From the **Type** list, select **CORS** and then click **Create**.
The CORS policy page appears.
The newly created CORS policy is enabled by default.
4. Optionally, in the **Description** field, enter a description.
The description can contain up to 512 characters.
5. Create a CORS group.
For more information, see [“Creating a CORS group” on page 16](#).
6. Click **Save**.
The CORS policy appears on the **Policies** page.
Use the filter type **CORS** to view all the available CORS policies in the organization.

Creating a CORS group

After you create a CORS policy, you must associated the policy with a CORS group.

1. On the **Policies** page, after you create a CORS policy, open the CORS policy, and then click **Create CORS Group**.
The **Create CORS Group** dialog box appears.

2. Enter the information for each field as mentioned in the following table:

Property	Description
Allowed Origins	<p>The list of origins permitted to access the managed API or managed API group. Origins are case insensitive.</p> <p>Allowed origins refer to the specific domains or origins that are permitted to access resources on a server. When an API makes a cross-origin request, the server checks if the origin of the request is on the allowed list. If the origin is allowed, the servers respond with the data, otherwise the request is blocked.</p> <p>Origin refers to the combination of the protocol, domain, and port. For example, <code>https://example.com:8080</code> and <code>http://example.com:8080</code> are considered different origins.</p> <p>If no CORS policy is configured, API Center automatically allows cross-origin requests.</p> <p>Default is <code>Access-Control-Allow-Origin: *</code> which means access from any origin is allowed by default of no CORS policy is configured.</p> <p>Note: A single CORS policy in API Center can't contain duplicate origins.</p>
Allowed Control Max Age	<p>The duration, in seconds, for caching the results of a preflight request.</p> <p>The Allowed Control Max Age header tells the browser how long it can cache the result of its preflight request. During this period, the browser doesn't send another preflight request for the same resource and can directly make the actual request.</p> <p>Maximum allowed control age is 86400 seconds.</p>
Allowed Method	<p>Select the methods to enable CORS support. Allowed methods contain method elements that specify the supported HTTP verbs. The allowed method ensures that only certain types of requests can be made cross-origin, reducing the risk of unauthorized actions.</p> <p>You can select one or all methods from the list of available methods. Available methods are GET, PATCH, POST, DELETE, and PUT.</p> <p>Default is <code>Access-Control-Allow-Methods: *</code>. If no CORS policy is configured, the value <code>*</code> indicates all methods.</p>
Allowed Headers	<p>Enter a comma-separated list of headers that the client must submit in the actual request of the resource. These are the list of headers allowed to be included in cross-origin requests. Allowed headers are not case sensitive.</p>
Exposed Headers	<p>List of acceptable headers to be exposed to the client scripts on a web browser, except for the CORS-safelists response headers.Exposed headers are not case sensitive.</p>

3. Click **Create**.

The CORS group appears in the **CORS Groups** section of the CORS policy.

Enabling a CORS policy

You can enable a CORS policy from the **Policies** page.

1. Click the **Actions** menu on the row of the CORS policy and select **Enable**.
2. On the **Enable Policy** dialog box, click **OK**.

Editing a CORS policy

You can edit a CORS policy from the **Policies** page.

1. Click the **Actions** menu on the row of the CORS policy and select **Edit**.
2. Edit the CORS policy and then click **Save**.

Disabling a CORS policy

You can disable a CORS policy from the **Policies** page.

1. Click the **Actions** menu on the row of the CORS policy and select **Disable**.
2. On the **Disable Policy** dialog box, click **OK**.

Deleting a CORS policy

You can delete a CORS policy from the **Policies** page.

1. Click the **Actions** menu on the row of the CORS policy and select **Delete**.
2. On the **Delete Policy** dialog box, click **OK** to delete the policy.

Editing a CORS group

If you are assigned the Admin role, you can edit a CORS group.

1. On the **Policies** page, click on the CORS policy name that you want to edit.
The CORS policy page opens.
2. Click the edit button on the row of the CORS group.
The **Update CORS Group** dialog box appears.
3. Edit the CORS group values and then click **Update**.
API Center applies the updates to the policy.

Deleting a CORS group

If you are assigned the Admin role, you can delete a CORS group.

You can't delete a CORS policy if the policy is associated with a managed API.

1. On the **Policies** page, click on the CORS policy name from which you want remove an associated CORS group.
The CORS policy page opens.
2. Click the delete button on the row of the CORS group.
A warning dialog box appears.
3. Click **Delete**.
API Center deletes the CORS group and the updates the CORS policy.

Rules and guidelines for CORS policies

Consider the following guidelines for CORS policies:

- You can't activate a managed API or managed API group if they contain a disabled CORS policy. Edit the managed API or managed API group and reassign a CORS policy.
- For a managed API with an anonymous security policy, if CORS policies are defined in both API Center and the Application Integration process, both CORS policies are applied to the managed API. These policies can either be identical or different. However, the request will succeed only if both CORS policies are successfully applied. If either policy fails, a failure response is returned.
In the case of an authenticated managed API, the CORS policy defined in API Center takes precedence and overrides the CORS policy defined in Application Integration.
- When testing a managed API that has an assigned CORS policy, ensure that the API Center host is included as one of the origins in the CORS policy configuration.
- To run an authenticated managed API with an assigned CORS policy, ensure the Authorization and Content-Type headers are included in the allowed headers list for the request origins. If the REST API was created using the top-down approach, add these headers to the CORS group's allowed headers list.
- According to W3C standards, ports 80 and 443 are the default ports.

CHAPTER 4

Rate limit policies

The rate limit policy controls the number of times API consumers can access an operation during a designated time frame. If the number of API calls exceeds the rate limit within the designated time frame, API consumers can't access the operation.

The default organization level rate limit is 1,000 requests per minute. The maximum rate limit that you can define is 10,000 requests per minute.

You can configure and associate rate limit policies with APIs and API operations when you design an API or create a managed API. When you create a managed API, you can associate user-level rate limit policy with a managed API to control the number of times a specific API consumer can access an API and its operations within a designated timeframe. You can associate only one user with one user-level rate limit policy.

You can associate an API-level or operation-level rate limit policy with any number of APIs or API operations. The **API References** area of a rate limit policy displays all the designed APIs, published APIs, and managed APIs that use the policy.

When you configure a rate limit policy, the policy is enabled by default. After configuring the policy, the policy manager can choose to enable or disable the policy.

You can change the values of an existing rate limit policy. Any change to the existing values reflect in all the API operations that use the rate limit policy. You must first disable the rate limit policy to change the values of the policy. After you save the changes, you can view the changes in API Center even when the API is not yet enabled. Enable the policy for the changes to take effect on the APIs in the runtime environment.

If you associate a rate limit policy at the API level, operations level, and user level, the rate limit policy with the minimum value takes precedence and the other values are ignored.

If a user-defined rate limit policy is selected at an API level and an operation inherits the API-level rate limit policy, API Center displays the selected policy name along with its configuration during the design of a REST API, managed API, and managed API group. If no policy is selected at the API level or operation level, API Center displays the organization-level rate limit policy as selected along with its configuration.

Creating a rate limit policy

Create a rate limit policy to assign to operations to control the number of times API consumers can access the operation during a designated time frame.

If you are assigned an Administrator or API Policy Manager role, you can create a rate limit policy. When you create a rate limit policy, the policy is enabled by default. You or other API Center user roles can then associate the rate limit policy while designing an API.

Note: Ensure that the managed API operation name for which you want to create the rate limit policy doesn't contain the keyword "_default" because "_default" is reserved for Informatica internal use.

1. On the **Policies** page, click **Create Policy**.
The **Create Policy** dialog box appears.
2. In the **Name** field, enter a name for the policy.
3. From the **Type** list, select **Rate Limit** and then click **Create**.
4. Optionally, in the **Description** field, enter a description.
5. In the **Number of Calls** field, specify the maximum number of times API consumers can access an operation during a designated time frame.
Enter a number from 1 through 1000000. The number of calls can't exceed the specified time frame by a multiple of 50.
6. In the **Time Frame** field, specify the time frame in seconds.
Enter a number from 1 through 86400.
7. Click **Save**.
The rate limit policy appears on the **Policies** page, under **Operational Policies**.

Organization level rate limit policy

An organization level rate limit policy is created by default while provisioning an API Center domain. The organization level rate limit policy is named as **system-org-level-rate-limit**.

If you do not apply a rate limit policy while designing or creating a managed API, this organization level rate limit policy gets applied to the API by default. You can associate the organization level rate limit policy with any number of APIs. The **API References** area of an organization level rate limit policy displays all the managed APIs that use the policy.

You can edit only the number of calls and time frame values of an organization level rate limit policy. Any change to the existing values reflects in all the API operations that use the organization level rate limit policy. You can't disable the organization level rate limit policy.

User-level rate limit policy

You can associate the user-level rate limit policy to control the number of times a specific API user can access a managed API and its operations within a designated timeframe. That is, you can configure the number of times a specific user can invoke the managed API.

When you try to invoke the managed API after reaching the configured rate limit, you receive an HTTP 429 status code.

You can associate only one user name with one user-level rate limit policy, regardless of whether the rate limit policy is enabled or disabled for the user.

You can associate a user-level rate limit policy only with an active state managed API. If you are assigned a user-level rate limit policy that is disabled in the policy configuration page, the policy status **Disabled** appears next to the **Policy Name** field. You can't activate a managed API that has a disabled user-level rate limit policy.

associated with it. However, you can activate a managed user if the policy is enabled for the managed API but disabled for a specific user.

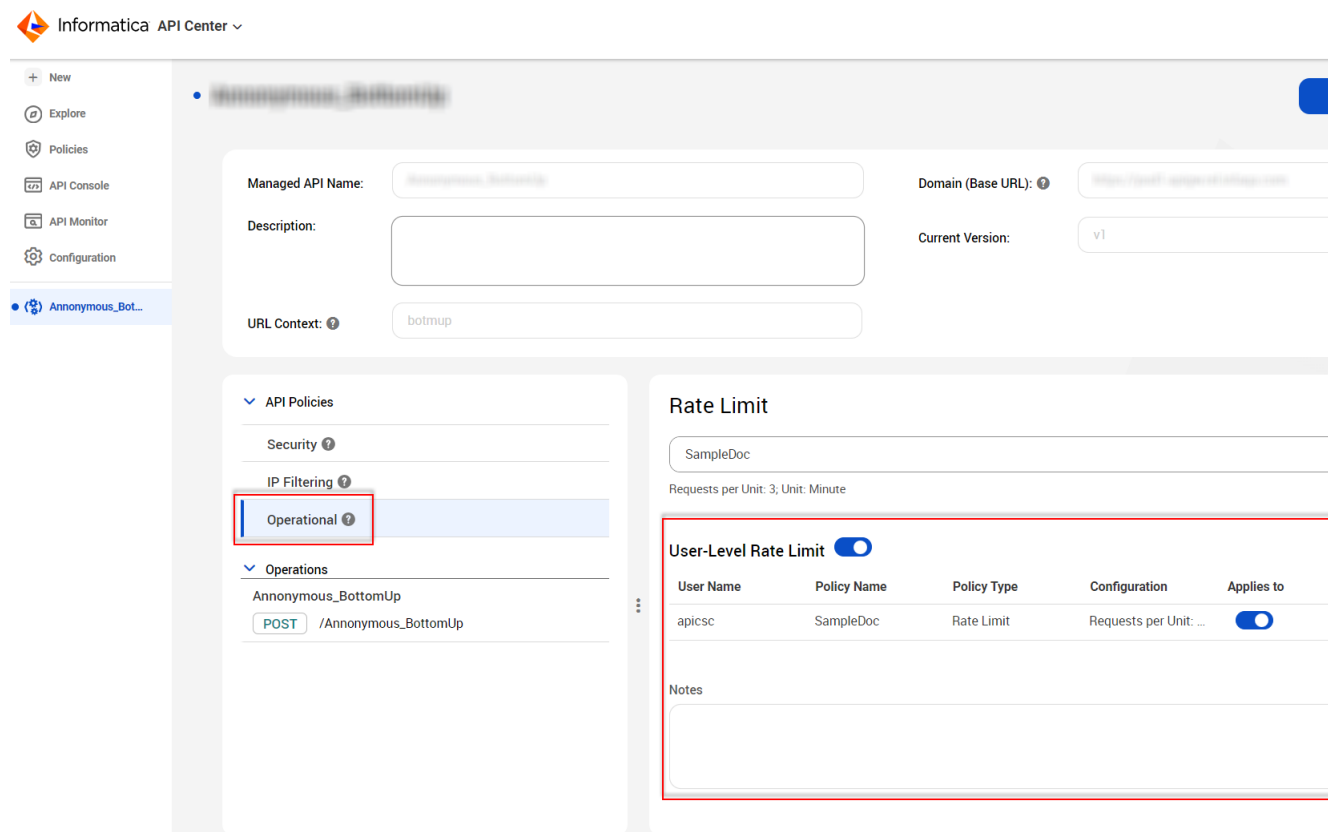
You can use the user-level rate limit policy with basic, OAuth 2.0, JSON web token (JWT), and session ID authentication methods.

To invoke a managed API using OAuth 2.0 authentication, you must create an OAuth 2.0 token using the credentials provided for the specific user name in the user-level rate limit policy. The managed API invocation succeeds only when you provide the correct OAuth 2.0 Client ID and OAuth 2.0 Client Secret for the specific user.

To invoke a managed API using JSON web token authentication, you must generate a JWT using the credentials provided for the specific user name in the user-level rate limit policy. The managed API invocation succeeds only when you provide the correct JWT for the specific user.

To invoke a managed API using session ID, pass the session ID header value as `IDS-SESSION-ID` and then run the API.

The following image shows how to configure a user-level rate limit policy for a managed API:



Creating a user-level rate limit policy

If you are assigned an Administrator or Deployer role, you can create a user-level rate limit policy for a managed API.

1. On the **API Console** page, click the **Managed APIs** tab.
2. Click the **Actions** menu on the row of the managed API to edit and select **Edit Managed API**.
3. In the **API Policies** area, click **Operational**.

4. Optionally, select the rate limit.

5. In the **User-Level Rate Limit** area, click the **+** button to add new user names from the available list of users.

You can associate only one user name with one user-level rate limit policy irrespective of the rate limit policy being enabled or disabled for the user.

If you do not want to associate the user-level rate limit policy to the entire managed API, you can toggle the slider and disable the rate limit policy.

6. Select the rate limit policy that you want to configure.

7. The policy type and configuration fields auto-populates based on the user name and policy type selection.

8. Optionally, toggle the **Applies to** slider if you do not want to associate the rate limit policy with the particular user.

9. Optionally, in the **Notes** field, enter a note.

10. Click **Save**.

If any validation errors occur, the **Validation** panel appears. Fix all errors listed on the **Validation** panel and click **Save** again.

API Center saves the managed API.

CHAPTER 5

Response caching policies

Response caching policies define how long API Center stores API responses in the cache. You can reduce redundant backend load by creating response caching policies for operations.

When response caching is enabled for an operation, API Center saves responses to invocations of the operation in the cache repository. When API consumers access the operation, API Center attempts to use saved responses in order to reduce time and resources. If a matching response is found in the cache, API Center returns the response from the cache.

API Center deletes responses from the cache after the defined timeout. When you disable the response caching policy for an operation, API Center deletes responses from the cache for the operation.

You can configure a maximum timeout of 3,600 seconds.

The **API References** area of a response caching policy displays all the designed APIs that use the policy.

When you run a managed API endpoint that has response caching configured, the response caching doesn't work if the upstream API response is in compressed format. The `Accept-Encoding` request header typically contains a comma-separated list of encoding formats, such as `gzip`, `br`, and `deflate`. The `Accept-Encoding` request header is added by default in Postman or web browser. To resolve the issue of honoring the response caching, use Postman to disable the `Accept-Encoding` header in your request, and then invoke the API. However, if you invoke the API from a web browser, `Accept-Encoding` is added automatically. As a result, the response caching policy might not be honored.

Creating a response caching policy

Create a response caching policy to assign to operations to define how long API Center stores API responses in the cache.

1. On the **Policies** page, click **Create Policy**.
2. In the **Name** field, enter a name for the policy.
3. From the **Type** list, select **Response Caching** and then click **Create**.
4. Optionally, in the **Description** field, enter a description.
5. In the **Caching Timeout** field, enter a timeout value from 1 through 3600 seconds.
6. Click **Save**.

The response caching policy appears on the **Policies** page, under **Operational Policies**.

CHAPTER 6

Privacy policies

You can configure a privacy policy for an API or operation to protect Personally Identifiable Information (PII) that API data contains.

API Center can issue warnings or block API requests and responses if the request or response payload contains the following information:

- Credit card number
- Email address
- IP address
- Phone number
- United States Social Security number

You can select different actions for requests and responses.

The **API References** area of a privacy policy displays all the designed APIs that use the policy.

Creating a Personally Identifiable Information (PII) policy

Create a Personally Identifiable Information (PII) policy to assign to APIs or operations to protect the sensitive data that you define.

1. On the **Policies** page, click **Create Policy**.
2. In the **Name** field, enter a name for the policy.
3. From the **Type** list, select **Personally Identifiable Information (PII)** and then click **Create**.
4. Optionally, in the **Description** field, enter a description.
5. In the **Privacy Settings** area, for each type of information to protect, select the action to take for the request and the response. You can select different actions for the request and the response.

Select one of the following actions:

- **No action.** Don't take any action.
- **Warning.** Issue a warning message that there was a privacy policy leakage. Don't block the request or response.
- **Block.** Block the request or response and issue a warning message that the action was blocked because of a potential privacy policy breach.

6. Click **Save**.

The PII policy appears on the **Policies** page, under **Privacy Policies**.

CHAPTER 7

IP filtering policies

IP filtering policies control access to managed APIs, managed API groups, and custom APIs within your organization. An IP filtering policy defines a set of access rules that determine the IP addresses that are allowed to access an API and the IP addresses that are blocked from accessing an API. You must be assigned the Deployer role to assign IP filtering policies to managed APIs.

When API Center receives a request to access a managed API, it checks the request IP address against the rules in the policy, in the order that the rules are listed in. API Center applies the first rule that matches the request IP address to the request.

Note: You can't use a combination of allow and deny IP filtering rules for the same API.

You can define IP filtering policies at both the domain level, managed API, and managed API group level. The **Domain IP Filtering** policy applies globally to every API in the organization by default. If you configure both managed API or managed API group and domain IP filtering policies, the managed API or managed API group policies take precedence over domain IP filtering policies.

IP filtering policy example

The following table lists rules that allow access to specific IP address ranges:

Access Type	From IP address	To IP address
Allow	9.10.11.12	9.10.25.27
Allow	5.6.7.8	5.6.15.17
Allow	1.2.3.4	1.2.12.14

The following table lists rules that deny access to all IP addresses:

Access Type	From IP address	To IP address
Deny	0.0.0.0	255.255.255.255

API-specific IP filtering policy

You can configure an IP filtering policy for a managed API, managed API group, or custom API.

The IP filtering policy designates the range of computer IP addresses that are allowed or that are denied permission to invoke the API. If an IP filtering policy is not defined for an API, or if API Center doesn't find any matches with the API-specific policy, API Center applies the domain IP filtering policy to the API.

Creating an IP filtering policy

Create an IP filtering policy to assign to managed APIs, managed API groups, or custom APIs to control access to the APIs.

1. On the **API Console** page, click **Managed APIs**.
2. Click to open the actions menu of the API and select **Edit Managed API**.
3. In the **API Policies** area, click **IP Filtering**.
4. Click **Add IP Filtering Rule**.
5. Select to allow or deny a range of addresses, and then fill in the IP range.
6. Optionally, enter a description of the rule.
7. Repeat steps 4 through 6 to create as many additional rules as required to define the policy.

Note: You can't use a combination of allow and deny rules.

You can change the rule evaluation order by moving rules up or down in the list.

8. Click **Save**.

To delete a rule, in the right-most column of the **IP Filtering Rules** list, rest on a row in the table and click **Delete**.

Domain-level IP filtering

You can use the **Domain IP Filtering** policy to manage IP-based access control for all managed APIs, managed API groups, and custom APIs within your organization. This policy defines access rules that either allow or deny IP addresses permission to invoke any API in your organization.

The **Domain IP Filtering** policy applies globally to every API in the organization by default. If you configure both managed API or managed API group and domain IP filtering policies, the managed API or managed API group policies take precedence over domain IP filtering policies.

When API Center receives an API invocation request, it checks the request's IP address against the IP filtering rules in the order they are listed. The first matching rule is applied to the request, either allowing or denying access.

You must use either allow rules or deny rules in the domain IP filtering policy. You can't use a combination of allow and deny rules.

Configuring a domain-level IP filtering policy provides centralized IP access control across all APIs in your organization, simplifying security management. You can combine the domain IP filtering policy with managed API, managed API group, or custom API-specific policies for greater flexibility. You can also control how policies are evaluated and enforced by ordering the rules.

Creating an IP filtering policy at the domain level

Create an IP filtering policy to assign to all managed APIs, managed API groups, and custom API in your organization.

1. On the **Configuration** page, click the **Domain IP Filtering** tab.
2. Click **Add IP Filtering Rule**.
3. Select to allow or deny a range of addresses, and then fill in the IP range.
4. Optionally, enter a description of the rule.

5. Repeat steps 2 through 4 to create as many additional rules as required to define the policy. You can change the rule evaluation order by moving rules up or down in the list.

Note: You can't use a combination of allow and deny rules.

6. Click **Save**.

To delete a rule, in the right-most column of the **IP Filtering Rules** list, rest on a row in the table and click **Delete**.

CHAPTER 8

Third-party authentication and authorization

You can use third-party authentication in API Center to handle user authentication and authorization.

API Center integrates with external identity providers, such as Okta or Azure Active Directory (AAD) to verify users' identities and grant them access to protected resources. The advanced identity management and authentication systems with third-party providers enhances security and scalability, and reduces administrative load in API Center.

If your organization is licensed and configured for SAML, you can use OAuth to start a REST API session using JSON web token (JWT) access tokens. Your organization's IDMC administrator can set up OAuth using Azure Active Directory or Okta as the identity provider so that users can start REST API sessions using JWT access tokens.

You can use your already existing OAuth provider or identity provider, such as Azure AD or Okta. You can generate the tokens using the authorization servers provided by these identity providers to access protected resources, such as APIs. You do not need to add all your third-party API users in IDMC.

Federated or third-party authorization server support with OAuth 2.0 server

You can implement OAuth 2.0 authentication using federated or third-party providers such as, Okta and Azure AD, to secure your APIs effectively. This configuration enhances API security by leveraging external OAuth identity providers. You can streamline the authentication process, making it easier for developers to integrate secure access controls.

You can enable your API consumers to access protected resources using tokens generated from their preferred OAuth provider without pre-provisioning all API consumers through the IDMC identity provider.

You can fetch JWT access tokens from the identity provider and include the tokens in login requests. Before you use JWT access tokens, complete the following tasks:

1. In IDMC, configure the organization to use SAML and set up users as SAML users.
2. Set up an OAuth identity provider. You can use identity providers such as Azure Active Directory and Okta.
3. In API Center, register the authorization server of the third-party identity service for authentication. For more information, see [“Register an authorization server” on page 31](#).
4. In API Center, assign the third-party users to specific APIs for authorization.

To log in, you need to obtain a JWT access token from the identity provider and include the token in a loginOAuth POST request. The token can be used for one REST API session. If the login request is successful, the response includes a session ID to use in subsequent API calls.

For information about OAuth setup using Azure Active Directory, see the following article: [Set up OAuth with Azure AD](#)

For information about OAuth setup using Okta, see the following article: [Set up OAuth with Okta](#)

For more information about identity provider configuration, see the identity provider's documentation.

For more information about registering identity providers and logging in using JWT access tokens, see [REST API Reference](#).

Register an authorization server

If you are assigned the Admin role, you can add your client's authorization server, generate the user-specific OAuth 2.0 client token, and use the token to grant users permissions for accessing a managed API or managed API groups.

Complete the following steps to register the authorization server of the third-party identity service for authentication.

1. On the **Third-Party Authentication** tab of the **Configuration** page, click **Register Authorization Server**.
The **Register Authorization Server** dialog box appears.
2. Enter the following details:

Field	Description
Issuer	The unique identifier or absolute URL of your organization's identity provider. Must use the same HTTPS scheme as the key URL and be a subset of the key URL. Maximum length is 255 characters.
JWKS URI	The URL where your identity provider publishes its public keys. This URL is crucial for verifying security during authentication.
IDP Attribute Name	A specific data field (claim) from the security token provided by your identity provider that API Center uses to identify the IDMC user. The default value is <code>sub</code> . <code>sub</code> is short for <code>subject</code> .
IDMC Attribute Name	The IDMC attribute name used to identify the IDMC user. Allowed values are <code>Alias Name</code> and <code>User ID</code> .
Verify JWKS URI	The URL must use the same HTTPS scheme as the issuer URL. Default is enabled. For an identity provider where the URL and HTTPS schema are different from the issuer URL, disable this option.

3. Click **Add**.
The added authorization server appears in the third-party authentication tab.

Edit an authorization server

If you are assigned the Admin role, you can edit an authorization server from the **Third-party Authentication** page.

1. On the **Configuration** page, click **Third-Party Authentication** tab.
2. Click the **Actions** menu on the row of the registered authorization server and select **Edit**.
3. Edit the authorization server details and then click **Edit**.

The authorization server details get updated.

Delete an authorization server

If you are assigned the Admin role, you can delete an authorization server from the **Third-party Authentication** page.

1. On the **Configuration** page, click **Third-Party Authentication** tab.
2. Click the **Actions** menu on the row of the registered authorization server and select **Delete**.
A delete confirmation dialog box appears.
3. Click **Delete**.

The authorization server gets deleted from API Center.

Define user permission

If you are assigned the Deployer or Admin role, you can define permissions for active SAML users that you configured in your organization. You can grant access to each user for accessing a managed API or managed API group in your organization.

1. On the **Configuration** page, click the **Authorization** tab.
2. On the **Third-party Authorization** tab, click **Define User Permission**.
3. Select a user from the list of available users.
All active SAML users for the organization appears.
4. Select the managed APIs and managed API groups for which you want to allow access permissions to the particular user.
The managed APIs and managed API groups are organized into distinct tabs, making it easier to locate and interact with the specific resources you need.
5. Click **Save**.

You can edit, delete, or disable user permissions as required.

INDEX

R

Rate limit policy
Organization level [20](#)

Rate limit policy (*continued*)
system-org-level-rate-limit [20](#)