

webMethods API Gateway User's Guide

Version 10.5

October 2019

This document applies to webMethods API Gateway 10.5 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 2016-2024 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA Inc. and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at <https://softwareag.com/licenses/>.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at <https://softwareag.com/licenses/> and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software AG Products / Copyright and Trademark Notices of Software AG Products". These documents are part of the product documentation, located at <https://softwareag.com/licenses/> and/or in the root installation directory of the licensed product(s).

Document ID: YAI-UG-105-20240212

Table of Contents

About this Documentation.....	7
Document Conventions.....	8
Online Information and Support.....	8
Data Protection.....	9
1 webMethods API Gateway.....	11
Introduction to webMethods API Gateway.....	12
Searching Data in API Gateway.....	13
Configuring the Number of APIs listed on a Page.....	16
Using Help in API Gateway.....	16
2 API Gateway Administration.....	19
Overview of Administration Tasks.....	20
General Configuration.....	20
Security Configuration.....	108
Destination Configuration.....	197
Audit Logging.....	217
Data Management.....	222
System Settings.....	226
External Accounts.....	229
3 User Management.....	235
Manage Users, Groups, and Teams.....	236
Manage Your User Settings and Preferences.....	257
4 APIs.....	261
Overview.....	263
Creating an API by Importing an API from a File.....	265
Creating an API by Importing an API from a URL.....	267
Creating an API from Scratch.....	268
API Mashups.....	286
Viewing API List and API Details.....	296
Filtering APIs.....	302
Activating an API.....	302
Deactivating an API.....	305
Publishing APIs.....	305
Unpublishing APIs.....	311
Modifying API Details.....	315
Updating APIs.....	316
API Mocking.....	319
Attaching Documents to an API.....	323
SOAP to REST Transformation.....	325
CentraSite Provided APIs.....	333

Versioning APIs.....	334
API Scopes.....	335
Exposing a SOAP API to Applications.....	344
API Grouping.....	345
API Tagging.....	346
Exporting APIs.....	347
Exporting Specifications.....	349
Deleting APIs.....	350
Example: Managing an API.....	351
5 Policies.....	363
Overview.....	364
Policy Validation and Dependencies.....	366
Managing Threat Protection Policies.....	371
System-defined Stages and Policies.....	382
Managing Global Policies.....	542
Managing API-level Policies.....	557
Managing Scope-level Policies.....	560
Managing Policy Templates.....	565
Supported Alias and Policy Combinations.....	576
6 Aliases.....	581
Overview.....	582
Creating a Simple Alias.....	582
Creating an Endpoint Alias.....	583
Creating an HTTP Transport Security Alias.....	586
Creating a SOAP Message Security Alias.....	589
Creating a webMethods Integration Server Service Alias.....	592
Creating an XSLT Transformation Alias.....	593
7 Applications.....	595
Overview.....	596
Creating an Application.....	597
Viewing List of Applications and Application Details.....	605
Regenerating API Access Key.....	606
Modifying Application Details.....	606
Registering an API with Consumer Applications from API Details Page.....	607
Registering APIs with Consumer Applications from Application Details Page.....	607
Suspending an Application.....	608
Activating a Suspended Application.....	608
8 API Packages and Plans.....	611
Overview.....	612
Creating a Package.....	612
Creating a Plan.....	614
Viewing List of Packages and Package Details.....	617
Modifying a Package.....	617
Deleting a Package.....	618

Activating a Package.....	618
Publishing a Package.....	619
Viewing List of Plans and Plan Details.....	620
Modifying a Plan.....	620
Deleting a Plan.....	621
9 Export and Import Assets and Configurations.....	623
Overview.....	624
Importing Asset and Configuration Archives.....	629
10 Asset Promotions.....	633
Manage Stages, Promotions, and Rollbacks.....	634
11 API Gateway Analytics.....	647
Analytics Dashboards.....	648
Runtime Events and Metrics Data Model.....	656
12 Microgateway Management.....	741
Overview.....	742
13 REST APIs in API Gateway.....	745
API Gateway Administration.....	746
Alias Management.....	754
Application Management.....	754
API Gateway Archive.....	756
API Gateway Availability.....	756
Document Management.....	757
Internal Service.....	757
Policy Management.....	758
Promotion Management.....	760
Public Services.....	761
API Gateway Search.....	762
Server Information.....	765
Service Management.....	765
Transaction Data.....	766
User Management.....	767
Backward compatibility support for REST APIs.....	768
14 Remove User Data from API Gateway.....	771
Removing User Data.....	772
15 Usage Scenarios.....	775
Change Ownership of Assets.....	776
Custom Policy Extension.....	786
Team Support.....	802
API First Implementation.....	826

Gateway Endpoints.....	834
SAML SSO.....	840
Secure API using OAuth2 with refresh token workflow.....	847

About this Documentation

■ Document Conventions	8
■ Online Information and Support	8
■ Data Protection	9

This documentation describes how you can use API Gateway and other API Gateway components to effectively manage APIs for services that you want to expose to applications, whether inside your organization or outside to partners and third parties.

To use this content effectively, you should have an understanding of the APIs that you want to expose to the developer community and the access privileges you want to impose on those APIs.

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Narrowfont	Identifies service names and locations in the format <i>folder.subfolder.service</i> , APIs, Java classes, methods, properties.
<i>Italic</i>	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
Monospace font	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the symbol.
[]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at <http://documentation.softwareag.com>. The site requires credentials for Software AG's Product Support site Empower. If you do not have Empower credentials, you must use the TECHcommunity website.

Software AG Empower Product Support Website

If you do not yet have an account for Empower, send an email to empower@softwareag.com with your name, company, and company email address and request an account.

Once you have an account, you can open Support Incidents online via the eService section of Empower at <https://empower.softwareag.com/>.

You can find product information on the Software AG Empower Product Support website at <https://empower.softwareag.com>.

To submit feature/enhancement requests, get information about product availability, and download products, go to [Products](#).

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the [Knowledge Center](#).

If you have any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory at https://empower.softwareag.com/public_directory.asp and give us a call.

Software AG TECHcommunity

You can find documentation and other technical information on the Software AG TECHcommunity website at <http://techcommunity.softwareag.com>. You can:

- Access product documentation, if you have TECHcommunity credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

1 webMethods API Gateway

■ Introduction to webMethods API Gateway	12
■ Searching Data in API Gateway	13
■ Configuring the Number of APIs listed on a Page	16
■ Using Help in API Gateway	16

Introduction to webMethods API Gateway

webMethods API Gateway enables an organization to securely expose APIs to external developers, partners, and other consumers for use in building their own applications on their desired platforms. It provides a dedicated, web-based user interface to perform all the administration and API related tasks such as creating APIs, defining and activating policies, creating applications, and consuming APIs. API Gateway gives you rich dashboard capabilities for API Analytics. APIs created in API Gateway can also be published to API Portal for external facing developers' consumption. webMethods API Gateway supports REST-based APIs, SOAP-based APIs, and WebSocket APIs, provides protection from malicious attacks, provides a complete run-time governance of APIs, and information about gateway-specific events and API-specific events.

Note:

Software AG recommends using API Gateway user interface for all the functionalities provided by API Gateway and not use the Integration Server user interface.

API Gateway provides the following key features:

Support for SOAP APIs, REST APIs, and WebSocket APIs

API Gateway supports REST-based APIs, SOAP-based APIs, and WebSocket APIs. This support enables organizations to leverage their current investments in SOAP-based APIs while adopting REST for new APIs. The API Gateway's SOAP to REST transformation feature enables an API provider to expose parts of the SOAP API or expose the complete SOAP API with RESTful interface. API Gateway allows you to customize the way the SOAP operations are exposed as REST resources.

Secure APIs

API Gateway protects APIs from malicious attacks initiated by external client applications. Administrators can secure traffic between API consumer requests and the execution of services on API Gateway by filtering requests coming from particular IP addresses and blacklisting specified IP addresses, detecting and filtering requests coming from particular mobile devices. You can avoid additional inbound firewall holes when the native APIs are hosted on webMethods ESB.

Policy enforcement

API Gateway provides complete run-time governance of APIs. API Gateway enforces access tokens such as API key check, OAuth2 token and operational policies such as security policies for run-time requests between applications and native services. API providers can enforce security, traffic management, monitoring, and SLA management policies, can transform requests and responses into expected formats, and collect events metrics on API consumption and policy evaluation. API Policies can be defined globally and applied to a set of APIs. With API Gateway you can also define policy templates that can be applied across APIs.

Mediation

API Gateway provides routing policies such as content-based, and context-based, for run-time requests between applications and native services. These policies perform routing and load balancing of incoming requests to an API.

Message transformation

API Gateway lets you configure an API and to transform the request and response messages to suit your requirements. To do this, you can specify an XSLT file to transform messages during the mediation process. You can also configure an API to invoke Integration Server services to pre-process or post-process the request or response messages.

Easy discovery and testing of APIs

API Gateway provides filter capabilities to quickly find APIs of interest. API descriptions and additional documentation, usage examples, and information about policies enforced at the API level provide more details to the developers that help them decide whether to adopt a particular API. Developers can use the provided samples and expected error and return codes to see how the API works.

Clustering support

Multiple instances of API Gateway can be clustered together to provide scalability and high availability.

Built-in usage analytics

API Gateway provides information about Gateway-specific events and API-specific events, details about which APIs are more popular than others. The Gateway-specific events information is available by way of dashboards to users. With this information, providers can understand how their APIs are being used, which in turn can help identify ways of improving their users' experience and increase API adoption.

Packages and Plans

API Gateway provides capabilities to create and manage packages and plans. This helps the API providers in providing tiered access to their APIs to allow different service levels and pricing plans. Users can view the details of the package, such as included APIs and associated plans. Plans provide information about pricing and quality of service terms defined within them. Consumers can subscribe to any plan available under the package, based on their business needs.

Functional Privileges

API Gateway allows you to assign functional privileges to a user or group (LDAP or local) using access profiles. The functional privileges are assigned to users of teams based on the team's requirements. You must have a functional privilege assigned to perform any of the key API Gateway features.

API Mashups

API Gateway allows you to consolidate services and expose them as a single service. You can create API mashups that extend an API operation by grouping it with other API operations available in API Gateway.

Searching Data in API Gateway

The search feature in API Gateway is a type-ahead search; a simple and easy to use search facility where you can type the text of interest to search. You can search for all items that contain one or

more specified keywords (that is, text strings) in the item's properties. Some of the properties are name, description, version, key, value, and so on in the API.


You can search for the following types of data:

- APIs
- Applications
- Aliases
- Packages
- Plans
- Threat protection
- Global policies
- Policy templates
- Users
- Groups
- Teams

To search for an item, type a string in the search box in the title navigation bar. A list of search result is displayed directly below the Search box. The number of matches found are displayed in five sections depending on the type they fit in: **APIs**, **Application**, **Alias**, **Packages**, and **Plans**. A minimum of five search results are displayed in each category. If there are no results as per the search string typed, a message displays saying so.

If you find what you are searching for in the search result box, click on it to view the details. You are navigated to the specific page that displays more information. For example, if you are searching for an API and click the displayed result, you are navigated to the specified API details page. If you are searching for an application and click the displayed result, you are navigated to the specified Application details page.

If you want to see all the search results click **Show all results** in the search result box. The Advanced search page is displayed. This is a dedicated page that displays extensive search results. In the Advanced search page, you can search or filter the results in the following ways: by team, by type, or keyword.

- **By Team:** Select the **Select All** check box in the **Search by team** section to display all results associated with the teams that you are assigned to. To view the results for a team, select the check box next to the required team. When no teams are selected, results for all teams assigned to you appear are displayed.
- **By type:** Select one or more types in the left navigation pane to see search results pertaining to the selected types. For example, if you select the type **APIs**, all the APIs that have the specified string is displayed. By default, all filters are selected. To remove a filter, you can clear the check box next to a filter from the left pane or click  next the filter you want to remove.

- **By keyword:** Type a keyword in the **Search by keyword** field, all the search results containing the specified keyword are displayed in the list. For example, if you type the keyword `petstore`, all search results containing the `petstore` would be filtered and displayed.

Note:

Search by keyword will not show any search results, if the field names have any special characters. The following special characters are not supported - ! ? & # \$ * % ; = ' " () / \ < >

The fields that does not support special characters are as follows:

- Maturity state
- Scope name
- Scope description
- API Operations info name
- API Resource path
- API Tags
- Application identifiers named values
- User Login ID
- User First name
- User Last name
- OAuth scope name
- OAuth Scope description

For example, if an API has a tag name `Test-001`, and you search APIs with the tag name `Test-001`, you will not get any search results.

Note:

You cannot search for REST resources and methods in a REST API. The search function only works for the name and description of the REST API. For example, you can search for a REST API named `LibraryAPI`. But you cannot search for a REST resource named `book` or a REST method `POST` within the REST API. However, the search function works for name, description, and operations of SOAP APIs.

You cannot search for resources and methods of an OData API.

There are a few configurable properties available for search. These properties can be configured in the file, `uiconfiguration.properties`, located at `SAGInstallDir\profiles\IS_default\apigateway\config\`. Edit the file as required. After modifying the properties file, you have to restart Integration Server for the changes to take effect.

You must type in a minimum number of characters in the global search box, to search for data. This property can be configured.

The following property is used to configure the minimum number of characters to search. The default value is 3.

```
apigw.search.minimum.num.chars=3
```

Note:

The value provided must be a number greater than 0. If you provide an invalid value, it takes up the default value of 3.

The following property is used to configure the number of search results to load for each type in the advanced search page. The default value is 10.

```
apigw.num.results.search=10
```

Note:

The value provided must be a number greater than 0. If you provide an invalid value, it takes up the default value of 10.

Configuring the Number of APIs listed on a Page

The default number of APIs that are listed in the Manage APIs and Manage applications page can be configured through the properties file located at `SAGInstallDir\profiles\IS_default\apigateway\config\uiconfiguration.properties`.

Edit the configuration file as required. You can configure the number of results to load for pagination. The default value is 20. The provided value should be a number greater than 0.

```
apigw.num.results.pagination=20
```

If you configure a random value or value that is not specified in the extended setting `paginationPossibleValues`, then API Gateway user interface, by default, displays 20. For details about the extended settings, see [“Configuring Extended Settings” on page 22](#).

You have to restart Integration Server for the changes to take effect.



You can configure the number of APIs that get listed per page in the Manage APIs or the Manage applications page. In each of these pages, you can use the pagination bar at the bottom of the page to navigate from one page to another, the first page, or the last page when there are more than 20 APIs in the list. To change the number of APIs listed in a page, select the required number in the **Show # results per page** field in the pagination bar at the bottom of the page. The API list now displays only those many APIs in one page as specified. For example, if you select **Show 10 results per page**, only 10 APIs are listed in one page.

This configuration that you change through the drop down is maintained as long as you are logged in to API Gateway. Once you log out, the value is reset to the default configured value in the `uiconfiguration.properties` file.

The value is set in the drop down is applicable for both APIs and applications listing. For example, if you change the show results to 10 in the Manage APIs drop down, then the number is retained for Manage applications page as well.

Using Help in API Gateway

API Gateway's built-in context-sensitive help gives an overview of the functionality of API Gateway.

You can access API Gateway Help link by expanding the menu options icon , in the title bar and selecting **Help**. This opens the introduction to the webMethods API Gateway page in the help system. You can browse the required topics in the navigation pane. Click on a topic to display the detailed information. You will also find the help links in the form of a help icon  on several pages

of the API Gateway user interface. Click the help icon  on the page to view the corresponding detailed information.

2 API Gateway Administration

■ Overview of Administration Tasks	20
■ General Configuration	20
■ Security Configuration	108
■ Destination Configuration	197
■ Audit Logging	217
■ Data Management	222
■ System Settings	226
■ External Accounts	229

Overview of Administration Tasks

This section describes the various aspects of the way API Gateway functions can be configured:

- **General configuration:** load balancing, extended settings, service faults, approval settings, proxy server aliases, and URL aliases.
- **Security configuration:** keystores and trustore, ports, SAML issuer, custom assertions, OAuth 2.0, Kerberos settings, JWT, and OpenID provider.
- **Destination configuration:** targets to which the events and performance metrics data is sent.
- **Data Management:** configure the event types and interval at which data can be archived or purged.
- **System setting configuration:** configure system-level configuration parameters and communicate them across nodes in the cluster.
- **External account configuration:** add and configure service registries, AWS accounts, and Integration Server instances.

General Configuration

You must have the API Gateway's manage general administration configurations functional privilege assigned to perform the following configurations in the general configuration section of API Gateway:

- Configure API Gateway to communicate with a load balancer that allows API Gateway to provide the endpoints of the API with the load balancer URL.
- Configure the extended settings which are advanced parameters required for your server to operate properly.
- Configure API fault settings for errors being returned by the API Gateway to the applications.
- Configure approval settings.
- Configure Proxy server aliases.
- Configure URL aliases.
- Configure the custom content-types.
- Configure cache to boost performance.
- Configure log levels and the destination where the aggregated logs would be collected.
- Configure API Gateway and Terracotta license file path.
- Configure cluster settings.
- Configure API callback request settings.
- Configure transaction alerts settings.

Clusters and Load Balancers

Load balancing enables the distribution of messages received across the API Gateway nodes. Clustering helps in attaining the high availability functionality as well as load balancing. In addition, a cluster provides fault tolerance that ensures recovery of events and metrics in case a node goes down. If you have a web service that is hosted at two or more endpoints, you can use the load balancing option to distribute requests across the nodes. Requests are distributed across multiple nodes and are routed based on the round-robin execution strategy.

In a load balanced system, calls from an application are distributed across two or more different instances of API Gateway, referred to as nodes. Each node is an instance of API Gateway running on an instance of Integration Server.

If you cluster API Gateway instances, you must configure API Gateway with the load balancer URL appropriately for it to report the API address at the load balancer URL instead of having direct access address with API Gateway.

Load Balancer URLs

Load balancer URLs are the URLs of the load balancer where the requests are sent by the applications. When an API is activated on a node of a cluster, the endpoint of the API is provided as the load balancer URL instead of the Gateway end point. API Gateway stores this load balancer URL endpoint. Since all nodes of an API Gateway cluster are synchronized with APIs, each API Gateway accepts the message when routed from the load balancer to any node in the cluster.

A load balancer URL consists of a host name (or IP address) and port number of the load balancer in the following format:

`http://hostname:portnumber`

or

`http://IP-address:portnumber`

For example, if the host name of the load balancer is ExampleHost, and its port number is 80, the load balancer URL would be `http://ExampleHost:80`

You must configure any one node in a cluster with the same load balancer URL. The load balancer URLs are automatically synchronized on all the nodes in a cluster.


Note:

- When a Load Balancer URL is updated in API Gateway, a logout and re-login or opening a new session reflects the updated artifacts URL under Specifications. The artifacts downloaded without a new session also show the updated endpoints.
- You can also configure the load balancer URL to use the Web application URL, HTTPS protocol, or WebSocket port configuration.

Configuring Load Balancer URLs

You have to configure the load balancer URLs to define the endpoints for API Gateway.

> To configure the load balancer URLs

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Load balancer**.
3. Provide the following information in the API runtime invocation URLs section:

Field	Description
Load balancer URL (HTTP)	Specifies the primary HTTP load balancer URL. For the URL, you can specify either the IP address or host name of the load balancer with the port number, as follows: <i>http://IP-address:portnumber or http://hostname:portnumber</i>
Load balancer URL (HTTPS)	Specifies the primary HTTPS load balancer URL. For the URL, you can specify either the IP address or host name of the load balancer with the port number, as follows: <i>http://IP-address:portnumber or http://hostname:portnumber</i>
Load balancer URL (WS)	Specifies the WebSocket load balancer URL. For the URL, you can specify either the IP address or host name of the load balancer with the port number, as follows: <i>http://IP-address:portnumber or http://hostname:portnumber</i>

4. Provide the following information in the Web application URLs section:

Field	Description
Web application load balancer URL	Specifies the Web application load balancer URL. If a value is not specified, then API Gateway uses the default hostname and port number during publish of an API Gateway asset from CentraSite to API Gateway. For example, <i>http://myHostname:9072</i> .


5. Click **Save**.

Configuring Extended Settings

You must have the API Gateway's manage user administration functional privilege assigned to configure the extended settings.

You can configure advanced parameter settings in the Extended settings section. These parameters affect the operation of your server. You must not change these settings unless requested to do so by Software AG Global Support. You can configure the watt parameter settings in this section.

➤ **To configure the extended settings**

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Extended settings**.
3. Click **Show and hide keys**. This displays all the configurable parameters.
4. You can configure any of the following parameters in the Extended keys section by providing the required values. The configured values are listed under Extended settings at the top of the page.

Parameter and Description

allowEGInvokeOnly

Specifies whether the SOAP APIs with Transport policy set to `http`, can be invoked using the reverse invoke method when you set the external port as `https`, and the Registration and Internal ports as `http`. Ensure you enable this setting in the system where the SOAP API is created.

Note:

This setting affects only the behavior of SOAP APIs. You can invoke the REST APIs using the reverse invoke method, irrespective of this setting, given the above said conditions are true.

Possible values:

- `true`. You can invoke SOAP APIs using the reverse invoke method if the external port is set as `https`, and the Registration and Internal ports are set as `http`.
- `false`. You cannot invoke SOAP APIs using the reverse invoke method.

allowExceedMaxWindowSize

Specifies whether the number of records retrieved by Elasticsearch in a single request exceeds the configured value or not.

Possible values:

- `true`. The number of records retrieved in a single request can exceed the maximum value configured in Elasticsearch. This is the default value.
- `false`. Displays an error message when the number of records retrieved in a single request exceeds the configured value.

apiDocumentsRestrictedExtension

Specifies the list of restricted file extensions to prevent users from uploading files with those extensions as the input document for an API. For example, a file with the `.exe` file extension

Parameter and Description

could contain executable code that run on demand when it is downloaded. If files with the .exe file extension are restricted, users cannot upload a file with the .exe extension in API Gateway.

By default, several standard file extensions are blocked, including any file extensions that are treated as executable files by Windows Explorer. The file extensions blocked by default are:

- .bat - Batch file
- .bin - Binary file
- .dll - Windows dynamic link library
- .exe - Executable program

You can remove any or all of the default file extensions.

apiDocumentsUploadSizeLimitInMB

Specifies the maximum document size, in MB, that can be uploaded as an input document for an API.

Default value is 5.

This prevents users from uploading huge files that might slow down the system.

apig_MENConfiguration_tickInterval

Specifies the time interval (in seconds) between each interval processor iteration.

The value, you provide, must be an evenly divisible fraction of the smallest policy interval, which is one minute.

Default value is 15.

Note:

Exercise caution when you modify this setting as this is a system level setting.

apig_rest_service_redirect

Specifies whether the incoming API requests must be redirected to the directives `/ws` and `/mediator` for providing Mediator compatible endpoints.

Possible values:

- `true`. The incoming API requests are redirected to the directives, `/ws`, `/gateway` or `/mediator` to provide Mediator compatible endpoints.
 - `false`. The incoming API requests are not redirected to any directive. This is the default value.
-

apig_schemaValidationPoolSize

Parameter and Description

Specifies the pool size for the XML schema parsers. API Gateway uses parsers to validate the XML payload against the XML schema when you have selected Schema in the **Validate API Specification** policy of the API.

The default value is 10. Provide a bigger value to increase the performance of API Gateway in validating the XML schema of APIs.

apiGroupingPossibleValues

Specifies the names of API groups. You can organize your APIs by associating them to the relevant API groups. The groups provided, by default, are:

- Finance Banking and Insurance
- Sales and Ordering
- Search
- Transportation and Warehousing

You can add, edit, or delete API groups based on your requirement.

apiKeyExpirationPeriod

Specifies the time for which an API Key is valid. You can provide the value in seconds, minutes, days, months, or years. For example, 8 seconds, 8s, 10 months, 10m, 15 minutes, 15min.

The expiration date is computed as follows:

- When a new application is created: Expiration date = The time when an application is created + The value specified in the apiKeyExpirationPeriod parameter.
- When an API access key is regenerated: Expiration date = The time when the API key is regenerated + The value specified in the apiKeyExpirationPeriod parameter.

If you do not specify a value, then the API key never expires.

apiKeyHeader

Specifies the HTTP header name from which API Gateway retrieves the API key from incoming client requests.

The default value is x-Gateway-APIKey.

apiMaturityStatePossibleValues

Specifies the API maturity state values that can be set for an API. You can search for APIs based on their maturity status.

The default values provided are Beta, Deprecated, Experimental, Production, and Test.

backupSharedFileLocation

Parameter and Description

Specifies the file location where the data backup file has to be archived. The default location is `SAGInstallPath/profiles/IS_default/workspace/temp/`.

The files are saved with the corresponding timestamp in the specified location. Only the run-time events are included in the archives.

clusterNotifierCacheInterval

Specifies the time interval after which data in the `ClusterNotifierCache` is considered stale and is removed from the cache.

The default value is 900 seconds. If you provide a non-numeric value, API Gateway interprets the value as the default value of 900 seconds. If you provide a value less than 60 seconds, API Gateway interprets the value as the lower limit of 60 seconds.

The `ClusterNotifierCache` maintains a data structure which has an entry for each active member in a cluster. This data structure is used to communicate changes on API definitions, applications, policies, and so on, between the cluster members. When a cluster member shuts down gracefully it removes its entry from the data structure. However, when a cluster member process is killed the entry remains, and other cluster members continue posting notifications to the entry. In order to avoid endless growing resulting in performance degradation the cluster data structure is monitored for absent cluster members. If a cluster member has not reacted within the configured `clusterNotifierCacheStaleInterval` it is regarded as stale, and its data is removed from the `ClusterNotifierCache`.

customCertificateHeader

Specifies the header name of the request header in which the client certificate can be passed. API Gateway checks for the existence of this header and fetches the certificate and identifies the application.

The default value is `X-Client-Cert`.

The certificate included in the custom header can be in the following formats:

- Base64 encoded PEM certificate with `BEGIN CERTIFICATE` and `END CERTIFICATE` delimiters
- Non-Base64 encoded PEM certificate with `BEGIN CERTIFICATE` and `END CERTIFICATE` delimiters.
- PEM certificate can be without `BEGIN CERTIFICATE` and `END CERTIFICATE` delimiters if a single certificate is added.
- URL encoded PEM certificate with `BEGIN CERTIFICATE` and `END CERTIFICATE` delimiters.
- URL encoded PEM certificate can be without the `BEGIN CERTIFICATE` and `END CERTIFICATE` delimiters if a single certificate is added.

Parameter and Description

PEM formatted certificates are defined to be Base64 encoded. Here, the certificate with delimiters are Base64 encoded to avoid the stripping off of the delimiters.

decodeAllDelimitersInURI

Specifies whether the encoded characters of URL in the incoming client requests must be decoded or not. The URLs are encoded as per the URI specs or user's requirements.

Possible values:

- `true`. The encoded characters in the URL of incoming requests are decoded.
- `false`. The encoded characters in the URL of the incoming client requests are not decoded. This is the default value.

defaultEncoding

Specifies the format for encoding the design time and run time invocation data.

The default value is UTF-8.

If you want to modify this value, Software AG recommends that you do it before starting API Gateway.

You can change the value in the `gateway-core.xml` file located in the folder:

`SAGInstallDir\IntegrationServer\instances\instance-name\packages\WmAPIGateway\config\resources\beans.`

The property to be modified is: `<entry key="defaultEncoding" value="UTF-8"/>`.

Note:

Changing this value after starting API Gateway might make API Gateway non-functional. If API Gateway is clustered, then the same value should be updated in all nodes of API Gateway.

defaultLanguage

Specifies the display language for the user interface of API Gateway. You can change the display language to your preferred language at any time.

The default value is `en`.

defaultSearchResultSize

Specifies the maximum number of transactions the API Gateway Search API returns in its response. If the search result exceeds the value of this setting, then you can navigate through the search results by specifying the range of records that you want to view. For example, if the value specified in the `defaultSearchSize` setting is `1000` and the count of your search result is `5000`, then only the first `1000` records are displayed. To view the consequent records, you can specify the number of the record from which you want to view, and the number of records that must be displayed. That is, to view the records from `1001` to `2000`, you can specify the range as follows:

Parameter and Description

```
POST http://localhost:5555/rest/apigateway/search
{
  "types": [
    "TRANSACTION_EVENTS" ],
  "scope": [
    { "attributeName": "responseCode",
      "keyword": "304"
    },
  ],
  "from": "1001"
  "size": "1000"
}
```

disableRemoteEntityReference

Specifies whether remote entity references are resolved when creating a SOAP API.

Possible values:

- `true`. Disables the resolving of remote entity references when creating SOAP APIs.
- `false`. Enables the resolving of remote entity references when creating SOAP APIs.

enableHotdeploy

Specifies whether the hot deploy functionality must be enabled or disabled. When this setting is enabled, you can modify APIs that are active.

Possible values:

- `true`. Enables the hot deploy function. This is the default value.
- `false`. Disables the hot deploy function.

Note:

Ensure to refresh your browser when you modify this setting to reflect the changes to the ongoing user sessions.

enableImportBackup

This parameter provides an option to secure an API before overwriting it.

Available values are:

- `true`. If you set the value to `true`, the existing API is restored in the event of an error during the import. This is the default value.
- `false`. If you set the value to `false`, the feature to secure an existing API before overwriting is disabled. If the overwrite fails due to an error during the import, the existing API has to be deleted.

enableTeamWork

Parameter and Description

Specifies whether the Team Support feature in API Gateway is enabled.

Possible values:

- `true` - enables the Team Support feature.
- `false` - disables the Team Support featured. This is the default value.

For details on the feature, see [“Team Support” on page 802](#).

esScrollTimeout

Specifies the time, in milliseconds, that the search results for each request must be kept active in Elasticsearch.

When the **allowExceedMaxWindowSize** setting is enabled, the **Scroll** feature of the Elasticsearch is enabled to allow Elasticsearch to accept multiple search requests and return multiple results. That is, you can perform multiple search requests using the same query till you get the desired number of records from Elasticsearch.

When you send a search request, Elasticsearch returns the result and keeps the result active for the time specified in the **esScrollTimeOut** setting. If a request exceeds the time specified in the **esScrollTimeOut** setting, then the subsequent search requests also fail with a *Invalid Scroll ID* error message. For more information on the Scroll feature, refer <https://www.elastic.co/guide/en/elasticsearch/reference/current/search-request-body.html#request-body-search-scroll>.

events.collectionPool.maxThreads

Specifies the maximum number of threads to be used for the event data collection pool.

Each thread in this pool is assigned a task of collecting the events like transactions, error and performance metrics, and processing them for sending to destinations such as API Gateway, Elasticsearch, API Portal, and so on.

Specifying more number of threads implies that the processing of events for sending to the desired destinations is faster. At the same time, it increases the usage of system resources, which could result in slower service execution.

This value must be greater than or equal to the value of **events.CollectionPool.minThreads**.

Default value is 8.

events.collectionPool.minThreads

Specifies the minimum number of threads to be used for the event data collection pool.

Each thread in this pool is assigned a task of collecting the events like transaction, error and performance metrics and processing them for sending to the desired destinations such as API Gateway, Elasticsearch, API Portal, and so on.

Specifying less number of threads implies that processing of events for sending to the desired destinations is slower.

Parameter and Description

Default value is 1.

events.collectionQueue.size

Specifies the size of the collection queue to be used during event data collection. This is used in connection with the **events.collectionPool.minThreads** and **events.collectionPool.maxThreads** settings.

When events like transaction, error events and performance metrics are generated during API invocations, they will be put in the collection queue for further processing. Each thread in the collection pool is assigned a task of collecting these events and processing them for sending to the desired destinations such as API Gateway, Elasticsearch, API Portal, and so on.

If the queue capacity is reached, then any additional event data would be lost. Hence it is better to increase this size when there is an increase in incoming traffic.

Specifying a large collection queue size and small collection thread pool size might cause delay in processing the events data but ensures all the data are processed.

On the other hand, specifying a small collection queue size and large collection thread pool size might cause faster processing of the events data but keeps the CPUs busier and at the same time when the traffic increases there is a possibility of data loss when the collection queue size is full.

Default value is 10000.

events.reportingPool.maxThreads

Specifies the maximum number of threads to be used for the event data reporting pool.

Each thread in this pool is assigned a task of sending the events like transaction, error and performance metrics to the desired destinations such as API Gateway, Elasticsearch, API Portal, and so on.

Specifying more number of threads implies that sending of events to the desired destinations is faster. At the same time it increases the usage of system resources, which could result in slower service execution.

This value must be greater than or equal to the value of **events.ReportingPool.minThreads**.

Default value is 4.

events.reportingPool.minThreads

Specifies the minimum number of threads to be used for the event data reporting pool.

Each thread in this pool is assigned a task of sending the events like transaction, error and performance metrics to the desired destinations such as API Gateway, Elasticsearch, API Portal, and so on.

Parameter and Description

Specifying less number of threads implies that sending of events to the desired destinations is slower.

Default value is 2.

events.reportingQueue.size

Specifies the size of the reporting queue to be used during event data reporting. This is used in connection with the **events.reportingPool.minThreads** and **events.reportingPool.maxThreads** settings.

When events like transaction, error events and performance metrics are generated during API invocations, they will be put in the collection queue for further processing. Each thread in the collection pool is assigned a task of collecting these events, processing them and put in the reporting queue for sending to the desired destinations such as API Gateway, Elasticsearch, API Portal, and so on.

If the reporting queue capacity is reached, then any additional event data would be lost. Hence it is better to increase this size when there is an increase in incoming traffic.

Specifying a large reporting queue size and small reporting thread pool size might cause delay in processing the events data but ensures all the data are processed.

On the other hand, specifying a small reporting queue size and large reporting thread pool size might cause faster processing of the events data but keeps the CPUs busier and at the same time when the traffic increases there is a possibility of data loss when the collection queue size is full.

Default value is 5000.

eventsRefreshInterval

Specifies the refresh interval for the events indices in seconds.

The default value is 10.

forwardInternalAPIsRequest

This parameter is required in the case of a paired gateway deployment scenario using an Advanced Edition license at DMZ.

Specifies whether the incoming requests are forwarded to internal APIs that are deployed in the green zone.

Possible values:

- `true`. API Gateway forwards the incoming requests to the internal APIs that are deployed in the green zone
- `false`. API Gateway does not forward the incoming requests. This is the default value.

Parameter and Description

Following are the internal APIs and their URIs for which this parameter is required and its value must be set to true:

- `getOAuthToken` - `/pub/apigateway/oauth2/getAccessToken`
- `OAuth Authorization` - `/pub/apigateway/oauth2/authorize`
- `getOpenIdToken` - `/pub/apigateway/openid/getOpenIDToken`
- `openIDCallbackService` - `/pub/apigateway/openid/openIDCallback`
- `getJWTToken` - `/pub/apigateway/jwt/getJsonWebToken`

forwardQueryParams

Specifies whether API Gateway should forward the query parameter sent by client and query parameters configured in Request Processing stage to the native service if the `$(sys_resource_path)` variable is not present in the Routing policy URL.

- `true`. API Gateway forwards the query parameters sent by client and query parameters configured in Request Processing stage to the native service even if the `$(sys_resource_path)` is not present in the Routing policy URL.
- `false`. API Gateway does not forward the query parameters to the native service if the `$(sys_resource_path)` is not available in the Routing policy.

invokeESB_asUser

Specifies the username of the user who can run IS services via policies configured in API Gateway.

If the `Run as User` value is included in the policy level for Invoke webMethods IS service section, then the value provided in the policy overrides the user provided in this setting.

If no value is provided in either of the above mentioned fields, then the user authenticated from the incoming request can invoke the IS service.

maxAllowedZipFileSize

Specifies the maximum size of the zip file that can be uploaded to create an API from the **Create API** screen.

Default value is 100000000.

maxBackupsLimit

Specifies the maximum number of backups that can be archived. The default value is 10. If you do not provide a value, then infinite number of archives can be stored.

The archives are saved in the location provided in the **backupSharedFileLocation** setting.

maxRegexLengthInSearchQuery

Parameter and Description

Specifies the maximum length of Regex parameter that you can use in Regex expression query.

Default value is 37000. This value can be increased based on the requirement.

maxWindowSize

Specifies the maximum number of search results that Elasticsearch can return for a single search request.

Using the **index.max_result_window** property in Elasticsearch, you can configure the maximum number of records to be retrieved in a single Elasticsearch request. To know more about the property, refer

<https://www.elastic.co/guide/en/elasticsearch/reference/current/index-modules.html>.

Default value is 10000.

If the value configured in the **index.max_result_window** setting of Elasticsearch is different from that of the default value, then it is recommended that you provide the same value in this field.

If you have a value greater than the configured value, then Elasticsearch displays an error message. Also, you must enable the **allowExceedMaxWindowSize** setting if the total number of results to be retrieved is more than the value specified here.

For example, if you have specified 1000 in this setting and the total number of records to be retrieved is 20000, then API Gateway sends 20 requests to retrieve all records provided the **allowExceedMaxWindowSize** setting is enabled. In the above example, if the **allowExceedMaxWindowSize** setting is not enabled, an error message is displayed as the value is lesser than the default value.

paginationPossibleValues

Specifies the list of possible values of the pagination size, that is, number of items listed per page.

The default values displayed for pagination options are 10, 20, 30, 40, 50. For example, if you select 20, then 20 items are displayed per page.

For example, if you change the values to 5, 10, 15, 20, 25 then pagination options displayed are 5, 10, 15, 20 and 25. So now when you select 15, the items displayed per page would be 15.

On modifying the value, you have to logout and re-login for the changes to reflect.

If you do not provide any value in the paginationPossibleValues field, the default value as configured by the parameter `apigw.num.results.pagination` in the configuration file located at `SAGInstallDir\profiles\IS_default\apigateway\config\uiconfiguration.properties` is considered.

pg_Active_OpenID_Provider

Parameter and Description

Specifies the unique identifier (ID) of an active OpenID Provider.

pg_Cache_autoScalerRunInterval

Specifies the run interval, in minutes, for the Auto Scaler thread.

Default value is 120 minutes.

The Auto Scaler thread checks for system memory load and adjusts the percentage of objects kept in the cache automatically.

pg_Cache_averageObjectSize

Specifies the value used to calculate the average size, in bytes, of the objects that can be loaded into cache.

The default value is 64.

Software AG recommends you not to modify this value.

pg_Cache_boundedCacheResizeRunInterval

Specifies the run interval, in minutes, for the bounded cache resize thread.

The default value is 30 minutes.

The bounded cache stores a predefined number of objects in memory. The cache value is configured in terms of percentage and it varies based on the new objects added to database. The bounded cache resize thread computes the memory size at the configured interval. For example, if you specify 50, the cache memory is calculated once every 50 minutes.

pg_Cache_maxCacheSize

Specifies the maximum number of objects that can kept in the unbounded cache memory.

The default value is 1048576.

Internally, unbounded cache is also a bounded cache with a configured maximum limit.

pg_Cache_minCachePercent

Specifies the minimum cache percent value. API Gateway maintains the configured memory size. That is, if the value that is computed by the auto scaler thread goes below this value, API Gateway resets the cache percent to the value specified for this setting.

The default value is 20.

The auto scaler thread computes the percentage of objects to be stored in cache. If it computes a lower percentage, say 2%, the value is reset based on the pg_Cache_minCachePercent value.

pg_Cache_minCacheSize

Specifies the minimum number of objects that are kept in the unbounded cache memory.

Parameter and Description

The default value is 1024.

For example, if there are 20 objects, and cache size is set to 100%, then the computed cache size is 20. When the actual size becomes lower than the value specified in this setting, then the value is reset to the `pg_Cache_minCacheSize` value.

pg_Cache_statisticsProcessorRunInterval

Specifies the run interval, in minutes, for the statistics processor thread. The statistics processor thread stores the cache statistics in Elasticsearch.

The default value is 15 minutes.

The cache statistics analytics page displays details about the cache statistics.

pg_JWT_isHTTPS

Specifies whether the transport protocol over which the JSON Web Tokens (JWTs) are granted authorization is restricted to HTTPS.

Possible values:

- `true`. Restricts the transport protocol to HTTPS. This is set by default.
- `false`. Allows HTTP and HTTPS transport protocols.

pg_oauth2_createDefaultScopes

Specifies whether default scopes must be created.

Possible values:

- `true`. When local authorization server is configured as the `default authorization server`, then API Gateway automatically creates a scope in Integration Server during API creation, and creates a scope mapping between the OAuth scope in the local authorization server and the API. This setting is useful in cases where a service is published from Centrasite.
- `false`. API Gateway does not create a scope automatically. Users must manually create and map scopes for the services published from Centrasite. This is the default value.

pg_oauth2_isHTTPS

Specifies whether the transport protocol over which the OAuth 2.0 access tokens are granted authorization is restricted to HTTPS.

Possible values:

- `true`. Restricts the transport protocol to HTTPS. This is set by default.
- `false`. Allows HTTP and HTTPS protocols.

pg_OpenID_isHTTPS

Parameter and Description

Specifies the transport protocol over which the OpenID (ID) tokens are granted authorization is restricted to HTTPS.

Possible values:

- `true`. Restricts the transport protocol to HTTPS. This is set by default.
 - `false`. Allows HTTP and HTTPS protocols.
-

pg_xslt_disableDoctypeDeclarations

Specifies whether the xslt doc type declarations must be disabled or not.

Possible values:

- `true`. Disables the xslt doc type declarations. This is the default value.
 - `false`. Enables the xslt doc type declarations.
-

pg_xslt_enableDOM

Specifies whether the DOM parsing must be enabled.

Possible values:

- `true`. Enables the DOM parsing.
 - `false`. Disables the DOM parsing and enables other parsers.
-

pg_xslt_enableSecureProcessing

Specifies whether the use of extensions must be disabled.

Possible values:

- `true`. Enables the use of extensions. This is the default value.
 - `false`. Disables the use of extensions.
-

pg.3pSnmpSender.sendDelay

This is an internal parameter. Do not modify.

pg.cs.snmpTarget.base64Encoded

This is an internal parameter. Do not modify.

pg.cs.snmpTarget.connTimeout

Specifies the number of milliseconds before an inactive connection to the SNMP target server is closed. If set to 0, the socket remains open indefinitely.

pg.cs.snmpTarget.maxRequestSize

Parameter and Description

Specifies the maximum size (in bytes) for SNMP traps.

The default value is 10485760.

pg.cs.snmpTarget.retries

Specifies the number of times to resend SNMP traps upon failure.

Default value is 1.

This parameter works with **pg.cs.snmpTarget.sendTimeOut** to determine the delay in re-sending SNMP traps to malfunctioning SNMP servers (that is, it retries*sendTimeOut).

This means that if the retries parameter is set to 3, and the **sendTimeOut** parameter is set to 500 milliseconds, there is a 1.5 second delay before the thread sending the alert is available to send another event. Malfunctioning event destinations could delay the amount of time it takes API Gateway to report events, or it could cause discarded events when the queue reaches its maximum level.

pg.cs.snmpTarget.sendTimeOut

Specifies the time (in milliseconds) to wait before the SNMP trap times out because the server destination is not responding.

This value schedules a timer that resends an SNMP event that has not yet completed when it expires. You must set a timeout value that ensures that the trap is sent to the SNMP server within the time frame. This parameter does not abort an event that is in progress. Set this parameter higher than the default when sending traps with large payloads. The default value is 500.

This parameter works with **pg.cs.snmpTarget.retries** to determine the delay in resending SNMP traps to malfunctioning SNMP servers (that is, it retries *sendTimeOut).

This means that if the retries parameter is set to 3, and the sendTimeOut parameter is set to 500 milliseconds, there is a 1.5 second delay before the thread sending the alert is available to send another event. Malfunctioning event destinations could delay the amount of time it takes API Gateway to report events, or it could cause discarded events when the queue reaches its maximum level.

pg.default.enable.oldVersion

Specifies whether the oldest version of an API has to be enabled when the version of an API is not specified.

For example, when you have an API that is versioned, the client can specify the version number in the URL to invoke that specific version of the API, that is, API-NAME/Version-number. When the client does not specify the version number, API Gateway defaults to the latest version of the API and invokes it. You can use this parameter to change this behavior such that API Gateway defaults to the oldest version of API and invokes it.

Parameter and Description

Available values are:

- `true`. When you set this parameter as `true` and invoke an API without specifying the version number, then API Gateway defaults to the oldest version of the specified API and invokes it.
- `false`. This is the default value. When you set this parameter as `false` and invoke an API without specifying the version number, then API Gateway defaults to the latest version of the specified API and invokes it.

pg.endpoint.connectionTimeout

Specifies the time interval (in seconds) after which an HTTP connection attempt times out.

The default value is 30 seconds.

This is a global property that applies to the endpoints of all APIs. If you prefer to specify a connection timeout for the endpoints of an API individually, set the `HTTP Connection Timeout` parameter in the API's Routing Protocols processing step, which would then take precedence over `pg.endpoint.connectionTimeout`.

The precedence of the Connection Timeout configuration is as follows:

- If you specify a value for the **Connection timeout** field in routing endpoint alias, then the **Connection timeout** value specified in the Endpoint alias section takes precedence over the timeout values defined at the API level and the global level.
- If you specify a value 0 for the **Connection timeout** field in routing endpoint alias, then API Gateway uses the value specified in the **Connection timeout** field in the routing protocol processing step of an API. The Read Timeout value specified at an API level takes precedence over the global configuration.
- If you specify a value 0 or do not specify a value for the **Connection timeout** field in the routing protocol processing step at the API level or specify a value 0 at an alias level, then API Gateway uses the value specified in this `pg.endpoint.connectionTimeout` property.
- If you do not specify any value for `pg.endpoint.connectionTimeout`, then API Gateway uses the default value of 30 seconds.

pg.endpoint.readTimeout

Specifies the time interval (in seconds) after which a socket read attempt times out. Default value: 30 seconds.

This is a global property that applies to all APIs. If you prefer to specify a read timeout for APIs individually, set the Read Timeout field in the API's Routing Protocols processing step, which would then take precedence over `pg.endpoint.readTimeout`.

The precedence of the Read Timeout configuration is as follows:

Parameter and Description

- a. If you specify a value for the **Read timeout** field in routing endpoint alias, then the **Read timeout** value specified in the Endpoint alias section takes precedence over the timeout values defined at the API level and the global level.
- b. If you specify a value 0 for the **Read timeout** field in routing endpoint alias, then API Gateway uses the value specified in the **Read Timeout** field in the routing protocol processing step of an API. The Read Timeout value specified at an API level takes precedence over the global configuration.
- c. If you specify a value 0 or do not specify a value for the **Read timeout** field in the routing protocol processing step at the API level or specify a value 0 at an alias level, then API Gateway uses the value specified in this `pg.endpoint.readTimeout` property.
- d. If you do not specify any value for `pg.endpoint.readTimeout`, then API Gateway uses the default value of 30 seconds.

pg.lb.failoverOnDowntimeErrorOnly

Specifies API Gateway's behavior of endpoints in a load-balanced routing scenario.

Possible values:

- `true`. Load balancing does not happen when the service fault encountered in the response is a downtime error. This is the default value.

For example, the following are some Downtime exceptions and for which fail overs happen: `ConnectException`, `MalformedURLException`, `NoRouteToHostException`, `ProtocolException`, `SocketTimeoutException`, `UnknownHostException`, `UnknownServiceException`, Web Service not available, The service cannot be found.

- `false`. Load balancing happens whenever a service fault is encountered in the response coming from endpoint 1 and API Gateway immediately tries the next configured endpoint. There is no distinction on the type of fault present in the response from endpoint.

pg.MTOMStreaming.cachedFiles.delete.interval

Specifies the interval (in seconds) to delete the cached MTOM files.

Default value is 3600 seconds. This property takes effect only when the **pg.suppress.IS.lcm** setting is set to `true`.

pg.nativeServer.validatePrivateIPs

Specifies whether to validate the native server endpoint against the private IP configuration during invocation of an API.

Possible values:

- `true`. If this is set to true, then the API invocation is not allowed if the native endpoint uses any private IPs other than the ones configured in `/rest/apigateway/configurations/whiteListingIPs`.

Parameter and Description

- `false`. If this is set to `false`, no private IP validation is done for the native endpoint.
Default value is `false` for on-prem and `true` for cloud installations.
-

pg.override.users.withsameloginid

Specifies whether API Gateway must validate and overwrite a user based on their login ID or UUID when importing or promoting the user or team from one stage to another, provided the logged-in credentials (login ID) of both instances are the same.

Possible values:

- `true`. API Gateway validates and overwrites the user with the user's login ID, if the user is not matched with the UUID during import or promotion of a user or teams. The default value is `true`.
 - `false`. API Gateway validates and overwrites the user with the UUID.
-

pg.removeSSNID

Specifies whether to include the set-cookie header in the response header. The set-cookie header contains `ssnid` value sent from the native service.

Possible values:

- `true`. The client that invokes API Gateway API, does not receive the set-cookie header that contains `ssnid` value.
 - `false`. The client that invokes API Gateway API, receives the set-cookie header that contains `ssnid` value if the native service sends this in the response.
-

pg.security.allowedhostnames

Validates the host header in an incoming request against the host names configured in this setting. Provide a comma-separated list of host names that you want the host header in the incoming request to be validated against.

This setting is required especially in scenarios where, when an API is invoked by adding a dummy host in the Host header. The invocation gets through the proxy API and API Gateway returns success response, which can pose as a vulnerability.

If the host in the request header is not in the list of host names configured in this setting, API Gateway returns a 400 Bad Request with an error response `Invalid Host header`.

If you do not include any host names and leave the setting blank, the host header in the incoming request is not validated.

pg.snmp.communityTarget.base64Encoded

Specifies whether to use a third-party SNMPv1 community-based connection.

The default value is `false`.

Parameter and Description

When this property is set to `true`, the Community name of 3rd Party SNMP destination configuration is expected as base64 encoded.

pg.snmp.communityTarget.maxRequestSize

Specifies the maximum size (in bytes) for SNMP traps.

The default value is 65535.

pg.snmp.communityTarget.retries

Specifies the number of times to resend SNMP traps upon failure.

Default value is 1.

This parameter works with **pg.snmp.communityTarget.sendTimeOut** to determine the delay in re-sending SNMP traps to malfunctioning SNMP servers (that is, it retries *sendTimeOut).

This means that if the retries parameter is set to 3, and the **sendTimeOut** parameter is set to 500 milliseconds, there is a 1.5 second delay before the thread sending the alert is available to send another event. Malfunctioning event destinations could delay the amount of time it takes API Gateway to report events, or it could cause discarded events when the queue reaches its maximum level.

pg.snmp.communityTarget.sendTimeOut

Specifies the time (in milliseconds) to wait before the SNMP trap times out because the server destination is not responding. This value schedules a timer that resends an SNMP event that has not yet completed when it expires. You must set a timeout value that ensures that the trap is sent to the SNMP server within the time frame. This parameter does not abort an event that is in progress. Set this parameter higher than the default when sending traps with large payloads.

The default value is 500.

This parameter works with **pg.snmp.communityTarget.retries** to determine the delay in re-sending SNMP traps to non-responsive SNMP servers (that is, it retries *sendTimeOut).

This means that if the retries parameter is set to 3, and the **sendTimeOut** parameter is set to 500 milliseconds, there is a 1.5 second delay before the thread sending the alert is available to send another event. Malfunctioning event destinations could delay the amount of time it takes API Gateway to report events, or it could cause discarded events when the queue reaches its maximum level.

pg.snmp.customTarget.connTimeout

Specifies the number of milliseconds before an inactive connection to the third-party SNMP server is closed. If set to 0, the socket remains open indefinitely.

pg.snmp.userTarget.maxRequestSize

Parameter and Description

Specifies the maximum size (in bytes) for SNMP traps.

The default value is 65535.

pg.snmp.userTarget.retries

Specifies the number of times to resend SNMP traps upon failure.

The default value is 1.

This parameter works with **pg.snmp.userTarget.sendTimeOut** to determine the delay in re-sending SNMP traps to malfunctioning SNMP servers (that is, it retries *sendTimeOut).

This means that if the retries parameter is set to 3, and the **sendTimeOut** parameter is set to 500 milliseconds, there is a 1.5 second delay before the thread sending the alert is available to send another event. Malfunctioning event destinations could delay the amount of time it takes API Gateway to report events, or it could cause discarded events when the queue reaches its maximum level.

pg.snmp.userTarget.sendTimeOut

Specifies the time (in milliseconds) to wait before the SNMP trap times out because the server destination is not responding. This value schedules a timer that resends an SNMP event that has not yet completed when it expires. You must set a timeout value that ensures that the trap is sent to the SNMP server within the time frame. This parameter does not abort an event that is in progress. Set this parameter higher than the default when sending traps with large payloads.

The default value is 500.

This parameter works with **pg.snmp.userTarget.retries** to determine the delay in resending SNMP traps to malfunctioning SNMP servers (that is, it retries *sendTimeOut).

This means that if the retries parameter is set to 3, and the **sendTimeOut** parameter is set to 500 milliseconds, there is a 1.5 second delay before the thread sending the alert is available to send another event. Malfunctioning event destinations could delay the amount of time it takes API Gateway to report events, or it could cause discarded events when the queue reaches its maximum level.

pg.soapToRest.typeConvertorEnabled

Specifies whether the key values in a SOAP request must be converted to their primitive type when a SOAP API is transformed to REST API. For example, if the XML is `<number>10</number>`, it is converted as "number" : 10.

Possible values:

- `true`. Values are converted to their primitive type. This is the default value.
 - `false`. Values are not converted to their primitive type.
-

Parameter and Description

pg.suppress.IS.lcm

Specifies whether to override IS lifecycle manager with API Gateway lifecycle manager.

Possible values:

- `true`. Set this to true to use the API Gateway lifecycle manager.
- `false`. Set this to false to use the IS lifecycle manager. By default, the value is false.

API Gateway lifecycle manager provides options to specify the interval for deleting of cached MTOM files using the **pg.MTOMStreaming.cachedFiles.delete.interval** setting.

pg.uddiClient.publish.maxThreads

Specifies the maximum allowed number of threads to publish the performance metrics data to CentraSite.

The default value is 2.

pg.uddiClient.uddiClientTimeout

Specifies the connection and read timeout, in milliseconds, for publishing performance metrics to CentraSite.

The default value is 5000.

pgmen.quotaSurvival.addLostIntervals

Specifies whether the lost intervals from the recovered quota should be added.

pgmen.quotaSurvival.interval

Specifies the number of seconds that elapses before the quota survival processor task runs.

portClusteringEnabled

Specifies whether the port configuration synchronization across API Gateway cluster nodes is enabled or disabled.

When as an API Gateway Administrator you create, update or delete a port definition, you might prefer to do it in one of the nodes in the cluster and have it instantly reflected on the other nodes without having to restart them. This setting controls whether you want to synchronize these port configuration changes across the API Gateway cluster nodes.

In this context clustered environment refers to clustering being enabled under **Administration > General > Clustering** section.

In a clustered environment, the possible values are:

- `true`. This enables port configuration synchronization across API Gateway cluster nodes. This is the default value

Parameter and Description

The ports that are not configured through the API Gateway UI (that is ports configured through Integration Server) are ignored by API Gateway UI and are removed on a restart.

- `false`. This disables port configuration synchronization across API Gateway cluster nodes. The ports that are not configured through the API Gateway UI (that is ports configured through Integration Server) are displayed in the API Gateway UI and they are not removed on a restart.

In a non-clustered environment, the possible values are:

- `true`. The ports that are not configured through the API Gateway UI (that is ports configured through Integration Server) are ignored by API Gateway UI and are removed on a restart.
- `false`. The ports that are not configured through the API Gateway UI (that is ports configured through Integration Server) are displayed in the API Gateway UI and they are not removed on a restart.

Note:

Restart all the cluster nodes, when you change this parameter, for the changes to take effect.

promotionListSortByTime

Specifies whether the list of promotions in the **Promotions** tab of the **Promotion management** page are sorted by the promotion time.

Possible values:

- `true`. Promotion list is sorted by promotion time with the latest on top.
- `false`. Promotion list is sorted by promotion name. This is the default value.

retainResponseStatus

Specifies whether the native service status has to be sent to the client when an Outbound Authentication policy action is applied to the API.

Possible values:

- `true`. The response status received from the native service is passed on to API clients.
- `false`. The response status received from the native service is not exposed to API clients. This implies that the 401 HTTP response status resulting from failing outbound authentications are not exposed to the API clients. This is the default value.

return408ForConnectionTimeout

Specifies the status code to be included in the response when a request to the native service times out.

Possible values:

Parameter and Description

- `true`. The response contains 408 error code uniformly when a request to the native service is timed out. This is the default value.
- `false`. The response does not contain 408 error code.

saveAuditlogsWithPayload

Specifies whether the audit logs have to be saved along with payload.

Possible values:

- `true`. The audit logs have to be saved along with payload. This is the default value.
- `false`. The audit logs are not saved along with payload.

sendClientRequestURI

Specifies whether the URI that is present in a request must be decoded before sending it to the native service.

Possible values:

- `true`. The URI is not decoded. The unicode characters in a request are encoded.
- `false`. The URI is decoded without change in the path of the URL. This is the default value.

setDefaultContentType

Specifies that default content type to be included in the GET and DELETE methods, if the content type is missing in request.

Possible values:

- `true`. The default content type **application/x-www-form-urlencoded** is added, if content type is missing in request. This is the default value.
- `false` -The default content type is not added. The content-type is sent as is.

strictResourceMatching

Specifies whether API Gateway must perform a strict matching of the resource path from runtime invocation with the API definition of resource paths.

Possible values:

- `true`. API Gateway uses the strict resource matching criteria and the matching fails if it encounters any other characters than that are specified. This is the default value.
- `false`. API Gateway matches the best resource instead of using the strict resource matching criteria.

transferEncodingChunked_handleAsStream

Parameter and Description

Specifies whether the request and response payloads should be handled as stream.

Possible values:

- `true`. API Gateway checks for the following conditions to handle the request and response payloads as stream:
 - Request payload is handled as stream, if the `Transfer-encoding: chunked` header is sent in the request.
 - Response payload is handled as stream, if the `application/octet-stream` accept header is sent in the request.
- `false`. The payloads are not handled as streams (even though the conditions specified above are met). This is the default value.

Note:

The response from the native endpoint need not contain the `Transfer-encoding: chunked` header for the response payload to be handled as stream.

transformerPoolSize

Specifies the maximum size for transform pool that consists XSLT transformers. To reduce performance impacts, you can reuse the transformers from the pool instead of creating them for every request.

useTypeInIndexNameForESDestination

Specifies the index format used when sending data from API Gateway to a configured Elasticsearch destination.

The default value is *true*.

Elasticsearch version 7.2 supports data with dedicated index for each of the event types. So, API Gateway sends data in different indexes for different event types if the value of this property is set to `true`. The event type is concatenated with the Elasticsearch destination index name. That is, the index name will be in the following format: `{IndexName}_{EventType}`. For example, `database_transactionalevents`; where *default* is the index name and *transactional events* in the event type.

If the value is set to `false`, the type name is not concatenated with the index name. In this case, the index created for each event is sub-indexed under a main index when data is sent to configured Elasticsearch destination. The value of this setting must be set to `false`, if the version of the destination Elasticsearch is 5.x or earlier.

validateTransportProtocolAgainstEGPort

Specifies whether the EG protocol in the transport protocol policy of APIs must be validated during the reverse invoke of APIs.

Possible values:

Parameter and Description

- `true`. Validation is performed to ensure an EG protocol is specified in transport protocol policy of APIs that are invoked using the reverse invoke method.
- `false`. No validation is performed. The default value is `false`.

wsdIPortLayout

Specifies which ports entries are exposed in a WSDL.

Possible values:

- `service-port`. All the port entries are exposed. This is the default value.
- `service-only`. Only one port (with servicename or version) is exposed. When this value is set, only the simple endpoint with the service name is generated.
- `mediator-comp`. When this value is set, the entries generated are in mediator compatibility mode. Note that custom endpoints do not appear in this case.

5. You can configure the watt parameters in the Watt keys section by providing the required values.

The configured watt keys are listed under Watt settings below the Extended keys at the top of the page.

Only the following watt parameters get synchronized across nodes in a cluster setup:

- `watt.security.ssl.client.ignoreEmptyAuthoritiesList`
- `watt.security.ssl.ignoreExpiredChains`
- `watt.server.url.alias.partialMatching`
- `watt.server.oauth.authServer.alias`
- `watt.server.oauth.requireHTTPS`
- `watt.net.http401.throwException`
- `watt.net.http501-599.throwException`
- `watt.server.SOAP.MTOMStreaming.enable`
- `watt.server.http.Strict-Transport-Security`
- `watt.server.rest.removeInputVariablesFromResponse`
- `watt.server.coder.responseAsXML`
- `watt.security.ssl.cacheClientSessions`
- `watt.server.enterprisegateway.ignoreXForwardedForHeader`

If you modify the other watt parameters in one API Gateway instance they do not get synchronized across other nodes in a cluster setup. You have to manually modify them in the other instances.


6. Click **Save**.

Configuring API Fault Settings

You must have the API Gateway's manage user administration functional privilege assigned to configure the API fault settings.

You can configure global error responses for all APIs to display the default error message.

» To configure the service fault settings

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > API fault**.
3. Select **Send native provider fault** to send the native API provider's fault content, if available. API Gateway ignores the default error message and sends whatever content it receives from the native API provider.

If you do not select this option then API Gateway sends the default error message.

4. Specify the error message text in the **Default error message** text box that is sent out when the **Send native provider fault** is not selected.
5. Click **Save**.

Approval Configuration

API Gateway allows you to configure an approval process to ensure that the requests are valid. If the requests are invalid, API Gateway enables approvers to reject the requests. API Gateway allows you to configure approvals for:

- Create application: To enforce approvals for creating an application in API Gateway.
- Register application: To enforce approvals for associating an application with APIs.
- Update application: To enforce approvals for modifying an application.
- Subscribe package: To enforce approvals in API Gateway to enable API Portal consumers for subscribing a package to a plan in API Portal.

In API Gateway, you can create an application and associate (register) the application created with APIs. If you have the API Gateway's manage general administration configurations functional privilege assigned, you can configure approvals for creating or registering an application. If you

have configured approvers, and if a user wants to create and register an application in API Gateway, then the application is created and registered with an API only if any one approver from the approvers group approves the create application and the register application requests. Similarly, you can configure approvals for updating an application or subscribing a package.

You can configure a set of different approvers for creating or registering applications. For example, if you have configured an approvers group, *group1* to approve or reject a create application request and an approvers group, *group2* to approve or reject a register application request, then when a user creates and registers an application in API Gateway, then the create application request must be approved by any one approver in *group1*. When the application is created, the register application request is sent to the approvers in *group2* and this register application request must be approved by any one approver in *group2*. When the request is approved, the application created is registered to an API. However, if any one user from *group2* rejects the request, then the application gets created but is not register to an API.

API Gateway approvals can also be used when API Gateway is integrated with API Portal and an API Portal API consumer uses the API get access token to register an application to an API in API Gateway. In this scenario, API Portal implicitly sends a request to API Gateway to create an application. When the approvers approve the create application request, an application is created in API Gateway and then API Gateway initiates a register application request and the approvers approve the register application request. The application is registered to an API which is published to API Portal. The consumer is now able to view and manage the API in API Portal.

Note:


You can customize the Approval page view by modifying the file OAuth_Approval.html located at `SAG_Root\IntegrationServer\instances\instance_name\packages\WmAPIGateway\templates`.

Similarly, you can configure approvals for subscribing a package in API Gateway, when an API consumer subscribes to a package in API Portal. API Gateway receives the package subscription request and the approvers in the approvers group approve or reject the request for subscribing a package. When the request is approved, the acknowledgement is sent to API Portal and the package is subscribed.

Configuring Approvals for Creating an Application

You have to configure the approval settings, to enforce approval for creating an application in API Gateway.

➤ To configure approvals for creating an application

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Approval configuration > Create application**.
3. Set the **Enable** toggle button to the on position to enable approval configuration to take effect.
4. Select the team of approvers from the **Approvers** drop-down list.

5. Select *Anyone* from the **Approved by** drop-down list.

This implies that, any one user of the selected team can approve or reject the requests. The requester need not wait for the approval of each approver in the approvers group.

Note:

If a user is associated with the selected team, then the user can approve or reject a pending request even if the user is not associated with the **Approvers** group.

6. Select **Configure approval initiate request mail template to be sent to approver**.

This is to configure the email template to be sent to the approver, for approving the request of creating an application.

7. Provide the following information in the **Configure approval initiate request mail template to be sent to approver** section:

Field	Description
Send notification	To send an email notification to the approver to approve the request for creating an application.
Subject	The subject line of the email to be sent.
Content	By default, the template appears. You can customize the email content.

Note:

The @ character acts as a place holder and the values are automatically generated by the system. For example, Hello @approver.name appears as Hello Joe in the email sent, where Joe is the approvers login ID.

Note:

The email notifications are sent only to the local API Gateway users.

8. Select **Configure request approved mail template to be sent to requester**.

This is to configure the email template to be sent to the requester to notify that the request for creating an application is approved.

9. Provide the following information in the **Configure request approved mail template to be sent to requester** section:

Field	Description
Send notification	To send an email notification to the requester that the request for creating an application is approved by the approver.
Subject	The subject line of the email to be sent.

Field	Description
Content	By default, the template appears. You can customize the email content. Note: The @ character acts as a place holder and the values are automatically generated by the system. For example, Approval of @event.type appears as Approval of Create Application in the email sent, where Create Application is the event.type.

10. Select **Configure rejection mail template to be sent to requester**.

This is to configure the email template to be sent to the requester to notify that the request for creating an application is rejected.

11. Provide the following information in the **Configure rejection mail template to be sent to requester** section:

Field	Description
Send notification	To send an email notification to the requester that the request for creating an application is rejected by the approver.
Subject	The subject line of the email to be sent.
Content	By default, the template appears. You can customize the email content. Note: The @ character acts as a place holder and the values are automatically generated by the system. For example, Approval of @event.type appears as Approval of Create Application in the email sent, where Create Application is the event.type.


12. Click **Cancel** to revert to the last saved changes or to abandon all the changes if the values are not saved.

13. Click **Save**.

Configuring Approvals for Registering Application

You have to configure the approval settings, to enforce approval for associating an application with APIs in API Gateway. The API Portal API get access token request is considered as a create and registering application event in API Gateway.

» To configure approvals for registering an application

1. Expand the menu options icon , in the title bar, and select **Administration**.

2. Select **General > Approval configuration > Register application**.
3. Set the **Enable** toggle button to the on position to enable approval configuration to take effect.
4. Select the team of approvers from the **Approvers** drop-down list.
5. Select the *Anyone* from the **Approved by** drop-down list.

This implies that, any user of the selected team can approve or reject the requests. The requester need not wait for the approval of each approver in the approvers group.

Note:

If a user is associated with the selected team, then the user can approve or reject a pending request even if the user is not associated with the **Approvers** group.

6. Select **Configure approval initiate request mail template to be sent to approver**.

This is to configure the email template to be sent to the approver for associating an application with APIs.

7. Provide the following information in the **Configure approval initiate request mail template to be sent to approve** section:

Field	Description
Send notification	To send an email notification to the approver to approve the request for associating an application with APIs.
Subject	The subject line of the email to be sent.
Content	By default, the template appears. You can customize the email content. Note: The @ character acts as a place holder and the values are automatically generated by the system. For example, Hello @approver.name appears as Hello Joe in the email sent, where Joe is the approvers login ID.

Note:

The email notifications are sent only to the local API Gateway users.

8. Select **Configure request approved mail template to be sent to requester**.

This is to configure the email template to be sent to the requester to notify that the request for associating an application with APIs is approved.

9. Provide the following information in the **Configure request approved mail template to be sent to requester** section:

Field	Description
Send notification	To send an email notification to the requester that the request for associating an application with APIs is approved by the approver.
Subject	The subject line of the email to be sent.
Content	By default, the template appears. You can customize the email content.

Note:
The @ character acts as a place holder and the values are automatically generated by the system. For example, Approval of @event.type appears as Approval of Register Application in the email sent, where Register Application is the event.type.

10. Select **Configure rejection mail template to be sent to requester**.

This is to configure the email template to be sent to the requester to notify that the request for registering an application is rejected.

11. Provide the following information in the **Configure rejection mail template to be sent to requester** section:

Field	Description
Send notification	To send an email notification to the requester that the request for associating an application with APIs is rejected by the approver.
Subject	The subject line of the email to be sent.
Content	By default, the template appears. You can customize the email content.

Note:
The @ character acts as a place holder and the values are automatically generated by the system. For example, Approval of @event.type appears as Approval of Register Application in the email sent, where Register Application is the event.type.


12. Click **Cancel** to revert to the last saved changes or to abandon all the changes if the values are not saved.

13. Click **Save**.

Configuring Approvals for Updating Application

You have to configure the approval settings, to enforce approval for modifying an existing application in API Gateway.

➤ **To configure approvals for updating an application**

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Approval configuration > Update application**.
3. Set the **Enable** toggle button to the on position to enable approval configuration to take effect.
4. Select the team of approvers from the **Approvers** drop-down list.
5. Select *Anyone* from the **Approved by** drop-down list.

This implies that, any one user associated with the selected team can approve or reject the requests. The requester need not wait for the approval of each approver in the approvers group.

Note:

If a user is associated with the team, then the user can approve or reject a pending request even if the user is not associated with the **Approvers** group.

6. Select **Configure approval initiate request mail template to be sent to approver**.

This is to configure the email template to be sent to the approver for approving the request of modifying an existing application.

7. Provide the following information in the **Configure approval initiate request mail template to be sent to approver** section:

Field	Description
Send notification	To send an email notification to the approver to approve the request for modifying an existing application.
Subject	The subject line of the email to be sent.
Content	By default, the template appears. You can customize the email content.

Note:

The @ character acts as a place holder and the values are automatically generated by the system. For example, Hello @approver.name appears as Hello Joe in the email sent, where Joe is the approvers login ID.

Note:

The email notifications are sent only to the local API Gateway users.

8. Select **Configure request approved mail template to be sent to requester**.

This is to configure the email template to be sent to the requester to notify that the request for modifying an existing application is approved.

9. Provide the following information in the **Configure request approved mail template to be sent to requester** section:

Field	Description
Send notification	To send an email notification to the requester that the request for modifying an existing application is approved by the approver.
Subject	The subject line of the email to be sent.
Content	By default, the template appears. You can customize the email content. <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: The @ character acts as a place holder and the values are automatically generated by the system. For example, Approval of @event.type appears as Approval of Update Application in the email sent, where Update Application is the event.type.</p> </div>

10. Select **Configure rejection mail template to be sent to requester**.

This is to configure the email template to be sent to the requester to notify that the request for modifying an existing application is rejected.

11. Provide the following information in the **Configure rejection mail template to be sent to requester** section:

Field	Description
Send notification	To send an email notification to the requester that the request for modifying an existing application is rejected by the approver.
Subject	The subject line of the email to be sent.
Content	By default, the template appears. You can customize the email content. <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: The @ character acts as a place holder and the values are automatically generated by the system. For example, Approval of @event.type appears as Approval of Update Application in the email sent, where Update Application is the event.type.</p> </div>


12. Click **Cancel** to revert to the last saved changes or to abandon all the changes if the values are not saved.

13. Click **Save**.

Configuring Approvals for Subscribing Package

You have to configure the approval settings in API Gateway, to enforce approval for subscribing a package to a plan in API Portal.

➤ To configure approvals for subscribing a package

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Approval configuration > Subscribe package**.
3. Set the **Enable** toggle button to the on position to enable approval configuration to take effect.
4. Select the team of approvers from the **Approvers** drop-down list.
5. Select *Anyone* from the **Approved by** drop-down list.

This implies that, any one user associated with the team can approve or reject the requests. The requester need not wait for the approval of each approver in the approvers group.

Note:

If a user is associated with the selected team, then the user can approve or reject a pending request even if the user is not associated with the **Approvers** group.

6. Select **Configure approval initiate request mail template to be sent to approver**.

This is to configure the email template to be sent to the approver for subscribing a package.

7. Provide the following information in the **Configure approval initiate request mail template to be sent to approve** section:

Field	Description
Send notification	To send an email notification to the approver to approve the request for subscribing a package.
Subject	The subject line of the email to be sent.
Content	By default, the template appears. You can customize the email content.

Note:

The @ character acts as a place holder and the values are automatically generated by the system. For example, Hello @approver.name appears as Hello Joe in the email sent, where Joe is the approvers login ID.

Note:

The email notifications are sent only to the local API Gateway users.

8. Select **Configure request approved mail template to be sent to requester**.

This is to configure the email template to be sent to the requester to notify that the request for subscribing a package is approved.

9. Provide the following information in the **Configure request approved mail template to be sent to requester** section:

Field	Description
Send notification	To send an email notification to the requester that the request for subscribing a package is approved by the approver.
Subject	The subject line of the email to be sent.
Content	By default, the template appears. You can customize the email content. <div style="background-color: #f0f0f0; padding: 5px; margin-top: 5px;"> <p>Note: The @ character acts as a place holder and the values are automatically generated by the system. For example, Approval of @event.type appears as Approval of Subscribe Package in the email sent, where Subscribe Package is the event.type.</p> </div>

10. Select **Configure rejection mail template to be sent to requester**.

This is to configure the email template to be sent to the requester to notify that the request for subscribing a package is rejected.

11. Provide the following information in the **Configure rejection mail template to be sent to requester** section:

Field	Description
Send notification	To send an email notification to the requester that the request for registering an application is rejected by the approver.
Subject	The subject line of the email to be sent.
Content	By default, the template appears. You can customize the email content. <div style="background-color: #f0f0f0; padding: 5px; margin-top: 5px;"> <p>Note: The @ character acts as a place holder and the values are automatically generated by the system. For example, Approval of @event.type appears as Approval of Subscribe Package in the email sent, where Subscribe Package is the event.type.</p> </div>


12. Click **Cancel** to revert to the last saved changes or to abandon all the changes if the values are not saved.

13. Click **Save**.

Managing Pending Requests

You can manage your pending requests in the Pending Requests section. You can approve or reject a pending request or delete the requests approved by you.

> To approve or reject a pending request

1. Expand the menu options icon , in the title bar, and select **Pending Requests**.

The Pending requests page appears.

2. Select **Pending Requests**.

A list of pending requests appears with the following information:

Field	Description
Requested by	The name of the requestor.
Requestor comments	The additional information or comments added by the requestor.
Event	The type of event for which the request is pending. The options are: <ul style="list-style-type: none">■ Create application■ Register application■ Update application■ Subscribe package
Request details	The details of the type of event requested. The details available for the following event types are: <ul style="list-style-type: none">■ Create application<ul style="list-style-type: none">■ Application name. The name of the application.■ Owner. The owner of the application.■ Identifiers. Identifiers by which messages from a particular consumer application is recognized at run time.

Field	Description
	<ul style="list-style-type: none"> ■ Contact emails. The email address of the owner of the application. ■ Version: The version of the application.
	<ul style="list-style-type: none"> ■ Register application <ul style="list-style-type: none"> ■ Application name. The name of the application. ■ API. The API to which the application is associated.
	<ul style="list-style-type: none"> ■ Update application <ul style="list-style-type: none"> ■ Application name. The name of the application. ■ Owner. The owner of the application. ■ Identifiers. Identifiers by which messages from a particular consumer application is recognized at run time. ■ Contact emails. The email address of the owner of the application. ■ Version: The version of the application.
	<ul style="list-style-type: none"> ■ Subscribe package: <ul style="list-style-type: none"> ■ Application name. The name of the subscription. ■ Owner. The owner of the subscription. ■ Identifiers. Identifiers by which messages from a particular consumer subscription is recognized at run time. ■ Contact emails. The email address of the owner of the subscription. ■ Version. The version of the subscription. ■ Plan name. The name of the plan. ■ Package name. The name of the API package.

3. Click the **Approve** or **Reject** icon to approve or reject requests.


Alternatively, you can select multiple pending requests to be approved or rejected simultaneously by selecting the check boxes adjacent to the names of the requests.

The request gets removed from the Pending requests page.

Deleting Requests

When you perform a task, for example, you create an application in API Gateway, then an approval request is generated if an approval is configured in API Gateway and this request that is waiting for approvers approval is listed in the Pending Request section. If you want to delete this request, you can delete it from the Pending Request section.

> To delete a request

1. Expand the menu options icon , in the title bar, and select **Pending Requests**.
2. Select **My requests**.
A list of requests appears with the detailed information of the request.
3. Click the **Delete** icon for the request that has to be deleted.
4. Click **Yes** in the confirmation dialog.

Outbound Proxy

When API Gateway executes a request against a remote server, it issues an HTTP or HTTPS request to the specified target server. If your API Gateway instance is behind a firewall, and must route these HTTP or HTTPS requests through a third party proxy server, you can configure proxy servers to which API Gateway routes these requests.

For API Gateway to use a proxy server, you must define a proxy server alias. The proxy server alias identifies a proxy server and a port on the server through which you want to route requests. You can configure API Gateway to route requests to one or more proxy server aliases for each type of outbound requests (HTTP, HTTPS).

All the configured and available proxy server aliases are listed in a table with the corresponding details of the port being used, the host on which it is, the protocol used, and so on. To use a proxy server you have to enable it.


Note:

Any modifications to the outbound proxy in Integration Server may not reflect in API Gateway UI. Hence, Software AG recommends that you do not configure or modify outbound proxy through Integration Server Administrator UI.

Configuring Proxy Server Alias

For API Gateway to use a proxy server, you must define a proxy server alias. Proxy server alias names must be unique across protocols, that is, you cannot have proxy server aliases of the same name but of different protocols.

➤ To configure proxy server alias

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Outbound proxy**.

This displays a list of available proxy server aliases and the corresponding details.


3. Click **Add proxy** and provide the following information:

Field	Description
Alias	The alias name of the proxy server.
Username	<i>Optional.</i> The username API must use when accessing this proxy server.
Password	<i>Optional.</i> A valid password associated with the username.
Hostname or IP address	The host name or IP address of the proxy server.
Port number	The port on which this proxy server listens for requests.
Protocol	The type of protocol (HTTP, HTTPS) to use for the host/port combination.
Set as default	Indicates whether this proxy server alias should be the default proxy server alias for its protocol type or not. Click Yes or No . Only one default proxy server alias can be set for each protocol type.

4. Click **Add**.



Modifying a Proxy Server Alias

➤ To modify a proxy server alias

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Outbound proxy**.
3. In the Outbound proxy list, select the proxy server alias that you want to edit.
4. Incorporate the required changes.
5. Click **Ok**.

Deleting a Proxy Server Alias

> To delete a proxy server alias

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Outbound proxy**.
3. In the Outbound proxy list, click  in the action column of the proxy server alias to be deleted.
4. Click **Yes** in the confirmation dialog box.

The proxy server alias is deleted from the list.

Proxy Bypass


If you are using proxy servers for outbound HTTP or HTTPS requests, you can optionally route selected requests directly to their target servers, bypassing the proxy servers listed in the outbound proxies list. For information on outbound proxies, see [“Outbound Proxy” on page 60](#).

To allow the outbound HTTP and HTTPS requests sent by API Gateway to bypass a particular proxy, you must add the proxy to the **Proxy bypass** list.

Adding a Proxy Bypass

The HTTP and HTTPS requests sent from API Gateway to remote servers are not directed through the proxies added in the **Proxy bypass** screen.

> To add a proxy bypass

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Proxy bypass**.
3. In the **Addresses** field, provide the host name or IP of the proxy server that the HTTP and HTTPS requests must bypass.

You can type the fully qualified host and domain name of each server to which you want the API Gateway to issue requests directly. Type the host name or the domain name exactly as they appear in the URLs that the server uses. To enter multiple names, separate each with commas. You can use the asterisk (*) to identify several servers with similar names. The asterisk matches any number of characters. For example, if you want to bypass requests made to localhost, www.yahoo.com, home.microsoft.com, and all hosts whose names begin with NYC, you would type:

```
localhost,www.yahoo.com,home.microsoft.com, NYC*.*
```

4. Click **Add**.




The specified proxy server is displayed in the **Addresses** list.

5. Click **Save**.

The list is saved.

Modifying a Proxy Bypass



> To modify a proxy bypass

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Proxy bypass**.
3. In the **Addresses** list, click  in the action column of the proxy server to be edited.
4. Incorporate the required changes.
5. Click .
6. Click **Save**.

The changes are saved.

Deleting a Proxy Bypass

> To delete a proxy bypass

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Proxy bypass**.
3. In the **Addresses** list, click  in the action column of the proxy server to be deleted.
4. Click **Save**.

The selected proxy server is deleted.

URL Aliases

A URL alias is an alias that you create to replace a portion of the URL to an API on API Gateway. The URL alias typically replaces the path portion of the URL which identifies the name and invocation endpoint of the API. For example, if the URL is `http://test:2225/gateway/RESTCalcService/1.0`, you can create a URL alias, `calc` for an API, then the name and invocation endpoint of the API on API Gateway is replaced with `calc`, for example, `http://test:2225/calc`. If the client sends a request that contains the matching alias, then the corresponding URL path is invoked.

In addition to associating a URL alias with a single resource, you can also map different resources for each port, therefore, based on the incoming port, the corresponding resource path is invoked. This makes it possible to define a single URL alias that resolves to different destinations based on the incoming port of the HTTP request.

URL aliases have several benefits:

- A URL alias may be easier to type than full URL path names.
- A URL alias is more secure than URL path names.

The **URL aliases** page displays a list of available URL aliases with the corresponding details including the default URL path, port it is mapped to, and so on. You must have the Manage aliases functional privilege assigned to manage the alias information.

In the **URL aliases** page, with the **Global gateway endpoint** section, you can also define global gateway endpoint. For more information about global gateway endpoint, see [“How do I Define Global Gateway Endpoint?” on page 837](#)

Note:

Any modifications to the URL aliases in Integration Server do not reflect in API Gateway. Hence, Software AG recommends that you do not modify the aliases through Integration Server Administrator. On migration from 10.0 to 10.1, the existing configuration in 10.0 is migrated to the API Gateway UI.

API Gateway only supports modification of URL aliases for the WmAPIGateway package. To modify URL aliases for any other packages, you must use Integration Server Administrator.

Creating URL Alias


When you create a URL alias, you create an association between an alias and an API on API Gateway.

Keep the following information in mind when creating a URL alias:

- An alias name must be unique across API Gateway.
- You can associate a single URL alias with multiple resources by specifying port mappings. A port mapping correlates the alias with a different URL alias based on the port on which the request was received.

- If you want to use URL alias with a REST resource you must enable partial matching of URL aliases.
- If you have enabled or intend to enable partial matching of URL aliases, do not define an alias that begins with another alias.

➤ **To create a URL alias**

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > URL aliases**.

This displays a list of available URL aliases and the corresponding details.

3. Click **Add URL alias** and provide the following information:

Field	Description
Alias	<p>The alias name of the proxy server.</p> <p>An alias name must be unique across API Gateway.</p> <p>The alias name cannot include a space, nor can it include the following characters: # % ? ' " < \</p> <p>The alias name cannot begin with the string http:// or https://</p>
Port number	<p>The port number to use when accessing the API.</p> <p>When API Gateway receives a request that contains a URL alias, API Gateway resolves the request destination by first determining if there is a URL path associated with the incoming port. If there is no URL path mapped to the port number, then API Gateway uses the default URL path for the alias as the request destination. The port mappings are always evaluated first.</p> <p>Because the URL path can be different for each port, using port mappings allows the use of a single URL alias with multiple destinations. Each port can have only one URL path mapping. You can add port mappings to multiple destinations by clicking the +Add button.</p>
URL path	<p>The URL path for the URL alias and for port number mapped for the URL alias.</p> <p>The URL path cannot include a space or the following characters: # % ? ' " < \</p>
Default URL path	The URL path to the API on API Gateway.

Field	Description
	<p>You must specify the default URL path if you do not define any port mappings for the URL alias. If the URL alias includes port mappings, the Default URL Path field is optional.</p> <p>The URL path cannot include a space or the following characters: # % ? ' " < \</p>

4. Click **Save**.

Using Port Mappings with URL Alias


For a URL alias, you can create one or more port mappings which associates a port with a resource. Port mappings allow the use of a single URL alias with multiple resource paths, because each port can be mapped to a different URL path.

When API Gateway receives a request that contains a URL alias, API Gateway resolves the request destination by first determining if there is a URL path associated with the incoming port. If there is no URL path mapped to the port number, then API Gateway uses the default URL path for the alias as the request destination. The port mappings are always evaluated first.

Adding Port Mapping to URL Alias

You can create multiple port mappings for a URL alias. Incoming requests with the URL alias received on a port that has no URL path mapping, resolves to the path specified in the **Default URL Path** field. If the alias does not specify a **Default URL Path** value, API Gateway uses the alias as the URL path.

➤ To add a port mapping to a URL alias

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > URL aliases**.
3. In the URL alias list, select the URL alias for which you want to define port mappings.
4. In the **Port number** list, select the port for which you want to specify an alternate path.
5. In the **URL path** field, specify the path to API. The URL path cannot include a space or the following characters: # % ? ' " < \
6. Click **+Add** to add the port mapping to the alias.
7. Click **Save**.



Deleting Port Mapping for URL Alias

You can delete any of the port mappings added to a URL alias.

Note:

If you intend to delete all of the port mappings for a URL alias, make sure the URL alias specifies a **Default URL Path** value.

› To delete a port mapping for a URL alias

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > URL aliases**.
3. In the URL alias list, select the URL alias for which you want to delete a port mapping.
4. In the **Port number** list, click  in the action column of the port mapping that you want to delete.
5. Repeat the preceding step for each port mapping that you want to delete for an alias.
6. Click **Save**.

Enabling Partial Matching of URL Aliases

In some cases, URL requests include identifiers for a particular API. Because these identifiers vary for each instance of an API, URL requests might not exactly match any of the defined URL aliases for a particular API. To enable you to define URL aliases for such APIs, API Gateway can use partial matching to process URL requests. A *partial match* occurs when a request URL matches or begins with only part of a URL alias.

When partial matching is enabled and API Gateway receives a request URL, an alias is considered a match if the entire alias matches all or the beginning of the request URL, starting with the first character of the request URL's path.

For example, if URL alias `calc` has a URL path of `gateway/RESTCalcService/1.0` and API Gateway receives the following request URL:

```
https://MyHost:9072/calc/deleteCalc
```

The request URL matches the `calc` alias and resolves to the path:

```
https://MyHost:9072/gateway/RESTCalcService/1.0/deleteCalc.
```

API Gateway retains the trailing characters of the request URL. In this case, API Gateway retains the `/deleteCalc`.

To enable API Gateway to use partial matching of URL requests, you must set the value of the parameter `watt.server.url.alias.partialMatching` to `true` in Integration Server (in the Integration


Server Administrator, go to **Settings > Extended** link). The default is `false`. For more information on configuring partial matching of URL aliases using Integration Server, see *webMethods Integration Server Administrator's Guide*.

Important:

If you change the setting of this parameter, you must restart Integration Server for the changes to take effect.

Modifying a URL Alias

> To modify a URL alias



1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > URL aliases**.
3. In the URL Alias list, select the URL alias that you want to edit.

Alternatively, in the URL Alias list, locate the row that contains the alias you want to modify, and click the **Edit** icon.

4. Incorporate the required changes.
5. Click **Save**.

Deleting a URL Alias

> To delete a URL alias

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > URL aliases**.
3. In the URL Alias list, locate the row that contains the alias you want to delete, and click .
4. Click **Yes** in the confirmation dialog box.

Example: Usage Scenarios of URL Aliases

This section explains how an API consumer can invoke an API for which a URL alias is created. This example uses the *RESTCalcService* API. Suppose, the *RESTCalcService* API is activated in API Gateway and following are the gateway endpoints for this API:

- `http://test:2225/gateway/RESTCalcService/1.0`


- <http://test:4000/gateway/RESTCalcService/1.0>
- <http://test:4010/gateway/RESTCalcService/1.0>
- <http://test:5555/gateway/RESTCalcService/1.0>

Also, the *RESTCalcService* consists of the *postCalc* resource path that adds two numbers.

If this API is published to the API consumer then the invocation endpoint for the consumer may appear as:

<https://test:5555/gateway/RESTCalcService/1.0/postCalc>

If you do not want to expose the API name and path information or if you want to shorten the invocation endpoint as it is complex, then you can create a custom URL alias. To create a URL alias:

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > URL aliases**.
3. Click **Add URL alias** and provide the following values:

Field	Value
Alias	calc
Default URL path	gateway/RESTCalcService/1.0/postCalc

4. Click **Save**.

Suppose, the URL alias name provided while creating a URL alias is `calc`. You can now expose the <https://test:5555/calc> invocation endpoint to the API consumer instead of <https://test:5555/gateway/RESTCalcService/1.0/postCalc>.

With the default URL path specified in alias configuration, the API consumer can use this URL alias for any ports of gateway endpoint shown in the API details page, for example,

- <http://test:2225/calc>
- <http://test:4000/calc>
- <http://test:4010/calc>
- <http://test:5555/calc>

Additionally, you can use port mapping, if you want to use the same alias for a different resource path. This can be done by providing a different resource path for each port, instead of the default URL path in alias configuration. To use the same alias for a different resource path:

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > URL aliases**.

3. Click **Add URL alias** and provide the following values:

Field	Value
Alias	calc
Port number	5555
URL path	gateway/RESTCalcService/1.0/postCalc

4. Click **+Add** to add port mappings to multiple destinations and provide the following values:

Field	Value
Port number	4000
URL path	gateway/RESTCalcService/1.0/deleteCalc

5. Click **Save**.

Note:

As the **Default URL path** is not provided, the incoming call for ports other than 4000 and 5555 fails. If the **Default URL path** is provided then the port mapping takes the first precedence. If the value of the port does not match, then the **Default URL path** is used.

For the alias `calc`, each port is mapped to a different resource, for the invocation endpoint:

- `https://test03:4000/calc`: RESTCalcService/1.0/deleteCalc resource is invoked.
- `https://test03:4010/calc`: error appears as the default URL is not provided and a path is not configured for the 4010 port.
- `https://test03:5555/calc`: RESTCalcService/1.0/postCalc resource is invoked.

Custom Content-types

An API Provider can configure custom content-types based on how the payloads in the incoming or outgoing requests have to be processed. If the native API consumes some custom content-type, the API Provider can configure a mapping between this custom content-type and a base content-type so that the schema validation, and identification in the policies are performed based on the base type.

For example, a native API consumes `application/xyz` content-type. The API provider creates an API for this native API and enforces the Validate API Specification policy and the API definition has schema mapping for `application/json`. When the request reaches API Gateway and as there is no content-type schema mapping for `application/xyz`, the schema validation is skipped. In such scenarios, if the API provider creates a custom content-type mapping in the API Gateway UI with the content type as `application/xyz` and base type as `JSON`, then the payload in the incoming request is validated against the `JSON` schema.

The following table explains the different identifiers and payload validation criteria that can be used for various content types that you can use to configure your custom content-types.

Content-type	Identifier	Payload validation
application/xml	XPath	XML schema
text/xml		
text/html		
multipart/form-data		
multipart/mixed		
application/json	JSONPath	JSON schema
application/json/badgerfish		
text/plain	Regex	Regex

Note:


The custom content-type feature is not supported for the SOAP to REST transformed APIs.

Configure Custom Content-types

When you configure a custom content-type, you specify what base-type the content type while processing the content.

You must have the API Gateway's manage general administration configurations functional privilege assigned to configure custom content-type.

➤ To configure custom content-type

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Custom Content-types**.
3. Provide the Content-type that has to be configured.
4. Select the **Base type** as one of the following:
 - **JSON**: Specifies that the base-type for the provided Content-type is set to JSON.
 - **XML**: Specifies that the base-type for the provided Content-type is set to XML.
 - **Text**: Specifies that the base-type for the provided Content-type is set to plain text.

5. Click .

You can configure multiple custom content-types by clicking Add.

Cache Configuration

In API Gateway version 10.1 or earlier, all assets are stored in memory and reside in memory indefinitely. Since the cache is loaded fully during startup, it takes a while to load all the elements, such as applications, APIs, policies, and so on in the cache. The Cache configuration functionality in API Gateway enables lazy loading of the assets and ensures that API Gateway startup is quick and is independent of the data size. If the memory size is not enough to contain all the entities API Gateway caches can be configured to load a certain percentage of the entities in memory.

API Gateway internally classifies cache based on the usage and size as follows:

- **Size bounded:** In-memory cache that forgets the least recently used (LRU) entry when the maximum number of entries is reached. This can be configured as percentage of the total number of entries in the data store.
- **Size unbounded:** When the cache configuration percentage is set to 100% manually or as computed by auto scale option, the cache is set as unbounded.

You can configure individual cache types in this section. You can set additional parameters that control the cache configuration in the **Administration > General > Extended Settings** section. For details see, [“Configuring Extended Settings” on page 22](#).

Note:

The cache configuration is synchronized across different nodes in a cluster.

Autoscaling mode

Depending on the load on the system the autoscaler thread computes the percentage of entities to be kept in memory. If the number of entities is less all the entities are stored in memory. If the number of entities does not fit into the memory then it computes the memory for each of the caches that are set to run in autoscale mode. The caches are reconfigured automatically based on the computed value. This can be controlled by setting the parameters `pg_Cache_autoScalerRunInterval` and `pg_Cache_minCachePercent`.

Percentage mode


Depending on the growth or the current size of the number of entities the cache size is computed and resized periodically. This is controlled by the setting the parameters `pg_Cache_boundedCacheResizeRunInterval` and `pg_Cache_minCacheSize`.

You can also collect the required statistics for the selected cache, which is displayed in the Global dashboard under **Analytics > Cache Statistics**. This statistics collection interval can be configured by setting the parameter `pg_Cache_statisticsProcessorRunInterval`.

Configuring Cache to Improve Performance

You must have the API Gateway's manage general administration configurations functional privilege assigned to configure cache.

> To configure cache to improve performance

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Caching > Cache configuration**.
3. Select the **Cache name**, from the drop-down list of available cache to be configured.
4. Select a value for the **Percentage of cache in memory**.

For example, you can set the value as 10%, 20%, and so on. If you do not select any value, the value is set to Auto scale by default.

5. Select **Collect statistics** to collect the required statistics for the selected cache.

6. Click  .

The configured cache is listed in a table in the cache configuration section. If the configuration is set to Auto scale and Collect statistics is not selected then, by default, the cache is not listed.

You can edit or delete the cache configuration entries by clicking the icons in the action column.

7. Click **Update**.

This saves the cache configuration.

Note:

To enhance the performance of basic authentication scenarios at runtime, `ACCESS_PROFILES` cache type is always configured to 100% during restart of API Gateway. You cannot reconfigure the `ACCESS_PROFILES` cache from API Gateway user interface and Software AG recommends users not to modify the value of `ACCESS_PROFILES` cache type using REST APIs.

Application Logs

API Gateway collects and stores logs from different sources such as API Gateway server logs, API Gateway UI logs, Internal Data Store logs, dashboard logs and platform logs. Each type of logged data comes from different components. API Gateway provides an easy way of configuring log settings to collate the logs from different sources at a single place in a common format. You also have an option to download these logs as an archive. This provides an easy access to logs collected from different sources when you have to look into the logs to troubleshoot any issues, such as error or performance problems.

You can use the log aggregation functionality to configure log aggregation such that the aggregated logs from different sources can be stored at a single location in a common format. API Gateway provides a single user interface, in the API Gateway dashboard under **Analytics > Application**

logs, where you can search and view the aggregated logs stored in the Internal Data Store. You can filter the logs as required and search for a particular log.

Note:

The log aggregation is done across all the configured cluster nodes. The log level configuration is propagated across all the nodes in a cluster.

Different components of API Gateway from which the logs are collected are:


- API Gateway user interface and server
- Internal Data Store
- Dashboard
- Platform

Configuring Log Levels

You can configure the log level settings to specify the level of detail you want the logs, which are collected from various API Gateway components, to contain. These logs once configured can be downloaded as an archive in the Download diagnostics section.

You must have the API Gateway's manage general administration configurations functional privilege assigned to configure log levels.

» To configure log levels

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Application logs**.
3. In the Log level configuration section you can configure the following parameters:

Log file type	Description
API Gateway	<p>This setting sets the API Gateway server log level. API Gateway server logs contain information on the API Gateway UI related activities such as creating and managing APIs, aliases, applications and so on. API Gateway server logs also contain information on the start-up activity of API Gateway server and its functioning, its state and activities related to connections to various components.</p> <p>Select a log level that you want to configure to collect required logs.</p> <p>The available log levels are: off, trace, debug, info, warn, error, fatal.</p>

Log file type	Description
	<p>Note: Changing the log level of API Gateway logs sets that value of log level for all the log components of API Gateway.</p>
Internal Data Store	<p>This setting sets the Internal Data Store log level. This contains log files of Elasticsearch included with API Gateway.</p> <p>Select a log level that you want to configure to collect required logs.</p> <p>The available log levels are: off, debug, info, warn, error, fatal.</p>
Dashboard	<p>This setting sets the log level for dashboard logs. These logs contain information regarding dashboard related activities and the kibana log files that are included in API Gateway.</p> <p>Select a log level that you want to configure to collect required logs.</p> <p>The available log levels are: silent, quiet, verbose.</p>
Security	<p>This setting enables API Gateway security logs. These logs contain information on security-related administrative and operational events. For example, changes to authorization, authentication, port settings, password restrictions, attempts to log on to API Gateway and to access API Gateway services, and so on.</p> <p>Select the checkbox to enable the collection of security logs.</p>
Session	<p>This setting enables API Gateway session logs. These logs contain information on sessions activated on API Gateway and provide data on when sessions start, their current status, their duration, and so on.</p> <p>Select the checkbox to enable the collection of session logs.</p>
Platform	<p>This setting sets API Gateway platform logs .</p> <p>Select a log level that you want to configure to collect required logs.</p> <p>The available log levels are: off, trace, debug, info, warn, error, fatal.</p>

4. Click **Save**.


This configures the log levels for the various log file types that can be downloaded as an archive.

Downloading the Log Files

You can download the log files as an archive. This contains all the logs collected from various API Gateway components, the thread dumps, and the server configurations, the details of which correspond to the log level configured. You can browse through the downloaded data to troubleshoot any issues, such as error or performance problems.

You must have the API Gateway's manage general administration configurations functional privilege assigned to download log files.

> To download the log files

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Application logs**.
3. Click **Download diagnostics** in the Download diagnostics section.
4. Click **Save File** in the file download dialog box.
5. Click **OK**.

The file is downloaded and saved in the zip format.

Configuring Log Aggregation


You can configure log aggregation such that the aggregated logs from different sources can be stored at a single location in a common format. You can select any of the following destinations to store the aggregated data:

- Internal Data Store
- Elasticsearch, in case you have an external Elasticsearch configured with API Gateway

Enabling log aggregation starts log collection from various sources. If the Internal Data Store is configured as the destination the aggregated log is displayed in the API Gateway dashboard. The dashboard provides a simple user interface where you can view the aggregated logs and details of the logs. You can filter or search for any specific log in the dashboard.

You must have the API Gateway's manage general administration configurations functional privilege assigned to configure the log aggregation and the destination for log aggregation.

> To configure log aggregation

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Application logs**.

- In the Log aggregation section, click the toggle button to change the status to , to enable the log aggregation.

Note:

Only when this is enabled, the log aggregation is done.

- In the Log aggregation section, select one of the following:

- **API Gateway store:** Select this to set Internal Data Store as destination to store the aggregated log.

If you select this option the aggregated logs are stored in the Internal Data Store and you can view them in the API Gateway dashboard.

- **External Elasticsearch.** Select this to set Elasticsearch as destination to store the aggregated log.

Note:

You must have an external Elasticsearch configured with API Gateway to store the data.

In addition provide the following information:

- **Protocol.** The type of protocol (HTTP, HTTPS) to use for the host and port combination.
- **Hostname.** Specifies the host name or IP address of the machine on which Elasticsearch resides.
- **Port.** Specifies the port where Elasticsearch server runs.
- **Indexname.** Specifies the index of the collected logs.
- **Username.** Specifies the Elasticsearch user ID for authenticating Elasticsearch when API Gateway communicates with it.
- **Password.** Specifies the password of the Elasticsearch instance to be used for establishing communication between API Gateway and Elasticsearch.

- Click **Save**.

The selected destination is now configured for log aggregation.

License Configuration

When you purchase webMethods API Gateway, your organization is granted a license to use it with certain features and functionality. The license expires after a time period specified by your purchase agreement.


Before you install API Gateway, you are provided with a license key file that you place in the file system of the machine on which API Gateway will run. This file contains the license key, which is a special code associated with your license. When you install API Gateway, the setup program asks you to provide the name and location of this file. The setup program then copies this file to

the `SAGInstall\directory\instances\instance_name\config` directory with the name `licenseKey.xml`. If this file is inadvertently deleted, API Gateway login fails.

Viewing Licensing Information

You must have the API Gateway's manage general administration configurations functional privilege assigned to configure licenses and view license information.

> To view license information

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > License configuration**.

This displays the current license information and an option to configure license

3. In the License information section you can view a list of features, with a check mark next to each feature you are licensed to use.
4. To view details of the licensing information, click **Details**.


API Gateway displays the following licensing information:

- **Sales information.** Displays the license details related to Sales such as Serial number of the license, the License key, the customer name and ID, number of contracts and details and so on.
- **Product information.** Displays the product details that comes with the current license such as license expiration date, operating system on which the product runs, product details such as code, name, version, and ID, the renewal date, and so on.
- **Integration Server.** Displays the Integration server details such as product code, version, concurrent sessions that can run, clustering and publishing capability, and so on.
- **License information.** Displays the license details such as Price unit, and quantity, installation type, license type, license version, and so on.
- **API Gateway.** Displays the features of API Gateway that are licensed.
- **Terracotta.** Displays the details of the License file, and expiration date of the license.

Configuring Licenses

You must have the API Gateway's manage general administration configurations functional privilege assigned to configure licenses and view license information.

> To configure the license

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > License configuration**.

This displays the current license information and an option to configure license.

3. In the Configure license section provide the following information:

- **API Gateway license file.** Provide the pathname for the API Gateway license file.

For example:

```
C:\Installations\IntegrationServer\instances\default\config\licenseKey.xml
```

- **Terracotta license file.** Provide the pathname for the Terracotta license key file if you have a Terracotta installed for clustering. An API Gateway requires Terracotta to be installed for a clustered API Gateway environment. The Terracotta license file contains the license information for all of your Terracotta components. You add this file to API Gateway by placing it into the `SAGInstalldirectory\common\conf` directory of the machine on which API Gateway runs. If the license file is located in a different directory than the specified directory for licenses you have to specify the pathname where the Terracotta license key file is located.

4. Click **Save**.

The license key file is saved in the location specified.

Enabling Clustering for API Gateway

API Gateway supports clustering to achieve horizontal scalability and reliability. Each API Gateway cluster node holds all the API Gateway components including UI, the API Gateway package and an Internal Data Store instance for storing assets. The synchronization of the nodes is performed through a Terracotta server array and Internal Data Store clustering that is defined across the Internal Data Store instances.

As each node of an API Gateway cluster offers the same functionality, nodes can be added or removed from an existing cluster. After you ensure that API Gateway has the appropriate environment, you can add it to the cluster by configuring the API Gateway to use clustering.

Keep the following points in mind when enabling clustering for API Gateway.

- Each API Gateway cluster node consists of an Internal Data Store instance for storing run-time assets and configuration items. For a cluster configuration, the Internal Data Store instances should also be clustered using standard Elasticsearch clustering properties, by modifying the `SAG_root/InternalDataStore/config/elasticsearch.yml` file on each instance.
- To be in the same cluster, API Gateways must use the same Terracotta server array URLs and the same cluster name.
- An enterprise can have more than one cluster. To isolate multiple clusters on the same network, each cluster must have a different cluster name or different Terracotta server array or both.



- Before you enable clustering using Terracotta, perform the following steps to ensure that Terracotta Server Array is ready for use by the API Gateways in the cluster.
 - Install the Terracotta Server Array if you have not done so already. For installation instructions, see *Installing Software AG Products*.

Note:

Before you install your Terracotta Server Array, you need to decide how many Terracotta Servers will make up the array and whether or not you will mirror those servers. To guide your decision, review the sections on high availability and architecture in the product documentation for the Terracotta Server Array or consult your Software AG representative.

- Configure the tc-config.xml file on the Terracotta Server Array. For details on installing and changing a Terracotta license, see *webMethods Integration Server Administrator's Guide*.
- You must have API Gateway administrator privileges to enable clustering.

> To enable clustering

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Clustering**.
3. Click the **Enabled** toggle button to  state to enable the cluster.
4. Provide the following information:

Field	Description
Cluster name	<p>Specifies the name of the cluster to which API Gateway belongs.</p> <p>Keep the following in mind when specifying the cluster name:</p> <ul style="list-style-type: none"> ■ The cluster name cannot include any periods ".". API Gateway converts any periods in the name to underscores when you save the cluster configuration. ■ The cluster name cannot exceed 32 characters. If it does, API Gateway uses the first 20 characters of the supplied name and then a hash for the remaining characters.
Session time out (minutes)	<p>Specifies the number of minutes an inactive session is retained in the clustered session store.</p> <p>The default is 60 minutes.</p> <p>Set the clustered session timeout value to be longer than the session timeout value, which governs how long a server keeps</p>

Field	Description
	session information in its memory. (API Gateway displays the session timeout on the Administration > System settings > Configuration screen.)
Action on startup	<p>Specifies how API Gateway responds when an error at start up prevents API Gateway from joining the cluster. Select one of the following:</p> <ul style="list-style-type: none"> ■ Start as stand-alone API Gateway. API Gateway starts as a stand-alone, unclustered API Gateway if it encounters any errors that prevent it from joining the cluster at start up. API Gateway continues to receive requests on inbound ports and serve requests. This is the default setting. ■ Shut down API Gateway. API Gateway shuts down if it encounters any errors that prevent it from joining the cluster at start up. ■ Enter quiesce mode on stand-alone. API Gateway starts as a stand-alone, unclustered API Gateway in quiesce mode if it encounters any errors that prevent it from joining the cluster at start up. When API Gateway is in quiesce mode, only the diagnostic port and quiesce port are enabled.
Terracotta server array URLs	<p>A comma separated list of the URLs for the Terracotta server array to be used with the cluster to which this API Gateway belongs.</p> <p>The URLs must be in the format: <i>host:port</i></p>

5. Click **Save**.
6. Restart API Gateway.
7. Log on as an Administrator and navigate to **Administration > General > Clustering** and verify that all nodes in the cluster are displayed under Cluster hosts.

Note:

Clustering must be enabled on all nodes in a cluster for them to be included in the cluster.


Configuring API Callback Processor Settings

You can configure the API callback processor setting **All API callback requests** so that API Gateway accepts all the requests from the client that contain the callback request URL and wrap the requests with its own URL before routing them to the native API. This lets API Gateway track the requests that the client sends to the native API and the callback messages that are sent by the native API to the client. In addition, you can use the settings **Allow HTTPS access only** and

Process only allowed IPs requests to avoid any external threats in case an unauthorized user tries to access the protected resource.

You must have manage general administration configurations functional privileges to configure callback processor settings.

➤ **To configure API callback processor settings**

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Callback processor settings**.
3. Select **All API callback requests**.

This enables API Gateway to accept all the API callback requests coming from the client and wraps these requests with its own URL before it routes these requests to the native API. This option is selected by default.

When this setting is disabled, the request from the client reaches the native API, as is, without the API Gateway wrapping it with its own URL. So, when the native API sends out the callback request to the client it directly reaches the client and API Gateway is unable to track such events.

4. Select **Allow HTTPS access only**.

This allows API Gateway to receive only HTTPS callback requests from the native API and processes the requests before routing them to the client. If a HTTP callback request comes in, API Gateway sends out an Access denied message to the client. This option is selected by default.

If this option is not selected then API Gateway accepts the HTTP callback requests and processes the requests before routing them to the client.

5. Select **Process only allowed IPs requests**. This allows API Gateway to receive the callback requests only from the IP addresses specified in the Trusted IP addresses list.

API Gateway allows callback requests only from the allowed IPs configured in Trusted IP address list. You can configure your native APIs machine IPs or the native API outbound proxy server IPs here, so API Gateway allows a request coming from the native API and would then be routed to the client.

If there are no trusted IPs configured and this option is selected, then API Gateway does not allow any requests.

6. Type the IP address in the **Trusted IP address** and **Add**. You can add multiple IP addresses.

API Gateway allows only requests coming from these IP addresses when the option **Process only allowed IPs requests** is selected.

7. Click **Save**.

Messaging

API Gateway Messaging is an umbrella term that encompasses the exchanging of messages across the multiple platforms in asynchronous style.

To configure API Gateway for JMS messaging, you need to:


- Create one or more JNDI provider aliases to specify where API Gateway can look up administered objects when it needs to create a connection to JMS provider or specify a destination for sending or receiving messages.
- Create one or more connection aliases that encapsulate the properties that API Gateway needs to create a connection with the JMS provider.

Creating a JNDI Provider Alias

Each JMS provider can store JMS administered objects in a standardize namespace called the Java Naming and Directory Interface (JNDI). JNDI is a Java API that provides naming and directory functionality to Java applications.

As the JMS client, API Gateway uses a JNDI provider alias to encapsulate the information needed to look up an administered object. When you create a JMS connection alias, you can specify the JNDI provider alias that API Gateway should use to look up administered objects (that is, Connection Factories and Destinations).

➤ To create a JNDI provider alias

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Messaging**.

API Gateway displays a list of all the currently defined JNDI provider alias definitions.

3. Click **Add JNDI provider alias**.
4. Provide the following information for the JNDI provider alias

Field	Description
JNDI alias name	The alias name that you want to assign to this JNDI provider.
Description	A description for this JNDI alias.
Predefined JNDI Templates	Select the JNDI template that you want to use.

Field	Description
	<p>The JNDI templates provide information that you can use to complete alias configuration for a specific provider. The available templates are: Broker, UM, filesystem, LDAP, JBoss, WebLogic, Qpid-AMQP (0-x)</p>
Initial context factory	<p>The class name of the JNDI provider.</p> <p>The JNDI provider uses the initial context as the starting point for resolving names for naming and directory operations. API Gateway displays the initial context factory for the provider depending on the predefined JNDI template selected.</p>
Provider URL	<p>The primary URL of the initial context for sessions with the JNDI provider.</p> <p>The URL specifies the JNDI directory in which the JNDI provider stores JMS administered objects.</p> <p>If you are using Software AG Universal Messaging, this is the Universal Messaging realm server in the format <code>nsp://UM_host : UM_port</code> (for example, <code>nsp://127.0.0.1:9000</code>).</p> <p>If you are using a cluster of Universal Messaging realm servers, supply a list of the URLs to each realm server in the cluster. Use a colon or semi-colon to separate each URL:</p> <ul style="list-style-type: none"> ■ Separate the URLs using a comma if API Gateway always attempts to connect to the first Universal Messaging server in the list, only trying the second Universal Messaging server in the list if the first server becomes unavailable. ■ Separate the URLs using a semicolon if API Gateway connects to a randomly chosen URL from the list. This may result in better distribution of clients across the servers in the cluster. <p>If you are using the filesystem you have to provide the filepath of the location of the file to be used.</p> <p>If you are using Qpid-AMQP option you have to provide the file path location where the <code>amqp.properties</code> file is saved.</p>
Provider URL failover list	<p>A list of the failover URLs of the initial context for sessions with the JNDI provider. If the connection to the primary JNDI provider becomes unavailable, API Gateway automatically attempts a connection to a JNDI provider specified in this list.</p> <p>Specify one URL per line.</p>

Field	Description
	<p>Keep the following points in mind when adding JNDI provider URLs to the failover list:</p> <ul style="list-style-type: none"> ■ The JNDI providers must be the same type as the primary provider. For example, if the primary provider is a webMethods Broker, then the JNDI providers in the failover list must also be webMethods Brokers. ■ The administered objects on the providers must be identical to the each other. ■ Once Integration Server connects to a JNDI provider, it continues to use that JNDI provider until the connection is lost or interrupted. ■ When you start or restart a connection alias, Integration Server attempts to connect to the primary JNDI provider. If the connection fails, Integration Server immediately attempts to connect to the first JNDI provider in the failover list. If the connection fails, Integration Server attempts to connect to the next JNDI provider in the list. ■ When using webMethods Broker as a JNDI provider, you can keep objects in sync between webMethods Brokers using a webMethods Broker territory. That way objects can automatically forward from on webMethods Broker to another within the territory. ■ When using a cluster of Universal Messaging realm servers as the JNDI provider, Software AG recommends that you do not specify a Provider URL Failover List value. The realm URLs specified in Provider URL function as the failover list.
Security principal	<p>The principal name, or user name supplied by API Gateway to the JNDI provider, if the provider requires one for accessing the JNDI directory.</p> <p>For information about whether or not the JNDI provider requires security principal information, consult the product documentation for the JNDI provider.</p>
Security credentials	<p>The credentials, or password, that API Gateway provides to the JNDI provider, if the provider requires security credentials to access the JNDI directory.</p> <p>For information about whether or not the JNDI provider requires security credentials, consult the product documentation for the JNDI provider.</p>

Field	Description
Other properties	<p>Any additional properties the JNDI provider requires for configuration. For example, you might need to specify the classpath for any additional .jar or class files that the JNDI provider needs to connect to the JNDI.</p> <p>When you select a predefined JNDI template, API Gateway populates this field with any additional properties and placeholder information required by the JNDI provider.</p> <p>For more information about additional properties or classes required by a JNDI provider and the location of those files, see the product documentation for the JNDI provider.</p>

5. Click **Add**.

The JNDI provider alias created is listed in a table under JNDI provider alias definitions in the Messaging page.

You can edit, export the alias definition or delete the JNDI provider alias as required. You can also perform a test lookup for a JNDI provider to ascertain it is working as expected.

Considerations while using Software AG Universal Messaging as a JMS provider

Keep the following points in mind when using Universal Messaging as the JMS provider:

- When using Universal Messaging, if the version of Universal Messaging is equivalent to or higher than the version of API Gateway, you do not need to add any client libraries to the Integration Server classpath.
- When using Universal Messaging, if the version of Universal Messaging is lower than the version of API Gateway, you have to add the JMS provider's client libraries to the Integration Server classpath. You can do it in one of the following ways:
 - Place the libraries in the server's classpath by placing them in the *Install directory*\instances*instance_name*\lib\jars\custom directory. For details on the procedure, see *webMethods Integration Server Administrator's Guide*.
 - Make the libraries available to all server instances by placing them in the *Install directory*\IntegrationServer\instances\lib\jars directory.
 - Isolate the jars within a package classloader by placing them in the following directory: *packageName*\code\jars. If you place the files in the package classloader, make sure to set the Class Loader property when configuring a JMS connection alias to this JMS provider.

Note:

If you have configured and are using Software AG Universal Messaging, on migration from an earlier version of API Gateway to API Gateway 10.5, for example, if you are migrating from API Gateway 10.3 to 10.5, then the JNDI provider alias created in the 10.3 version is found to point to the 10.3 Universal Messaging endpoint. You can resolve this by either copying the 10.3 Universal Messaging client jars to 10.5 Integration Server's classpath or by

manually changing the **Provider URL** of the **JNDI Provider Alias** in 10.5 Integration Server to point to the 10.5 Universal Messaging endpoint.

- Keep the Universal Messaging client libraries up to date. If you install a Universal Messaging fix that updates the client libraries, make sure to copy the updated Universal Messaging client library files to the Software AG_directory /common/lib directory used by API Gateway.


For details on other supported JMS providers, see *webMethods Integration Server Administrator's Guide*.

Creating a JMS Connection Alias

A JMS connection alias specifies the information that API Gateway needs to establish an active connection between API Gateway and the JMS/AMQP provider. API Gateway uses a JMS connection alias to send messages to and receive messages from the JMS/AMQP provider. When you create a JMS connection alias, keep the following points in mind:

- You can use JNDI to retrieve administered objects (Connection Factories and Destinations) and then use the Connection Factory to create a connection. If you intend to use a JNDI provider, you need to configure one or more JNDI provider aliases before creating a JMS connection alias.
- Each JMS connection alias has an associated transaction type. Within API Gateway, certain functionality must be completed within a non-transacted session. For example, to use API Gateway to send or receive large message streams, you must specify a JMS connection alias whose transaction type is set to `NO_TRANSACTION`.

> To create a JMS connection alias

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Messaging**.

API Gateway displays a list of all the currently defined JNDI provider alias definitions, JMS connection alias definitions and individual SOAP JMS trigger controls.

3. Click **Add JMS connection alias** under JMS connection alias definitions section.
4. In the General Settings section, provide the following information:

Field	Description
Connection alias name	Name of the connection alias. This name should be unique as each connection alias represents a connection factory to a specific JMS provider.
Description	A description of the JMS connection alias.

Field	Description
Transaction type	<p>Specifies whether the sessions that use this JMS connection alias are transacted.</p> <p>Available values are:</p> <ul style="list-style-type: none"> ■ NO_TRANSACTION. Indicates that sessions that use this JMS connection alias are not transacted. ■ LOCAL_TRANSACTION. Indicates that sessions that use this JMS connection alias are part of a local transaction. ■ XA_TRANSACTION. Indicates that sessions that use this JMS connection alias are part of an XA transaction.
Connection client ID	The JMS client identifier associated with the connections established by this JMS connection alias.
Username	<p><i>Optional.</i> User name needed to acquire a connection from the connection factory.</p> <p>For more information about whether or not the JMS provider requires a user name and password to obtain a connection, refer to the product documentation for the JMS provider.</p>
Password	<p><i>Optional.</i> Password needed to acquire a connection from the connection factory.</p> <p>For more information about whether or not the JMS provider requires a user name and password to obtain a connection, refer to the product documentation for the JMS provider.</p>

5. In the Connection protocol settings section, provide the following information:

Field	Description
JNDI provider alias name	The alias to the JNDI provider that you want this JMS connection alias to use to look up administered objects.
Connection factory lookup name	<p>The lookup name for the connection factory that you want to use to create a connection to the JMS provider specified in this JMS connection alias.</p> <p>If the JMS connection alias connects to Universal Messaging, specify the Universal Messaging connection</p>

Field	Description
	<p>factory that you created when you set up your environment.</p> <p>If you are using SonicMQ as the JMS provider, specify the lookup name that refers to the serializable Java object file containing the SonicMQ object definitions. Include the .sjo extension as part of the lookup name.</p>
<p>Create administered objects on demand (Universal Messaging)</p>	<p>Specifies whether API Gateway creates administered objects on the JNDI provider if the object is not found when API Gateway looks up the object.</p> <p>Select the check box if you want API Gateway to create a destination or connection factory when an JNDI lookup for the object fails.</p>
<p>Enable follow the master (Universal Messaging)</p>	<p>Specifies whether connections created from this JMS connection alias always connect to the master realm server in the Universal Messaging cluster. This setting affects producer and consumer connections created using this JMS connection alias.</p> <p>You can do one of the following:</p> <ul style="list-style-type: none"> ■ Select the Enable follow the master check box to indicate that API Gateway connects to the master realm server in the Universal Messaging cluster when API Gateway uses this JMS connection alias to send or receive messages. ■ Clear the Enable follow the master check box to disable the follow the master behaviour for this JMS connection alias when API Gateway uses this JMS connection alias to send or receive messages. <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: Enable follow the master option is available to JMS connection aliases that use Universal Messaging as the JMS provider.</p> </div>
<p>Monitor webMethods connection factory</p>	<p>Specifies how API Gateway monitors the connection factory object for changes, if at all. This only applies if a JMS connection alias connects to the webMethods Broker using a webMethods Connection Factory object in a JNDI server.</p> <p>Select one of the following available options:</p>

Field	Description
	<ul style="list-style-type: none"> ■ No. Indicates that API Gateway does not monitor the connection factory. This is the default option. ■ Poll for changes (specify interval). Monitors the connection factory by polling the changes at an interval that you specify. ■ Poll for changes (interval defined by webMethods Connection Factory). Monitors the connection factory at an interval determined by the refresh interval specified for the webMethods Connection Factory object. ■ Register change listener. Monitors the connection factory by registering an event listener.

6. In the Advanced settings section, provide the following information:

Field	Description
Class loader	<p>The name of the class loader that you want to use with this JMS connection alias. API Gateway uses the specified class loader when performing certain activities with the JMS connection alias (send a message, receive a message, create a connection, create a destination, and so on.)</p> <p>By default, API Gateway uses the server class loader. However, you can specify the class loader for a package instead. This can help prevent conflicts between the jars required for different JMS providers.</p>
Maximum CSQ size (messages)	<p>The maximum number of messages that can exist in the client side queue for this JMS connection alias. API Gateway writes messages to the client side queue if the JMS provider is not available when messages are sent. Each JMS connection alias has its own client side queue.</p> <p>Specify -1 if you want the client side queue to be able to contain an unlimited number of messages. That is, specify -1 if you do not want to set a maximum limit.</p> <p>If you specify 0, API Gateway does not write messages to the client side queue for this JMS connection alias.</p>
Drain CSQ in order	<p>Specifies whether API Gateway drains the client side queue by sending the messages to the JMS provider in the same order in which API Gateway placed the messages in the client side queue.</p>

Field	Description
	<p>Select the check box if you want API Gateway to send messages from the client side queue in the same order in which API Gateway originally placed the messages in the client side queue.</p> <p>When the Drain CSQ in order check box is selected, after the connection to the JMS provider is re-established, API Gateway continues to write new messages to the client side queue until the client side queue is completely drained. If the Drain CSQ in order check box is not selected, after the connection to the JMS provider is re-established, API Gateway sends new messages directly to the JMS provider while it drains the client side queue.</p> <p>Note: You can also specify the number of messages API Gateway retrieves from the client side queue for delivery to the JMS provider at one time. By default, API Gateway sends 25 messages at a time.</p>
<p>Create temporary queue</p>	<p>Specifies whether API Gateway creates a temporary queue on the JMS provider to handle request-reply operations that do not specify a replyTo destination.</p> <p>Select the check box if you want API Gateway to create a temporary queue. Clear the check box if you do not want API Gateway to create a temporary queue.</p> <p>You must select the Create temporary queue check box if you want to select the Enable request-reply listener for temporary queue check box.</p>
<p>Enable request-reply listener for temporary queue</p>	<p>Specifies whether API Gateway creates a single dedicated MessageConsumer for receiving synchronous replies delivered to the temporary queue for this JMS connection alias.</p> <p>When this check box is selected, API Gateway creates a dedicated consumer for receiving replies to requests published using this JMS connection alias.</p> <p>When this check box is cleared, API Gateway creates a new JMS MessageConsumer for each reply message.</p>
<p>Enable destination management with designer (Broker and Universal Messaging)</p>	<p>Specifies whether users can use Designer to create, list, and modify destinations on the webMethods Broker or when working with JMS triggers.</p>

Field	Description
	<p>Select the check box if you want Designer users to be able to create, list, and modify destinations using a JMS trigger that uses this JMS connection alias.</p> <p>Note: Software AG recommends that you prevent Designer users from managing destinations in a production environment.</p>
Create new connection per trigger	<p>Specifies whether API Gateway creates a separate connection to the JMS provider for each JMS trigger.</p> <p>Select the check box if you want API Gateway to create a separate connection for each JMS trigger that uses this JMS connection alias.</p> <p>If you want a concurrent JMS trigger that uses this JMS connection alias to use multiple connections to the JMS provider, you must configure the alias to create a separate connection per trigger.</p>

7. In the Producer caching section, provide the following information to configure pools for caching of JMS Session and MessageProducer objects:

Field	Description
Caching Mode	<p>Specifies whether to enable caching of JMS Session and MessageProducer objects for this connection alias.</p> <p>Select one of the following:</p> <ul style="list-style-type: none"> ■ DISABLED. Indicates that API Gateway does not cache JMS Session or MessageProducer objects. ■ ENABLED PER DESTINATION. Enable caching of JMS Session and MessageProducer objects. <p>For a non-transacted JMS connection alias, API Gateway caches a Session object and a MessageProducer object. For a transacted JMS connection alias, API Gateway caches a Session object.</p>
Minimum pool size (unspecified destinations)	<p>The minimum number of entries in the default session pool.</p> <p>The default is 1.</p>
Maximum pool size (unspecified destinations)	<p>The maximum number of entries in the default session pool.</p>

Field	Description
	The default is 30.
Minimum poolsize per destination	The minimum number of entries in each destination-specific pool.
Maximum poolsize per destination	<p>The maximum number of entries in each destination-specific pool.</p> <p>A value of 0 (or blank) indicates that API Gateway does not create separate pools for any of the destinations associated with the JMS connection alias.</p>
Destination lookup names (semicolon delimited)	<p>A semicolon delimited list of the lookup names for the destinations for which you want API Gateway to create separate pools.</p> <p>Note: This field appears only when the JMS connection alias specifies JNDI Lookup for creating the connection to the JMS provider.</p>
Idle timeout (ms)	<p>The number of milliseconds before API Gateway removes an inactive pool entry. The timeout value applies to the default session pool and the destination-specific pools.</p> <p>A value of 0 indicates that pool entries never expire. A value of -1 indicates that API Gateway uses the system default as defined by the <code>watt.server.jms.producer.pooledSession.timeout</code> parameter.</p> <p>The default value is 60000 milliseconds.</p>

8. In the Producer retry section, provide the following information to configure automatic retry of `pub.jms:send` services that use this JMS connection alias to send a message to the JMS provider:

Field	Description
Maximum Retry attempts	<p>The maximum number of times that API Gateway automatically retries a <code>pub.jms:send</code> service that fails because of a transient error.</p> <p>The default value is 0. A value of 0 indicates that automatic retry is disabled for this JMS connection alias.</p>
Retry interval (ms)	The number of milliseconds that API Gateway waits between retry attempts.

Field	Description
	The default is 1000 milliseconds (1 second).

Note:

If the JMS connection alias is transacted or uses a connection factory to which the multi-send guaranteed policy is applied, API Gateway ignores the producer retry values.

9. In the Enhanced logging section, provide the following information to configure additional logging for the sending or receiving of JMS messages that use this messaging connection alias:

Field	Description
Logging type	<p>Specifies where API Gateway writes log messages when enhanced logging is enabled for the message producers or consumers that use this JMS connection alias to send or receive messages.</p> <p>You can select one of the following:</p> <ul style="list-style-type: none"> ■ SERVER LOG. Write enhanced logging messages to the server log. If you specify the server log as the destination, make sure to increase the logging level for the 0168 Messaging (Enhanced Logging) server log facility to at least Info. ■ MESSAGING AUDIT LOG. Write enhanced logging messages to the messaging audit log.
Enable producer message ID tracking	Select to indicate that API Gateway writes additional log messages when a message producer uses this JMS connection alias to send messages to a destination in Producer Message ID Tracking: Include Destinations.
Enable consumer message ID tracking	Select to indicate that API Gateway writes additional log messages for messaging consumers (triggers) that use this JMS connection alias to receive messages. API Gateway writes additional log message for the JMS triggers listed in Consumer Message ID Tracking: Include Triggers
Producer message ID tracking: include destinations (semicolon delimited)	<p>The destination names for which API Gateway performs additional logging when sending messages to the destination.</p> <p>Use a semicolon (;) to separate each destination name. Leave this field blank if you want API Gateway to perform enhanced logging for every destination to which this JMS connection alias sends messages.</p>

Field	Description
Consumer message ID tracking: include triggers (semicolon delimited)	<p>The fully qualified name of the JMS triggers for which API Gateway performs additional logging during trigger processing.</p> <p>Use a semicolon (;) to separate each trigger name. Leave this field blank if you want API Gateway to perform enhanced logging for every JMS trigger that uses this JMS connection alias to receive messages.</p>

10. Click **Add**.

The JMS connection alias created is listed in a table under JMS connection alias definitions in the Messaging page.

Web Services Endpoint Alias

A web service endpoint alias represents the network address and, optionally, any security credentials to be used with web services. You can use the network address properties to enable dynamic addressing for web services. The security credentials can be used to control both transport-level and message-level security for web services.

In web service descriptors, an endpoint alias is associated with a binder. API Gateway uses a binder to collect the definitions for addresses, communication protocols, and data formats for a particular port type in one container to collect the definitions for addresses, communication protocols, and data formats for a particular port type in one container.

For a provider web service descriptors, the endpoint alias is used to construct the `location=` attribute of the address element (which is contained within the port element) when WSDL is requested for the web service. The security credentials might be used when constructing a response to a web service request. When you create a provider web service descriptor, you can specify an existing endpoint alias, which is displayed (and can be changed) from the default binder of the web service descriptors.

For a consumer web service descriptor and its associated web service connectors (WSC), the alias information (including the addressing information and any security credentials), is used at run time to generate a request and invoke an operation of the web service.

API Gateway uses message addressing endpoint aliases to send responses to endpoints other than the one that initiated or sent the request. That is, when WSAddressing is enabled and the request SOAP message contains a non-anonymous ReplyTo or FaultTo endpoint, API Gateway uses the message addressing endpoint alias to determine the authentication details to be used to send a response to the ReplyTo and FaultTo endpoints.

An endpoint alias is usually created for one or more of the following reasons:

Dynamic endpoint addressing. As the actual value of the endpoint is looked up at run time, using an endpoint alias saves you from having to specify or change the server information each time you use the web service.

Security. An endpoint alias is required in order to configure WS-Security for web service providers and consumers.

When you create web service endpoint aliases, keep the following points in mind:

- Alias names must be unique within the specified usage (provider or consumer) and protocol. This can result in multiple endpoint aliases with the same name. For example, you can have a provider alias named `aliasOne` for the HTTP protocol. You could also have a consumer alias named `aliasOne` for the HTTP protocol and a provider alias named `aliasOne` for the HTTPS protocol.

API Gateway saves web service endpoint aliases at the location, *Integration Server_directory*\instances*instance_name*\config\endpoints. The host name and port are required for a provider HTTP or HTTPS web service endpoint alias, but are optional for a consumer HTTP or HTTPS web service endpoint alias.

If API Gateway on which a consumer web service descriptors reside, is located behind a firewall and the web service request needs to be routed through a proxy server, you can assign a proxy alias to the consumer web service endpoint alias. You can identify default provider web service endpoint aliases for HTTP and HTTPS. If a provider web service descriptor contains a binder set to the default alias, API Gateway uses the information in the default alias when constructing the WSDL for the descriptor.

Creating an Endpoint Alias for a Provider Web Service Descriptor

If a provider web service descriptor binder specifies the JMS transport, you must assign a web service endpoint alias to the binder. For a web service descriptor that uses SOAP over JMS, the provider web service endpoint alias provides the following information: JMS message header information for the request message, such as delivery mode, time to live, and the destination for replies. Integration Server uses this information to populate the binding elements in the WSDL generated for the web service descriptor.

The SOAP-JMS trigger that listens for SOAP over JMS messages for the web service descriptor. The SOAP-JMS trigger also provides the JMS connection information needed to create a connection on the JMS provider. Integration Server uses the information provided by the SOAP-JMS trigger to construct most of the JMS URI (the web service descriptor determines the `targetService`). The JMS URI appears in the WSDL document as the value of the `location="` attribute for the address element within the port element. WS Security Properties that specify the information needed by the SOAP processor to decrypt and verify the inbound SOAP request and/or encrypt and sign the outbound SOAP response and the details for adding the timestamp information.


Keep the following information in mind when creating a web service endpoint alias for a JMS binder in a provider web service descriptor:

- You can associate the web service endpoint alias with
 - A SOAP-JMS trigger that already exists.
 - A WS endpoint trigger that you create at the same time you create the endpoint alias.
- If you use a SOAP-JMS trigger in the web service endpoint alias and subsequently assign the alias to a JMS binder in a provider web service descriptor, the web service descriptor has a

dependency on the SOAP-JMS trigger. Consequently, at start up or when reloading the package containing the web service descriptor, API Gateway must load the SOAP-JMS trigger before loading the web service descriptor. If the SOAP-JMS trigger and web service descriptor are not in the same package, you need to create a package dependency for the package that contains the web service descriptor on the package that contains the SOAP-JMS trigger.

- If you rename the SOAP-JMS trigger assigned to an alias, you need to update the alias to use the renamed trigger.

➤ To create a provider web service endpoint alias for use with JMS

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Web services**.

API Gateway displays a list of all the currently defined endpoint aliases.

3. Click **Add web service endpoint alias**.
4. In the Webservice endpoint alias properties section, provide the following information:

Field	Description
Alias	Name of the JMS provider web service endpoint alias. The alias name cannot include the following special characters: # ©\ & @ ^!%* :\$. / \ \ ` ; , ~ + =) (} { } [] > < "
Description	A description for the endpoint alias.
Type	Specifies the type - Provider

5. In the JMS transport properties section, provide the following information:

Field	Description
Delivery mode	The message delivery mode for the request message. This is the delivery mode that web service clients must specify in the JMS message that serves as the request message for the web service. You can select one of the following modes: <ul style="list-style-type: none"> ■ PERSISTENT. Specifies that the request message should be persistent. The message is not lost if the JMS provider fails.

Field	Description
	<ul style="list-style-type: none"> ■ NON_PERSISTENT. Specifies that the request message is not persistent. The message might be lost if JMS provider fails.
Time to live	<p>The number of milliseconds that can elapse before the request message expires on the JMS provider.</p> <p>If you specify a value 0, it indicates that the message does not expire.</p>
Priority	<p>Specifies the message priority.</p> <p>The JMS standard defines priority levels from 0 to 9, with 0 as the lowest priority and 9 as the highest.</p>
Reply to name	<p>Name or lookup name of the destination to which the web service sends a response (reply) message.</p> <p>Specify a name if the JMS connection alias used by the SOAP-JMS trigger connects to the webMethods Broker natively. Specify a lookup name if the JMS connection alias uses JNDI to retrieve a connection factory that is then used to connect to the JMS provider.</p>
Reply to type	<p>Type of destination to which the web service sends the response (reply) message.</p> <p>Specify the destination type if the following are true:</p> <ul style="list-style-type: none"> ■ The web service descriptor to which the endpoint alias is assigned use the In-Out message exchange pattern. ■ The JMS connection alias specified by the SOAP-JMS trigger connects to the webMethods Broker natively. On the webMethods Broker, a queue and topic can have the same name. You must specify Reply To Type to indicate to which destination the reply will be sent. <p>Select one of the following destination types:</p> <ul style="list-style-type: none"> ■ Queue. Specifies that the web service sends the response message to a particular queue. ■ Topic. Specifies that the web service sends the response message to a particular topic.

6. In the JMS WSDL options section, provide the following information:

Field	Description
Include connection factory name	When selected, includes the connection factory name in the JMS URI.
Include JNDI parameters	When selected, includes the JNDI parameters in the JMS URI.

Note:

The JMS URI appears in the WSDL document as the location attribute value for the address element contained within the port element.

7. In the WS security properties section, provide the following information:

Field	Description
Keystore alias	Alias of the keystore containing the private key used to decrypt the inbound SOAP request or sign the outbound SOAP response. Note: The provider must have already given the consumer the corresponding public key.
Key alias	Alias of the private key used to decrypt the request or sign the response. The key must be in the keystore specified in Keystore alias.
Truststore alias	The alias for the truststore that contains the list of CA certificates that API Gateway uses to validate the trust relationship.
Timestamp precision	Specifies whether the timestamp is precise to the second or millisecond. If you set the precision to milliseconds, API Gateway uses the timestamp format yyyy-MM-dd'T'HH:mm:ss:SSS'Z'. If you set the precision to seconds, API Gateway uses the timestamp format yyyy-MM-dd'T'HH:mm:ss'Z'. If you do not select a precision value, API Gateway uses the value specified for the <code>watt.server.ws.security.timestampPrecisionInMilliseconds</code> parameter.
Timestamp time to live (seconds)	Specifies the time-to-live value for the outbound message in seconds.

Field	Description
	<p>API Gateway uses the time-to-live value to set the expiry time in the Timestamp element of outbound messages. The Timestamp Time to Live value must be an integer greater than 0.</p> <p>If you do not specify a Timestamp time to live value, API Gateway uses the value specified for the <code>watt.server.ws.security.timestampTimeToLive</code> parameter.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: The Timestamp time to live value should be greater than the Time to live value specified under JMS transport properties.</p> </div>
<p>Timestamp maximum skew (seconds)</p>	<p>Specifies the maximum number of seconds that the web services client and host clocks can differ and still allow timestamp expiry validation to succeed.</p> <p>Specify a positive integer or zero.</p> <p>API Gateway uses the timestamp maximum skew value only when you implement WS-Security via a WS-Policy. API Gateway validates the inbound SOAP message only when the creation timestamp of the message is less than the sum of the timestamp maximum skew value and the current system clock time.</p> <p>If you do not specify a timestamp maximum skew value, API Gateway uses the value specified for the <code>watt.server.ws.security.timestampMaximumSkew</code> parameter.</p>

8. Click **Add**.

The web service endpoint alias created is listed in a table under Web service endpoints list.

Creating an Endpoint Alias for a Consumer Web Service Descriptor

A web service endpoint alias for use with a consumer web service descriptor that has a JMS binder specifies how and where API Gateway sends a request message when executing a web service descriptor.

When creating a consumer web service descriptor, API Gateway extracts the JMS information from the WSDL document and saves it with the binder information in the web service descriptor. However, as indicated in the SOAP over Java Message Service standard, the only JMS information required in the WSDL is the lookup variant and the destination name. Consequently, it is possible that some information necessary to connect to the JMS provider is absent from the WSDL. API

Gateway uses the information in a JMS consumer web service endpoint alias to replace or supplement the JMS information specified in the WSDL document.

When creating a consumer web service descriptor, the message addressing properties define the WS-addressing headers information that can be attached to the SOAP message.


Keep the following points in mind when creating a web service endpoint alias for use with a consumer web service descriptor with a SOAP over JMS binding:

- A JMS consumer web service endpoint alias can specify one of the following options to connect to a JMS provider:
 - JNDI provider alias and a connection factory
 - JMS connection alias

Only specify a JNDI provider alias and connection factory, or JMS connection alias, if information for connecting to the JMS provider was not included in the WSDL document used to create the consumer web service descriptor or if you want to overwrite the connection information included in the WSDL document.

- If you want to use the client side queue with the web service descriptor to which the alias is assigned, you must specify a JMS connection alias as the way to connect to the JMS provider.
- Information in the JMS consumer web service endpoint alias can supplement or replace the JMS URI information obtained from a WSDL.

➤ To create a consumer web service endpoint alias for use with JMS

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Web services**.
API Gateway displays a list of all the currently defined endpoint aliases.
3. Click **Add web service endpoint alias**.
4. In the Webservice endpoint alias properties section, provide the following information:

Field	Description
Alias	Name of the JMS provider web service endpoint alias. The alias name cannot include the following special characters: # © \ & @ ^ ! % * : \$. / \ \ ` ` ; , ~ + =) (} { } [] > < "
Description	A description for the endpoint alias.
Type	Specifies the type - Consumer

Field	Description
Execute ACL	<p>Specifies the ACL that governs which user groups on your server can use this web service endpoint alias.</p> <p>Select an ACL from the drop-down list. By default, only members of groups governed by the Internal ACL can use this alias.</p>

5. In the JMS transport properties section, provide the following information depending upon whether you want to connect to the JMS provider using a connection factory or a JMS connection alias:

Field	Description
JNDI properties	<p>Select this option if you want to connect to the JMS provider using a connection factory.</p> <p>Provide the following information:</p> <ul style="list-style-type: none"> ■ JNDI provider alias. The alias for the JNDI provider that API Gateway uses to look up administered objects. ■ Connection factory name. The lookup name for the connection factory to use to create a connection to the JMS provider. <div style="border: 1px solid gray; padding: 5px; margin-top: 10px;"> <p>Note: You have to specify a connection factory only if the WSDL document used to create the consumer web service descriptor does not specify a connection factory or you want to overwrite the connection factory.</p> </div>

JMS connection alias	<p>Select this option if you want to connect to the JMS provider using a JMS connection alias.</p> <p>Provide the following information:</p> <ul style="list-style-type: none"> ■ JMS connection alias. The name of the JMS connection alias that you want API Gateway to use to connect to the JMS provider.
-----------------------------	---

6. In the WS security properties section, provide the following information:

Field	Description
Username	The user name to include with the UsernameToken

Field	Description
Password	The password to include with the UsernameToken (must be plain text)
Retype password	Re-type the above password.
Keystore alias	<p>Alias to the keystore that contains the private key used to:</p> <ul style="list-style-type: none"> ■ Sign outbound SOAP requests ■ Include an X.509 authentication token for outbound SOAP requests ■ Decrypt inbound SOAP responses <p>Note: To verify messages from this consumer, the web services provider must have a copy of the corresponding public key.</p>
Key alias	<p>Alias to the private key used to sign or include X.509 authentication token for outbound SOAP messages or decrypt inbound SOAP responses.</p> <p>The key must be in the keystore specified in Keystore alias.</p>
Truststore alias	The alias for the truststore that contains the list of CA certificates that API Gateway uses to validate the trust relationship.
Timestamp precision	<p>Specifies whether the timestamp is precise to the second or millisecond.</p> <p>If you set the precision to milliseconds, API Gateway uses the timestamp format yyyy-MM-dd'T'HH:mm:ss:SSS'Z'. If you set the precision to seconds, API Gateway uses the timestamp format yyyy-MM-dd'T'HH:mm:ss'Z'.</p> <p>If you do not select a precision value, API Gateway uses the value specified for the <code>watt.server.ws.security.timestampPrecisionInMilliseconds</code> parameter.</p>
Timestamp time to live (seconds)	<p>Specifies the time-to-live value for the outbound message in seconds.</p> <p>API Gateway uses the time-to-live value to set the expiry time in the Timestamp element of outbound messages.</p>

Field	Description
	<p>The Timestamp Time to Live value must be an integer greater than 0.</p> <p>If you do not specify a Timestamp time to live value, API Gateway uses the value specified for the <code>watt.server.ws.security.timestampTimeToLive</code> parameter.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: The Timestamp time to live value should be greater than the Time to live value specified under JMS transport properties.</p> </div>
<p>Timestamp maximum skew (seconds)</p>	<p>Specifies the maximum number of seconds that the web services client and host clocks can differ and still allow timestamp expiry validation to succeed.</p> <p>Specify a positive integer or zero.</p> <p>API Gateway uses the timestamp maximum skew value only when you implement WS-Security via a WS-Policy. API Gateway validates the inbound SOAP message only when the creation timestamp of the message is less than the sum of the timestamp maximum skew value and the current system clock time.</p> <p>If you do not specify a timestamp maximum skew value, API Gateway uses the value specified for the <code>watt.server.ws.security.timestampMaximumSkew</code> parameter.</p>

7. Click **Add**.

The web service endpoint alias created is listed in a table under Web service endpoints list.

JMS Triggers

JMS Triggers are meant only for SOAP APIs. These triggers are created automatically when you create a Web Service Provider endpoint and act as Listener for that Web Service Provider endpoint.

API Gateway provides ways for managing JMS triggers and the resources used by JMS triggers. You can only update a trigger and cannot create a trigger. Specifically, you can use the controls provided by API Gateway to:

- Increase or decrease the maximum number of server threads used for JMS triggers
- Change the maximum execution threads for concurrent JMS triggers
- Change the destinations to which the trigger subscribes

- Change the JMS connection alias used by the trigger
- Delay the frequency with which a JMS trigger polls for more messages

The Individual SOAP JMS trigger controls section displays all the JMS triggers that exist on the API Gateway along with a summary of each trigger. The summary includes the current status, state, and thread usage of the trigger as well as configuration information such as the JMS connection alias used by the trigger, the destination to which the trigger subscribes, and the processing mode of the trigger.

JMS trigger status and state

The state of a JMS trigger indicates whether the trigger is enabled, disabled, or suspended. The status indicates whether trigger is running.

A JMS trigger can have one of the following states:

- **Enabled.** The JMS trigger is available. A JMS trigger must be enabled for it to receive and process messages. An enabled trigger can have a status of “Not Running” which means that it would not receive and process messages. Reasons that an enabled JMS trigger can be disabled include: a disabled JMS connection alias, an exception thrown by the trigger, and trigger failure at startup.
- **Disabled.** The JMS trigger is not available. Integration Server neither retrieves nor processes messages for the JMS trigger. The JMS trigger remains in this state until you enable the trigger.

A JMS trigger can have a status of “Running” or “Not Running (reason)” where reason identifies why the trigger is not running, such as “Not Running (trigger disabled)”.


You can enable a trigger by clicking the toggle button . The toggle button changes to  to depict that the trigger is enabled.

You can disable the trigger by clicking the toggle button . The toggle button changes to  to depict that the trigger is disabled.

Updating JMS triggers

When API Gateway creates a JMS trigger as part of creating a provider web service endpoint alias, API Gateway uses default values for some of the trigger properties and placeholders for other properties. You can modify the default and placeholder values in the SOAP JMS trigger controls

» To update a JMS trigger

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Messaging**.

The SOAP JMS trigger controls section displays a list of all the currently defined JMS triggers and the corresponding details.

3. Click a JMS trigger.

The Update JMS trigger screen appears.

4. In the JMS destinations and message selectors section you can modify the following information as required:
 - a. **Destination name.** Type the name of the queue from which the JMS trigger receives messages.
5. In the Settings section you can modify the following information as required:
 - a. **JMS connection alias.** Specify the JMS connection alias used by the trigger.
6. In the Properties section you can select one of the following Processing modes:
 - **Concurrent.** Specifies that API Gateway processes multiple messages for this trigger at one time. You can modify the following information for this processing mode:
 - **Max execution threads.** Specify the maximum number of messages that API Gateway can process concurrently.
 - **Connection count.** Specify the number of connections you want the JMS trigger to make to the webMethods Broker.

Note:

The Connection count value must be less than or equal to the Max Execution Threads value.

- **Serial.** Specifies that API Gateway processes messages received by the trigger one after the other.
7. Click **Update**.

The trigger is saved with the modified values.

Transaction Alerts

API Gateway supports core as well as transaction-based licensing model. When API Gateway uses a transaction-based licensing model, then each service invocation is considered as a transaction and API Gateway keeps a track of these transactions. API Gateway transactions for the current month is compared with the maximum number of transactions allowed in a month. The maximum number of transactions that are allowed in a month is specified by the license agreement as represented in the licenseKey.xml file available in the `InstallDir\IntegrationServer\instances\instance_name\config` directory.

When the calculated transaction count is higher than the maximum transactions allowed per month, users are notified through an email or through a notification that appears in the API

Gateway UI. You must have the manage user administration functional privilege assigned to view the usage details in the **Analytics > API usage details** page.

Note:

API Gateway also uses core-based license model. However, only when API Gateway uses a transaction-based licensing model, the **General > Transaction alerts** and the **Analytics > API usage details** pages appear in API Gateway.

Configuring Criteria for Transaction Alert Notification


➤ To configure transaction alert criteria

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Transaction alerts**.

This displays a list of available alert criteria and the corresponding details.

3. Click **Add Criteria** and provide the following information:

Field	Description
Notify at	<p>Select the usage percentage at which the notification is to be sent.</p> <p>Note: If API Gateway is enabled with transaction based license, then by default, two API Gateway UI notifications appear at 90% and at 100% marks. You cannot delete these default notifications, however, you can modify these default notifications.</p>
Notify through	<p>Specifies the way in which a user is notified about the transaction alert.</p> <p>Select one of the following options:</p> <ul style="list-style-type: none"> ■ Email. The transaction alert information is sent through an email. ■ API Gateway UI. The transaction alert information is shared through API Gateway user interface. ■ Both. The transaction alert information is shared both through email as well as API Gateway user interface.
Email	A valid email address to which the transaction alerts are sent.



Field	Description
	You can add multiple email addresses by clicking  . Additionally, you can edit (by clicking the edit icon) or delete (by clicking the delete icon) the email address.

4. Click **Save**.

The criteria is saved and listed in the table of available alert criteria.



Modifying License Alert Configurations

> To modify a transaction alert criteria

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Transaction alerts**.
3. In the Transaction alert configurations list, locate the alert that you want to modify and click  in the Action column.
4. Incorporate the required changes.
5. Click **OK**.

Deleting a Transaction Alert Configuration

> To delete a transaction alert criteria

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Transaction alerts**.
3. In the Transaction alert configurations list, locate the row that contains the alert criteria you want to delete, and click .

The alert criteria is deleted from the Transaction alert configurations list.

Security Configuration

You must have the API Gateway's manage security configurations functional privilege assigned to perform the following tasks in the security configuration section of API Gateway:

- Configure the keystores and truststores required for incoming message-level security and transport-level security.
- Configure ports of API Gateway.
- Configure the SAML issuer to use in API Gateway outbound authentication to fetch the SAML token from the STS (Security Token Service).
- Configure the custom assertions to use in inbound authentication of API Gateway.
- Configure Kerberos settings.
- Manage master password.
- Configure JSON web token(JWT), OAuth, and OpenID authorization servers and third-party providers.

Keystore and Truststore

Keystores and truststores are secure files with industry-standard file formats. The keystore file stores the private keys and SSL certificates and the truststore file stores the trusted roots for the certificates.

A keystore file contains one or more pairs of a private key and signed certificate for its corresponding public key. The keystore should be strongly protected with a password, and stored (either on the file system or elsewhere) so that it is accessible only to administrators.

The truststore file functions as a database containing all the public keys for CAs (Certificate Authorities) within a specified trusted directory.

To enable the two-way SSL for an API Gateway service, you must add a valid, authorized X.509 certificate along with the private key in a keystore file, and the certificate of the client or partner in the truststore file. These keystore and truststore files have to be referred to in the HTTPs port that is used to access the API Gateway service.


API Gateway has a sample keystore that contains self-signed certificates, which are located in `InstallDir\IntegrationServer\instances\default\packages\WmAPIGateway\config\resources\security`. The sample self-signed certificates are specific to localhost and hence Software AG recommends not to use them for configuring SSL communication with API Gateway in a production environment.

Note:

Any modifications to the keystore and truststore aliases in Integration Server do not reflect in API Gateway. Hence, Software AG recommends that you do not modify the aliases through the Integration Server Administrator. On migration from 10.0 to 10.1, the existing configuration in 10.0 is migrated to the API Gateway UI.

Configuring Keystore Information

➤ **To configure Keystore information**

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Security**.

A list of existing keystores and truststores and corresponding details are displayed.

3. In the Keystores section, click **Add keystore**.
4. In the Create keystore section, provide the following information:

Field	Description
Alias	<p>A text identifier for the keystore file.</p> <p>The alias name can contain only letters, numbers and underscores. It can not include a space, hyphen and special characters.</p> <p>The keystore contains the private keys and certificates (including the associated public keys) for an Integration Server, partner application, or Integration Server component.</p>
Select file	Select a keystore file of the specified type using Browse. Select the required file and upload it.
Password	Password for the saved keystore file associated with this alias.
Type	The certificate file format of the keystore file, which by default is JKS for keystores. You can also use PKCS12 format for a keystore.
Description	Optional. A text description for the keystore alias.

5. Click **OK**.

All the key aliases in the uploaded file are listed.

6. Type a password for the required key alias.
7. Click **Save**.


The keystore is configured and the alias listed in the keystore alias table.

Note:

If a wrong password has been provided for the keystore or one of its aliases, API Gateway saves the keystore but it is not loaded. The keystore alias is displayed as the loaded icon with an X mark in the keystore listing.

Modifying Keystore Information

> To modify keystore information

1. Expand the menu options icon , in the title bar, and select **Administration**.

2. Select **General > Security**.

A list of keystores and truststores and corresponding details are displayed.

3. Click the keystore alias to be updated.

The update keystore section is displayed.

4. Modify the fields as required.

5. Click **Save**.

The set of available aliases is displayed.

Note:

If the keystore file is not updated during the edit, then clicking **Save** closes the form. If a different keystore file is uploaded, then the list of aliases in the file is loaded and you are prompted to configure the passwords for the aliases.

6. Type a password for the alias to be configured.


7. Click **Save**.

The keystore is updated.

Deleting Keystore Information


Be careful while deleting the keystore information. If the keystore and one of its key aliases is configured in the keystore settings and the keystore gets deleted, then the configuration would have issues.

> To delete keystore information

1. Expand the menu options icon , in the title bar, and select **Administration**.

2. Select **General > Security**.

A list of keystores and truststores and corresponding details are displayed.


3. Click  in the action column of the keystore to be deleted.

4. Click **Yes** in the confirmation dialog.

The keystore is deleted from the list.

Configuring Truststore Information

> To configure truststore information

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Security**.

A list of existing keystores and truststores and corresponding details are displayed.

3. In the Truststores section, click **Add truststore**.
4. In the Create truststore section, provide the following information:

Field	Description
Name	<p>A name for the truststore file.</p> <p>The alias name can contain only letters, numbers and underscores. It can not include a space, hyphen and special characters.</p> <p>The truststore contains the trusted CA certificates for an Integration Server, partner application, or Integration Server component.</p>
Upload truststore file	<p>Select a truststore file of the specified type using Browse. Select the required file and upload it.</p> <p>Note: Supports only JKS file format.</p>
Password	<p>Password that is used to protect the contents of the truststore.</p> <p>This password must have been defined at truststore creation time using a keystore utility.</p> <p>Make sure you have the truststore password available when managing its corresponding truststore alias.</p>
Description	<p>Optional. A text description for the truststore alias.</p>

5. Click **Save**.


The truststore is configured and the alias listed in the truststore alias table.

Note:

If a wrong password has been entered for the truststore, API Gateway saves the truststore but it is not loaded. The truststore alias is displayed as the loaded icon with an X mark in the truststore listing.

Modifying Truststore Information

> To modify truststore information

1. Expand the menu options icon , in the title bar, and select **Administration**.

2. Select **General > Security**.

A list of keystores and truststores and corresponding details are displayed.

3. Click the truststore alias to be updated.

The update truststore section is displayed.

4. Modify the fields as required.


5. Click **Save**.

The truststore is updated.

Deleting Truststore Information


Be careful while deleting the truststore information. If the truststore settings and the truststore gets deleted, then the configuration would have issues.

> To delete keystore information

1. Expand the menu options icon , in the title bar, and select **Administration**.

2. Select **General > Security**.

A list of keystores and truststores and corresponding details are displayed.

3. Click  in the action column of the truststore to be deleted.


4. Click **Yes** in the confirmation dialog.

The truststore is deleted from the list.

Configuring Keystore and Truststore Information for Inbound Messages

You might want to configure API Gateway to refer to a default keystore, truststore, or both, before deploying any SOAP message flows that require signature, encryption, X.509 authentication, and so on, as configured in the Inbound Authentication - Message policy. The default keystore and truststore are that you want API Gateway to use for the incoming secured messages.

➤ To configure keystore and truststore settings for inbound messages

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Security**.

A list of existing keystores and truststores loaded during startup, and those created in API Gateway and the corresponding details appears.

3. To configure API Gateway's default keystore and truststore alias for incoming secured messages, provide the following information in the Configure keystore and truststore settings for inbound messages section:

Field	Description
Keystore alias	Select a keystore that API Gateway uses for incoming message-level security. Lists all available keystores. If you have not configured any keystore, the list is empty.
Key alias (signing)	Select the alias for the private key to sign the outgoing response from API Gateway to the original client. This alias value validates the inbound requests to API Gateway and signs the outgoing response from API Gateway to the original client. This field is auto-populated based on the selected keystore alias. It lists all the aliases available in the chosen keystore. If there are no configured keystores, this field is empty.
Truststore alias	The alias for the truststore that contains the list of CA certificates that API Gateway uses to validate the trust relationship with the client.

4. Click **Save**.

Post-requisites

While securing the SOAP APIs using WS-Security policies, perform the following:


1. Restart the server after configuring keystore and truststore information for the configuration to take effect.
2. Deactivate the APIs that have Inbound Authentication - Message policy enforced.

3. Update the keystore and truststore configuration.
4. Activate the APIs that were deactivated.

Configuring Keystore and Truststore Information for Outbound Connections

You might want to configure API Gateway to refer to a default truststore that you want API Gateway to use for securing outgoing SSL connections. The keystore and key alias can be configured for outgoing two-way SSL connections. During the SSL handshake between API Gateway and the native API, the server certificate, which is sent by the native API, has to be validated against a truststore in API Gateway.

➤ To configure keystore and truststore settings for outbound secured connections

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Security**.

A list of existing keystores and truststores loaded during startup, and those created in API Gateway and the corresponding details appears.

3. To configure API Gateway's default keystore and truststore alias for outgoing secured connections, provide the following information in the Configure keystore and truststore settings for outbound connections section:

Field	Description
Keystore alias	<p>Select a keystore that API Gateway uses for outgoing secured connections.</p> <p>Lists all available keystores. If you have not configured any keystore, the list is empty.</p>
Key alias	<p>Select the alias for the private key for an outbound connection from API Gateway to the native API.</p> <p>This field is auto-populated based on the selected keystore alias. It lists all the aliases available in the chosen keystore. If there are no configured keystores, this field is empty.</p>
Truststore alias	<p>The alias for the truststore that contains the list of CA certificates that API Gateway uses to validate the trust relationship with the native API.</p> <p>If you do not configure any truststore alias, it implies that API Gateway does not validate the certificates provided by native APIs.</p>

4. Click **Save**.

Ports

API Gateway listens for requests on ports that you specify. Each port is associated with a specific type of protocol, HTTP or HTTPS. In addition to these port types, API Gateway also provides three ports; API Gateway external port, API Gateway internal listener port, and the WebSocket listener port.

You can specify one or more HTTP or HTTPS ports on which API Gateway and the deployed APIs are available for consumption. API Gateway, by default, is available on the primary HTTP port. The primary HTTP port is the port specified on the Integration Server's Security > Ports page.

If your API Gateway is behind an internal firewall and is not allowed to accept communications from external clients through the DMZ, then you can configure the API Gateway instance in DMZ with an external port to listen to requests from external clients and using reverse invoke route them to the internal servers. The API Gateway internal listener port or the WebSocket listener port pulls the requests from the registration port of API Gateway in DMZ thus safeguarding from any malicious attacks.

External clients send requests to API Gateway. API Gateway external port listens to this client information from each request and evaluates the request against any API Gateway rules that have been defined. It then passes requests that have not violated a rule to the API Gateway internal port or the WebSocket listener port. These listener ports process the requests and send the responses to API Gateway, which then passes the responses back to the client.

The Ports page displays the following information about the configured ports:

Field	Description
Ports	<p>Specifies the port number.</p> <p>Click on the port number to edit the port configuration.</p> <p>API Gateway does not allow you to update an active port as long as the port is enabled. If you want to update an active port, API Gateway first automatically disables the port, then updates the configured details. On successful update, API Gateway enables the port back. In case, when updating the port, if an error occurs, API Gateway displays the error message that stops from updating the active port.</p>
Alias	Specifies the port alias that can be used across to refer to the corresponding port.
Protocol	Specifies the protocol used by the port to communicate.
Type	Specifies the type of port.
Enabled	Specifies the status of the port; whether it is enabled or disabled.
Accessmode	Allows to configure whether the port must allow or deny access to ESB services and folders, by default. For information on how to configure the access mode for a port, see “Configuring Access Mode for a Port” on page 135 .

Field	Description
IP Accessmode	Allows to configure whether the port must allow or deny access to the external hosts, or must follow global configuration for allowing or denying external hosts. For information on how to configure the IP access mode of a port, see “Configuring IP Access Mode for a Port” on page 133 .
Primary port	<p>Specifies whether the port is used as a primary port.</p> <p>✓ specifies that it is set as a primary port and ✗ depicts it is not set as a primary port. Only the HTTP and HTTPS ports can be set as primary ports once they are enabled.</p> <p>You can click ✗ for a port to set it as a primary port.</p> <p>Note: You can not disable or delete the primary port. Also, you can not modify the primary port details as long as the port is set as primary port.</p>
Description	Provides a short description of the port.
Action	Specifies the actions that can be performed on the port.


Note:

Any modifications to the ports in Integration Server may not reflect in API Gateway UI. Hence, Software AG recommends that you do not configure or modify the ports through Integration Server Administrator UI.

Adding an HTTP Port

The HTTP port enables API Gateway to authenticate the client and server securely and exchange the data. In addition, you can configure the type of client authentication that you want the server to perform. Client authentication allows you to verify the identity of the client.

➤ To add an HTTP port

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Security > Ports**.
The ports page lists all the ports configured with API Gateway, if any.
3. Click **Add ports**.
4. Select the type of port as **HTTP** and click **Add**.
5. Provide the following information:

Field	Description
HTTP listener configuration	
Port	Specify the number you want to use for the port. Select a number that is not already in use on this host machine.
Alias	Specifies an alias for the port that is unique for this API Gateway. An alias must be between 1 and 255 characters in length and include one or more of the following: letters (a -z, A-Z), numbers (0-9), underscore (_), period (.), and hyphen (-).
Description (optional)	Provide a description of the port.
Bind address (optional)	Specifies the IP address to which to bind this port. Specify a bind address if your machine has multiple IP addresses and you want the port to use this specific address. If you do not specify a bind address, API Gateway picks one for you.
Backlog	Specifies the number of requests that can remain in the queue for an enabled port before API Gateway begins rejecting requests. The default is 200. The maximum value is 65535.
Keep alive timeout	Specifies when to close the connection if the server has not received a request from the client within this timeout value (in milliseconds) or when to close the connection if the client has explicitly placed a close request with the server.
Private threadpool configuration. Specifies whether to create a private thread pool for this port or use the common thread pool.	
Enable	Select to enable the private threadpool configuration for this port.
Threadpool min	Specifies the minimum number of threads for this private threadpool. The default value is 1.
Threadpool max	Specifies the maximum number of threads for this private thread pool. The default value is 5.
Thread priority	Specifies the Java thread priority. The default value is 5.
Security configuration	
Client authentication	Specifies the type of client authentication you want API Gateway to perform for requests that arrive on this HTTPS port. Select one of the following:

Field	Description
	<ul style="list-style-type: none"> <li data-bbox="607 260 1474 363">■ Username/Password. API Gateway does not request client certificates. The server looks for user and password information in the header of requests coming from an external client. <li data-bbox="607 390 1474 632">■ Digest. API Gateway uses password digest to authenticate all requests. If the client does not provide the authentication information, API Gateway returns an HTTP WWW-Authenticate header with digest scheme to the client requesting for authentication information. If the client provides the required authentication information, API Gateway verifies and validates the request. <li data-bbox="607 659 1474 968">■ Request Kerberos Ticket. API Gateway looks for a Kerberos ticket in the HTTP Authorization header using the Negotiate authentication scheme. If it does not find the ticket, API Gateway uses user name and password for basic authentication. If the client does not provide any authentication information, API Gateway returns an HTTP WWW-Authenticate header with negotiate scheme to the client requesting for authentication information. If the client provides the required authentication information, API Gateway verifies and validates the request. <li data-bbox="607 995 1474 1304">■ Require Kerberos Ticket. API Gateway looks for a Kerberos ticket in the HTTP Authorization header using the Negotiate authentication scheme. If it does not find the ticket, API Gateway fails the authentication. If the client does not provide any authentication information, API Gateway returns an HTTP WWW-Authenticate header with negotiate scheme to the client requesting for authentication information. If the client provides the required authentication information, API Gateway verifies and validates the request.

You have to enable Kerberos by providing the following Kerberos properties with details that are used for handling service requests that come with a Kerberos ticket:

- **JAAS context.** Specify the custom JAAS context used for Kerberos authentication.
- **Principal.** Specify the name of the principal to use for Kerberos authentication.
- **Principal password.** Specify the password for the principal to use to authenticate the principal to the KDC.
- **Retype principal password.** Retype the principal password.
- **Service principal name.** Specify the name of the principal used with the service that the Kerberos client wants to access.

Field	Description
	<p>Note: API Gateway supports the <i>username</i> format for Service Principal Names (SPNs). This format represents the principal name as a named user defined in LDAP used for authentication to the KDC.</p>

- Click **Add**.

The port is created and is listed in the ports table.

Important:

The default access mode of the port is set to *Allow by default*. This implies that the port allows connections to all ESB services and folders. Users must note that this setting allows access to all enterprise assets hosted in internal Integration Server. There is a potential security risk for the IS assets that are secured by Anonymous Access Control Lists (ACL) or if the installation is exposed to the public internet. Hence, you can set the access mode of the port to *Deny by default* before enabling it. When you change the access mode, you add the required services and folders to the *Allow* list. For more information, see [“Configuring Access Mode for a Port” on page 135](#).

Also, the global IP access mode will be applied to the newly created HTTP ports. You can modify the IP access mode as per your requirement. For information on modifying IP access mode of ports, see [“Configuring IP Access Mode for a Port” on page 133](#).


- Click the  icon in the **Enabled** column next to the port to enable the port.

The port is enabled and a success message appears.

Adding an HTTPS Port

The HTTPS port enables API Gateway to authenticate the client and server securely and encrypt the data exchanged. By default, the HTTPS listener uses the certificates for the default SSL key. In addition, you can configure the type of client authentication that you want the server to perform. Client authentication allows you to verify the identity of the client.

> To add an HTTPS port

- Expand the menu options icon , in the title bar, and select **Administration**.
- Select **Security > Ports**.

The ports page lists all the ports configured with API Gateway, if any.

- Click **Add Ports**.

4. Select the type of port as **HTTPS** and click **Add**.
5. Provide the following information:

Field	Description
HTTPS listener configuration	
Port	Specify the number you want to use for the port. Select a number that is not already in use on this host machine.
Alias	Specifies an alias for the port that is unique for this API Gateway. An alias must be between 1 and 255 characters in length and include one or more of the following: letters (a -z, A-Z), numbers (0-9), underscore (_), period (.), and hyphen (-).
Description (optional)	Provide a description of the port.
Bind address (optional)	Specifies the IP address to which to bind this port. Specify a bind address if your machine has multiple IP addresses and you want the port to use this specific address. If you do not specify a bind address, API Gateway picks one for you.
Backlog	Specifies the number of requests that can remain in the queue for an enabled port before API Gateway begins rejecting requests. The default is 200. The maximum value is 65535.
Keep alive timeout	Specifies when to close the connection if the server has not received a request from the client within this timeout value (in milliseconds) or when to close the connection if the client has explicitly placed a close request with the server.
Private threadpool configuration. Specifies whether to create a private thread pool for this port or use the common thread pool.	
Enable	Select to enable the private threadpool configuration for this port.
Threadpool min	Specifies the minimum number of threads for this private threadpool. The default is 1.
Threadpool max	Specifies the maximum number of threads for this private thread pool. The default is 5.
Thread priority	Specifies the Java thread priority. The default is 5.
Security configuration	
Client authentication	Specifies the type of client authentication you want API Gateway to perform for requests that arrive on this HTTPS port.

Field	Description
-------	-------------

Select one of the following:

- **Username/Password.** API Gateway does not request client certificates. The server looks for user and password information in the header of requests coming from an external client.
- **Digest.** API Gateway uses password digest to authenticate all requests. If the client does not provide the authentication information, API Gateway returns an HTTP WWW-Authenticate header with digest scheme to the client requesting for authentication information. If the client provides the required authentication information, API Gateway verifies and validates the request.
- **Request Kerberos Ticket.** API Gateway looks for a Kerberos ticket in the HTTPS Authorization header using the Negotiate authentication scheme. If it does not find the ticket, API Gateway uses user name and password for basic authentication. If the client does not provide any authentication information, API Gateway returns an HTTP WWW-Authenticate header with negotiate scheme to the client requesting for authentication information. If the client provides the required authentication information, API Gateway verifies and validates the request.
- **Require Kerberos Ticket.** API Gateway looks for a Kerberos ticket in the HTTPS Authorization header using the Negotiate authentication scheme. If it does not find the ticket, API Gateway fails the authentication. If the client does not provide any authentication information, API Gateway returns an HTTP WWW-Authenticate header with negotiate scheme to the client requesting for authentication information. If the client provides the required authentication information, API Gateway verifies and validates the request.

You have to enable Kerberos by providing the following Kerberos properties with details that are used for handling service requests that come with a Kerberos ticket:

- **JAAS context.** Specify the custom JAAS context used for Kerberos authentication.
- **Principal.** Specify the name of the principal to use for Kerberos authentication.
- **Principal password.** Specify the password for the principal to use to authenticate the principal to the KDC.
- **Retype principal password.** Retype the principal password.

Field	Description
	<ul style="list-style-type: none"> ■ Service principal name. Specify the name of the principal used with the service that the Kerberos client wants to access. <div style="background-color: #f0f0f0; padding: 5px; margin: 5px 0;"> <p>Note: API Gateway supports the <i>username</i> format for Service Principal Names (SPNs). This format represents the principal name as a named user defined in LDAP used for authentication to the KDC.</p> </div> <ul style="list-style-type: none"> ■ Request Client Certificate. API Gateway requests client certificates for all requests. If the client does not provide a certificate, the server prompts the client for a userid and password. The server checks whether the certificate exactly matches a client certificate on file and is signed by a trusted authority. If so, the client is logged in as the user to which the certificate is mapped in API Gateway. If not, the client request fails, unless central user management is configured. ■ Require Client Certificate. API Gateway requires client certificates for all requests. The server behaves as described for Request Client Certificates, except that the client must always provide a certificate. ■ Use Identity Provider. API Gateway uses an OpenID Provider to authenticate requests. API Gateway redirects all requests sent to this port to the OpenID Provider specified in Identity Provider.

Listener specific credentials

Keystore alias	<p>Specifies a user-specified, text identifier for an API Gateway keystore.</p> <p>The alias points to a repository of private keys and their associated certificates. Although each listener points to one keystore, there can be multiple keys and their certificates in the same keystore, and more than one listener can use the same keystore alias.</p>
Key alias (signing)	<p>Specifies the private key of keystore.</p>
Truststore alias	<p>Specifies the public certificates of truststore.</p> <p>The alias points to a repository of public certificates.</p>

6. Click **Add**.


The port is created and is listed in the ports table.

Important:

The default access mode of the port is set to *Allow by default*. This implies that the port allows connections to all ESB services and folders. Users must note that this setting allows access

to all enterprise assets hosted in internal Integration Server. There is a potential security risk for the IS assets that are secured by Anonymous Access Control Lists (ACL) or if the installation is exposed to the public internet. Hence, you can set the access mode of the port to *Deny by default* before enabling it. When you change the access mode, you add the required services and folders to the *Allow* list. For more information, see [“Configuring Access Mode for a Port” on page 135](#).

Also, the global IP access mode will be applied to the newly created HTTPS ports. You can modify the IP access mode as per your requirement. For information on modifying IP access mode of ports, see [“Configuring IP Access Mode for a Port” on page 133](#).


7. Click the  icon in the **Enabled** column next to the port to enable the port.

The port is enabled and a success message appears.

Adding an API Gateway External Port

The API Gateway external and registration ports work as a pair. One port is not functional without the other.

➤ To add an API Gateway external port

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Security > Ports**.
The ports page lists all the ports configured with API Gateway, if any.
3. Click **Add Ports**.
4. Select the type of port as **API Gateway external** and click **Add**.
5. Provide the following information:

Field	Description
API Gateway external listener configuration. Provide the following details to configure the HTTP listener set up.	
External port	Specifies the port number you want to use for the external port. Use a number that is not already in use. This is the port that clients connect to through your outer firewall.
Alias	Specifies an alias for the port.

Field	Description
	An alias must be between 1 and 255 characters in length and include one or more of the following: letters (a -z, A-Z), numbers (0-9), underscore (_), period (.), and hyphen (-).
Description (optional)	A description of the port.
Protocol	Specifies the protocol to use for this port (HTTP or HTTPS). If you select HTTPS, additional security and credential boxes appear for which you have to provide the required values.
Bind address (optional)	Specifies the IP address to which to bind this port. Specify a bind address if your machine has multiple IP addresses and you want the port to use this specific address. If you do not specify a bind address, API Gateway picks one for you.
Backlog	Specifies the number of requests that can remain in the queue for an enabled port before API Gateway begins rejecting requests. The default is 200. The maximum value is 65535.
Keep alive timeout	Specifies when to close the connection if the server has not received a request from the client within this timeout value (in milliseconds) or when to close the connection if the client has explicitly placed a close request with the server. The default value is 20000ms
Private threadpool configuration.	Specifies whether to create a private thread pool for this port or use the common thread pool.
Enable	Select to enable the private threadpool configuration for this port.
Threadpool min	Specifies the minimum number of threads for this private threadpool. The default value is 1.
Threadpool max	Specifies the maximum number of threads for this private thread pool. The default value is 5.
Thread priority	Specifies the Java thread priority. The default value is 5.
Security configuration.	Provide the following details to configure security parameters
Client authentication	For the external port, specify the type of client authentication required. Select one of the following: <ul style="list-style-type: none"> ■ Username/Password. API Gateway does not request client certificates. The server looks for user and password information in the header of requests coming from an external client.

Field	Description
	<ul style="list-style-type: none"> <li data-bbox="500 260 1383 499">■ Digest. API Gateway uses password digest to authenticate all requests. If the client does not provide the authentication information, API Gateway returns an HTTP WWW-Authenticate header with digest scheme to the client requesting for authentication information. If the client provides the required authentication information, API Gateway verifies and validates the request. <li data-bbox="500 529 1383 842">■ Request Kerberos Ticket. API Gateway looks for a Kerberos ticket in the HTTPS Authorization header using the Negotiate authentication scheme. If it does not find the ticket, API Gateway uses user name and password for basic authentication. If the client does not provide any authentication information, API Gateway returns an HTTP WWW-Authenticate header with negotiate scheme to the client requesting for authentication information. If the client provides the required authentication information, API Gateway verifies and validates the request. <li data-bbox="500 871 1383 1178">■ Require Kerberos Ticket. API Gateway looks for a Kerberos ticket in the HTTPS Authorization header using the Negotiate authentication scheme. If it does not find the ticket, API Gateway fails the authentication. If the client does not provide any authentication information, API Gateway returns an HTTP WWW-Authenticate header with negotiate scheme to the client requesting for authentication information. If the client provides the required authentication information, API Gateway verifies and validates the request.

You have to enable Kerberos by providing the following Kerberos properties with details that are used for handling service requests that come with a Kerberos ticket:

- **JAAS context.** Specify the custom JAAS context used for Kerberos authentication.
- **Principal.** Specify the name of the principal to use for Kerberos authentication.
- **Principal password.** Specify the password for the principal to use to authenticate the principal to the KDC.
- **Retype principal password.** Retype the principal password.
- **Service principal name.** Specify the name of the principal used with the service that the Kerberos client wants to access.

Note:

API Gateway supports the *username* format for Service Principal Names (SPNs). This format represents the principal name as a

Field	Description
	<p>named user defined in LDAP used for authentication to the KDC.</p> <ul style="list-style-type: none"> ■ Request Client Certificate. API Gateway requests client certificates for all requests. If the client does not provide a certificate, the server prompts the client for a userid and password. The server checks whether the certificate exactly matches a client certificate on file and is signed by a trusted authority. If so, the client is logged in as the user to which the certificate is mapped in API Gateway. If not, the client request fails, unless central user management is configured. ■ Require Client Certificate. API Gateway requires client certificates for all requests. The server behaves as described for Request Client Certificates, except that the client must always provide a certificate. ■ Use Identity Provider. API Gateway uses an OpenID Provider to authenticate requests. API Gateway redirects all requests sent to this port to the OpenID Provider specified in Identity Provider.

Listener specific credentials (optional). This section appears only if you select the HTTPS option in the Protocol field of the API Gateway external listener configuration section. Provide the following details to configure listener specific credentials.

Keystore alias Specifies a user-specified, text identifier for an API Gateway keystore.

The alias points to a repository of private keys and their associated certificates. Although each listener points to one keystore, there can be multiple keys and their certificates in the same keystore, and more than one listener can use the same keystore alias.

Key alias (signing) Specifies the private key of keystore.

Truststore alias Specifies the public certificates of truststore.

The alias points to a repository of public certificates.

API Gateway registration listener configuration . Provide the following details to configure listener specific credentials.

Registration port Specifies the number you want to use for the registration port.

Use a number that is not already in use. It is best not to use a standard port such as 80 (the standard port for HTTP) or 443 (the standard port for HTTPS) because the external firewall allows access to those ports from the outside world.

You can add multiple registration ports by clicking **+Add**.

Field	Description
Alias	<p>Specifies an alias for the port.</p> <p>An alias must be between 1 and 255 characters in length and include one or more of the following: letters (a-z, A-Z), numbers (0-9), underscore (_), period (.), and hyphen (-).</p>
Description (optional)	A description of the port.
Protocol	<p>Specifies the protocol to use for this port (HTTP or HTTPS).</p> <p>If you select HTTPS, additional security and credential boxes appear for which you have to provide the required values.</p>
Bind address (optional)	<p>Specifies the IP address to which to bind this port.</p> <p>Specify a bind address if your machine has multiple IP addresses and you want the port to use this specific address. If you do not specify a bind address, API Gateway picks one for you.</p>
Security configuration. Provide the following details to configure security parameters.	
Client Authentication	<p>For the external port, specify the type of client authentication required..</p> <p>Select one of the following:</p> <ul style="list-style-type: none"> ■ Username/Password . API Gateway does not request client certificates. The server looks for user and password information in the header of requests coming from an external client. ■ Request Client Certificate. <i>This option appears only if you select the HTTPS option in the Protocol field of the API Gateway registration listener configuration section.</i> API Gateway requests client certificates for all requests. If the client does not provide a certificate, the server prompts the client for a userid and password. The server checks whether the certificate exactly matches a client certificate on file and is signed by a trusted authority. If so, the client is logged in as the user to which the certificate is mapped in API Gateway. If not, the client request fails, unless central user management is configured. ■ Require Client Certificate. <i>This option appears only if you select the HTTPS option in the Protocol field of the API Gateway registration listener configuration section.</i> API Gateway requires client certificates for all requests. The server behaves as described for Request Client Certificates, except that the client must always provide a certificate.
Listener specific credentials (optional). This section appears only if you select the HTTPS option in the Protocol field of the API Gateway external listener configuration section. Provide the following details to configure listener specific credentials.	

Field	Description
Keystore alias	Specifies a user-specified, text identifier for an API Gateway keystore. The alias points to a repository of private keys and their associated certificates. Although each listener points to one keystore, there can be multiple keys and their certificates in the same keystore, and more than one listener can use the same keystore alias.
Key alias (signing)	Specifies the private key of keystore.
Truststore alias	Specifies the public certificates of truststore. The alias points to a repository of public certificates.

6. Click **Add**.

The port is created and is listed in the ports table.

Important:

The global IP access mode will be applied to the newly created external and registration listener ports. You can modify the IP access mode as per your requirement. For information on modifying IP access mode of ports, see [“Configuring IP Access Mode for a Port” on page 133](#).


7. Click the  icon in the **Enabled** column next to the external and registration ports to enable them.

The port is enabled and a success message appears.

Configuring the API Gateway Internal Listener

The API Gateway Internal listener processes the requests received from the API Gateway External port and sends responses to API Gateway. This procedure describes how to connect the Internal listener to API Gateway.

> To configure the API Gateway Internal listener

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Security > Ports**.

The ports page lists all the ports configured with API Gateway, if any.

3. Click **Add Ports**.
4. Select the type of port as **API Gateway internal** and click **Add**.

5. Provide the following information:

Field	Description
Protocol	Specifies the protocol to use for this port (HTTP or HTTPS). If you select HTTPS, additional security and credential boxes appear for which you have to provide the required values.
Description (optional)	A description of the port.
Alias	Specifies an alias for the port. An alias must be between 1 and 255 characters in length and include one or more of the following: letters (a -z, A-Z), numbers (0-9), underscore (_), period (.), and hyphen (-).
Max connections	Specifies the number of connections maintained between API Gateway Internal port and API Gateway. The default value is 5.
Private threadpool configuration.	Specifies whether to create a private thread pool for this port or use the common thread pool.
Enable	Select to enable the private threadpool configuration for this port.
Threadpool min	Specifies the minimum number of threads for this private threadpool. The default value is 1.
Threadpool max	Specifies the maximum number of threads for this private thread pool. The default value is 5.
Thread priority	Specifies the Java thread priority. The default value is 5.
API Gateway external server	
Host	Specifies the host name or IP address of the machine on which the server is running.
Port	Specifies the port number of the registration port on the Server.
Registration credentials (optional)	
User name	Specifies the name of the user on API Gateway that the internal server should connect as.
Password	Specifies the password of the user on API Gateway that the internal server should connect as.
External client security	

Field	Description
Client authentication	<p>Specifies the type of client authentication the internal server performs against external clients. External clients pass their authentication information to API Gateway, which in turn passes it to the internal server.</p> <p>Select one of the following:</p> <ul style="list-style-type: none"> ■ Username/Password . API Gateway does not request client certificates. Instead it looks for user and password information in the request header. ■ Digest. The Internal Server looks for password digest information in the request header. ■ Request Client Certificate. API Gateway requests client certificates for requests from external clients. If the client does not provide a certificate, the server prompts the client for a userid and password. The server checks whether the certificate exactly matches a client certificate on file and is signed by a trusted authority. If so, the client is logged in as the user to which the certificate is mapped in API Gateway. If not, the client request fails, unless central user management is configured. ■ Require Client Certificate. API Gateway requires client certificates for requests from external clients. If the external client does not supply a certificate, the request fails.

- Click **Add**.

The port is created and is listed in the ports table.

Important:

The default access mode of the port is set to *Allow by default*. This implies that the port allows connections to all ESB services and folders. Users must note that this setting allows access to all enterprise assets hosted in internal Integration Server. There is a potential security risk for the IS assets that are secured by Anonymous Access Control Lists (ACL) or if the installation is exposed to the public internet. Hence, you can set the access mode of the port to *Deny by default* before enabling it. When you change the access mode, you add the required services and folders to the *Allow* list. For more information, see [“Configuring Access Mode for a Port” on page 135](#).

- Click the  icon in the **Enabled** column next to the port to enable the port.

The port is enabled and a success message appears.


Configuring the WebSocket Listener

The API Gateway WebSocket listener processes the requests from the clients. This procedure describes how to create the WebSocket listener to API Gateway.

Note:

WebSocket Secure port is not supported.

➤ **To configure the WebSocket listener**

1. Expand the menu options icon , in the title bar, and select **Administration**.

2. Select **Security > Ports**.

The ports page lists all the ports configured with API Gateway, if any.

3. Click **Add Ports**.

4. Select the type of port as **WS** and click **Add**.

5. Provide the following information:

Field	Description
Port	Specify the number you want to use for the port. Select a number that is not already in use on this host machine.
Alias	Specifies an alias for the port. An alias must be between 1 and 255 characters in length and include one or more of the following: letters (a -z, A-Z), numbers (0-9), underscore (_), period (.), and hyphen (-).
Description (optional)	A description of the port.
Bind address (optional)	Specifies the IP address to which to bind this port. Specify a bind address if your machine has multiple IP addresses and you want the port to use this specific address. If you do not specify a bind address, API Gateway selects one for you.
Backlog	Specifies the number of requests that can remain in the queue for an enabled port before API Gateway begins rejecting requests. The default is 200. The maximum value is 65535.
Keep alive timeout	Specifies when to close the connection if the server has not received a request from the client within this timeout value (in milliseconds) or when to close the connection if the client has explicitly placed a close request with the server.


6. Click **Add**.

The port is created and is listed in the ports table.

Important:

The default access mode of the port is set to *Allow by default*. This implies that the port allows connections to all ESB services and folders. Users must note that this setting allows access to all enterprise assets hosted in internal Integration Server. There is a potential security risk for the IS assets that are secured by Anonymous Access Control Lists (ACL) or if the installation is exposed to the public internet. Hence, you can set the access mode of the port to *Deny by default* before enabling it. When you change the access mode, you add the required services and folders to the *Allow* list. For more information, see [“Configuring Access Mode for a Port” on page 135](#).

Also, the global IP access mode will be applied to the newly created WebSocket listener ports. You can modify the IP access mode as per your requirement. For information on modifying IP access mode of ports, see [“Configuring IP Access Mode for a Port” on page 133](#).

7. Click the  icon in the **Enabled** column next to the port to enable the port.


The port is enabled and a success message appears.

Configuring IP Access Mode for a Port

You can configure the access of a port by internal and external IP hosts or apply the global IP access settings.

This section explains how to apply global IP settings for a port.

➤ To apply global IP settings for a port

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Security > Ports**.


The ports page lists all ports configured with API Gateway, if any.
3. Click the **IP Accessmode** button for the port that you want to configure the IP access mode.

The options to configure the port access mode are displayed.
4. Select **Global**. The global IP settings are applied to the selected port.
5. Click **Save**.

The changes are saved.

Allowing Access to All IP Hosts

The following procedure describes how to change the IP access settings for an individual port to Allow by Default and deny some hosts.

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Security > Ports**.

The ports page lists all ports configured with API Gateway, if any.

3. Click the **IP Accessmode** button for the port that you want to configure the IP access mode.

The options to configure the port access mode are displayed.

4. Select **Allow by default**.
5. In the Deny List field, provide the names of hosts for which you want to deny access to the port and click Add.

You can specify the host names (for example, `workstation5.webmethods.com`) or IP addresses (for example, `132.906.19.22` or `2001:db8:85a3:8d3:1319:8a2e:370:7348`) in the above fields. Repeat this step to add the required host names and IP addresses to the list. Also, you can also edit or delete the entered values by clicking the respective action next to the required value.

The host names or IP addresses can include upper and lower case alphabetic characters, digits (0-9), hyphens (-), and periods (.) but cannot include spaces. For IPv6, IP addresses can also include colons (:) and brackets ([]).

You can use the following pattern-matching characters to identify several clients with similar host names or IP addresses.


Char	Description	Example
*	Matches any number of characters	<code>r*.webmethods.com</code>
?	Matches any single character	<code>workstation?.webmethods.com</code>

6. Click **Save**.

The changes are saved.

Denying Access to All IP Hosts

The following procedure describes how to change the IP access settings for an individual port to Deny by Default and allow some hosts.

1. Expand the menu options icon , in the title bar, and select **Administration**.

2. Select **Security > Ports**.

The ports page lists all ports configured with API Gateway, if any.

3. Click the **IP Accessmode** button for the port that you want to configure the IP access mode.

The options to configure the port access mode are displayed.

4. Select **Deny by default**.

5. In the Allow List field, provide the names of hosts for which you want to allow access to the port and click Add.

You can specify the host names (for example, `workstation5.webmethods.com`) or IP addresses (for example, `132.906.19.22` or `2001:db8:85a3:8d3:1319:8a2e:370:7348`) in the above fields. Repeat this step to add the required host names and IP addresses to the list. Also, you can also edit or delete the entered values by clicking the respective action next to the required value.

The host names or IP addresses can include upper and lower case alphabetic characters, digits (0-9), hyphens (-), and periods (.) but cannot include spaces. For IPv6, IP addresses can also include colons (:) and brackets ([]).

You can use the following pattern-matching characters to identify several clients with similar host names or IP addresses.

Char	Description	Example
*	Matches any number of characters	<code>r*.webmethods.com</code>
?	Matches any single character	<code>workstation?.webmethods.com</code>

6. Click **Save**.

The changes are saved.


Configuring Access Mode for a Port

This section describes how to allow or deny access of ports by ESB services and folders. If you allow access by default, you can specify the services for which the access has to be denied; and, if you deny access by default, you can specify the services for which the access has to be allowed.

Important:

When performing the following procedure, do not log into the server through the port you want to change, if you are selecting Deny by default. The procedure involves temporarily denying access to all services through the port. If you log on through the port you want to change and then deny access to all services through it, you will be locked out of the server. Instead, log on through a different existing port or create a new port to log on through.

➤ To configure access mode for a port

1. Expand the menu options icon , in the title bar, and select **Administration**.

2. Select **Security > Ports**.

The ports page lists all the ports configured with API Gateway, if any.

3. Click the **Accessmode** button for the port that you want to configure the access mode.

The options to configure the port access mode are displayed.

4. Select one of the following options:

- **Allow by default.** To allow access of the port, by default.

- **Deny by default.** To deny access of the port, by default.

The port is enabled or denied for access by all ESB services and folders.

5. Optional. Perform one of the following:

- If you have selected **Allow by default**, provide the ESB services or folder for which you want to deny access to the port in the **Add Folders and Services to Deny List** field and click **Add**. Repeat this step to add the required folders and services to the list. You can also edit or delete the entered values by clicking the respective action next to the required value.

- If you have selected **Deny by default**, provide the ESB services or folder for which you want to allow access to the port in the **Add Folders and Services to Allow List** field and click **Add**. Repeat this step to add the required folders and services to the list. You can also edit or delete the entered values by clicking the respective action next to the required value.

6. Click **Save**.

The changes are saved.

API Gateway services to be exposed for API Portal and client communication

If you have configured port access restrictions to allow access only to the APIs hosted on the API Gateway (say with /gateway/, /ws/ , and so on), then ensure that you also provide access to the following APIs in case the APIs are protected by security policies such as OAuth, OpenId or JWT. Allowing access to these endpoints is important for API Portal and API consumers to access API Gateway to retrieve the tokens.

- pub.apigateway.oauth2:getAccessToken
- secure.apigateway.oauth2:approve
- pub.apigateway.oauth2:authorize
- pub.apigateway.oauth2:authorize
- pub.apigateway.openid:getOpenIDToken

- `pub.apigateway.openid:openIDCallback`
- `pub.apigateway.jwt:getJsonWebToken`
- `pub.apigateway.jwt:certs`
- `pub.apigateway.jwt:configuration`
- `pub.apigateway.jwt:thirdPartyConfiguration`

Additionally, the following REST API endpoints are exposed by API Gateway, which are required from the API Portal to access API Gateway. This is to ensure that while you only allow required REST API endpoints, API Portal functionalities continue to work without any impact.

API Portal invokes the following two internal APIs of API Gateway:

- Token request endpoint (`apigateway.accesstokens`)
- JWT request endpoint (`apigateway.jwt:getJsonWebToken`)

Global IP Access Settings For Ports

This section describes how to specify the global IP access setting for ports. The server uses this setting to determine IP access for ports that do not have a custom IP access setting. The default global setting is Allow by Default.

When you create a port, you can customize IP access for it, or you can specify that it use the global IP access setting for the server. If you use the global IP access setting and later change it, the server will use the new global setting for the port. For example, as shipped, the server uses Allow by Default as the global IP access setting (with no hosts explicitly denied). If you create a new port 6666 and do not customize IP access for it, the server uses Allow by Default for port 6666. If you later change the global IP access to Deny by Default, the server will then use Deny by Default for port 6666. If you later customize IP access to port 6666, subsequent changes to the global setting will have no effect on port 6666.

For any given port, you can specify IP access one of two ways:

- **Deny by Default.** Set up the port to deny requests from all hosts except for ones you explicitly allow. Use this approach if you want to deny most hosts and allow a few.
- **Allow by Default.** Set up the port to allow requests from all hosts except for ones you explicitly deny. Use this approach if you want to allow most hosts and deny a few.

You can specify access settings globally (for all ports) or individually (for one port). For more information, refer to respective sections listed in the following table:

Type of access	Section to refer
Configuring Global IP Access	
Deny by Default	“Allowing Connections from Specified Hosts” on page 138

Type of access	Section to refer
Allow by Default	“Denying Connections from Specified Hosts” on page 139
Configuring IP Access for Individual Ports	
Allow by Default	“Allowing Access to All IP Hosts” on page 134
Deny by Default	“Denying Access to All IP Hosts” on page 134

To customize IP access for individual ports, see [“Allowing Access to All IP Hosts” on page 134](#) and [“Denying Access to All IP Hosts” on page 134](#).


Allowing Connections from Specified Hosts

The following procedure describes how to change the global IP access setting to Deny by Default and specify some hosts to allow.

Important:

If you inadvertently lock all hosts out of the server, you can correct the problem by following the steps given in the [“If You Inadvertently Deny IP Access to All Hosts” on page 140](#) section.

➤ To allow connection from specified IP hosts

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Security > Ports**.

The ports page lists all the ports configured with API Gateway, if any.

3. Click **Configure Global IP Access Settings**.

The options to configure IP access settings at global level are displayed.

4. Select **Deny by default**.

Note:

If you select *Deny by default*, ensure that you configure the IP that must be allowed to access ports. Otherwise, ports cannot be accessed by external hosts.

5. In the Allow List field, provide the names of hosts for which you want to allow access to the port and click Add.

You can specify the host names (for example, `workstation5.webmethods.com`) or IP addresses (for example, `132.906.19.22` or `2001:db8:85a3:8d3:1319:8a2e:370:7348`) in the above fields. Repeat this step to add the required host names and IP addresses to the list.

The host names or IP addresses can include upper and lower case alphabetic characters, digits (0-9), hyphens (-), and periods (.) but cannot include spaces. For IPv6, IP addresses can also include colons (:) and brackets ([]).

Note:

IP addresses are harder to spoof, and therefore more secure.

You can use the following pattern-matching characters to identify several clients with similar host names or IP addresses.

Char	Description	Example
*	Matches any number of characters	r*.webmethods.com
?	Matches any single character	workstation?.webmethods.com


6. Click **Save**.

The changes are saved.

Denying Connections from Specified Hosts

The following procedure describes how to change the global IP access setting to Allow by Default and specify some hosts to deny.

> To deny connection from specified IP hosts

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Security > Ports**.

The ports page lists all the ports configured with API Gateway, if any.

3. Click **Configure Global IP Access Settings**.

The options to configure IP access settings at global level are displayed.

4. Select **Allow by default**.
5. In the Deny List field, provide the names of hosts for which you want to allow access to the port and click Add.

You can specify the host names (for example, workstation5.webmethods.com) or IP addresses (for example, 132.906.19.22 or 2001:db8:85a3:8d3:1319:8a2e:370:7348) in the above fields. Repeat this step to add the required host names and IP addresses to the list.

The host names or IP addresses can include upper and lower case alphabetic characters, digits (0-9), hyphens (-), and periods (.) but cannot include spaces. For IPv6, IP addresses can also include colons (:) and brackets ([]).

You can use the following pattern-matching characters to identify several clients with similar host names or IP addresses.

Char	Description	Example
*	Matches any number of characters	r*.webmethods.com
?	Matches any single character	workstation?.webmethods.com

6. Click **Save**.

The changes are saved.

If You Inadvertently Deny IP Access to All Hosts

If you select *Deny by Default* in **Configure Global IP Access Settings**, all hosts are restricted from accessing the server ports. Hence, API Gateway cannot communicate with Integration Server.

For example, if you have defined five ports and those ports are configured to use the global IP access settings. In such case, if you change the global IP access setting to **Deny By Default**, API Gateway cannot communicate with hosts through any ports.

This section explains the steps to reset your setting and allow API Gateway to communicate with Integration Server.

➤ To modify the global IP access settings

1. Log off from API Gateway.
2. Open the diagnostic port of the Integration Server by entering `http://hostname:diagnostic port` in your browser's address bar.

For example, `10.2.100.112:9999`. By default, the diagnostic port is `9999`.

3. Navigate to the **Security > Ports**.

The Port List appears.

4. Perform any of the following:
 - Click **Change Global IP Access Restrictions** and make required changes. For more information, see [“Allowing Connections from Specified Hosts”](#) on page 138.
 - Click **Allow** in the **IP Access** column of the required port.
5. Restart Integration Server.
6. Refresh your browser and log on to API Gateway and modify global IP access settings.

Configuring Restriction to IP Address based on Authentication

You must have the *Manage Security Configuration* functional privilege to configure this restriction.

You can configure the restriction to client IP address based on authentication failure in API Gateway to prevent malicious attack that occurs when a client floods a server with many requests in an attempt to interfere with server processing. This restriction prevents the malicious attack by blocking or denying the unauthenticated client from accessing the APIs, when API Gateway fails to authenticate the client. Using API Gateway, you can limit the number of times a client fails to authenticate the API in a specified time interval. The reason for authentication failure can be either of the following:

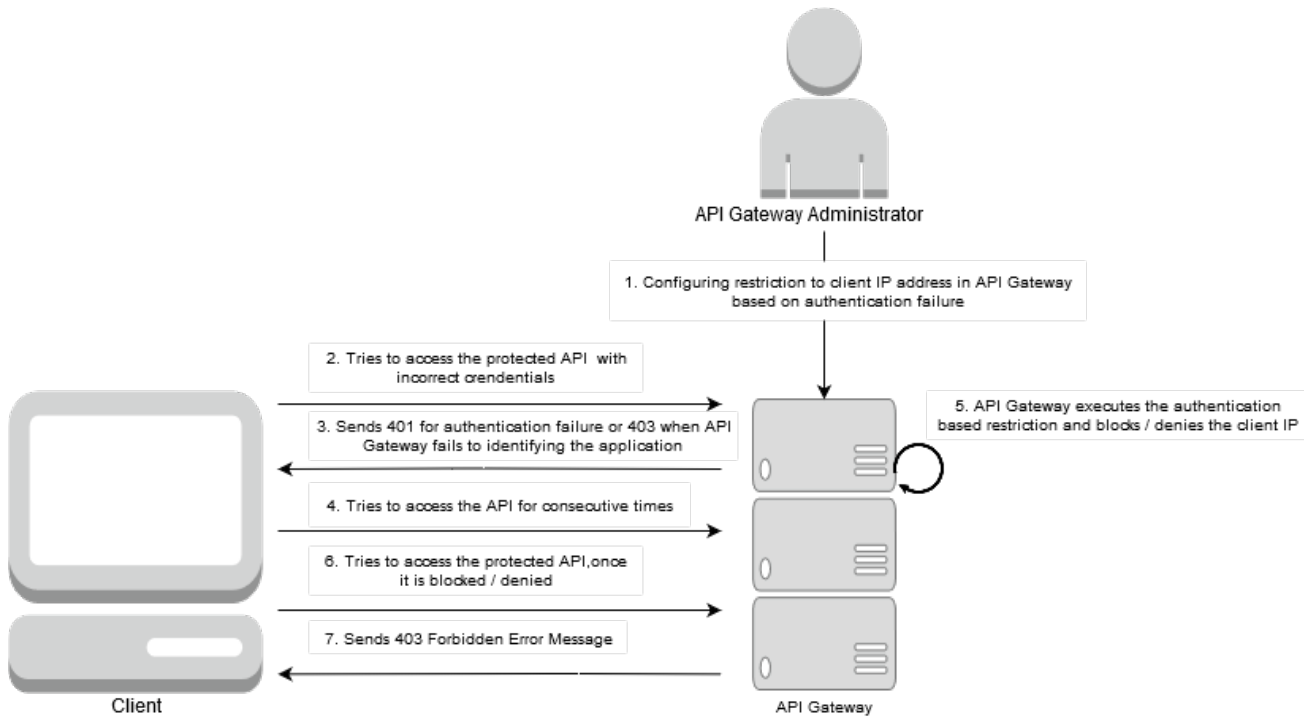
- when API Gateway fails to authenticate the client (**or**)
- when API Gateway fails to identify the client and its respective application.

When the above mentioned authentication failure occurs, API Gateway sends the *401* or *403* error message to the client.


When API Gateway detects that the failed authentication limit has been exceeded, it blocks or denies that particular client IP address from accessing any of the APIs and sends the *403 Forbidden error* to the client.


Note:


- If an API is enforced with Identify and Access Application policy, and if the invocation fails due to non-preemptive authentication failure. API Gateway does not take such non-preemptive authentication failure into count and increase the failed authentication count.
- When you use Load Balancer for configuring high availability between the API Gateway instances, API Gateway honours the X-Forwarded-For (XFF) header from the client. As the XFF header has the actual client IP address, API Gateway can block or deny the problematic client from accessing the protected API based on your configuration.



> **To configure restriction to IP address based on authentication**

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Security > Global IP Access Settings**.
3. Click **Authentication based restrictions - Block/Deny by IP address** section and provide the following information.

Field	Description
Enable	<p>Specifies whether restriction to IP address based on authentication is enabled.</p> <p>Click the toggle button to change the state to  to enable IP address restriction.</p> <p>By default this option is disabled.</p>
Maximum failed authentication	Specifies the maximum number of failed authentication that API Gateway can accept from a specific IP address in a given time interval.
In (seconds)	Specifies the time interval, in seconds, in which maximum authentication failure can be permitted.

Field	Description
Action when limit exceeds	<p>Specifies the action to be performed when the number of failed authentication from an IP address exceeds the specified limits.</p> <p>Select one of the following:</p> <ul style="list-style-type: none"> ■ Add IP address to deny list - Permanently denies the IP address from accessing any APIs. ■ Block the IP address - Temporarily blocks the IP address from accessing any APIs for specified time interval. <ul style="list-style-type: none"> ■ In (seconds). Specify the time interval for which you want to block the IP address.
Denied IP list	<p>Specifies the list of IP addresses that are denied from access.</p> <p>Click  in the Action column to remove an IP address from the denied list.</p>


4. Click **Save**.

The configuration is saved.

SAML Issuer

If a native API is enforced with the SAML policy, API Gateway uses this configuration to communicate to STS (Security Token Service) to retrieve the SAML token.

> To add a SAML issuer

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Security > SAML issuer**.

The SAML issuer page lists all the issuers configured along with the Endpoint URI corresponding to each SAML issuer, if any.


3. Click **Add SAML issuer**.
4. In the Add SAML issuer section, provide the following information:

Field	Description
Name	Name of a SAML token issuer used by API Gateway.

Field	Description
	This value must match the value of the Issuer field in the SAML assertion.
Normal client	Selecting this sets the client that requests the SAML token.
Act as delegation	Selecting this delegates the SAML request to another user (delegator). The delegator uses a signature element to authenticate the SAML request.
Issuer policy	Specifies the name of an issuer policy to be used to communicate with SAML issuer. <ul style="list-style-type: none"> ■ If a value is specified for the Issuer policy field, then the selected issuer policy is applied to all APIs that are using the SAML authentication. ■ If a value is NOT specified for this field, then a default issuer policy based on the WSS Username or Kerberos communication mode is applied to all APIs.
Communicate using. Specifies the mode of communication.	
WSS Username	Specifies that WSS Username mode is used to obtain the SAML assertion to access the API. The WSS username token supplied in the header of the SOAP request that the consumer application submits to the API.
Kerberos	Specifies that Kerberos mode is used to obtain the SAML token and assertion to access the API. Transports the Kerberos token over the Transport Layer Security (TLS) protocol to provide additional security features.
Authenticate using. Specify the type of authentication you want to use while communicating with the SAML issuer.	
For the Authentication type WSS Username , authenticate using the following:	
Custom credentials	Specifies the values provided in the policy required to communicate the SAML issuer. Provide the following information: <ul style="list-style-type: none"> ■ Username. Specify a username. ■ Password. Specify a password. ■ Domain. Specify a domain.
For the Authentication type Kerberos , authenticate using any of the following:	

Field	Description
Custom credentials	<p>Specifies the values provided in the policy required to communicate the SAML issuer.</p> <p>Provide the following information:</p> <ul style="list-style-type: none"> ■ Client principal. A valid client LDAP user name. ■ Client password. A valid password of the client LDAP user. ■ Service principal. A valid Service Principal Name (SPN). The specified value is used by the client to obtain a service ticket from the KDC server. ■ Service principal nameform. Specifies the format in which you want to specify the principal name of the service that is registered with the principal database. Select one of the following: <ul style="list-style-type: none"> ■ Username. Represents the principal name as a named user defined in LDAP used for authentication to the KDC. ■ Hostbased. Represents the principal name using the service name and the host name, where host name is the host computer.
Delegate incoming credentials	<p>Specifies the values provided in the policy required by the API providers to select whether to delegate the incoming Kerberos token or act as a normal client.</p> <p>Provide the following information:</p> <ul style="list-style-type: none"> ■ Client principal. A valid client LDAP user name. ■ Client password. A valid password of the client LDAP user. ■ Service principal. A valid Service Principal Name (SPN). The specified value is used by the client to obtain a service ticket from the KDC server. ■ Service principal nameform. Specifies the format in which you want to specify the principal name of the service that is registered with the principal database. Select one of the following: <ul style="list-style-type: none"> ■ Username. Represents the principal name as a named user defined in LDAP used for authentication to the KDC. ■ Hostbased. Represents the principal name using the service name and the host name, where host name is the host computer.

Field	Description
Incoming HTTP basic auth credentials	<p>Specifies the incoming HTTP basic authentication credentials in the transport header of the incoming request for client principal and client password.</p> <p>Provide the following information:</p> <ul style="list-style-type: none"> ■ Service principal. A valid Service Principal Name (SPN). The specified value is used by the client to obtain a service ticket from the KDC server. ■ Service principal nameform. Specifies the format in which you want to specify the principal name of the service that is registered with the principal database. Available values are: <ul style="list-style-type: none"> ■ Username. Represents the principal name as a named user defined in LDAP used for authentication to the KDC. ■ Hostbased. Represents the principal name using the service name and the host name, where host name is the host computer.
Endpoint URI	Provide the endpoint URI of the STS.
SAML version	<p>Specify the SAML version to be used for authentication.</p> <p>Available values are: SAML 1.1, SAML 2.0</p>
WS-Trust version	<p>Specify the WS-Trust version that API Gateway must use to send the RST to the SAML issuer.</p> <p>Available values are: WS-Trust 1.0, WS-Trust 1.3</p>
Applies to	<p>Specify the scope for which this security token is required.</p> <p>For example, the APIs to which this token is applied.</p>
Signing configurations	
Keystore alias	<p>Specify the keystore to be used by API Gateway while sending the request to the STS.</p> <p>A keystore is a repository of private keys and corresponding public certificates.</p>
Key alias (signing)	Specify the key alias, a private key used to sign the request sent to STS.
Encryption configurations	
Truststore alias	Select the truststore that should be used by API Gateway while sending the STS request.

Field	Description
	Truststore is a repository that holds all the trusted public certificates.
Certificate alias (Encryption)	Select the certificate from the truststore used to encrypt the request that is sent to the STS.
Request security token template parameters.	Defines extensions to the <wst:RequestSecurityToken> element for requesting specific types of keys, algorithms, or key and algorithms, as specified by a given policy in the return token(s).
Key	Specifies the key type of the security token template.
Value	Specifies a value for the request token.
	You can add multiple key and values by clicking  .

5. Click **Add**.

This adds the SAML issuer and it is listed in the SAML issuers list.

Custom Assertions

API Gateway uses WS-Security (WSS) to provide message-level security and protection for SOAP message requests from a client to an API, and SOAP message responses from an API to a client. By default, API Gateway supports the WSS policies like Username, X.509 certificate, Security Assertion Markup Language (SAML), Kerberos, Encryption, and so on, for the request or response SOAP messages, or both.

API Gateway also provides an extension to define and use custom policy assertions. Custom assertions allow the API providers to extend and provide additional security policies that are not available by default in API Gateway.

In WS-Security, custom assertions are used for expressing individual security requirements, constraints, or both. The individual policy assertions can be combined to create security policies that ensure secure and reliable exchanges of SOAP messages between a client and a SOAP API.

API Gateway supports the following assertion types for enforcing a custom security policy:

Binding Assertions

These assertions specify the security mechanism that is to be used by the client or API such as the keys being used, algorithms, and so on. Common properties used by other assertions are also defined in the security binding assertion.

API Gateway supports the following WS-SecurityPolicy binding assertions:

Binding Assertion	Description
Transport Binding	<p>This assertion is used when the message is protected at the transport level. In this binding, messages are exchanged only through a defined medium, for example, HTTPS.</p> <p>Note: By default, API Gateway uses the transport binding for Kerberos authentication.</p>
Asymmetric Binding	<p>This assertion is used when both the initiator and the recipient possess security tokens. In this binding, initiator uses its private key to sign and the recipient's public key to encrypt. Recipient uses its private key to decrypt and initiator's public key to verify the signature.</p> <p>Note: By default, API Gateway uses the asymmetric binding for the security policies.</p>
Symmetric Binding	<p>This assertion is used when only the initiator or recipient has a security token. In this binding, both the signing and encrypting of messages is done using a single security token.</p>

Token Assertions

These assertions specify the types of tokens to be used to authenticate and secure SOAP messages.

API Gateway supports the following WS-SecurityPolicy token assertions:

Token Assertion	Description
Username Token	<p>When using this assertion, the message-level security is implemented using a WSS username token. The assertion authenticates a client using the username and password in the SOAP request. If validation of the username token succeeds, then API Gateway passes the message to the API. If validation fails, then API Gateway returns a SOAP fault.</p>
X509 Token	<p>When using this assertion, the message-level security is implemented using an X.509v3 certificate. The assertion authenticates a client using the X.509v3 certificate in the SOAP request. If validation of the X.509v3 certificate succeeds, then API Gateway passes the message to the API. If validation fails, then API Gateway returns a SOAP fault.</p>
Kerberos Token	<p>When using this assertion, the message-level security is implemented using a Kerberos token. The assertion authenticates a client using the Kerberos token in the SOAP request. If validation of the Kerberos token succeeds, then API Gateway passes the message to the API. If validation fails, then API Gateway returns a SOAP fault.</p>

Token Assertion	Description
SAML Token	<p>When using this assertion, the message-level security is implemented using a SAML (Security Assertions Markup Language) token. SAML is a standard data format for exchanging authentication and authorization data between the client and the SOAP API. If validation of the SAML token succeeds, then API Gateway passes the message to the API. If validation fails, then API Gateway returns a SOAP fault.</p>
	<p>Note: API Gateway supports both the SAML 1.1 and 2.0 standards.</p>

Policy Assertions

API Gateway allows you to even define a complete custom policy assertion. For example, a policy assertion might specify a symmetric binding and the security token types that are used to digitally sign or encrypt SOAP messages between the client and API.

Creating a Custom Assertion

Pre-requisites:

You must have the API Gateway's manage security configurations functional privilege assigned to add a custom assertion.


You might want to create a custom assertion when you want to:

- Enforce symmetric binding with an authentication mechanism that is not available by default in API Gateway.
- Support signing and encryption at the desired level.
- Modify the predefined encryption algorithm and security layout properties.
- Enforce custom authentication tokens that are not available by default in API Gateway.

Important:

When creating a custom assertion, make sure that both the syntax and the semantics of the assertion element are valid and in compliance with the Web Services Security Policy specification.

➤ To create a custom assertion

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Security > Custom assertions**.

API Gateway displays a list of all the currently defined policy assertions.

3. Click **Add assertion**.

4. Select the assertion type. The available options are:

- **Binding**
- **Token**
- **Policy**

5. Provide the following information:

Field	Description
Name	<p>Name of the custom assertion.</p> <p>For a binding or token assertion type, this is the display name of the assertion in the Binding Assertion field or Custom Token Assertion of the Inbound Authentication - Message policy.</p> <p>For a policy assertion type, this is the display name of the assertion in the Issuer Policy field of the Add SAML Issuer configuration page.</p>
Select file	<p>Click Browse and select the policy assertion file to be uploaded.</p> <p>The Assertion element text box displays the data from the assertion file.</p> <p>If you have uploaded the policy assertion file and want to replace it, click the Delete icon.</p>
Assertion element	<p>If you have not uploaded the policy assertion file, provide the XML representation of assertion.</p>

6. Click **Add**.

The custom assertion is added. You can create as many custom assertions you require.

Post-requisites:

To enforce the custom binding or token assertion in an API, select the assertion in the appropriate fields of the **Inbound Authentication - Message** policy:

- **Binding Assertion**
- **Custom Token Assertion**


To enforce the custom policy assertion in an API, select the assertion and the corresponding SAML issuer in the appropriate fields:

- **Issuer Policy** field of the **Add SAML Issuer** configuration page.
- **Authentication scheme** field of the **Outbound Authentication - Message** policy.

Viewing Custom Assertion List and Assertion Configuration

You can view the list of configured custom assertions in API Gateway. In addition, you can view and modify the configuration in the individual custom assertion details page.

➤ To view a list of custom assertions and assertion configuration

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Security > Custom assertions**.

A list of all available custom assertions appears.

You can delete a custom assertion by clicking its **Delete** icon.

3. Click any custom assertion to view the configuration details.

The custom assertion details page displays the XML element.


Modifying Custom Assertion

Pre-requisites:

You must have the API Gateway's manage security configurations functional privilege assigned to modify a policy assertion.

You might want to modify a custom assertion to change the currently defined security settings, such as, authentication scheme, signing and encryption, algorithms and supporting tokens, of SOAP messages.

➤ To modify a custom assertion

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Security > Custom assertions**.

A list of all available custom assertions appears.

3. Select the required custom assertion that you want to modify.
4. Modify the fields as required.
5. Click **Save**.

The custom assertion is updated.

Post-requisites:

When you are ready to put the policy assertion into effect in an API, select it in the appropriate field of **Inbound Authentication - Message** policy.



Deleting Custom Assertion

Pre-requisites:

You must have the API Gateway's manage security configurations functional privilege assigned to delete a policy assertion.

You delete a policy assertion to remove it from API Gateway permanently.

> To delete a policy assertion

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Security > Custom assertions**.
A list of all available custom assertions appears.
3. Click  in the action column of the custom assertion to be deleted.
4. Click **Yes** in the confirmation dialog.

The custom assertion is deleted from API Gateway.

Example: Custom Assertions

API Gateway, by default, uses the asymmetric binding assertion with X.509v3 token for implementing SOAP message protection. If you would like to enforce any authentication (other than the predefined authentications shipped with API Gateway), include additional WSS custom assertions, sign and encrypt SOAP messages, and define custom properties, such as the algorithms and layout of security header, you can create custom assertions that would construct the custom policy file to suit your specific security requirements.

Following is a policy file that API Gateway generates when a WSS username token is enforced by the Inbound Authentication Message policy for an API.

```
<wsp:Policy wsu:Id="9dbda2fb-9cef-4ff9-bc70-115c942a3b76"
  xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/
    oasis-200401-wss-wssecurity-utility-1.0.xsd">
  <wsp:ExactlyOne>
    <wsp>All>
      (L01) <sp:AsymmetricBinding xmlns:sp=
        "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
        xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
        <wsp:Policy>
          <sp:InitiatorToken>
            <wsp:Policy>
              <sp:X509Token sp:IncludeToken
```



```

        "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702
        /IncludeToken/Never">
    <wsp:Policy>
        <sp:WssX509V3Token10 />
    </wsp:Policy>
</sp:X509Token>
</wsp:Policy>
</sp:InitiatorToken>
<sp:RecipientToken>
    <wsp:Policy>
        <sp:X509Token sp:IncludeToken=
            "http://docs.oasis-open.org/ws-sx/ws-securitypolicy
            /200702/IncludeToken/Never">

            <wsp:Policy>
                <sp:WssX509V3Token10 />
            </wsp:Policy>
        </sp:X509Token>
    </wsp:Policy>
</sp:RecipientToken>
<sp:AlgorithmSuite>
    <wsp:Policy>
        <sp:TripleDesRsa15 />
    </wsp:Policy>
</sp:AlgorithmSuite>
<sp:Layout>
    <wsp:Policy>
        <sp:Strict />
    </wsp:Policy>
</sp:Layout>
<sp:ProtectTokens/>
</wsp:Policy>
</sp:AsymmetricBinding>
<sp:SupportingTokens xmlns:sp=
    "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
    <wsp:Policy>
        <sp:UsernameToken sp:IncludeToken=
            "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/
            IncludeToken/AlwaysToRecipient"/>

    </wsp:Policy>
</sp:SupportingTokens>
</wsp:All>
</wsp:ExactlyOne>
</wsp:Policy>

```

You might have a requirement to change the policy assertion that is available by default in API Gateway. For example, you might want to generate the above security policy using a symmetric binding instead of the default asymmetric binding, and modify the username token that is defined by default as a supporting token to a signed supporting token. You could then create custom policy assertions to achieve these specific requirements.

Important:

When adding a custom policy assertion, make sure that both the syntax and the semantics of the assertion are valid and in compliance with the Web Services Security Policy specification.

Symmetric Binding Assertion

You might want to use a symmetric binding (instead of the default asymmetric binding) when only API Gateway possess the X.509v3 token for authentication. You might also want to sign and

encrypt the SOAP messages, modify the encryption algorithm, and include timestamp on the SOAP messages. You would then create a custom binding assertion with the specific property lines:

```
<sp:SymmetricBinding
  xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
  xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
  <wsp:Policy>
    <sp:ProtectionToken>
      <wsp:Policy>
        <sp:X509Token sp:IncludeToken=
          "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/
          IncludeToken/AlwaysToRecipient">
          <wsp:Policy>
            <sp:WssX509V3Token10/>
            <sp:WssX509PkiPathV1Token10/>
          </wsp:Policy>
        </sp:X509Token>
      </wsp:Policy>
    </sp:ProtectionToken>
    <sp:AlgorithmSuite>
      <wsp:Policy>
        <sp:TripleDesRsa15/>
      </wsp:Policy>
    </sp:AlgorithmSuite>
    <sp:Layout>
      <wsp:Policy>
        <sp:Strict/>
      </wsp:Policy>
    </sp:Layout>
    <sp:IncludeTimestamp/>
    <sp:ProtectTokens/>
    <sp:OnlySignEntireHeadersAndBody/>
    <sp:SignBeforeEncrypting/>
  </wsp:Policy>
</sp:SymmetricBinding>
```

You could create custom assertions to include one or more of the following security requirements:

Supporting Token Assertions

You might want to sign the supporting token for example, WSS username token, and use `SignedSupportingTokens` assertion. You might also want to specify that the signed username token must always be included in the messages sent to the recipient. You would then create a custom token assertion with the specific property lines:

```
<sp:SignedSupportingTokens xmlns:sp=
  "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
  <wsp:Policy>
    <sp:UsernameToken sp:IncludeToken=
      "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/
      IncludeToken/AlwaysToRecipient"/>
  </wsp:Policy>
</sp:SignedSupportingTokens>
```

WSS Token Assertions

You might want to include WSS10 and WSS11 assertions to provide additional SOAP message security. You would then create two separate custom token assertions with the specific property lines:

Wss10 assertion:

```
<sp:Wss10
  xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"

  xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
  <wsp:Policy>
    <sp:MustSupportRefIssuerSerial/>
  </wsp:Policy>
</sp:Wss10>
```

Wss11 assertion:

```
<sp:Wss11
  xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
  xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
  <wsp:Policy>
    <sp:MustSupportRefIssuerSerial/>
    <sp:MustSupportRefThumbprint/>
    <sp:RequireSignatureConfirmation/>
  </wsp:Policy>
</sp:Wss11>
```

After you have defined these custom assertions in API Gateway, execution of a policy that is configured with all of these custom assertions in the Inbound Authentication - Message policy, would construct the custom security policy file as follows:

```
<wsp:Policy wsu:Id="1e747a18-b55d-4e99-ac67-80a8eafd76b3"
  xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/
    oasis-200401-wss-wssecurity-utility-1.0.xsd">
  <wsp:ExactlyOne>
    <wsp:All>
      <sp:SymmetricBinding xmlns:sp=
        "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
        <wsp:Policy>
          <sp:ProtectionToken>
            <wsp:Policy>
              <sp:X509Token sp:IncludeToken=
                "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/
                  IncludeToken/AlwaysToRecipient">
                <wsp:Policy>
                  <sp:WssX509PkiPathV1Token10/>
                </wsp:Policy>
              </sp:X509Token>
            </wsp:Policy>
          </sp:ProtectionToken>
          <sp:AlgorithmSuite>
            <wsp:Policy>
              <sp:TripleDesRsa15/>
            </wsp:Policy>
          </sp:AlgorithmSuite>
        </sp:Layout>
      </wsp:Policy>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

```

    </wsp:Policy>
  </sp:Layout>
  <sp:OnlySignEntireHeadersAndBody/>
</wsp:Policy>
</sp:SymmetricBinding>
<sp:SignedSupportingTokens xmlns:sp=
  "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
  <wsp:Policy>
    <sp:UsernameToken sp:IncludeToken=
      "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/
      IncludeToken/AlwaysToRecipient"/>
  </wsp:Policy>
</sp:SignedSupportingTokens>
<sp:Wss11 xmlns:sp=
  "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
  <wsp:Policy>
    <sp:MustSupportRefIssuerSerial/>
    <sp:MustSupportRefThumbprint/>
    <sp:RequireSignatureConfirmation/>
  </wsp:Policy>
</sp:Wss11>
<sp:Wss10 xmlns:sp=
  "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
  <wsp:Policy>
    <sp:MustSupportRefIssuerSerial/>
  </wsp:Policy>
</sp:Wss10>
<sp:EncryptedParts xmlns:sp
  "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
  <sp:Body/>
</sp:EncryptedParts>
</wsp:All>
</wsp:ExactlyOne>
</wsp:Policy>

```

Kerberos Settings

Kerberos is an authentication protocol that uses symmetric encryption and a trusted third party system to validate the identity of clients. The Kerberos protocol provides authentication over open and insecure networks in which communication between the hosts can be intercepted.

You can use API Gateway to configure Kerberos authentication for API requests. API Gateway provides support for using Kerberos authentication for inbound and outbound HTTP and HTTPS requests at the transport and the message level.

Kerberos authentication system consists of the a Kerberos client that needs to access and use Kerberos services, a trusted third-party system, specifically a Key Distribution Center (KDC) and a server that hosts APIs that are accessible using Kerberos authentication.

Note:


You can configure the kerberos settings through API Gateway and Integration Server UI. But, Software AG recommends to use API Gateway UI to configure or modify introspection endpoint.

Configuring API Gateway to Use Kerberos

Before you configure API Gateway to use Kerberos authentication, ensure that:

- A working Key Distribution Center (KDC) is set up.
- The KDC is configured as an LDAP directory, for authenticating incoming requests with Kerberos tickets.
- The Kerberos client is registered with the principal database of the KDC.
- The API that you want to access is registered with the KDC.
- A valid Kerberos configuration file is available.

➤ To configure API Gateway to use Kerberos

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Security > Kerberos**.
3. Click **Edit**.
4. Provide or modify the following information as required:

Field	Description
Realm	Optional. The domain name of the Kerberos server, in uppercase letters. Note: A value specified for Realm overwrites the realm set in the KDC configuration file specified in Kerberos configuration file .
Key distribution center	Optional. The host name of the machine on which the KDC resides. A value specified for Key distribution center overwrites the default key distribution center set in the KDC configuration file specified in Configuration file .
Configuration file	The location of the Kerberos configuration file that contains the Kerberos configuration information, including the locations of KDCs, defaults for the realm and for Kerberos applications, and the host names and Kerberos realms mappings.
Use subject credentials	Specifies whether API Gateway requires a Kerberos V5 Generic Security Services (GSS) mechanism to obtain the necessary credentials from an existing subject set up by the JAAS authentication module. Here, subject represents the user or service being authenticated in the JAAS login context.

5. Click **OK**.

Master Password Management

In the normal course of its operations API Gateway may connect to native APIs, applications, databases, and other systems such as, API Portal, CentraSite, or external entities such as Email servers and databases. API Gateway is required to provide a password to each of these systems before connecting to them. API Gateway uses this password to identify itself or authenticate to the other systems.

When you configure API Gateway to connect to an application or subsystem, for example a database, you specify the password that API Gateway must send to the database server in order to connect to it. Later, when an API Gateway user makes a request that requires the database, API Gateway sends the configured password to the database server and connects to it. In API Gateway, you would be using passwords while enforcing security related policies, while connecting to various destinations such as, API Portal, CentraSite, Email, and SNMP, while configuring the security-related aliases, configuring outbound proxy servers, and so on.

To protect these passwords API Gateway encrypts them. By default, it encrypts them using Password-Based Encryption (PBE) standard, also known as PKCS5. This encryption method requires the use of an encryption key or master password that you specify. The encrypted passwords are stored in a file. The master password is also encrypted, and by default, is stored in a file. For greater security, you can change the master password in API Gateway at regular intervals or you can configure API Gateway to prompt for the master password at server startup instead.

Points to remember regarding master password:

- When the master password is updated in one node, it is not synchronized across other nodes in the cluster. The master password has to be updated manually in all the nodes.
- During export or import of assets, ensure that the master password is identical across stages and on different instances of API Gateway.

Backing up the Password and Master Password Files

You should regularly back up the files API Gateway uses to maintain the passwords and the master password. In API Gateway instance's home directory (`APIGateway_directory\instances\instance_name`), these files are:

- `config/txnPassStore.dat`: Stores encrypted passwords.
- `config/empw.dat`: Stores encrypted master password.
- `config/configPassman.cnf`: Specifies password configuration settings.
- `config/passman.cnf`: Non-editable version of `configPassman.cnf`.

Always back up and restore these files together. If you change the name or location of the password store or the master password store, make sure your backup procedure backs up the correct files.


Updating the Master Password

When you first install API Gateway, the master password is *manage*. For security purposes, you should change the master password immediately after installation and again on a regular basis. You should also change it when there are personnel changes.

The default expiry interval for a master password is 90 days. Once the master password expires, the status of the password changes to Inactive. As the expiration date nears, the API Gateway displays the password expiration status on the API Gateway and sends warning messages to the console stating that it is time to change the master password. If API Gateway is configured for e-mail notification, API Gateway also sends e-mail messages with this information to the configured addresses.

You must have the Manage security configurations functional privilege assigned to update the master password.

➤ To update the master password

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Master password**.
3. Click **Update master password**.
4. Type the current password in the **Current password** field.
5. Type the new password in the **New password** field.
6. Re-type the new password to confirm the new password in the **Confirm new password** field.
7. Click **Update**.

This updates the master password.


Managing Master Password Expiry Interval

When you first install API Gateway, it is configured to use PBE to encrypt passwords, and has a master password of *manage* with an expiry interval of 90 days. You can see the current expiry date by looking at the Administration > Security > Master password screen.

The expiry interval is the time between password changes. If you do not change the master password by the expiry date, API Gateway continues to operate using the existing password indefinitely. If you specify an interval value 0, the password does not expire and no warnings are sent to API Gateway or the server log.

You must have the Manage security configurations functional privilege assigned to update the master password expiry interval.

➤ To update the master password expiry interval

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Master password**.
3. Provide a value for the expiry interval.
4. Click **Update**.

This updates the expiry interval.

Advanced Configuration to Manage Master Password

The configPassman.cnf file contains additional configuration settings for password encryption. The file consists of a number of properties, some of which are commented out in the default configuration.

Note:

The configPassman.cnf file has a companion file, passman.cnf. If you make changes to configPassman.cnf file, API Gateway automatically updates passman.cnf to reflect these changes when you initialize API Gateway. Never update passman.cnf directly.

As shipped, the configPassman.cnf file specifies that passwords will be stored in file config/txnPassStore.dat and encrypted using Password-Based Encryption (PBE). In addition, it specifies that the master password will be stored in file config/empw.dat. Properties that can be used to specify other settings are commented out.

If you want to change these optional settings, you must edit the configPassman.cnf file. The file must always specify the following:

- Encryption method for passwords.
- Location of the file that contains the passwords.
- Method API Gateway uses to obtain the master password.

The following sections describe the configPassman.cnf file in detail and how to change password and master password settings.

Working with Password Settings

This section describes how to use the configPassman.cnf file to change settings for passwords.

Controlling Name and Location of Password File

The default file name and location for the password file is in the server instance's home directory under config/txnPassStore.dat. To change it, locate and modify the following property:

```
outbound.password.field.fileName=config/txnPassStore.dat
```


This property must always be present and uncommented. If you want to change the file name or location, change the right hand side only. You can specify an absolute or relative path. In the path name, use the forward slash (/) only; the backward slash (\) is not supported.

Controlling Encryption of Outbound Password File

The default encryption method for the password file is Password-Based Encryption (PBE). To change it, locate the following properties and uncomment a different method. One and only one of these properties must always be uncommented.

```
default.encrypted=EntrustPbePlus ---- This denotes PBE encryption -- most secure
```

```
#default.encrypted=Base64 ---- This denotes Base64 encoding -- not secure
```

```
#default.encrypted=None ---- This denotes Clear text -- not secure
```

Working with Master Password Settings

By default, the master password is stored in the file `config/empw.dat` under the server instance's home directory, but if you prefer, you can configure API Gateway to prompt for the master password at server initialization. The following sections describe how to tell API Gateway which method to use.

Storing the Master Password in a File

To store the master password in a file, use the following properties:

```
master.password.storeInFile=true
```

This controls whether API Gateway stores the masterpassword in a file (true) or prompts for it at server initialization (false). If this value is set to true, make sure the `master.password.field.attemptsLimit` properties are commented out.

```
master.password.field.fileName=config/empw.dat
```

This indicates the location of the master password store. Use the forward slash (/) only; the backward slash (\) is not supported.

```
master.password.field.repeatLimit=3
```

This indicates the number of password changes required before you can reuse a password.

Prompting for the Master Password at Server Initialization

To prompt for the master password at server initialization, use the following properties. Use these properties only if you want API Gateway to prompt for the password at server initialization (that is, you specified false for `master.password.storeInFile`). If you do not want API Gateway to prompt for the password at server initialization, make sure these two properties are commented out.

```
#master.password.field.useGUI=true
```

Specify true to prompt for the password in a pop-up window. If you select this method, you can start the server from the Windows start menu. This is default if `master.password.storeInFile` is set to false.

```
#master.password.field.attemptsLimit=3
```

This indicates the number of unsuccessful login attempts permitted before API Gateway rejects the request.

You cannot configure API Gateway to prompt for the master password at server initialization if:

- API Gateway runs as a Windows service.
- API Gateway runs as a background application on UNIX.

Restoring the Password and Master Password Files

If your API Gateway is configured to encrypt passwords using Password- Based Encryption (PBE), your API Gateway will have a master password, which is the key used to encrypt the other passwords. You have to provide the master password whenever you want to change to a new encryption key. In addition, some installations are configured so that API Gateway prompts for the master password when API Gateway initializes; without the password, API Gateway starts up in safe mode. Therefore, if you lose or forget your master password, you have to restore it or reset it, depending on the circumstances.

You can restore passwords if either of the following is true:

- Your master password and other passwords are stored in files and you have recent backups of both and the `passman.cnf` file.
- API Gateway is configured to prompt for the master password, you have a recent backup of the password file and the `passman.cnf` file, and you know the master password for that backup.

➤ To restore the master password and other password files

1. Determine which files you need to restore.

If your master password is not stored in a file, that is, your API Gateway prompts you for a master password at server startup, then you can restore just the password file and the `passman.cnf` file. Otherwise, you must restore the master password file, the password file, and the `passman.cnf` file from backups.

2. Determine the name and location of the files.

The `passman.cnf` file is always `config/passman.cnf` located under the server instance's home directory (`APIGateway_directory\instances\instance_name`). By default, the master password file is `config/empw.dat` and the password file is in `config/txnPassStore.dat`. If you are not sure of the location of these files on your system, look at the file `config/configPassman.cnf`.

3. Shut down API Gateway.

4. Copy the replacement files to appropriate directory.
5. Restart API Gateway.

Note:

Always back up and restore the master password file (if you use one), the password file, and the passman.cnf file together.

Resetting the Master Password

You can use the API Gateway Administration > Security > Master password section to reset the master password and all the stored passwords in the unlikely event the master password or the other passwords are lost or corrupted.


The reset procedure clears the stored passwords and resets the master password to *manage*.

You must have the Manage security configurations functional privilege assigned to reset the master password.

You must reset the passwords if any of the following is true:

- Your master password and passwords are stored in files and you do not have recent backups of the master password file, the password file, and the passman.cnf file.
- API Gateway is configured to prompt for the master password and you do not have recent backups of the password file and the passman.cnf file.
- API Gateway is configured to prompt for the master password and you have lost or forgotten the master password.

> To reset the master password

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Master password**.
3. Click **Reset**.
4. Click **Yes** in the confirmation dialog.

This clears the stored passwords and resets the master password to *manage*.

OAuth, JWT, and OpenID Configuration

This section describes the Open Authorization (OAuth), JSON Web Token (JWT), and OpenID Connect (OpenID) authentication protocols that you can use to identify and authorize a client application. The application is first identified based on the criteria provided in the strategy configured. A strategy is a way to authenticate the incoming request and provides multiple

authentication mechanisms or multiple authorization servers for a single authentication scheme. API Gateway identifies the application and validates the token submitted through the strategy configured in the application.

OAuth Authentication Use case and Workflow

The Open Authorization is a flexible authorization framework for securing application access to protected resources of APIs. OAuth 2.0 uses access tokens that are presented by client applications (on behalf of the end users) to access the protected resources.

The OAuth authorization framework includes the following terms:

Roles

- *Resource Owner* (or the End User). This is the holder of the protected resources that the client application accesses. The resource owner is typically a person (usually the end user), but could also be an application.
- *Client Application* (or the Client). This is the application that is requesting access to protected resources on behalf of the end user.
- *Resource Server* (or API Gateway). This is the server that stores the protected resources the application is trying to access. API Gateway acts as a resource server.
- *Authorization Server*. This is the server that acts as an interface between the client application and end user, authenticates the end user, and issues access tokens after proper authorization. API Gateway can be configured to act as an OAuth 2.0 authorization server. You can configure API Gateway for use with a third-party OAuth 2.0 authorization server, such as PingFederate.

Authorization Grant Types

- *Authorization Code*. This is the grant type used to obtain access tokens (and optionally refresh tokens) and is optimized for confidential clients.

Note:

API Gateway supports the confidential client authentication using Authorization headers. The confidential clients have to authenticate using their credentials, the client id and client secret combination. Few points to consider using the Authorization Code grant type:

- If the property `watt.server.oauth.token.endpoint.auth=session` (the default value) and the confidential client already has a session when it comes to the token endpoint, the access to the endpoint is granted even if there are no credentials in the header.
- If the property `watt.server.oauth.token.endpoint.auth=credentials` or if the client does not already have a session, the confidential client must provide the `client_secret` in the Authorization header.
- API Gateway does not support the client secret in the body of the request for the Authorization code grant.

For details about the properties mentioned, see *webMethods Integration Server Administrator's Guide*

- *Implicit*. This is the grant type used to obtain access tokens and is optimized for public clients. It does not support the issuance of refresh tokens.

- **Client Credentials.** This is the grant type used to obtain access tokens for client-only authentication.
- **Resource Owner Password Credentials.** This is the grant type used to obtain access tokens when the resource owner has a trust relationship with the client, and clients are capable of obtaining the resource owner's credentials.

Clients

- **Confidential.** A confidential client is an application that is capable of keeping a client password confidential to the world. This client password is assigned to the client app by the authorization server. This password is used to identify the client to the authorization server, to avoid fraud. An example of a confidential client could be a web app, where no one but the administrator can get access to the server, and see the client password.
- **Public.** A public client is an application that is not capable of keeping a client password confidential. For instance, a mobile phone application or a desktop application that has the client password embedded inside it. Such an application could get cracked, and this could reveal the password. The same is true for a JavaScript application running in the users browser. The user could use a JavaScript debugger to look into the application, and see the client password.

API Gateway can be used as an authorization server and as a resource server.

API Gateway as a Resource Server

When API Gateway acts as a resource server, it hosts the protected resources, and accepts and responds to the client applications' requests that include an access token. The client application sends the access token in the Authorization request header field using the Bearer authentication scheme. The resource server validates the access token locally or remotely if it cannot validate locally.

If the token is valid and the client application has privileges to access the protected resources, the resource server executes the request. If the access token is invalid, it rejects the request.

API Gateway as an Authorization Server

When API Gateway acts as an authorization server, it receives authorization requests from client applications. The authorization server handles the interactions between the client application, resource server, and resource owner for approval of the request.

As an authorization server API Gateway issues tokens to client applications on behalf of a resource owner for use in authenticating subsequent API calls to the resource server. The resource server hosts the protected resources, and can accept or respond to the protected resource requests using access tokens. If the client application is authorized to access the protected resources, the resource server executes the request. The authorization server retains the information about the access tokens it issues, including the user information. When a client presents an access token to the resource server, the resource server sends the token to the authorization server to ensure that the token is valid and that the requested service is within the scope for which the access token was issued. A *scope* is the definition of the resources that the client application can access on behalf

of a resource owner. If the client application does not have privileges to access the resources, the resource server rejects the request.

Using API Gateway with an External Authorization Server

When API Gateway is the resource server, you must specify an authorization server. As an alternative to using API Gateway as the authorization server, you can use a third-party server as the authorization server. This allows API Gateway to validate access tokens issued by third-party servers and also allow to dynamically create clients in the third-party server.

Note:

- Before you configure API Gateway to use a third-party authorization server, make sure that the authorization server is compliant with the RFC 7662, OAuth 2.0 token introspection.
- From API Gateway release 10.3 onwards API Gateway supports multiple authorization servers.

To use an external authorization server, you must configure your third-party authorization server. This includes, but is not limited to, the following:

- To introspect the token, you should have a JWKS URI or you should create a client account that API Gateway uses to call the authorization server's introspection endpoint.

Make a note of the `client_id` and `client_secret` values. You provide this information as part of defining the external authorization server alias for the API Gateway resource server.

Make a note of the URL for the introspection endpoint. You provide this information as part of defining the external authorization server alias in the API Gateway resource server.

Validation of JWT token of the external authorization server happens in the following ways:

Local Introspection	Remote Introspection
<p>Validation of the JWT token happens within the gateway in the following methods:</p> <ul style="list-style-type: none"> ■ Using JWKS URI. <p>The external authorization server's signature is verified by using the public certificate in the JWKS URI.</p> <p>API Gateway's cache has a key as <i>kid</i> claim and its value is the certificate corresponding to the <i>kid</i> claim. The cache is populated on every restart of API Gateway by invoking the JWKS URI.</p> <p>In the runtime, while validating the token using the local introspection, the <i>kid</i> value from the incoming JWT is fetched and the corresponding certificate is retrieved from</p>	<p>Validation of the JWT token happens with the authorization server. Therefore, token caching is not possible in remote introspection.</p> <p>It has an introspection endpoint, which is used to validate the token. In addition, the client id and client secret are used to protect the endpoint, so that anonymous users cannot access the resource. To invoke an endpoint, you require a user; Gateway user is the one you can use to invoke the endpoint.</p>

Local Introspection	Remote Introspection
<p>the cache and the signature validation happens.</p>	<ul style="list-style-type: none"> <li data-bbox="240 352 446 384">■ Using RSA. <p data-bbox="289 413 854 548">The external authorization server's signature in the JWT is verified by the truststore defined in the local introspection configuration.</p> <li data-bbox="240 575 483 606">■ Using HMAC. <p data-bbox="289 636 854 980">If the authorization server uses HMAC algorithm, that means the signature validation of the JWT is performed using a shared key between the authorization server and API Gateway. You must specify the HMAC shared secret when creating the strategy of the application. The HMAC shared secret in the application is used to validate the authorization server's signature present in JWT.</p>

- Create the required scopes.
- Configure an alias to the authorization server.

Currently, API Gateway, by default, can be used with the following third-party authorization servers, but are not limited to, that are RFC 7662, OAuth 2.0 token introspection compliant:

- Okta
- PingFederate

You can also use other third-party authorization servers like Google, keycloak, and so on.

Authorizations for applications created from API Portal

When you create applications through API Portal, you must specify the required authorization server using the **watt.server.oauth.authServer.alias** settings in the Administration section of API Gateway.

If API Gateway is the authorization server, then provide `local` as the value of the **watt.server.oauth.authServer.alias** setting. Else, provide the name of the corresponding authorization server. For information on extended settings, see [“Configuring Extended Settings” on page 22](#).

Use case 1: OAuth Authentication with API Gateway as a Resource server as well as an Authorization server

This describes the high level workflow for the scenario where API Gateway is a resource server as well as an Authorization server.

1. Configure API Gateway as an internal authorization server.

Ensure you configure OAuth scopes while configuring the authorization server. For a complete procedure on configuring API Gateway as an internal authorization server, see [“Configuring the Internal Authorization Server” on page 180](#).

2. Map the scopes.

For a complete procedure on mapping scopes, see [“Mapping OAuth or OpenID Scopes” on page 191](#).

3. Enforce the Identify and authorize application policy on the API.

Ensure to select OAuth2 token. For details of the Identify and authorize application policy see, [“Identify and Authorize Application” on page 398](#).

4. Associate an application with the API.

You can create a new application or use an existing one. Ensure that the application associated contains the strategy for OAuth authentication. While creating a strategy you can associate it with the scopes that are available to be used while using dynamic client registration. For a complete procedure on creating an application with a strategy, see [“Creating an Application” on page 597](#). Refer the [“Authorizations for applications created from API Portal” on page 167](#) section for information on configuring authorization server for the applications created from API Portal.

5. Activate the API.

User on invoking the API uses the OAuth identification method to access the protected resource.

6. Get OAuth token required to access the application. For the procedure on retrieving an OAuth token, see [“Retrieving OAuth Token” on page 171](#).

7. Use the access token to invoke the API.

Use case 2: OAuth Authentication with API Gateway as a Resource server and an external Authorization server

This describes the high level workflow for the scenario where API Gateway is the resource server with a third-party authorization server. This is generally used in an environment where there is an existing authorization server, which is used with API Gateway as a resource server.

1. Configure a Provider if you are using the Dynamic client registration. Else you can proceed to step 2.

For a complete procedure on configuring a provider, see [“Adding a Provider” on page 183](#)

2. Configure an external authorization server.

Ensure you configure OAuth scopes while configuring the authorization server. For a complete procedure on configuring an external authorization server, see [“Adding an External Authorization Server” on page 186](#).

3. Map the scopes.

For a complete procedure on mapping scopes, see [“Mapping OAuth or OpenID Scopes” on page 191](#).

4. Enforce the Identify and authorize application policy on the API.

Ensure to select OAuth2 token. For details of the Identify and authorize application policy see, [“Identify and Authorize Application” on page 398](#).

5. Associate an application with the API

You can create a new application or use an existing one. Ensure that the associated application contains the strategy for OAuth authentication and contains the client ID. While creating a strategy you can associate it with the scopes that are available to be used while using dynamic client registration. If the authorization server supports dynamic client registration, then you can select the option Generate credentials. If the application is already created in authorization server, then provide the respective client id. For a complete procedure on creating an application with a strategy, see [“Creating an Application” on page 597](#). Refer the [“Authorizations for applications created from API Portal” on page 167](#) section for information on configuring authorization server for the applications created from API Portal.

6. Activate API.

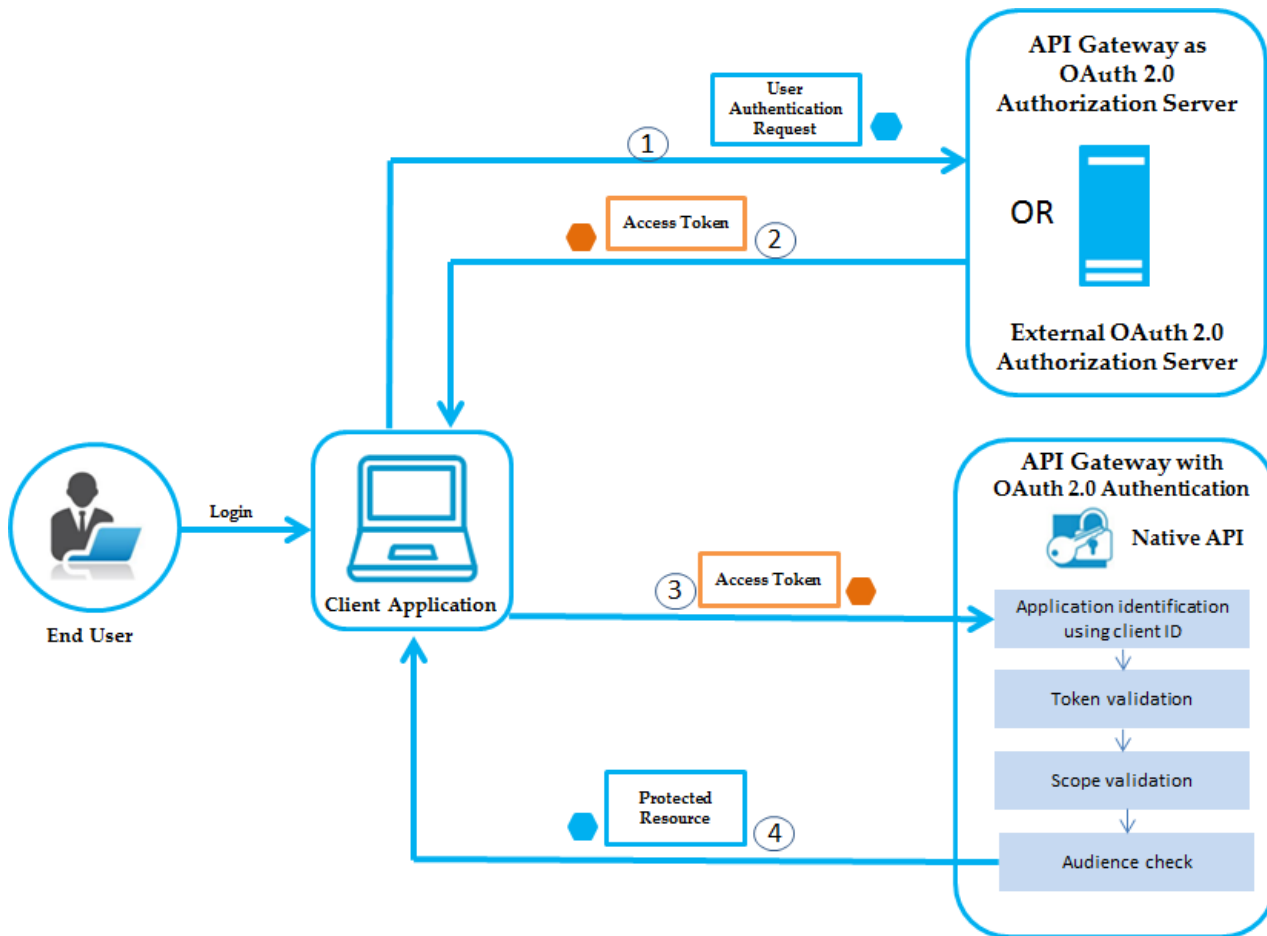
User on invoking the API uses the OAuth identification method to access the protected resource.

7. Get the access token from the authorization server. For the procedure on retrieving an OAuth token, see [“Retrieving OAuth Token” on page 171](#).

8. Use the access token to invoke the API.

OAuth Authorization Workflow

The flow of authorization requests and responses between the end user, client application, authorization server, and resource server is as depicted in the following figure.



The OAuth authorization workflow is as follows:

1. The end user logs in, the client application sends the authentication request to the authorization server to obtain an access token.
2. Authorization server validates the request and generates an access token for the client.
3. Client uses this access token to send HTTP requests to API Gateway.
4. API Gateway then performs the following:
 - a. Identifies the application using the clientId.
 - b. Validates the token locally or remotely if it is not possible locally.
 - c. Checks if the requested resource is part of the scopes in the token.
 - d. Checks the audience.

If all the above are validated, API Gateway provides access to the protected resource. If the access token is expired, authorization server returns a specific error response. The client application can then use Refresh Token to request a new access token. The Authorization Server returns a new access token that can be used to access the protected resource.

Retrieving OAuth Token

You must retrieve an OAuth token to access an API that is OAuth protected.

> To retrieve an OAuth token

1. Open your REST client.
2. Make a POST call to the following URL, with the hostname of the system where API Gateway is installed in place of localhost:

```
http://localhost:5555/invoke/pub.apigateway.oauth2/getAccessToken
```

For example

```
http://10.2.120.14:5555/invoke/pub.apigateway.oauth2/getAccessToken
```

3. Provide the following payload, with the required client id and client secret, in the **Request** section:

```
{
  "grant_type": "client_credentials",
  "client_id": "client id",
  "client_secret": "client secret"
}
```

You can find Client id and Client secret in the **Authentication** section of the **Application details** page.

For example

```
{
  "grant_type": "client_credentials",
  "client_id": "0abcd80e-f009-4a38-b52e-e663b2e18e5b",
  "client_secret": "3bd9c383-813e-40d4-b876-67c4da7c71cc"
}
```

The access token that can be used to access the required application is displayed in the **Response** section.

Sample response

```
{
  "access_token": "c9a39e14e6a84be0b228bc9bcb76ad99",
  "token_type": "Bearer",
  "expires_in": 3600
}
```

JWT Authentication Use case and Workflow

JSON Web Token is a JSON-based open standard ([RFC 7519](#)) means of representing a set of information to be securely transmitted between two parties. A set of information is the set of claims (claim set) represented by the JWT. A claim set consists of zero or more claims represented by the

name-value pairs, where the names are strings and the values are arbitrary JSON values. The claims in a JWT are encoded as a JSON object that is used as the payload of a JSON Web Signature (JWS) structure, enabling the claims to be digitally signed. JWTs can be signed using a shared secret (with HMAC algorithm), or a public or private key pair using RSA.

API Gateway can generate a JWT token itself or validate the JWT token generated by a trusted third-party server. API Gateway uses the RSA-based JWT to provide stronger integrity protection to JWTs when API Gateway is the issuer of the token. The JSON-based access tokens contain one or more claims. A claim is any piece of information that serves as a unique identifier, and that the token issuer who generated the token has verified. API Gateway extracts the claims from the JWT, identifies the application and then authorizes access to the protected resource.

Note:

JWT authentication is supported for both REST and SOAP APIs. API Gateway does not support Base64 encoded JWT tokens.

Use case 1: JWT authentication with API Gateway as a JWT issuer

This describes the high level workflow for the scenario where API Gateway can generate the JSON Web Token itself.

1. Configure API Gateway as an internal authorization server.

For a complete procedure on configuring API Gateway as an internal authorization server, see [“Configuring the Internal Authorization Server” on page 180](#).

2. Enforce the Identify and authorize application policy on the API.

Ensure to select JWT. For details of the Identify and authorize application policy see, [“Identify and Authorize Application” on page 398](#).

3. Associate an application with the API.

You can create a new application or use an existing one. Ensure that you add the required claims while creating the application, which you would use to validate the access token. For a complete procedure on creating an application with a strategy, see [“Creating an Application” on page 597](#).

4. Activate the API.

User on invoking the API uses the JWT identification method to access the protected resource.

5. You get the JWT in one of the following ways (with or without claims), which you can pass as a bearer token to invoke the API.

- **Retrieve JWT Token** - For a complete procedure on retrieving a JWT token, see [“Retrieving JWT Token” on page 174](#).

- **Retrieve JWT Token with Claims** - For a complete procedure on retrieving a JWT token with claims, see [“Retrieving JWT Token with Claim” on page 175](#).

Use case 2: API Gateway with an external JWT issuer

This describes the high level workflow for the scenario where API Gateway accepts JSON Web Token generated by a trusted third-party server.

1. Configure an external authorization server.

For a complete procedure on configuring an external authorization server, see [“Adding an External Authorization Server” on page 186](#).

2. Enforce the Identify and authorize application policy on the API.

Ensure to select JWT. For details of the Identify and authorize application policy see, [“Identify and Authorize Application” on page 398](#).

3. Associate an application with the API.

You can create a new application or use an existing one. Ensure that you add the required claims while creating the application, which you would use to validate the access token and the external authorization server that would be the JWT issuer. For a complete procedure on creating an application with a strategy, see [“Creating an Application” on page 597](#). Refer the [“Authorizations for applications created from API Portal” on page 167](#) section for information on configuring authorization server for the applications created from API Portal.

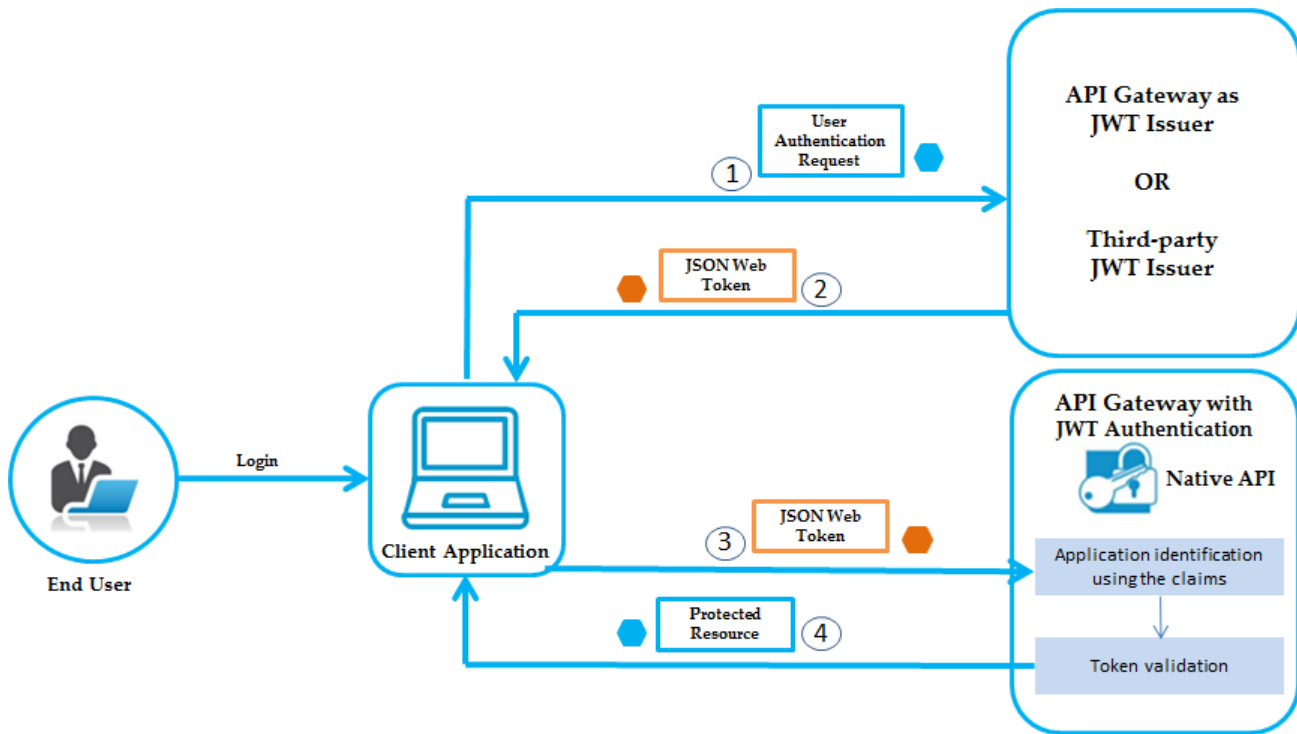
4. Activate the API.

User on invoking the API uses the JWT identification method to access the protected resource.

5. Pass the JWT as a bearer token to invoke the API.

JWT Authorization Workflow

The flow of authorization requests and responses between the end user, client application, JWT issuer, and resource server is as depicted in the following figure.



The JWT authorization workflow is as follows:

1. The end user logs in, the client application sends an authentication request to API Gateway or to any third-party JWT issuer, to obtain a JWT token.
2. If API Gateway is the JWT issuer, then it validates the user or the application. If the user or application credentials are valid, API Gateway generates the JSON token using a private key that was specified in the JWT configuration, and sends the generated token to the client.

If the user credentials are invalid, API Gateway returns a specific error response.

3. Client sends the generated JSON token in the HTTP Authorization request header as a Bearer token to access the protected API in API Gateway.
4. API Gateway first identifies the application based on claims from the JWT, then validates the JWT using the public certificate of the issuer (the issuer can be API Gateway or a third-party issuer) and provides access to the protected resources.

If the validation fails, API Gateway returns a specific error response.

Note:

If API Gateway has generated the JSON token, it validates the signature using a public certificate that was specified in the JWT configuration. Else, if the HTTP request is sent from a third-party JWT issuer, API Gateway validates the token using a public certificate or the JWKS URI of the issuer.

Retrieving JWT Token

You can retrieve JWT using one of the following ways:

- **Retrieve with static payload:** This method is used to retrieve an access token for a general access.
- **Retrieve using an Application Id:** This method is used to retrieve an access token to be used for a particular application.

➤ **To retrieve a JWT token**

1. Open your internet browser.
2. Perform one of the following steps to retrieve access token with static payload:
 - To retrieve the access token with static payload, provide the following URL in the browser, with the IP of API Gateway in place of local host:

```
http://localhost:5555/rest/pub/apigateway/jwt/getJsonWebToken
```

- To retrieve the access token for an application, provide the following URL, with the IP of API Gateway and required application Id:

```
http://localhost:5555/rest/pub/apigateway/jwt/getJsonWebToken?
app_id=applicationId
```

For example,

```
https://localhost:5556/rest/pub/apigateway/jwt/getJsonWebToken?
app_id=9502c862-9e67-4726-bc13-598df42c7fb6
```

The JWT token is displayed:



The subject claim of the token generated by making a GET call will be the username of user who calls the JWT endpoint.

Note:

You must use HTTPS protocol when retrieving JWT token. If you want to use the HTTP protocol, you must set the `pg_JWT_isHTTPS` setting in the **Administration > Extended Settings** to *false*.

Retrieving JWT Token with Claim

When you retrieve a JWT token for a particular application, the application is authenticated using the application identifiers provided in the request, such as, APIKey or Username or Host name, and then a token is generated with application id as a subject.

For example, consider multiple developers using an application to retrieve an access token. In such a scenario, each user can have a claim that can be used to identify the user who made a particular transaction.

> To retrieve a JWT token with claim

1. Open your REST client.
2. Make a POST call to the following URL, with the IP address of the system where API Gateway is installed in place of localhost:

```
http://localhost:5555/gateway/security/getJsonWebToken
```

For example,

```
http://localhost:5555/rest/pub/apigateway/jwt/getJsonWebToken
```

3. Provide your claim identifiers in the **Request** section:

```
{ "claimsSet": { "identifier": "value" } }
```

For example,

```
{ "claimsSet": { "name": "username", "company": "organization" } }
```

Note:

Before invoking this service, ensure that the authorization server is configured and the scope mapping is done.

The access token is displayed in the **Response** section. The subject claim of the token generated by making a POST call will be the ID of the identified application.

OpenID Authentication Use case and Workflow

OpenID Connect is an open standard and decentralized authentication protocol that extends on the OAuth 2.0 authorization framework. It combines the capability of Open ID in verifying the client's identity and OAuth's capability of accessing the client's resources.

In case of OpenID support in API Gateway, you can use the OpenID authentication protocol to identify and authorize a client application to access the protected resources in one of the following ways (these are explained in detail in the Astacus section.):

- Use just the access tokens (that is OAuth token) to invoke the protected resources.
- Use the ID token (that gives information about the user) to invoke the protected resources in one of the following ways:
 - Present the ID token to exchange it for an access token and use the access token to access the protected resources.
 - Use the ID token directly to access the protected resources.

API Gateway does not act as an OpenID Connect server but can validate the tokens issued by other OpenID Connect servers.

The following internal API is used for getting an access token for an ID token.

exchangeIDToken

Method: **POST**

URL: `http://host:port/gateway/security/exchangeIDToken`

Payload

```
{
  "gatewayScopes": ["OktaTenant1:inventory"],
  "idToken": "eyJhbGciOiJIUzU1NiIsImtpZCI6IjQwYzZiMDliNDQ5NjczNDUzYzNkYTYi
  "expiry": 3000
}
```

For details on scopes, see [“Mapping OAuth or OpenID Scopes” on page 191](#).

Note:

The `getOpenIDtoken` call is deprecated and is no more available from the API Gateway release 10.3 onwards.

Use case 1: OpenID authentication using OpenID Connect Provider

This describes the high level workflow for using the OpenID authentication protocol to identify and authorize a client application to access the protected resources.

1. Configure a Provider if you are using the Dynamic client registration. Else you can proceed to step 2.

For a complete procedure on configuring a provider, see [“Adding a Provider” on page 183](#).

2. Configure an external authorization server.

Ensure you configure the external authorization server with the introspection URL and OAuth scopes. For a complete procedure on configuring an external authorization server, see [“Adding an External Authorization Server” on page 186](#).

3. Map the scopes.

For a complete procedure on mapping scopes, see [“Mapping OAuth or OpenID Scopes” on page 191](#).

4. Enforce the Identify and authorize application policy on the API.

Ensure to select OpenID Connect or JWT as options. For details of the Identify and authorize application policy see, [“Identify and Authorize Application” on page 398](#).

5. Associate an application with the API.

You can create a new application or use an existing one. Ensure that the application associated contains the strategy for OpenID authentication. While creating a strategy you can associate

it with the scopes that are available to be used while using dynamic client registration. For a complete procedure on creating an application with a strategy, see [“Creating an Application” on page 597](#).

6. Activate the API.

User on invoking the API uses the access token or the ID token provided by the provider to access the protected resource.

7. User can access the protected resources in one of the following ways:

- The user presents the access token to API Gateway and on validation accesses the protected resource.
- The user presents the ID token to API Gateway to exchange it for an access token (if the user has configured the OpenID Connect option in step 4). The client then presents the access token to API Gateway and on validation accesses the protected resource.

The following internal API is used for getting an access token for an ID token.

exchangeIDToken

Method: **POST**

URL: `http://host:port/gateway/security/exchangeIDToken`

Payload

```
{
  "gatewayScopes": ["OktaTenant1:inventory"],
  "idToken": "eyJhbGciOiJIUzI1NiIsImtpZCI6IjQwYzZiMDliNDQ5NjczNDUzYzNkYTYi
  "expiry": 3000
}
```

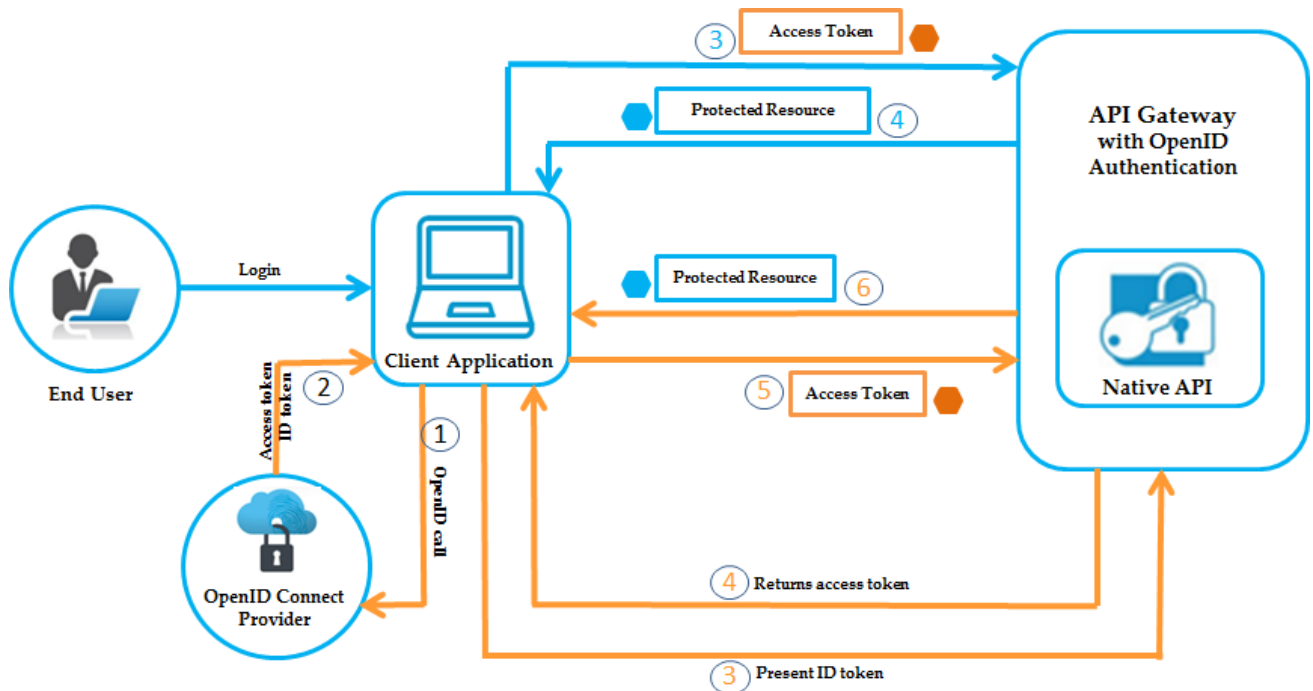
- The user presents the ID token as a JWT directly to API Gateway (if the user has configured the JWT option in step 4), and on validation accesses the protected resource.

OpenID Authorization Workflow

The OpenID Connect support in API Gateway provides two different ways for a client to access a protected resource depending on whether the provider has provided an access token or an ID token. The workflow diagram depicts both these cases. The first 2 steps are same in both the cases, the arrows in blue depict the flow where an access token is used to access the protected resource, and the arrows in orange depict the flow where an ID token is used to access the protected resource.

OpenID authorization workflow using the OpenID Connect Provider

The flow of authorization requests and responses between the end user, client application, OpenID Connect provider, and resource server is as depicted in the following figure. The client application makes an OpenID call to the OpenID Connect provider and receives an access token or an ID token in the response. It uses these tokens to access the protected resources.



OpenID authorization workflow using the access token provided by the Open ID Connect Provider

1. The client makes an OpenID call to the OpenID connect Provider.
2. The OpenID Connect Provider provides an access token to the client.
3. The client application presents the access token received from the OpenID Connect Provider to send HTTP requests to API Gateway.
4. API Gateway then performs the following:
 - a. Identifies the application using the clientId.
 - b. Validates the token locally or remotely if it is not possible locally.
 - c. Checks if the requested resource is part of the scopes in the token.
 - d. Checks the audience.

API Gateway provides access to the protected resource if all the validations are done. If the access token is valid, API Gateway provides access to the protected resource. If the access token is expired, authorization server returns a specific error response. The client application can then use Refresh Token to request a new access token. The Authorization Server returns a new access token that can be used to access the protected resource.

OpenID authorization workflow using the ID token provided by the Open ID Connect Provider

1. The client makes an OpenID call to the OpenID Connect Provider.
2. The OpenID Connect Provider provides an ID token to the client.

3. The client application presents the ID token received from the OpenID Connect Provider to API Gateway.
4. API Gateway validates the ID token and returns an access token to the client application.

The following internal API is used for getting an access token for an ID token.

exchangeIDToken

Method: **POST**

URL: `http://host:port/gateway/security/exchangeIDToken`

Payload

```
{
  "gatewayScopes": ["OktaTenant1:inventory"],
  "idToken": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjQwYzZiMDliNDQ5NjczNDUzYzNkYTY"
  "expiry": 3000
}
```

For details on mapping scopes, see [“Mapping OAuth or OpenID Scopes” on page 191](#).

5. The client then uses this access token to send HTTP requests to API Gateway.
6. API Gateway then performs the following:
 - a. Identifies the application using the clientId.
 - b. Validates the token locally or remotely if it is not possible locally.
 - c. Checks if the requested resource is part of the scopes in the token.
 - d. Checks the audience.

API Gateway provides access to the protected resource if all the validations are done. If the access token is valid, API Gateway provides access to the protected resource. If the access token is expired, authorization server returns a specific error response. The client application can then use Refresh Token to request a new access token. The Authorization Server returns a new access token that can be used to access the protected resource.

Note:

The user can present the ID token directly as a JWT to access the protected resources in case the ID token is provided on configuring the JWT property in the Identify and authorize application policy enforced on the API.

Configuring the Internal Authorization Server


Pre-requisites:

You must have the API Gateway's manage security configurations functional privilege assigned to add an authorization server.

You have to configure API Gateway with the required information to act as an internal authorization server for OAuth or JWT depending on what authentication protocol you want to use to identify



and authorize a client application. You can also define the required scopes that provide a way to limit the amount of access that is granted to an access token.

➤ **To configure an internal authorization server**

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Security > OAuth/JWT/OpenID**.
3. In the Internal Authorization servers section, click **local**.

This is the internal authorization server available that you can configure with required information to act as an internal authorization server for OAuth, JWT or OpenID authentication protocols.

4. The name field is pre-populated with the name of the internal authorization server, local, which is non-editable.
5. The description for the internal authorization server is pre-populated with the description available. You can modify the description as required.
6. Click **JWT configuration** to configure API Gateway as a JWT issuer.



Alternatively you can expand or collapse a section, using the down arrow () and the up arrow () and that appear next to the section name.

7. Provide the following information as required:

Field	Description
Token issuer	Name of the JWT token issuer used by API Gateway. Note: The Token issuer value is case-sensitive.
Algorithm	The cryptographic algorithm to sign JSON Web Tokens (JWTs). Supported values are: RS256, RS384, and RS512.
Expiry duration	The duration (in minutes) for which the token is valid. For example, the value 60 denotes that the access token will expire in one hour from the time the token was generated.
Audience	<i>Optional.</i> The intended recipient of the token. The application that receives the token must verify that the audience value is correct and reject any tokens intended for a different audience.

Field	Description
Keystore alias	Alias of the keystore containing the private key that is used to sign JWTs. The Keystore alias field contains a list of the available keystore aliases in API Gateway. If there are no configured keystore aliases, this field displays the <code>DEFAULT_IS_KEYSTORE</code> .
Key alias	Alias of the private key used to sign JWTs. The Key alias field contains a list of the available aliases in the selected keystore. If there are no configured keystores, this field is empty.

- Click **OAuth configuration** to configure API Gateway as an OAuth authorization server.

Alternatively you can expand or collapse a section, using the down arrow () and the up arrow () and that appear next to the section name.

- Provide the following information as required:
 - **Authorization code expiration interval.** Specifies the time (in seconds) during which the authorization code issued by the authorization server is valid. Valid values are between 1 and 2147483647. The default value is 600.
 - **Access token expiration interval.** Specifies the time (in seconds) for which the access tokens issued by the authorization server are valid. The default value is 3600. Value of -1 specifies that the access token does not expire.
- Click **OAuth tokens**.

This lists the available OAuth tokens with the following details:

- **Client ID.** Specifies the ID of the client application that requested the access token.
- **Owner ID.** Specifies the ID of the owner who issues the access token.
- **Access token.** Specifies the access token
- **Refresh token.** You can use to generate a new access token if the existing access token is expired.
- **Remaining refresh limit.** Displays the remaining attempts for refreshing the access token.
- **Action.** Revokes the access tokens, which means those tokens cannot be used to invoke the protected resource.

Note:

By default, API Gateway lists only 5 records and provides pagination to explore more tokens. You can also use the search and filter options to find the OAuth tokens. Here, the search and filter options are case sensitive.

You can remove all the expired OAuth access tokens using the following API.

Method : **GET**

URL: `hostname:port/invoke/pub.oauth/removeExpiredAccessTokens`

Note:

You can schedule the cleanup of the expired OAuth access tokens as required.

11. Click **OAuth scopes**.

OAuth 2.0 scopes provide a way to limit the amount of access that is granted to an access token. For example, an access token issued to a client application may be granted READ and WRITE access to the protected resources, or just the READ access. You can implement your APIs to enforce any scope or a combination of scopes as required. So, if a client receives a token that has READ scope, and it tries to invoke an API endpoint that requires WRITE access, the invocation fails.

You can provide the meaning to the scope in OAuth/OpenID scopes management section.

12. Type the scope that is registered in the authorization server and click **+Add**.

You can include multiple scopes.

13. Click **Update**.

This updates the internal authorization server details with the required information.

Adding a Provider


Pre-requisites:

You must have the API Gateway's manage security configurations functional privilege assigned to add a provider.

The OAuth 2.0 configuration in API Gateway is split into two sections - Providers and Authorization servers.

You have to add a provider and configure the authorization provider metadata information in this section for API Gateway to communicate with this provider during dynamic client registration only. If there is any deviation from the actual OAuth specification then the provider has to be configured for these deviations.

> To add a provider

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Security > JWT/OAuth/OpenID > Providers**.
3. Click **Add provider** and provide the following information:

Field	Description
Name	<p>Name of a third-party provider. For example, Amazon.</p> <p>You can also use one of the following pre-configured third-party providers that is shipped with the API Gateway installation:</p> <ul style="list-style-type: none"> ■ OKTA ■ PingFederate <div style="background-color: #f0f0f0; padding: 10px; margin-top: 10px;"> <p>Note: Considerations while using the PingFederate providers:</p> <ul style="list-style-type: none"> ■ If you want to use the pre-configured PingFederate provider, you have to use the Admin APIs for dynamic client registration for registering clients. ■ If you want to use the DCR API, you can create a provider to use DCR API. But, you cannot update or delete the clients created using the DCR API. </div>

Client metadata field mapping. Specifies the mapping of dynamic client registration specification to that of the client implementation of the provider.

The **Client metadata field mapping** fields are required when you are adding a third-party provider that is not shipped with API Gateway.

Specification name The client metadata attributes in accordance with the dynamic client registration specification as defined in RFC 7591.

The available values are:

- **redirect_uris.** Redirection URL that the authorization server uses to redirect the authorization code once the authorization request is approved by end user.

Note:

If you do not specify this attribute, API Gateway automatically generates the URL.

- **token_endpoint_auth_method.** The client authentication method at the token endpoint.
- **grant_types.** The grant type of authorization flow to obtain authorization codes, ID tokens, and refresh tokens.
- **application_type**
- **response_types.** The type of response that the client application uses at the authorization endpoint.
- **client_name.** Name of the client to use to represent the client application to the end user during authorization.

Field	Description
	<ul style="list-style-type: none"> ■ client_uri. URL of the client application. ■ logo_uri. URL of an image to use to represent the client application to the end user during authorization. <p>Note: The logo_uri is currently not supported in API Gateway.</p> <ul style="list-style-type: none"> ■ scope. List of user-authorized scopes that the client uses for requesting access tokens. <p>Note: If you do not specify this attribute, the authorization server registers the client with a default set of scopes.</p> <ul style="list-style-type: none"> ■ contacts. The means (for example, Email address) by which end users can contact the client for support requests. ■ tos_uri. URL of the service document for the client that describes a contractual relationship between the end-user and the client that the end-user accepts when authorizing the client. <p>Note: The tos_uri is currently not supported in API Gateway.</p> <ul style="list-style-type: none"> ■ jwt_uri. URL of the JSON Web Key (JWK) Set document containing the client's public keys. <p>Note: The jwt_uri is currently not supported in API Gateway.</p> <ul style="list-style-type: none"> ■ client_id. Identifier that is unique to the client application. ■ client_secret. The password or phrase for the client application to use to authorize communication with the end user.

Implementation name The client metadata attributes that are used by the authorization server, but are not in accordance with the dynamic client registration specification.

Example:

- For the **redirect_uris** field, provide the value *redirectUris*.
- For the **grant_types** field, provide the value *grantTypes*.
- For the **client_name** field, provide the value *name*.
- For the **logo_uri** field, provide the value *logoUrl*.

Field	Description
	<ul style="list-style-type: none"> ■ For the client_id field, provide the value <i>clientId</i>. ■ For the client_secret field, provide the value <i>secret</i>.
<p>Extended request parameters. Specifies the additional client metadata attributes that are specific to the authorization server, and are not specified in the dynamic client registration specification.</p> <p>In PingFederate (For example):</p> <pre>forceSecretChange = true</pre>	
Type	<p>Specifies the client metadata attribute type.</p> <p>The available values are: Client read, Client registration, Client update, Client delete.</p>
Key	The client metadata attribute key that is specific to the authorization server.
Value	A value for the client metadata attribute key. When sending requests to the authorization server, this value is appended to all requests.

You can add multiple request parameters by clicking **+ Add**.

4. Click **Save**.

The provider is added and displayed in the list of providers.


Adding an External Authorization Server

Pre-requisites:

You must have the API Gateway's manage security configurations functional privilege assigned to add an authorization server.

As an alternative to using API Gateway as the authorization server, you can use a third-party server as the authorization server. To use an external authorization server, you must configure your third-party authorization server.

➤ To add an external authorization server

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Security > OAuth/JWT/OpenID**.

The available and configured internal authorization servers and external authorization servers are listed in the respective sections.

3. Click **Add authorization server** in the External authorization servers section.
4. Type a name for the external authorization server.
5. Type a description for the external authorization server that is being configured.
6. Provide the discovery endpoint in the **Discovery URL** field and click **Discover**.

When you specify the discovery URL, API Gateway fetches data from the URL response and auto-populates the fields with the fetched data.

7. Click **Introspection** and provide the following information to validate the incoming tokens.

Field	Description
	Local introspection. Provide the following information to validate the tokens locally.
Issuer	Name of the token issuer.
JWKS URI	<p>Specifies JSON Web Key Signature endpoint to retrieve the corresponding public certificates for performing local introspection.</p> <p>API Gateway's cache has a key as kid claim and its value is the certificate corresponding to the kid claim. The cache is populated on every restart of API Gateway by invoking the JWKS URI. In the runtime, while validating the token using the local introspection, the kid value from the incoming JWT is fetched and the corresponding certificate is retrieved from the cache and the signature validation happens.</p>
Truststore alias	<p>Specify the alias of the truststore on API Gateway that holds the Certificate Authority (CA) certificate of third-party authorization server.</p> <p>This is required if the JWKS URI is not available for the authorization server and you want to configure this certificate directly.</p>
Certificate alias	<p>Alias of the certificate used to validate the token.</p> <p>The Certificate alias field contains a list of the available aliases in the selected truststore. If there are no configured truststores, this field is empty.</p>
	Remote introspection. Provide the following information to validate the tokens remotely if local introspection cannot be done.
Introspection endpoint	URL of the token introspection endpoint of a third-party OAuth 2.0 authorization server. API Gateway uses the introspection endpoint to check that access tokens used in client requests are currently active and are valid to invoke the protected resources.


Field	Description
Gateway user	The name of the Gateway user that API Gateway uses to invoke the token introspection endpoint.
Client ID	ID of the introspection client on the authorization server that API Gateway uses to introspect the access tokens.
Client secret	Password of the introspection client that API Gateway uses to introspect the access tokens.

8. In the Dynamic client registration section, provide the following information if you want to dynamically create a client from API Gateway when required.

Note:

The dynamic client registration is not supported for external authorization servers when you publish an application from CentraSite to API Gateway.

You would use this configuration only if you do not intend to use any of the existing clients.

Field	Description
Enabled	<p>Specifies whether dynamic client registration is enabled.</p> <p>Click the toggle button to change the state to  to enable dynamic client registration.</p> <p>By default this option is disabled.</p>
Provider name	Select the name of the third-party provider.
Client registration URL	Specifies the corresponding REST endpoint URLs for the client configuration of REST APIs.
Authentication type	<p>Specifies the type of authentication scheme that API Gateway would use to communicate with the external authorization server for client management.</p> <p>Select one of the following authentication type:</p> <ul style="list-style-type: none"> ■ Basic. Specifies the username and password information that would be passed in the authorization header of HTTP request for client authentication. ■ Username. The username to access the protected resources of REST APIs. ■ Password. A valid password associated with the username. ■ Token. Specifies the token information that would be added as a bearer token in the HTTP request for client authentication.

Field	Description
	<ul style="list-style-type: none"> ■ Token type. The type of token that would be contained in the HTTP request. ■ Token. The token that would be contained in the HTTP requests. ■ Refresh token. Specifies the refresh token information that would be added as a bearer token in the HTTP request for client authentication. ■ Refresh token. The refresh token that you would get from the external authorization server for the registered client ID and client secret. ■ Client ID. The client ID that you want to specify from the external authorization server. ■ Client secret. A valid client secret associated with the client ID. ■ Client credentials. Specifies the client information for which the application is created in the external authorization server. ■ Scope. The scope of the client application that you want to specify from the external authorization server. ■ Client ID. The client ID that you want to specify from the external authorization server. ■ Client secret. A valid client secret associated with the client ID. ■ None . Specifies that you could create the client dynamically in the external authorization server without using any type of authorization.
Supported grant types	<p>Specifies the list of grant types that are supported by API Gateway. Basically, grant types are the ways to get an access token from the external authorization server.</p> <p>Provide the grant type, in the Supported grant types field and click +Add.</p> <p>You can add more than one grant by clicking +Add.</p>

9. In the SSL Configuration section, provide the following information for SSL configuration, if the authorization server wants the 2-way SSL for the requests.

Field	Description
Keystore alias	<p>Alias of the keystore containing the private key that is used for a secured communication between API Gateway and the authorization server.</p> <p>You can view all the keystore aliases available in API Gateway. If there are no configured keystore aliases, the list box contains only the default keystore, <i>DEFAULT_IS_KEYSTORE</i>.</p>

Field	Description
Key alias	Alias for the private key to use to validate the HTTP requests from the client. You can view all the aliases available in the selected keystore. If there are no configured keystores, this list box is empty.
Truststore alias	Alias of the truststore on API Gateway that holds the Certificate Authority (CA) certificate of third-party authorization server. Note: You need to select a truststore alias only when all of the following are true: <ul style="list-style-type: none"> ■ The client account on the third-party authorization server is configured to use mutual (two-way) SSL, and ■ The authorization server's Certificate Authority certificate is not in the set of well-known authorities trusted by the JVM in which API Gateway runs.

10. In the Metadata section, provide the following information for the authorization server metadata, which is used for the communication to portal.

Field	Description
Access token URL	The endpoint URL on the authorization server through which the client application exchanges the authorization code, client ID, and client secret, for an access token.
Authorize URL	The endpoint URL on the authorization server through which the end user authenticates and grants authorization to the client application.
Refresh token URL	The endpoint URL on the authorization server through which the client application refreshes an expired access token.

11. Click **Scopes**.

OAuth 2.0 scopes provide a way to limit the amount of access that is granted to an access token. For example, an access token issued to a client application may be granted READ and WRITE access to the protected resources, or just the READ access. You can implement your APIs to enforce any scope or a combination of scopes as required. So, if a client receives a token that has READ scope, and it tries to invoke an API endpoint that requires WRITE access, the invocation fails.

You can provide the meaning to the scope in OAuth/OpenID scopes management section.

12. Provide the scope, in the Scope field that is registered in the authorization server and click **+Add**.

You can add more than one scope by clicking



13. Click **Save**.

The external authorization server is added. You can add as many authorization servers as required, but only one is the default at any given time.


Mapping OAuth or OpenID Scopes

You must have the API Gateway's manage security configurations functional privilege assigned to manage scopes.

You have to map the scope that you have defined in the authorization server with the APIs in API Gateway to authorize the access tokens to be used to access the protected resources. You can map either a complete API or parts (resources or methods) of an API to the scope.

For example, if there is a scope you have defined for an external authorization server, such as `readonly`, then the access tokens which contain `readonly` as their scope, should access only the GET resources. So, you can create an API Scope for the GET resources in an API or for multiple APIs and then map this `readonly` scope to all those API Scopes. Now this access token can invoke only the GET resources. If it tries to invoke any POST or PUT resource it fails. As another example you can consider mapping a business scope such as, `inventory`, that you have defined in the authorization server; you can map all the resources required for the `inventory` business to this scope.

➤ To map a scope


1. Expand the menu options icon , in the title bar, and select **OAuth/OpenID scopes**.
2. Click **Map scope**.
3. Provide the following information in the Authorization server scope section:

Field	Description
Select authorization server scope	Specifies the scope linked to the authorization server. Type a search word and select the required scope from the search list populated.
Name	Displays the name of the authorization server scope selected. This is populated by default and is non-editable.
Description	A brief description for the scope being mapped.
Audience	Provide a value or URI, the intended recipient of the authorization server scope.

Field	Description
	The application that receives the token verifies that the audience value is correct and rejects any tokens intended for a different audience.

- Click **API scopes**.
- Specify an API scope that is to be linked to the authorization server.

Alternatively, you can type a search word and select the required API scope from the search list populated.

The API scopes added are listed in the Selected API scopes table. You can click the delete icon , in the corresponding column, to delete an API scope from the list.

- Click **Save**.


This maps the authorization server scope to the selected API scopes and lists the authorization scope in the scopes list.


Viewing Scope Mapping Details

You must have the API Gateway's manage security configurations functional privilege assigned to manage scopes.

You can view the scope details and modify the scope details as required from the OAuth/OpenID scopes page.

> To view scope mapping details

- Expand the menu options icon , in the title bar, and select **OAuth/OpenID scopes**.

A list of available scopes appears. The details, such as, name and description of the scope is displayed in the form of a table. You can delete a scope by clicking the delete icon .

- Click a scope.

The scope details page appears. This page displays the details such as the authorization server name, the server scope, the API scopes that are linked to the server scope and the API scope details such as the API to which the scope is associated, the description of the API and API version number.

You can modify the scope by clicking the **Edit** button and modifying the required values.


Note:

You can edit or delete the APIs from the scope mapping, only if the APIs are assigned to your team(s).

Viewing Provider List and Provider Configuration



You can view the list of integrated third-party providers and their configuration details, modify the provider configuration, and delete a provider in the Providers section.

➤ To view a list of providers and provider configuration

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Security > JWT/OAuth/OpenID > Providers**.

The Providers section displays a list of all the defined third-party providers in API Gateway.

You can also perform the following operations in the Authorization servers section.

- You can view the configuration details of the provider by clicking the required provider. The provider details page displays the client metadata field mapping and extended request parameter configuration information of the selected provider.
- You can edit the provider configuration by clicking the required authorization server and modifying the details as required.
- You can reset the configuration to system default value by clicking  in the Action column for the respective provider.
- You can delete the required authorization server by clicking  in the Action column for the respective provider.


Modifying the Provider Configuration

Pre-requisites:

You must have the API Gateway's manage security configurations functional privilege assigned to modify a provider.

You might want to modify a provider to change the currently defined configuration settings.

➤ To modify a provider configuration

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Security > JWT/OAuth/OpenID > Providers**.

The Providers section displays a list of all the available providers in API Gateway.


3. Click the name of the partner provider you want to modify.
4. Modify the fields as required.
5. Click **Save**.

The provider is updated.

Viewing Authorization Server List and Server Configuration


You can view the list of authorization servers and their configurations details, modify the server configuration, and delete an authorization server in the Authorization servers section.

> To view a list of authorization servers and server configuration

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Security > JWT/OAuth/OpenID**.

The Authorization servers section displays a list of all the internal and external authorization servers in API Gateway under respective sections.

You can also perform the following operations in the Authorization servers section.

- You can view the configuration details of the authorization server by clicking the required authorization server. The authorization server details page displays the client information, scope information, and token information of the selected authorization server.
- You can edit the server configuration by clicking the required authorization server and modifying the details as required.
- You can delete the required authorization server by clicking  in the Action column for the respective authorization server.


Modifying Authorization Server Configuration

Pre-requisites:

You must have the API Gateway's manage security configurations functional privilege assigned to modify an authorization server.

You might want to modify an OAuth 2.0 authorization server to change the currently defined configuration settings.

> To modify the authorization server configuration

1. Expand the menu options icon , in the title bar, and select **Administration**.

2. Select **Security > JWT/OAuth/OpenID**.

The Authorization servers section displays a list of all the internal and external authorization servers in API Gateway.

3. Click the name of the authorization server you want to modify.
4. Modify the fields as required.
5. Click **Save**.

The authorization server is updated.


Deleting an Authorization Server

Pre-requisites:


You must have the API Gateway's manage security configurations functional privilege assigned to delete an authorization server.

You delete an authorization server to remove it from API Gateway permanently.

> To delete an authorization server

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Security > JWT/OAuth/OpenID**.

The Authorization servers section displays a list of available internal and external authorization servers in API Gateway.

3. Click  in the action column of the authorization server to be deleted.
4. Click **Yes** in the confirmation dialog.

The authorization server is deleted from API Gateway.

Deleting a Provider

Pre-requisites:


You must have the API Gateway's manage security configurations functional privilege assigned to delete a provider.

You delete a provider to remove it from API Gateway permanently.


Important:

You must not delete a provider if it is being used by an authorization server.

> To delete a provider

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Security > JWT/OAuth/OpenID**.

The Providers section displays a list of available providers in API Gateway.


3. Click  in the Action column of the provider to be deleted.
4. Click **Yes** in the confirmation dialog.

The provider is deleted from API Gateway.

Configuring Communication Details for Microgateway

When you want Microgateway to use API Gateway as an OAuth2 authorization server, the communication channel between Microgateway and API Gateway has to be set up. The access token is then introspected in the Microgateway using remote introspection. To enable this you have to configure communication details, such as the introspection endpoint, client ID and client secret, in API Gateway, which are then used by Microgateway to introspect the tokens in API Gateway.

> To configure the communication details for Microgateway

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Security > JWT/OAuth/OpenID > Microgateway**.
3. In the **Introspection endpoint** field, provide the URL of the introspection endpoint.

For example: `http://localhost:5555/invoke/pub.oauth/introspectToken`. The endpoint can have `http` or `https` depending on the protocol used and the hostname and port are the details of the host used.

Microgateway uses the introspection endpoint to check that access tokens used in client requests are currently active and are valid to invoke the protected resources.

4. In the **Client ID** field, provide an ID, which specifies the ID of the introspection client on the authorization server that Microgateway uses to introspect the access tokens.
5. In the **Client secret** field, provide the Client secret, which specifies the password of the introspection client that Microgateway uses to introspect the access tokens.
6. Click **Save**.

The information provided here is stored in the configuration properties file and provisioned as part of the asset provisioning during Microgateway startup.

Destination Configuration

API Gateway can publish events and performance metrics data to the configured destinations. Event type data provides information about activities or conditions that occur on API Gateway. The performance data provides information on average response time, total request count, fault count, and so on, for the APIs that it hosts.


You must have the API Gateway's manage destination configurations functional privilege assigned to configure the following destinations to which the event types and performance metrics data is published:

- API Gateway
- API Portal
- Transaction logger
- CentraSite
- Database
- Digital Events
- Elasticsearch
- Email
- SNMP

Configuring Events for API Gateway Destination

You have to configure the API Gateway destination so that the events, performance metrics, and audit log data can be published to API Gateway. By default, error events, lifecycle events, policy violation event, and performance data are published to API Gateway.

➤ To configure events for API Gateway destination

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Destinations**.
3. Select **API Gateway** to configure the event types for this destination.
4. In **Event types**, select the type of events to publish to API Gateway.

The available event types are:

- **Error:** Occurs each time an API invocation results in an error.
 - **Lifecycle:** Occurs each time API Gateway is started or shut down.
 - **Policy violation:** Occurs each time an API invocation violates the policy enforcement that was set for the API.
5. Select **Report performance data** to publish performance metrics data.
 6. In the **Publish interval** box, enter a time interval (in minutes) how often API Gateway must publish performance metrics. Enter a value from 1 through 60. The default is 60 minutes.
 7. In the **Audit log data** section, select the required management areas for which the audit logs should be recorded in the API Gateway destination.

Audit logs provide a record of system transactions, events, and occurrences in API Gateway. You can configure audit logging to show the following events:

- **API management**
- **Application management**
- **Team management**
- **Group management**
- **Package management**
- **Promotion management**
- **Approval management**
- **Alias management**
- **Analytics management**
- **Policy management**
- **Plan management**
- **User management**

Note:


By default, audit logging is enabled for all of the above-listed management areas in the API Gateway destination.

8. Click **Save**.

Configuring API Portal Destination

You have to configure API Portal as a destination to establish communication channel between API Gateway and API Portal to exchange data.

➤ To configure API Portal destination

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Destinations**.
3. Select **API Portal > Configuration** to configure API Portal as the destination.
4. Provide the following information in the Basic information section:

Field	Description
Name	Name of the portal being configured. This name will be referenced in API Portal under the list of providers and hence this name should be unique across your API Gateway instances in case multiple API Gateways are publishing APIs to the same API Portal.
Version	Version of the portal being configured.

5. Provide the following information in the Portal configuration section for Gateway to send data to API Portal:

Field	Description
Base URL	URL of the API Portal instance in the format <code>http://host:port</code>
Tenant	The tenant details of API Portal. By default, default tenant is considered if the tenant information is not provided.
Username	Username credential to access API Portal instance.
Password	Password credential to access API Portal instance.

6. Provide the following information in the Gateway configuration section for API Portal to create applications, request or revoke access tokens, subscriptions, and so on:

Field	Description
Base URL	URL of the API Gateway instance. This is pre-populated.
	<p>Note: When a load balancer URL is updated in API Gateway, the API Gateway base URL is be updated to same in this field.</p>

Field	Description
Username	Username credential of the API Gateway Administrator to access API Gateway instance.
Password	Password credential to access API Gateway instance.

7. Click **Publish**.

This establishes a communication channel between API Gateway and API Portal.


Configuring Events for API Portal Destination

Pre-requisites:

You have to configure API Portal to communicate with API Gateway before you select the events and metrics for publishing to API Portal.

You have to configure the API Portal destination so that the events and performance metrics data can be published to API Portal.

> To configure events for API Portal destination

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Destinations**.
3. Select **API Portal > Events** to configure the event types for this destination.
4. In **Event types**, select the type of events that you want API Gateway to publish to API Portal.

The available event types are:

- **Error**: Occurs each time an API invocation results in an error.
 - **Lifecycle**: Occurs each time API Gateway is started or shut down.
 - **Policy violation**: Occurs each time an API invocation violates the policy enforcement that was set for the API.
5. Select **Report performance data** to publish performance metrics data.
 6. In the **Publish interval** box, enter a time interval (in minutes) how often API Gateway must publish performance metrics. Enter a value from 1 through 60. The default is 60 minutes.
 7. Click **Save**.

Post-requisites:

After performing the event configurations, select **API Portal** as a **Destination** in the Policy Properties page for each policy, to receive the transaction event logs for the assigned policies.


Configuring Transaction Logger Destination

You have to configure transaction logger as a destination to receive the API Gateway transaction event logs. The transaction log data is written to a file or a database based on the configurations. The events are sent to the log file for the Log Invocation policy.

Note:

Any modifications to the API Gateway transaction logger destination in Integration Server do not reflect in API Gateway UI. Hence, Software AG recommends that you do not configure or modify API Gateway transaction logger destination through the Integration Server Administrator UI.

> To configure transaction logger destination

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Destinations**.
3. Select **Transaction logger**.
4. Provide the following information:

Field	Description
Name	The default name of the transaction log, API Gateway Transaction Logger.
Enable	Click the Enable activation toggle button to enable the logger to start writing the log entries to the database or the file.
Mode	Specifies whether the logger is to write entries to the destination synchronously or asynchronously. <ul style="list-style-type: none"> ■ Synchronous: In this mode, the logger writes entries directly to the destination. ■ Asynchronous: In this mode, the logger writes entries to a queue, then later writes the entries from the queue to the destination. Each logger has its own queue.
Guaranteed	Provides data about guaranteed delivery transactions. You can use guaranteed delivery log entries to do the following: <ul style="list-style-type: none"> ■ Track when transactions start and their current status.

Field	Description
	<ul style="list-style-type: none"> See the names of guaranteed delivery processes that are running. Track whether the processes completed successfully or failed. <p>The default value is No.</p>
Destination	<p>Specifies whether the logger is to write entries to a file or database.</p> <ul style="list-style-type: none"> Database: If this option is selected, the logger writes entries to the database. <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: You must ensure that in webMethods Integration Server, a pool alias is associated to the ISCoreAudit functional alias. For more information on pool alias, see <i>webMethods Integration Server Administrator's Guide</i>.</p> </div> <ul style="list-style-type: none"> File: If this option is selected, the logger writes entries to the file in <i>Integration Server_directory\instances\instance_name\logs\APIGateway</i> directory. The default value is file.
Maximum queue size	<p>Specifies the maximum number of entries the queue can hold. Specify numerals only (for example, do not include commas or periods).</p> <p>The default value is <i>100000</i> records.</p> <p>Choose a value that accommodates your system's average volume for log entries. If your logging volume has sudden spikes, the queue can usually catch up by writing the pending entries during lulls. If the queue size is reached, then any additional log entries would be lost. Ensure to check the availability of disk space or memory of API Gateway server before you change the default value.</p>
Maximum retries	<p>Specifies the maximum times the logger must retry writing the entry to the destination if the first attempt fails because of a transient error. A transient error is an error that arises from a temporary condition that might be resolved or corrected quickly, such as the unavailability of a resource due to network issues or failure to connect to a database. The default value is 3.</p>
Wait between retries	<p>Specifies the waiting time before the logger can reconnect and rewrite the entries to the destination in case of failure. The default value is 5 seconds.</p>

- Click **Save** to save the specified transaction log configuration value.

Note:

For these modifications to take effect, you must restart the API Gateway server.

You can click **Cancel** to revert to the last saved changes or to abandon all the changes if the values are not saved.

Post-requisites:

After performing the events configurations, you must select **Audit Log** as a **Destination** in the Policies properties page for each policy, to receive the transaction event logs for the assigned policies.

Configuring CentraSite Destination


You have to configure CentraSite as a destination to establish a communication channel between API Gateway and CentraSite to exchange data.

Once an API Gateway asset is published from CentraSite to API Gateway, the CentraSite communication and SNMP configuration details automatically appear in the **CentraSite** destination of API Gateway. If the same API Gateway asset is unpublished from the API Gateway instance at a later time, the CentraSite communication and SNMP configuration details are automatically removed from the **CentraSite** destination.

Note:

If you have already configured CentraSite as a destination in API Gateway, then publishing another API Gateway asset from any CentraSite instance to API Gateway fails and displays an error. In API Gateway, only one CentraSite instance can be configured as a destination at a time. To reconfigure another CentraSite instance, you need to use the **Force reset CentraSite communication and SNMP details** option.

➤ To configure CentraSite destination

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Destinations**.
3. Select **CentraSite > Configuration** to configure the CentraSite communication and SNMP details.

The Communication section displays the following information:

Field	Description
Protocol	Specifies the communication protocol used to establish communication between API Gateway and CentraSite.
Hostname	Specifies the host name or IP address of the machine on which CentraSite Application Server Tier (CAST) is running.
UDDI port	Specifies the port on which CAST is listening. The default port number for CAST is 53307.

Field	Description
Username	Specifies the CentraSite user ID for authenticating CentraSite when API Gateway communicates with CentraSite. This implies the user ID of a user who has the CentraSite Administrator role or the API Gateway Administrator role in CentraSite.
Target name	Specifies the name of the API Gateway asset as defined in CentraSite.

The SNMP section displays the following information:

Field	Description
Transport	Specifies the wire transport protocol that is used by the SNMP Listener. Supported values are: TCP and UDP.
Hostname	Specifies the CentraSite host name or IP address to which the SNMP listener binds.
Port	Specifies the port to which the SNMP listener binds. The default port number for CentraSite's SNMP server is 8181.
Username	Specifies the SecurityName that is used by the SNMP Listener.
Authorization protocol	Specifies the authorization protocol that is used by the SNMP Listener for decoding the incoming trap. Supported values are: MD5 and SHA.
Privacy protocol	Specifies the privacy protocol that is used by the SNMP Listener for decoding the incoming trap. Supported values are: AES128, AES, AES192, AES256, 3DES, and DESEDE.

4. Select **Send SNMP traps to CentraSite** to publish data about the runtime events and metrics to the CentraSite SNMP server.
5. Select **Force reset CentraSite communication and SNMP details**, and click **Reset** to delete the current configuration and restore the system configuration.
6. Click **Save**.

This establishes a communication channel between API Gateway and CentraSite.

Note:

The transaction events are published from API Gateway to CentraSite using SNMP. The runtime metrics are published from API Gateway to CentraSite using a UDDI client.


Configuring Events for CentraSite Destination

Pre-requisites:

You have to configure CentraSite to communicate with API Gateway before you select the events for publishing to CentraSite.

You have to configure the CentraSite destination so that events and performance metrics data of APIs in API Gateway can be published to CentraSite. However, you will be able to publish the data to CentraSite destination only for APIs published from CentraSite to API Gateway.

➤ To configure events for API Gateway destination

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Destinations**.
3. Select **CentraSite > Events** to configure the event types for this destination.
4. In **Event types**, select the type of events that you want API Gateway to publish to CentraSite.

The available event types are:

- **Error**: Occurs each time an API invocation results in an error.
 - **Lifecycle**: Occurs each time API Gateway is started or shut down.
 - **Policy violation**: Occurs each time an API invocation violates the policy enforcement that was set for the API.
5. Select **Report performance data** to publish performance metrics data.
 6. In the **Publish interval** box, enter a time interval (in minutes) to specify how often API Gateway must publish performance metrics. Enter a value from 1 through 60. The default is 60 minutes.
 7. Click **Save**.

Post-requisites:

After performing the event configurations, select **CentraSite** as a **Destination** in the Policy Properties page for each policy, to publish the transaction and monitoring event logs for the assigned policies. API Gateway sends these events to CentraSite through SNMP.

Important:

As a best practice, Software AG recommends you not to use the CentraSite destination for transaction events with large data payloads. This is because, the SNMP server using which the events are published from API Gateway to CentraSite does not handle transaction events with large data payloads.

Configuring Events for Database Destination

You have to configure the database destination so that the events, performance metrics, and audit log data can be published to the configured database.


API Gateway supports the following databases:

- DB2
- Oracle
- SQL Server

Note:

API Gateway functional alias has to be configured to receive the events in the Integration Server JDBC Pools configuration.

> To configure events for Database destination

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Destinations**.
3. Select **Database** to configure the event types for this destination.
4. In **Event types**, select the type of events that you want API Gateway to publish to the database.

The available event types are:

- **Error:** Occurs each time an API invocation results in an error.
 - **Lifecycle:** Occurs each time API Gateway is started or shut down.
 - **Policy violation:** Occurs each time an API invocation violates the policy enforcement that was set for the API.
5. Select **Report performance data** to publish performance metrics data.
 6. In the **Publish interval** box, provide a time interval (in minutes) to specify how often API Gateway must publish performance metrics. Provide a value from 1 through 60. The default is 60 minutes.
 7. In the **Audit log data** section, select the required management area for which the audit logs should be recorded in the Database destination.

Audit logs provide a record of system transactions, events, and occurrences in API Gateway. You can configure audit logging to show the following events:

- **API management**

- **Approval management**
- **Application management**
- **Alias management**
- **Team management**
- **Analytics management**
- **Group management**
- **Policy management**
- **Package management**
- **Plan management**
- **Promotion management**
- **User management**

Note:

By default, audit logging is disabled for all the mentioned management areas in the Database destination.

8. Click **Save**.

Post-requisites:

After performing the database configurations, select **Database** as a **Destination** in the Policy Properties page for each policy, to receive the transaction event logs for the assigned policies.

Configuring Events for Digital Events Destination


Pre-requisites:

API Gateway communicates with the Digital Event Services (DES) component which runs within the API Gateway server. For more information on configuring DES using Command Central, see *Using Digital Event Services to Communicate between Software AG Products*.

Digital Event Services (DES) enables API Gateway to communicate through digital events. Digital events are typed and serialized data structures used to communicate application or system runtime information. The application information could be state of a business process step or any associated business data. The system information could be the amount of memory used by an application.

You have to configure the Digital Event Services destination so that the events, performance metrics, and audit log data can be published to Digital Event Services.

➤ To configure events for Digital Event Services destination

1. Expand the menu options icon , in the title bar, and select **Administration**.

2. Select **Destinations**.
3. Select **Digital Event Services > Events** to configure the event types for this destination.
4. In **Event types**, select the type of events that you want API Gateway to publish to Digital Event Services.

The available event types are:

- **Error**: Occurs each time an API invocation results in an error.
 - **Lifecycle**: Occurs each time API Gateway is started or shut down.
 - **Policy violation**: Occurs each time an API invocation violates the policy enforcement that was set for the API.
5. Select **Report performance data** to publish performance metrics data.
 6. In the **Publish interval** box, enter a time interval (in minutes) how often API Gateway must publish performance metrics. Enter a value from 1 through 60. The default is 60 minutes.
 7. In the **Audit log data** section, select the required management area for which the audit logs should be recorded in the Digital Event Services destination.

Audit logs provide a record of system transactions, events, and occurrences in API Gateway. You can configure audit logging to show the following events:

- **API management**
- **Approval management**
- **Application management**
- **Alias management**
- **Team management**
- **Analytics management**
- **Group management**
- **Policy management**
- **Package management**
- **Plan management**
- **Promotion management**
- **User management**

Note:

By default, audit logging is disabled for all of the above-listed management areas in the Digital Event Services destination.

8. Click **Save**.


Post-requisites:

After performing the server configurations, select **Digital Events** as a **Destination** in the Policy Properties page for each policy, to receive the transaction events logs for the assigned policies.

Configuring Elasticsearch Destination

You have to configure Elasticsearch as a destination to establish a communication channel between API Gateway and Elasticsearch to exchange data.

> To configure Elasticsearch destination

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Destinations**.
3. Select **Elasticsearch > Configuration** to configure Elasticsearch as the destination.
4. Provide the following information in the Basic information section:

Field	Description
Protocol	Specifies the communication protocol used to establish communication between API Gateway and Elasticsearch.
Hostname	Specifies the host name or IP address of the machine on which Elasticsearch is running.
Port	Specifies the port where Elasticsearch server runs. The default port number is 9240.
Index name	Specifies the index name for Elasticsearch, where the data is stored.
Username	Specifies the Elasticsearch user ID for authenticating Elasticsearch when API Gateway communicates with it.
Password	Specifies the password of the Elasticsearch instance to be used for establishing communication between API Gateway and Elasticsearch.

Note:

You can provide the username and password for the Elasticsearch instances having configured with Basic authentication. You can also provide HTTPS enabled Elasticsearch instance.

5. Click **Test**.

This tests the communication channel between API Gateway and the configured Elasticsearch.

6. Click **Save** to save the specified email configuration value.

You can click **Cancel** to revert to the last saved changes or to abandon all the changes if the values are not saved.


Configuring Events for Elasticsearch Destination

Pre-requisites:

You have to configure Elasticsearch to communicate with API Gateway before you select Elasticsearch as a destination.

You have to configure Elasticsearch as a destination so that the events, performance metrics, and audit log data can be published to Elasticsearch.

> To configure Elasticsearch destination

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Destinations**.
3. Select **Elasticsearch > Events** to configure the event types for this destination.
4. In **Event types**, select the type of events that you want API Gateway to publish to Elasticsearch.

The available event types are:

- **Error**: Occurs each time an API invocation results in an error.
 - **Lifecycle**: Occurs each time API Gateway is started or shut down.
 - **Policy violation**: Occurs each time an API invocation violates the policy enforcement that was set for the API.
5. Select **Report performance data** to publish performance metrics data.
 6. In the **Publish interval** box, enter a time interval (in minutes) to specify how often API Gateway must publish performance metrics. Enter a value from 1 through 60. The default is 60 minutes.
 7. In the **Audit log data** section, select the required management area for which the audit logs should be recorded in the Elasticsearch destination.

Audit logs provide a record of system transactions, events, and occurrences in API Gateway. You can configure audit logging to show the following events:

- **API management**
- **Application management**
- **Team management**
- **Group management**
- **Package management**
- **Promotion management**
- **Approval management**
- **Alias management**
- **Analytics management**
- **Policy management**
- **Plan management**
- **User management**

Note:

By default, audit logging is disabled for all of the above-listed management areas in the Elasticsearch destination.

8. Click **Save**.


Post-requisites:

After performing the event configurations, select **Elasticsearch** as a **Destination** in the Policy Properties page for each policy, to publish the transaction and monitoring event logs for the assigned policies.

Configuring Email Destination

You have to configure email as a destination to receive alert notifications. The alerts are sent to the email ID specified in the Log Invocation, Monitor Service Performance, Monitor Service Level Agreement, and Throttling Traffic Optimization policies. Before configuring email as a destination, you must perform the following email server configurations to establish a connection between API Gateway and the email server.

> To configure Email destination

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Destinations**.
3. Select **Email > Configuration**.

4. Provide the following information:

Field	Description
SMTP server	The server name or the IP address of the SMTP server that API Gateway uses to send the messages.
Port	The port that API Gateway uses to connect to the specified SMTP server.
Transport layer security	<p>The SSL encryption type that API Gateway uses when communicating with an email server. Use one of the following transport layer security options:</p> <ul style="list-style-type: none"> ■ None. Default. Use an unsecure mode when API Gateway is communicating with an email server. When you specify <i>None</i>, the email server authenticates the email client using only the username and password. ■ Explicit. Use explicit security when API Gateway is communicating with an email server. With this type of security, API Gateway initially establishes an unsecure connection with the email server and then upgrades to the secure mode. <p>With explicit transport layer security, API Gateway communicates with the email servers that support SSL encryption and also with those that do not support SSL encryption. If the email server does not support transport layer security, API Gateway disconnects the connection which was established earlier with the email server. You can then establish an unsecure connection by selecting <i>None</i> in the Transport layer security field and enable the port.</p> ■ Implicit. Use implicit security when API Gateway communicates with an email server. With this type of security, API Gateway always establishes an encrypted connection with the email server. Only clients that support SSL are allowed to access API Gateway.
Truststore alias	The truststore that API Gateway uses while sending the request to a native API. Truststore is a repository that stores all the trusted public certificates.
Default email charset	The character set that API Gateway uses to be recognized by the system. Type the character set in the Default email charset field. By default, API Gateway uses UTF-8 character set for the messages.
From email	The email address from which you want to send the alerts.
Test email	The email address to which you want to send the test email. This email address can be the same as the From email address.

Field	Description
Internal email	The email address to which you want to send critical log entry messages. Typically, you would specify the e-mail address for the API Gateway Administrator.
Service email	The email address to which you want to send service failure message. In a development environment, you might direct these messages to the developer. In a production environment, you might direct these messages to the API Gateway Administrator.

5. Click **Test**.

This tests the communication channel between API Gateway and the configured email server.

6. Click **Save** to save the specified email configuration value.

You can click **Cancel** to revert to the last saved changes or to abandon all the changes if the values are not saved.


Post-requisites:

After performing the server configurations, select **Email** as a **Destination** in the Policies properties page for each policy, to receive the email alerts for the assigned policies.

Configuring Email Templates

API Gateway provides a default email template to send email alerts. You can compose and save the subject line as well as the email content for reuse. You can also customize the template to suit your needs.

> To configure Email Templates

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Destinations**.
3. Select **Email > Templates** to configure the event templates.
4. Specify the following for the Log Invocation, Monitor Service Level Agreement, Monitor Service Performance, and Throttling Traffic Optimization events:
 - **Subject:** The subject line of the email to be sent.
 - **Content:** By default, the template appears. You can customize the email content.

The template consists of the following default information for the Log Invocation event:

Note:

The @ character is a place holder and the values are automatically generated by the system. For example, Status: @status appears as Status: SUCCESS in the email. You can use the existing parameters multiple times or delete the parameter if the parameter is not required from the available parameters in the template. However, you cannot add new parameters.

```
The transaction event parameters from the API Gateway Metrics and
Event Notification engine are:
Runtime_Policy: @policy_action_name
API: @api_name
Version : @version
Operation or Resource_Name: @operation_resource_name
Native endpoint: @native_endpoint
Event generation time: @description
Consumer_Name: @consumer_name
Consumer_ID: @consumer_ID
Status: @status
Coorelation_ID:@correlationID
Error origin: @errorOrigin
```

The template consists of the following default information for the Monitor Service Level Agreement, Monitor Service Performance, and Throttling Traffic Optimization events:

```
The monitor event parameters from the API Gateway Metrics and
Event Notification engine are:
Runtime_Policy: @policy_action_name
API: @api_name
Version : @version
Operation or Resource_Name: @operation_resource_name
Native endpoint: @native_endpoint
Action type: @actionType
Attribute: @attribute
Consumer_Name: @consumer_name
Consumer_ID: @consumer_ID
Alert Message: @alertMessage
```

Additionally, you can click  to abandon the changes and revert to the default template.


You can click  to review the changes before adding the changes to the email content.

5. Click **Save**.

Configuring SNMP Destination

You have to configure SNMP as a destination to send the events to a third-party SNMP server.

> To configure SNMP destination

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Destinations**.

3. Select **SNMP**.
4. Provide the following information:

Field	Description
Send TRAPs to 3rd party SNMP server	Specifies API Gateway to use the SNMP traps to capture events that you can send to a third-party SNMP server.
SNMP target type	<p>Specifies target type of the SNMP server. The available options are:</p> <ul style="list-style-type: none"> ■ User: To send the events information to a SNMP user. ■ Community: To send the events information to a group of SNMP users that form a community. <p>By default, User is selected.</p>
Hostname	<p>The host name or IP address of the machine where the CentraSite SNMP or the third-party SNMP server is installed and running.</p> <p>Note: The host name cannot be localhost.</p>
Port	The SNMP trap receiver port that is listening for SNMP requests and packets.
Transport	<p>The protocol used by SNMP to send traps. The available options are: TCP and UDP. By default, TCP is selected.</p> <p>If you select UDP, ensure that the SNMP server is in the same subnet, or configure the routers to get packets across subnet boundaries. The maximum PDU size when running in UDP mode is 64 Kb which might be restrictive when sending large request and response payloads for Transaction Events.</p>
Username	<p>If User is selected as the SNMP target type, then the Username field specifies the SNMPv3 user name to use when connecting to the receiver.</p> <p>If Community is selected as the SNMP target type, then the Community name field specifies the name to be used to interact with the SNMP receiver. This value must match the value that you set in the SNMP receiver.</p>
Use authorization	<p>Specifies whether an authorization key is required. You cannot edit the authorization fields unless Use authorization option is selected. The authorization fields are:</p> <ul style="list-style-type: none"> ■ Password: The key to be used for authorization. ■ Protocol: The authorization protocol is MD5

Field	Description
Use privacy	Specifies whether a privacy (encryption) key is required. You cannot edit the privacy fields unless Use privacy option is selected. The privacy fields are: <ul style="list-style-type: none"> ■ Password: The key to be used for privacy. ■ Protocol: The privacy protocol is DES.

- Click **Save** to save the specified SNMP configuration value.

You can click **Cancel** to revert to the last saved changes or to abandon all the changes if the values are not saved.

Post-requisites:

After performing the server configurations, select **SNMP** as a **Destination** in the Policies properties page for each policy, to receive the transaction events logs for the assigned policies.

Note:

Metrics is not supported for third-party SNMP server.


Configuring Events for SNMP Destination

Pre-requisites:

You have to configure a SNMP server to communicate with API Gateway before you select SNMP as a destination for the events and metrics for publishing to a third-party SNMP server.

You have to configure SNMP as a destination so that the event types data can be published to a third-party SNMP server.

➤ To configure events for SNMP destination

- Expand the menu options icon , in the title bar, and select **Administration**.
- Select **Destinations**.
- Select **SNMP > Events** to configure the event types for this destination.
- In **Event types**, select the type of events that you want API Gateway to publish to SNMP.

The available event types are:

- **Error:** Occurs each time an API invocation results in an error.
- **Lifecycle:** Occurs each time API Gateway is started or shut down.

- **Policy violation:** Occurs each time an API invocation violates the policy enforcement that was set for the API.

5. Click **Save**.

Post-requisites:

After configuring the events for SNMP destination, you need to select **SNMP** as a **Destination** in the Policy Properties page of the API to publish the transaction and monitoring events for the API.

Audit Logging

The audit logging feature of API Gateway provides audit information for different categories of system transactions, events, and occurrences of specific events (for example, login attempts) over a period of time. You can use audit logs to view a detailed record of various auditable events that occurred on the API Gateway objects, user login and logout operations, and identify the users who are responsible for the changes. You can configure which audit events to log for a specific destination based on your auditing requirements.

You can configure API Gateway to log the auditable events for following destinations:

- API Gateway
- Database
- Digital Event Services (DES)
- Elasticsearch

The following auditable events can be configured to write to the API Gateway audit logs:

- **API management events**

API management consists of the following events:

- Creation, modification, and deletion of an API object.
- Activation and deactivation of an API.

- **Approval management events**

Approval management consists of the following events:

- Approval and rejection of a request to create, register, and modify an application.
- Approval and rejection of a request to subscribe a package in API Portal.

- **Application management events**

Application management consists of the following events:

- Creation, modification, and deletion of an Application object.

- **Alias management events**

Alias management consists of the following events:

- Creation, modification, and deletion of an Alias object.

■ **Team management events**

Team management consists of the following events:

- Creation, modification, and deletion of teams.

■ **Analytics management events**

Analytics management consists of the following events:

- Archiving, purging, and restoring of analytics data in the database.

■ **Group management events**

Group management consists of the following events:

- Creation, modification, and deletion of a Group object.

■ **Policy management**

Policy management consists of the following events:

- Creation, modification, and deletion of a global Policy object.
- Creation, modification, and deletion of an API level Policy object.
- Activation and deactivation of a global policy.
- Activation and deactivation of an API level policy.

■ **Package management events**

Package management consists of the following events:

- Creation, modification, and deletion of a Package object.

■ **Plan management events**

Plan management consists of the following events:

- Creation, modification, and deletion of a Plan object.

■ **Promotion management events**

Promotion management consists of the following events:

- Creation, modification, and deletion of a Stage object.
- Promotion of an API stage.
- Rollback operation of an API stage.

■ **User management events**

User management consists of the following events:

- A user logs in or fails to log in to API Gateway.
- A user logs out of API Gateway.
- Creation, modification, and deletion of a User object.

API Gateway writes the audit logging data to the **Audit logs** dashboard (in the API Gateway user interface, go to **Analytics > Audit logs**). You can view and download audit logs.

Best Practices for API Gateway Audit Logging

API Gateway's audit logging feature has been implemented on an event-driven approach. By default, the API Gateway destination is enabled to log the auditable events for all areas of management, such as APIs, policies, users, and so on. As a best practice, Software AG recommends that you enable audit logging for the required management areas in other supported destinations: Database, Digital Events, and Elasticsearch. This practice is especially important when you want to provide the audit log data to external sources for analytics and anomaly detections.

Configuring Audit Logs

You have to configure which events you need to audit for a destination so that API Gateway logs the auditable events data to the specific destination. You can configure API Gateway to log the auditable events data to the following destinations:

- API Gateway
- Database
- Digital Event Services (DES)
- Elasticsearch


The following events are available for audit log reports:

- API management
- Approval management
- Application management
- Alias management
- Team management
- Analytics management
- Group management
- Policy management
- Package management

- Plan management
- Promotion management
- User management

By default, all of the auditable events are logged into the API Gateway destination.


➤ **To configure audit logs**

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Destinations**.
3. Select the required destination to log the auditable events.
4. In the **Audit log data** section, select the required management areas to monitor, audit, and report the data.
5. Click **Save**.

Viewing Audit Logs

You can use the audit log reports to view the data of auditable events.

➤ **To view audit logs**

1. Expand the menu options icon , in the title bar, and select **Analytics**.
The dashboard displays the API Gateway-wide analytics based on the metrics monitored.
2. Select **Audit logs**.
3. In the drop-down list, choose the time interval in which you want to view the data of auditable events.

The available options are:

- Last 2 days
- Last 7 days
- Last 30 days
- Last 60 days
- Last 90 days
- Custom

4. If you select **Custom**, type the **From Date** and **To Date** to specify the time interval that best suits your needs.
5. Click **Apply filter** to filter the analytics based on the time interval chosen.


You can view logs for API Gateway auditable events in the **Audit logs** dashboard.

You can also download the API Gateway audit log in a text file and view the auditable events data.

Downloading Audit Logs

You can download the audit log reports to examine the data of auditable events.

> To download audit logs

1. Expand the menu options icon , in the title bar, and select **Analytics**.
- The dashboard displays the API Gateway-wide analytics based on the metrics monitored.
2. Select **Audit logs**.
3. In the drop-down list, choose the time interval in which you want to view the data of auditable events.
4. If you select **Custom**, type the **From Date** and **To Date** to specify the time interval that best suits your needs.
5. Click **Download** to download and view the detailed report.

API Gateway generates a compressed file of the audit logs and downloads it to the default download folder configured in your browser..

The compressed file is named `auditlogs_N.zip`. The compressed file contains one or more simple text files, where each text file contains 10,000 audit log records.

API Gateway audit log are listed below.

Column	Detail
id	Unique identifier of the event that produced the audit record.
eventType	Type of event (audit log) that produced the record.
creationDate	Date and time the event entry was written to the log.
event_create_ts	Time in milliseconds when the event was generated in API Gateway.

Column	Detail
objectType	Type of object (for example, User, API, Application, and so on) on which the event occurred.
action	Type of action (for example, Create, Update, Delete, and so on) that was performed on the object.
object	Unique identifier of the API Gateway object on which the action was performed.
message	Message that describes the event that occurred.
user	Name of a user on the API Gateway instance that triggered the event.
sourceMachine	The host name of the machine on which the API Gateway instance is running.
clientIPAddress	IP address of the machine on which the API Gateway instance is running.
payload	The request payload defined for the event.
status	Current status of the event (Success or Failure).

Data Management

In API Gateway, the size of the database grows rapidly as API transactions and events continue to occur on a day-by-day basis. An unexpected database growth can eventually degrade the system performance over a period of time and generate unwanted errors in the API Gateway Data Store.

API Gateway provides a comprehensive data management solution for archiving, purging, and restoring events data. It allows you to manage and protect critically important data, and also conserve database disk space, thereby improving the performance of API Gateway.

Archive and Purge

The ability to purge and archive logging and monitoring data is critical to data management. API Gateway provides an integrated framework for archiving and purging the logging and monitoring data. By archiving and purging data from the API Gateway Data Store, you can maintain a healthy and optimal database, as well as keep your data archived for future use.

The following data in API Gateway Data Store can be archived or purged using **Archive & Purge**:

- **Audit logs:** Archives and purges the auditable event data.
- **Error events:** Archives and purges the runtime error event data.
- **Lifecycle events:** Archives and purges the API Gateway's lifecycle data.

- **Monitor events:** Archives and purges the runtime monitoring data.
- **Performance metrics:** Archives and purges the runtime performance metrics data.
- **Policy violation events:** Archives and purges the policy compliance violation data.
- **Threat protection events:** Archives and purges the threat protection data.
- **Transaction events:** Archives and purges the API transactional data.
- **Application logs:** Archives and purges the aggregated logs.


Archiving Data

Pre-requisites:

To archive data in the API Gateway Data Store, you must have the Manage purge and restore runtime event functional privilege in API Gateway.

Archiving is the process of moving data that is no longer actively used for long-term retention so that it can be used at a later time. You can archive data based on the type of data or the age of the data in API Gateway Data Store.

➤ To archive data

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Manage Data**.
3. Click **Archive & purge**.
4. Select the event types for which the data has to be archived.

The available options are:

- ALL
- Audit logs
- Error events
- Lifecycle events
- Monitor events
- Performance metrics
- Policy violation events
- Threat protection events
- Transaction events

- Application logs
5. Select **Archive**.
 6. Select one of the following options to archive the required data.
 - Select **Range**. Select a period during which you want the data to be archived.
 - To archive selected types of data from a particular date till the current date, select the required date in the **From date** field.
 - To archive selected types of data from the beginning (events start date) till a particular date, select the required date in the **To date** field.

API Gateway archives the selected type of data for the given date range.

 - Select **Duration**. Type the maximum time after which you want the data to be archived.
- API Gateway archives the selected types of data after the time specified in years, months, days, hours, minutes, or seconds (1y, 1m, 1d, 1H, 1M, 1S).
7. Click **Submit**.

The required data on the API Gateway Data Store is archived.

By default, the archives are stored in the following location: `<installation_location>/profiles/IS_default/workspace/temp/default`.

You can modify this location using the **backupsharedFilelocation** property in the **Extended settings** section.

You can provide the maximum number of backups that can be archived using the **maxBackupslimit** property in the **Extended settings** section. The default value of this setting is 10. If you do not provide a value in this field, then infinite number of archives can be stored.


Purging Data

Pre-requisites:

To purge data from the API Gateway Data Store, you must have the Manage purge and restore runtime event functional privilege in API Gateway.

Purging is the process of systematically deleting unwanted data from the database. You can purge data based on the type of data or the age of the data in the API Gateway Data Store.

➤ To purge data

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Manage Data**.

3. Click **Archive & purge**.
4. Select the event types for which the data has to be purged.

The available options are:

- ALL
- Audit logs
- Error events
- Lifecycle events
- Monitor events
- Performance metrics
- Policy violation events
- Threat protection events
- Transaction events
- Application logs

5. Select **Purge**.
6. Select one of the following options to purge (delete) the required data.

- Select **Range**. Select a period during which you want the data to be purged.
 - To purge selected types of data from a particular date till the current date, select the required date in the **From date** field.
 - To purge selected types of data from the beginning (events start date) till a particular date, select the required date in the **To date** field.

API Gateway deletes the selected types of data for the given date range. The rest of the data is retained.

- Select **Duration**. Type the maximum time after which you want the data to be purged.

API Gateway deletes the selected types of data after the time specified in years, months, days, hours, minutes, or seconds (1y, 1m, 1d, 1H, 1M, 1S). The rest of the data is retained.

7. Click **Submit**.

The required data is purged from the API Gateway Data Store.

Restore

Data restore in API Gateway protects the database against data loss caused by a variety of failures, for example, accidental deletion of data, malicious attacks, power outages, system failures or malfunctions, and so on.


Restoring Data

Pre-requisites:

To restore data in the API Gateway Data Store, you must have the Manage purge and restore runtime event functional privilege in API Gateway.

Restoring is the process of copying backup data from an archive and restoring it in the database to replace lost or damaged data. You can restore the archived data in the API Gateway Data Store as required.

> To restore data

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **Manage Data**.
3. Click **Restore**.
4. In the **Archives** drop-down list, select the archived file you want to restore.
5. Click **Submit**.

The selected data is restored in API Gateway Data Store.

System Settings

API Gateway user interface provides capability to configure system-level configuration parameters and communicate these changes across nodes in the cluster. You must have the API Gateway's manage system settings functional privilege assigned to configure the following settings:

- **Dashboard setting:** configure a port on which the dashboard runs.
- **System setting:** configure the API Gateway time out interval.
- **Logging setting:** modify the logging configuration.
- **SAML SSO:** configure the SAML settings for single sign-on.

Note:

For these configuration to take effect you have to restart API Gateway instance on every node after modifying any of these settings.


Modifying API Gateway Configuration Parameters

You can modify the port on which the dashboard runs, the API Gateway timeout interval, and the logging setting for API Gateway in this section.

Note:

For these modifications to take effect you must restart the API Gateway instance on each node.

> To modify API Gateway configuration parameters

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **System Settings > Configuration**.
3. To change the port on which the dashboard runs, type the required port number in the **Port** field in the Dashboard setting section.

You have to provide a port that is not in use in the system. If the provided port is already used by other system then the dashboard does not start.
4. To change the API Gateway timeout interval, type the required value in the **API Gateway timeout (minutes)** field in the System setting section.
5. To change the logging configuration, you can enable the logs based on the **Log level** by selecting the required log level.

The available log levels are **ERROR, WARN, INFO, DEBUG, TRACE**.


This changes the logging configuration for the UI logs.

6. Select **Store log in server** to save the logs in the server. Select one of the following modes:
 - **Store at interval:** Provide the log storage interval. Stores the logs in the server at the specified time interval.
 - **Burst mode:** Store the logs for every event.
7. Click **Save**.

Configuring SAML Settings for Single Sign-on

Security Assertion Markup Language is an XML-based standard for exchanging authentication and authorization data between security domains. You can configure SAML settings for single sign-on for API Gateway in the Systems Settings section.

> To configure SAML settings for single sign-on

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **System Settings > SAML SSO**.
3. Set the toggle button to the on position to enable SAML.
4. Provide the SAML redirect URI the format being a correct URL address.
For example, `/saml/sso/login`.
5. Select **Send signed SAML auth request** if you want to send out the signed SAML authorization request to the Identity Provider (IDP).
6. Select **Require signed assertion from IDP** to receive a signed assertion from IDP.
7. Provide the **Service provider identity** which is the hostname of the machine or localhost.
8. Provide the location where the IDP metadata is stored in the **IDP metadata location** field.
The location is the file URL of IDP metadata stored locally.
9. Provide the location where the Gateway metadata is stored in the **Gateway metadata location**.
The location is the file URL of IDP metadata stored locally for example,
`http(s)://APIGatewayinstance:9072/gatewayui/saml/sso/metadata`
10. Provide the following information in the **Keystore properties** section:
 - **Keystore location:** Location where the keystore file is stored.
 - **Keystore type:** The file type of the keystore file. The file type can be either **JKS** or **PKCS12**.
 - **Keystore password:** Password to access the keystore file.
11. Provide the following information in the **Default key properties** section:
 - **Default key alias:** Provide the alias for the specific key and associated certificate within the keystore.
 - **Default key alias password:** Provide the password to access the default key alias.
12. Provide the following information in the **Sign key properties** section:
 - **Sign key alias:** Provide the alias of the default key used to digitally sign requests sent to the service provider.
 - **Sign key alias password:** Provide the password to access the sign key alias.
13. Provide the following information in the **Encrypt key properties** section:

- **Encrypt key alias:** Provide the alias of the default key used to encrypt the request that is sent to the service provider.
- **Encrypt key alias password:** Provide the password to access the encrypt key alias.

14. Click **Save**.

External Accounts

API Gateway user interface provides capability to add and configure external accounts such as service registries, AWS accounts and Integration Server instances and communicate these changes across nodes in the cluster. An API Gateway administrator can add and remove the configured external accounts.

Service Registry

A service registry is essentially a catalog of services. Applications that expose services can register their services with one or more service registries. Applications that need to consume a service look up a service registry and obtain the address of an application server that provides the service. Using service registries with API Gateway adds resilience, scalability, and security to the application stack.

API Gateway uses service registries in the following ways:

- You can publish APIs defined in API Gateway to service registries. This enables other applications to use the service registry to dynamically locate an API Gateway instance that provides that API.
- You can set an API Gateway route to a service registry endpoint. This enables API Gateway to use the service registry to dynamically locate an application instance and route the request to it.

Service Registries supported by API Gateway

API Gateway currently supports the following service registries.

- *Eureka*

Eureka is a REST-based service for locating services for the purpose of load balancing and failover of middle-tier servers. It has been primarily designed for applications in the AWS cloud.

- *Service Consul*


Service Consul is a tool for discovering and configuring services in IT infrastructure.

Adding a Service Registry

You must have the **Manage service registries** functional privilege assigned to perform this task.

You have to add and configure a service registry in API Gateway before you reference it in an API. In cluster deployments of API Gateway, you can add a service registry on any API Gateway instance—other API Gateway instances are synced automatically.

➤ **To add and configure a service registry**

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Click **Service Registries**.
3. Click **Add service registry**.

Field	Description
Name	Name used by API Gateway for the instance of service registry that you are adding.
Description	Description of the instance of service registry.
Type	Type of service registry. Available values are: SERVICE_CONSUL and EUREKA .
Endpoint configuration	
Endpoint URI	The base URI for the service registry. This should include the IP address or the FQDN and the port on which the service registry accepts requests.
Service discovery path	The relative path of the service registry discovery service. API Gateway appends this path (and the service ID) to the Endpoint URI to generate a service discovery request for the service registry. Note: A service registry discovery service returns the address (IP address or FQDN) of an application instance for the requested service ID.
Service registration path	The relative path of the service registry registration service. API Gateway appends this path (and the service ID) to the Endpoint URI to generate a service registration request for the service registry. Note: API Gateway uses this service to register (publish) a service with the service registry.
Service de-registration path	The relative path of the service registry de-registration service. API Gateway appends this path (and the service ID) to the Endpoint URI to generate a service de-registration request for the service registry.

Field	Description
	<p>Note: API Gateway uses this service to de-register (unpublish) a service with the service registry.</p>
<p>Connection timeout (seconds)</p>	<p>Specifies the time (in seconds) after which a connection attempt times out while communicating with the service registry.</p> <p>The precedence of the Connection Timeout configuration is as follows:</p> <ol style="list-style-type: none"> If you specify a value for the Connection timeout field in routing endpoint alias, then the Connection timeout value specified in the Endpoint alias section takes precedence over the timeout values defined at the API level and the global level. If you specify a value 0 for the Connection timeout field in routing endpoint alias, then API Gateway uses the value specified in the Connection timeout field in the routing protocol processing step of an API. The Read Timeout value specified at an API level takes precedence over the global configuration. If you specify a value 0 or do not specify a value for the Connection timeout field in the routing protocol processing step at the API level or specify a value 0 at an alias level, then API Gateway uses the value specified in this <code>pg.endpoint.connectionTimeout</code> property. If you do not specify any value for <code>pg.endpoint.connectionTimeout</code>, then API Gateway uses the default value of 30 seconds.
<p>Read timeout (seconds)</p>	<p>Specifies the time (in seconds) after which a socket read attempt times out while communicating with the service registry.</p> <p>The precedence of the Read Timeout configuration is as follows:</p> <ol style="list-style-type: none"> If you specify a value for the Read timeout field in routing endpoint alias, then the Read timeout value specified in the Endpoint alias section takes precedence over the timeout values defined at the API level and the global level. If you specify a value 0 for the Read timeout field in routing endpoint alias, then API Gateway uses the value specified in the Read Timeout field in the routing protocol processing step of an API. The Read Timeout value specified at an API level takes precedence over the global configuration. If you specify a value 0 or do not specify a value for the Read timeout field in the routing protocol processing step at the API

Field	Description
	level or specify a value 0 at an alias level, then API Gateway uses the value specified in this <code>pg.endpoint.readTimeout</code> property.
	d. If you do not specify any value for <code>pg.endpoint.readTimeout</code> , then API Gateway uses the default value of 30 seconds.
SSL configuration	
Keystore alias	List of keystores that are configured in API Gateway. This is used when the service registry is SSL enabled.
	Lists all available keystores. If you have not configured an Integration Server keystore the list is empty.
Key alias	Lists all the private keys that are present in the selected keystore alias. This is used when the service registry is SSL enabled.
Truststore alias	A truststore contains the certificates that are trusted by API Gateway. If the service registry is SSL enabled its certificate should be added to the selected truststore.
Basic authentication details	
Username	The basic authentication user name.
Password	The basic authentication password.
Other configuration	
Heartbeat interval	<i>(This setting is applicable only to Eureka service registries.)</i> The interval at which the API Gateway sends a heartbeat message to the Eureka server to renew its leases. Eureka expects clients, such as API Gateway, to renew the lease by sending heartbeats as per the heartbeat interval configured in the Eureka server.
Headers	
Key	An HTTP header key that is included in all the HTTP requests sent to the service registry.
Value	A value for the given HTTP header key.

- Click **Add**.

The service registry is added to API Gateway.



Removing a Service Registry

- You cannot remove a service registry if any API has been published to it.

- You can remove only the non-default service registries that are not used by any API.
- You must have the **Manage service registries** functional privilege assigned to perform this task.

In cluster deployments of API Gateway, you can remove the service registry on any API Gateway instance—other API Gateway instances are synced automatically.

➤ To remove a service registry


1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Click **Service Registries**.
3. Click  in the row that has the service registry that you want to remove.
4. Click **Yes** to confirm.

The service registry is removed from API Gateway.

Configuring Integration Server Instance for API Implementation

To implement an API to Integration Server, you must provide the details of the Integration Server instance(s) in API Gateway.

➤ To configure an Integration Server details

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Click **External accounts**.
3. Click **Integration servers**.

The list of configured Integration server instances appears.

4. Click **Add new Integration Server**.

The options to add new Integration Server details appear.

5. Provide the following details:

Field	Description
Name	Name for the Integration Server instance being added.
Description	Description for the configuration.

Field	Description
Integration Server URL	URL of the Integration Server.
User name	User credentials required to access the Integration Server instance.
Password	Password required to access the Integration Server instance.
Keystore alias	The text identifier for the Integration Server keystore file. The keystore contains the private keys and certificates (including the associated public keys) of Integration Server.
Key alias	The alias for a specific key in the specified keystore.

The screenshot shows the 'Administration' page of the webMethods API Gateway. The navigation bar includes 'General', 'Security', 'Destinations', 'Manage data', 'System settings', and 'External accounts'. The left sidebar lists 'Service registry', 'AWS configuration', and 'Integration Servers'. The main content area displays a form for adding a new Integration Server instance. The form fields are as follows:

Field	Value
Name*	IS1
Description	Integration Server Instance
Integration Server URL*	http://localhost:5555
Username*	Administrator
Password*	*****
Keystore alias	[Dropdown menu]
Key alias	[Dropdown menu]

At the bottom of the form, there are three buttons: 'Cancel', 'Test', and 'Add'.

6. To validate the connectivity of the specified Integration Server instance, click **Test**.

The connection with the given server is tested and a success message appears.

7. Click **Add**.

The server details are saved.

3 User Management

- Manage Users, Groups, and Teams 236
- Manage Your User Settings and Preferences 257

Manage Users, Groups, and Teams

You can use API Gateway to define user information on the API Gateway server. The definition of user contains the login ID, password, and group membership.

Alternatively, you can set up API Gateway to access the information from a local user management system or you can use webMethods Integration Server to configure the Lightweight Directory Access Protocol (LDAP) external directory that your site uses for user information.

Note:

Central User Management is not supported with API Gateway.

webMethods Integration Server uses user information to authenticate clients and determine the server resources that a client is allowed to access. If the server is using basic authentication (username and password) to authenticate a client, it uses the login ID and passwords defined in user accounts to validate the credentials a client supplies.

API Gateway enables you to define user and group information to the API Gateway server. The user definition contains the user login ID, password, and group membership. The group definition contains the group name and a list of users in the group. After creating users and groups, users can be given the required functional privileges based on the teams that they are part of. A user can have different set of functional privileges in the teams that they are part of. For example, a user can have administrative privileges in a team and view privileges in another.


You can add and manage user information from the User Management page. This page lists all the basic information for the following:


- **Users:** User personas who can access API Gateway and perform tasks. A predefined user is an Administrator who has administrator privileges.
- **Groups:** The group membership identifies the groups to which a user belongs. User can create a group, associate users to the group, and delete a group in API Gateway.
- **Teams:** Users who share a common role or responsibility can be grouped as teams. When the Team feature is enabled, The members of teams can access the API Gateway assets of their teams and they can perform actions on these assets based on the functional privileges assigned to their teams.
- **Account settings:** You can define the password restrictions, password expiry and the account lock settings here.
- **LDAP configuration:** You can configure API Gateway to use LDAP and manage LDAP directories here.

Adding a User

You must have the API Gateway's manage user administration functional privilege assigned to add a user to API Gateway.

➤ **To add a user**

1. Expand the menu options icon , in the title bar, and select **User management**.
2. Click **Users**.
3. Click **Add user**.
4. Provide the following information in the Basic information section:

Field	Description
Login ID	A unique ID using which the user can log on to the account.
First name	A first name that contains letters, numbers, or a combination of all. You can also use special characters: . (dot), _ (underscore), and @ (at). Other special characters and spaces are not allowed. The user name is case sensitive.
Last name	A last name that contains letters, numbers, or a combination of all. You can also use the special characters: . (dot), _ (underscore), and @ (at). Other special characters and spaces are not allowed. The user name is case-sensitive.
Password	A password that contains letters, numbers, special characters, or a combination all. Spaces are not allowed. The password is case-sensitive.
Confirm password	Retype your password to confirm.
Email addresses	A valid email address of the user. You can add multiple email addresses by clicking  .
Allow digest authentication	Allow Digest Access Authentication to authenticate the API as described in RFC2617.

5. Click **Continue to associate Groups >**.

Alternatively, you can click **Groups** to go to the Groups section and associate the user to groups. You can search for the group name in the **Name** field and associate the user to the group selected. You can associate a user to multiple groups by clicking +.

Click **Save** to save the user details at this stage and provide the group information for the user at a later time.

6. Provide the group name in the **Name** field to which the user is added.

7. Click **Save**.

Note:


After adding an API Gateway user, you must include the user in a group that is associated with a team.

Modifying User Details

You must have the API Gateway's manage user administration functional privilege assigned to modify user details.

You can modify the basic or the group information of a user. You can add the user to a different group or delete the user from an existing group.

> To modify the user details

1. Expand the menu options icon , in the title bar, and select **User management**.

2. Click **Users**.

A list of available users appears.

3. Select the login ID of the user to be modified.

The User details tab appears.

4. Click **Edit**.

This opens the basic information of the user.

5. Modify the basic information of the user.

Note:

Select **Active** if you want to make the user an active user.

6. Modify the group details of the user.


You can modify or delete the name of the existing group that appears.

7. Click **Save**.

Deleting a User

Deleting a user removes the user from all the associated groups.

> To delete a user

1. Expand the menu options icon , in the title bar, and select **User management**.
2. Click **Users**.
A list of available users appears.
3. Click the delete icon for the user that has to be deleted.
4. Click **Yes** in the confirmation dialog.

User Groups

API Gateway is shipped with the following predefined groups:

- Administrators
- API-Gateway-Administrators
- API-Gateway-Providers

By default, API Gateway's Administrator user, is part of Administrators and API-Gateway-Administrators group.

The table lists the privileges based on the user group.

Privileges	API Gateway Administrator	API Provider
Manage APIs	Y	Y
Manage aliases	Y	Y
Manage policy templates	Y	N
Activate/Deactivate APIs	Y	Y
Manage global policies	Y	N
Manage threat protection configurations	Y	N
Manage applications	Y	Y
Activate/Deactivate global policies	Y	N
Publish API to service registry	Y	Y
Manage packages and plans	Y	Y
Activate/Deactivate packages	Y	Y
Publish to API Portal	Y	Y

Privileges	API Gateway Administrator	API Provider
View administration configurations	Y	N
Execute service result cache APIs	Y	Y
Manage user administration	Y	N
Change ownership/teams	Y	N
Manage general administration configurations	Y	N
Manage destination configurations	Y	N
Manage promotions	Y	Y
Manage scope mapping	Y	N
Manage security configurations	Y	N
Manage system settings	Y	N
Manage service registeries	Y	N
Import assets	Y	Y
Export assets	Y	Y
Manage purge and restore runtime events	Y	N
Manage microgateways	Y	N

Authentication and Authorization

API Gateway is primarily accessed using API Gateway user interface, which supports Basic authentication and SAML SSO.

You can also use REST APIs to manage API Gateway. To invoke the APIs, you must have the required functional privileges.


Note:

You cannot delete predefined users, groups, and teams but you can delete the groups and access profiles that are created in API Gateway.

Adding a Group

You can add the required users to a group.

➤ [To add a group](#)

1. Expand the menu options icon , in the title bar, and select **User management**.
2. Select **Groups**.
3. Click **Add group**.

The Group details tab appears.

4. Provide the following information in the Basic information section:

Field	Description
Name	Name of the user group to add.
Description	A description for the user group.

Click **Save** to save the group details at this stage and provide the group information for the user at a later time.

5. Click **Continue to associate Users >**.

Alternatively, you can click **Users** to go to the Users section.

6. Provide the user's login ID in the **Login ID** field.


You can search users based on the characters provided in user name and email id. Select the required user from the list displayed.

7. Click **Save**.

Modifying a Group

You can modify details for a selected group. You can add users to a group or remove them if required.

> To modify a group

1. Expand the menu options icon , in the title bar, and select **User management**.
 2. Click **Groups**.
- A list of groups appears.
3. Select the group to be modified.

The Group details tab appears.

4. Click **Edit**.

This opens the details of the selected group.

5. Modify the basic information of the user.

6. Modify the user details of the group.


Edit or delete (by clicking the delete icon) the name of the existing user that appears.

7. Click **Save**.

Deleting a Group

You can delete a group from the list that appears in the Groups section of the User Management page. Once a group is deleted, the user associated to the group is unable to perform the tasks associated to that group. Deleting a group does not delete the users associated with the group.

> To delete a group

1. Expand the menu options icon , in the title bar, and select **User management**.2. Click **Groups**.

A list of groups appears.

3. Click the **Delete** icon for the group that has to be deleted.**Note:**

You cannot delete predefined groups.


4. Click **Yes** in the confirmation dialog.

API Gateway Functional Privileges

The following table lists the functional privileges and their description:

Functional Privilege	Description
Select all	To select all the listed functional controls.
Manage APIs	To create and manage APIs.
Activate/ Deactivate APIs	To activate, deactivate and manage APIs.
Manage applications	To create and manage applications and register applications with the APIs.

Functional Privilege	Description
Manage aliases	To create and manage aliases.
Manage global policies	To apply a global policy to all APIs or the selected set of APIs.
Activate/Deactivate global policies	To activate, deactivate, and manage global policies.
Manage policy templates	To apply one or more policy templates to an API.
Manage threat protection configurations	To prevent malicious attacks on applications that typically involve large, recursive payloads, and SQL injections.
Publish API to service registry	To publish and unpublish APIs to service registry.
Manage packages and plans	To create packages and plans, associate a plan with a package, and associate APIs with a package. In addition, you can view the list of packages, package details, APIs, and plans associated with the package.
Activate/ Deactivate packages	To activate, deactivate, and manage packages.
Publish to API Portal	To publish and unpublish assets to API Gateway.
View administration configurations	To view administration configurations.
Manage general administration configurations	To create and manage administration configurations.
Manage security configurations	To create and manage security configurations.
Execute service result cache APIs	To execute service result cache API.
Manage destination configurations	To publish events and performance metrics data to the configured destinations.
Manage system settings	To create and manage system settings.
Manage user administration	To create and manage users.
Manage promotions	To create stages and manage promotions.
Manage service registries	To create and manage service registries
Change ownership/ teams	To change ownership of an asset or teams.
Manage scope mapping	To manage OAuth and OpenID scopes.

Functional Privilege	Description
Import assets	To import already exported APIs, application, policies, aliases, or other assets and configurations using the Import option in the user menu ().
Export assets	To export assets to your local system.
Manage purge and restore runtime event	To purge and restore events from the API Gateway store by setting the required date or duration in the API Gateway.
Manage microgateways	To manage the Microgateways connected to the API Gateway instance.

Creating Teams

This use case explains how to create teams by assigning the required functional privileges and users to them.

This use case begins when you have identified the list of users who must given access to an asset or a particular set of assets and end when you have created a team including the identified users.


In this example, a team with developers called *DevTeam* is created with the *Dev* user group as the team members, *User1* as the Team administrator, and all privileges under Manage API, Policies and Applications are assigned to the team.

Before you begin

Ensure that you have:

- API Gateway Manage user administration privilege.
- The user group, *Dev* is created. For information on how to create a user group, see [“Adding a Group” on page 240](#).

> To create teams

1. Expand the menu options icon , in the title bar, and select **User management**.
2. Click **Teams**.
3. Click **Add Team**.
The **Create Team** page appears.
4. Provide the name and description of the team in respective fields.
5. In the **Team Administrators** section, provide any or both of the following:

- Login Id of the user who want to assign as a team administrator in the **Login ID** field.
- Name of the API Gateway user group or the LDAP group that you want to assign as team administrator in the **Group name** field.

You can search users or user groups based on the characters provided in the above fields. Select the required user from the list displayed.

For the example consider in this use case, the team is named as *DevTeam* and the *User1* is specified as the team administrator.

Create Team
Create a team by providing the basic information, functional controls and add the required groups to the team. ⓘ

Team details

Basic information

Functional privileges

Groups

Name*

DevTeam

Description

All developers

Team Administrator

Users

u ✕

Users	Action
User1	

Groups (all users of the groups are team administrators)

Type a keyword

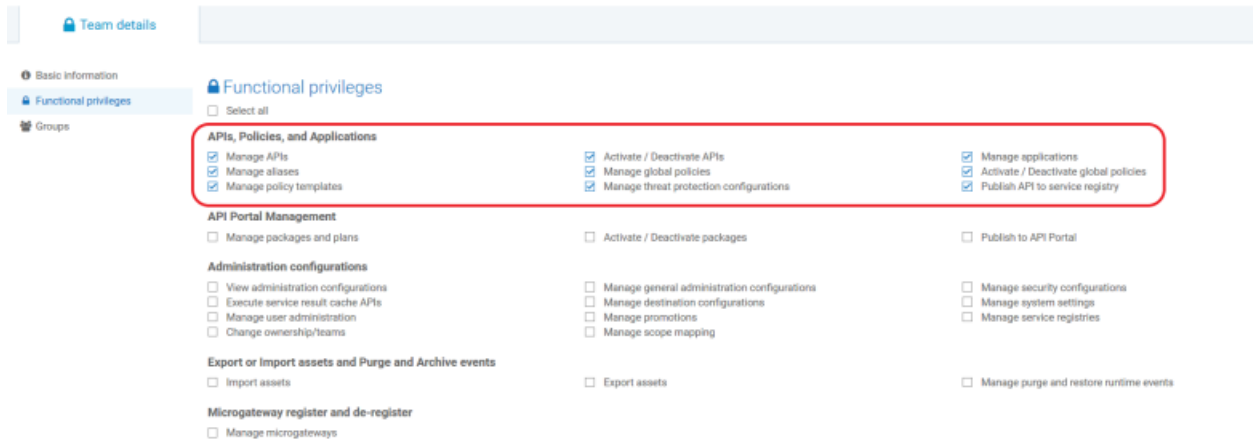
[Continue to assign functional privileges >](#)

6. Click **Continue to assign functional privileges >**.

The **Functional privileges** list appears.

7. Select the functional privileges to be assigned to the team members. For information on the available functional privileges, see [“API Gateway Functional Privileges” on page 242](#).

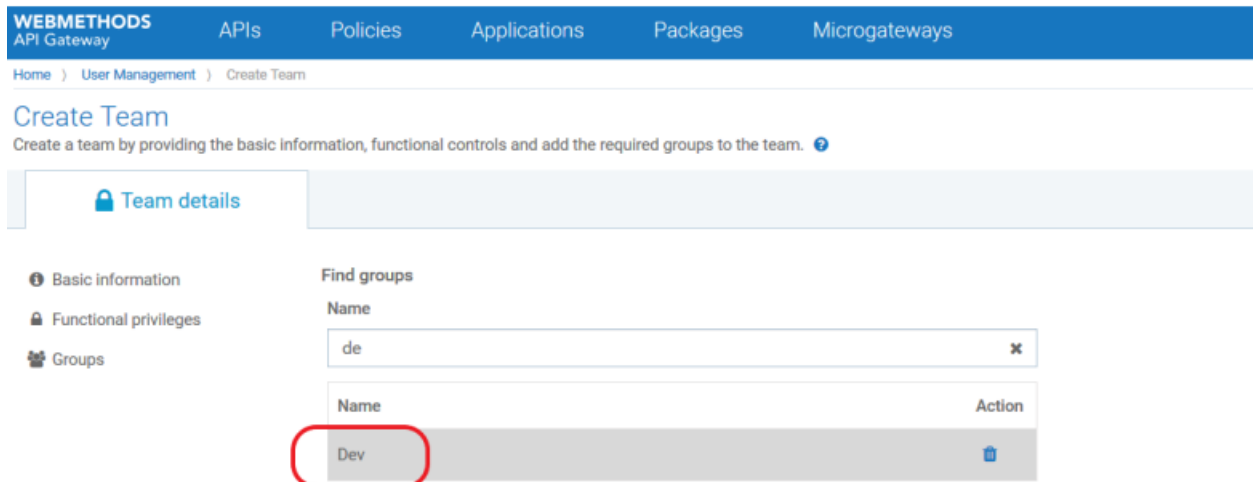
For this use case, you need to assign all privileges required to manage the APIs and applications assigned to the team. So, all functional privileges under the **APIs, Policies, and Applications** section are selected.



- Click **Continue to assign groups** >.

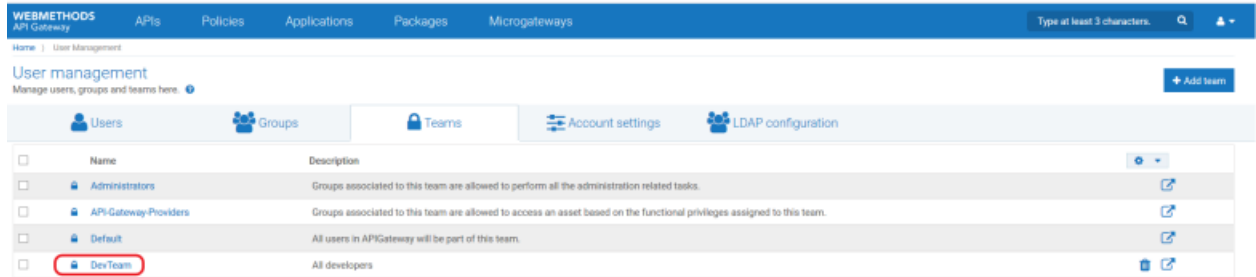
The **Find groups** section appears.

- In the **Name** field, provide the name of the user group that you want to add as team members. For this use case, the *Dev* user group that has all developers is selected.



- Click **Save**.

Team with specified details is created. As per the values provided in our use case, the *DevTeam* is created and appears in the list of teams.




After team creation, you can assign assets to the team.

Setting Password Restrictions

For security purposes, API Gateway places length and character restrictions on passwords for administrator and non-administrator users.

➤ To set password restrictions

1. Expand the menu options icon , in the title bar, and select **User management**.
2. Click **Account settings > Password restrictions**.
3. Provide the following information to set the required password restrictions.

Field	Description
Enable password change	Specifies whether users are allowed to change their passwords. This is selected by default.
Password enforcement mode	Specifies whether Administrator users are allowed to choose passwords that are not impacted by the password restriction settings. When this property is set to <code>strict</code> , API Gateway enforces the password restrictions. When set to <code>lax</code> , the password restrictions are not enforced.
Minimum password length	Specifies the minimum number of characters (alphabetic characters, digits, and special characters combined) the password must contain. The default value is 8.
Maximum password length	Specifies the maximum number of characters (alphabetic characters, digits, and special characters combined) the password must contain.


Field	Description
	Maximum number of characters that a password can have is 128. The default value is 64.
Minimum number of uppercase characters	Specifies the minimum number of uppercase alphabetic characters the password must contain. The default value is 0.
Minimum number of lowercase characters	Specifies the minimum number of lowercase alphabetic characters the password must contain. The default value is 0.
Minimum number of digits	Specifies the minimum number of digits the password must contain. The default value is 0.
Minimum number of special characters (neither alphabetic nor digits)	Specifies the minimum number of special characters, such as asterisk (*), period (.), and question mark (?) the password must contain. Note: The use of special characters is regulated by the following restrictions: <ul style="list-style-type: none">■ A password cannot begin with an asterisk (*).■ Passwords cannot contain quotation marks ("), backslashes (\), ampersands (&), or less-than signs (<). Use the <code>watt.server.illegalUserChars</code> configuration property to restrict the use of additional characters. The default value is 0.
Maximum number of identical characters in a row	Specifies the maximum number of identical characters in a row a password can contain. The default value is 3.
Number of old passwords to remember (per user)	Specifies the maximum number of previously set passwords that API Gateway saves for a user (excluding the current password). You cannot choose a password that matches any of the stored passwords. Maximum number of saved passwords is 12. The default value is 0.

4. Click **Save**.

Setting Password Expiry Restrictions

For security purposes, API Gateway allows administrators to set password expiration requirements on passwords for administrator and non-administrator users. An administrator user receives a reminder email to reset the password before certain number of days, as specified in the Password expiration settings page.

› To set password expiry restrictions

1. Expand the menu options icon , in the title bar, and select **User management**.
2. Click **Account settings > Password expiry settings**.
3. Provide the following information to set the required password expiry restrictions.

Field	Description
Enabled	<p>Specifies whether to enable the password expiry settings.</p> <p>This option is disabled by default. Select Enabled to enable the password expiry settings.</p>
Expiration interval (days)	<p>Specifies the number of days after which a password expires, if not changed.</p> <p>The value should be a non-zero integer.</p> <p>The default value is 90.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: Upon save, when this option is enabled, any password that is set before the expiration interval are considered expired and have to be reset. For example, if you changed your password 10 days ago and now, the Administrator changes the Expiration interval to 5 days, then your password has expired and needs to be reset.</p> </div>
Days prior to password expiry for email reminders	<p>Specifies the number of days prior to password expiry that API Gateway starts sending the reminder emails for password reset. The emails are sent daily until the user either updates the password or changes the expiration interval.</p> <p>The default value is 3.</p> <p>Set the value to 0 to prevent API Gateway from sending the reminder emails for soon to expire passwords.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note:</p> </div>


Field	Description
	API Gateway uses the SMTP server and port details specified in Integration Server in the Email Notification section on Resource Settings screen (Settings > Resources).
Expiration notice email addresses	Specifies the list of email addresses to which API Gateway sends an email notification informing that the user password is about to expire or has already expired. You can add multiple email addresses by clicking +Add .
Applies to users	Specifies the users to whom these settings apply. You can add multiple users by clicking +Add .

4. Click **Save**.

Configuring Account Locking Settings

For security purposes, it is important to lock a user account when the user fails to provide the correct password after a specified number of failed login attempts to API Gateway. A locked user account remains locked for a specific period of time, after which the account gets unlocked. API Gateway allows administrators to configure the account locking settings for administrator and non-administrator users. You can set the values for number of attempts by a user before locking the account and also the duration of the lock interval.

> To configure account locking settings

1. Expand the menu options icon , in the title bar, and select **User management**.
2. Click **Account settings > Account locking settings**.
3. Provide the following information to configure the required account locking settings.

Field	Description
Enabled	Specifies whether to enable the account locking settings. This option is disabled by default. Select Enabled to enable the account locking settings.
Maximum login attempts	Specifies the number of attempts in the specified time interval (minutes, hours, or days) to provide the correct password before locking the account. The default value is None.


Field	Description
Lockout duration	Specifies the duration (minutes, hours, or days) for which the account remains locked. The default value is None.
Apply account locking policy to	Specifies the list of users to whom the account locking settings apply. Specify one of the following: <ul style="list-style-type: none"> ■ All users. Indicates the account locking rules apply to all user accounts. ■ All users except predefined users. Indicates that account locking rules apply to all user accounts except the predefined user accounts (Administrator).


4. Click **Save**.

Unlocking User Accounts

API Gateway unlocks a user account after the specified locked duration. However, as an Administrator, you can manually unlock user accounts within the lockout duration configured.

> To unlock locked user accounts


1. Expand the menu options icon , in the title bar, and select **User management**.
2. Click **Account settings > Account locking settings**.

The Locked users section displays all the locked users.
3. From the list of locked user accounts, click  to unlock the user account.
4. Click **Save**.

Restricting User Accounts

API Gateway provides an option to restrict the user accounts, who are part only of the *Default* team and not any other team. You can enable this option to restrict those user accounts from logging into API Gateway.

> To restrict user accounts who belongs only to the default team

1. Expand the menu options icon , in the title bar, and select **User management**.
2. Click **Account settings > Account locking settings**.
3. In the **Login Restrictions** section, select **Restrict login for users who are not member of any team other than Default** if you want to restrict the user accounts associated to the *Default* team and not any other team.
4. Click **Save**.

Configuring API Gateway to Use LDAP

If your site uses Lightweight Directory Access Protocol (LDAP) for user and group information, you can configure API Gateway to obtain user and group information from the external directory.


LDAP protocols are designed to facilitate sharing information about resources on a network. Typically, they are used to store profile information (login ID, password, and so on.). You can also use them to store additional information. API Gateway uses LDAP for performing external authentication.

Using your existing LDAP information allows you to take advantage of a central repository of user and group information. System administrators can add and remove users from the central location. Users do not need to remember a separate password for webMethods applications; they can use the same user names and passwords that they use for other applications. Remember to use your LDAP tools to administer users or groups stored in an external directory.

To configure the server to use LDAP, you need to:

- Instruct API Gateway to use the LDAP protocol.
- Define one or more configured LDAP servers that API Gateway is to use for these users.
- If an LDAP provider is SSL-enabled, you can set the `watt.server.ssl.trustStoreAlias` property to point to the truststore alias that contains the certificates required to establish a secure connection with the LDAP server.

➤ To specify LDAP as the external provider

1. Expand the menu options icon , in the title bar, and select **User management**.
2. Click **LDAP configuration**.
3. Under Provider select **LDAP**.
4. Provide the following information:


Field	Description
Cache size (number of users)	<p>Specifies the maximum number of LDAP users API Gateway can keep in memory in the user cache.</p> <p>The default value is 10.</p> <p>Once the limit is reached, API Gateway selects users for removal from the cache based on how long they have been idle. As a result, activity can extend the time a user remains in the cache.</p>
Credential time-to-live (minutes)	<p>Specifies the number of minutes an LDAP user's credentials (userid and password) can remain in the credential cache before being purged.</p> <p>The default is 60 minutes.</p> <p>When a user first attempts to log in, API Gateway creates a user object and checks the user's credentials against the LDAP directory. API Gateway stores the credentials so that subsequent requests to authenticate are made against the cached credentials, not the LDAP directory.</p>

5. Click **Save**.

Managing LDAP Directories

You can manage the LDAP directories in the LDAP directories section. You can view all the LDAP directories configured listed in a table here with their directory URL details. You can create, update, delete and prioritize the LDAP directories here.

> To add an LDAP directory

1. Expand the menu options icon , in the title bar, and select **User management**.
2. Click **LDAP configuration**.
3. In the LDAP directories section, click **Add LDAP directory**.
4. Provide the following information to add an LDAP directory.

Field	Description
Directory URL	<p>Specifies the complete URL of the LDAP server.</p> <p>The URL has the format <i>protocol ://hostname :portnumber</i> where</p>

Field	Description
	<ul style="list-style-type: none"> <li data-bbox="548 258 1369 323">■ The <i>protocol</i> is LDAP for standard connections or LDAPS for secure connections <li data-bbox="548 350 1369 485">■ The <i>host</i> is the host name or IP address of the LDAP server. The <i>port</i> is the port on which the server is running. The port is optional. If omitted, the port defaults to 389 for LDAP, or 636 for LDAPS <p data-bbox="548 516 1369 617">For example, specifying the URL <code>ldaps://ldapserv1:700</code> would create a secure connection to the LDAP server running on the non-standard port 700 on the host called <code>ldapserv1</code>.</p> <p data-bbox="548 644 1369 919">If you specify <code>ldaps</code>, API Gateway attempts to make a secure connection to the directory server using an SSL socket. If the directory server is configured to use SSL, it has a server certificate in place to identify itself to clients. This certificate must be signed by an authority to prove its validity that is, the server certificate is signed by a CA). By default, API Gateway only trusts certificates signed by a signing authority whose CA certificate is in the API Gateway's trusted CAs directory.</p>
Principal	<p data-bbox="548 947 1369 1012">Specifies the user ID API Gateway should supply to connect to the LDAP server.</p> <p data-bbox="548 1043 1117 1073">For example, <code>o=webm.com</code> or <code>dc=webm,dc=com</code>.</p> <p data-bbox="548 1100 1369 1201">This user should not be the Administrator account, but a user that has permission to query groups and group membership. If your LDAP server allows anonymous access, leave this field blank.</p>
Credentials	<p data-bbox="548 1228 1369 1293">Specifies the password API Gateway should supply to connect to the LDAP server, that is, the Principal's password.</p>
Connection timeout (seconds)	<p data-bbox="548 1325 1369 1390">Specifies the number of seconds API Gateway waits while trying to connect to the LDAP server.</p> <p data-bbox="548 1417 1289 1482">After this time has passed, API Gateway tries for the next configured LDAP server on the list.</p> <p data-bbox="548 1509 862 1539">The default is 5 seconds.</p>
Minimum connection pool size	<p data-bbox="548 1570 1369 1671">Specifies the minimum number of connections allowed in the pool that API Gateway maintains for connecting to the LDAP server.</p> <p data-bbox="548 1698 1369 1864">When API Gateway starts, the connection pool initially contains this minimum number of connections. API Gateway adds connections to the pool as needed until it reaches the maximum allowed, which is specified in the Maximum Connection Pool field.</p>


Field	Description
	The default value is 0.
Maximum connection pool size	<p>Specifies the maximum number of connections allowed in the pool that API Gateway maintains for connecting to the LDAP server.</p> <p>When API Gateway starts, the connection pool initially contains the minimum number of connections as specified in the Minimum Connection Pool field. API Gateway adds connections to the pool as needed until it reaches the maximum allowed.</p> <p>The default value is 10.</p>
Directory name.	Specifies the directory name to be built on selecting any of the following criteria.
Synthesize DN	<p>Builds a distinguished name by adding a prefix and suffix to the user name. The Synthesize DN method can be faster than the Query DN method because it does not perform a query against the LDAP directory. However, if your LDAP system does not contain all users in a single flat structure, use the Query DN method instead.</p> <p>DN prefix</p> <p>A string that specifies the beginning of a DN you want to pass to the LDAP server.</p> <p>DN suffix</p> <p>A string that specifies the end of a DN you want to pass to the LDAP server.</p> <p>For example, if the prefix is <code>cn=</code> and the suffix is <code>,ou=Users</code> and a user logs in specifying <code>bob</code>, then API Gateway builds the DN <code>cn=bob,ou=Users</code> and sends it to the LDAP server for authentication.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: Be sure to specify all the characters required to form a proper DN. For instance, if you omit the comma from the suffix above, that is, you specify <code>ou=Users</code> instead of <code>,ou=Users</code>, API Gateway builds an invalid DN <code>cn=bobou=Users</code>.</p> </div>
Query DN	<p>Builds a query that searches a specified root directory for the user.</p> <p>Use this method instead of the Synthesize DN method if your LDAP directory has a complex structure.</p> <p>UID property</p>

Field	Description
	<p>A property that identifies an LDAP userid, such as "cn" or "uid".</p> <p>User root DN</p> <p>Provide the full distinguished name. For example, if you specify <code>ou=users,dc=webMethods,dc=com</code>, API Gateway issues a query that starts searching in the root directory <code>ou=users</code> for a common name that matches the name the user has logged in with.</p>
Default group	<p>Specifies the API Gateway group with which the user is associated.</p> <p>The user is allowed to access APIs that members of this API Gateway group can access. This access is controlled by the ACLs with which the group is associated.</p> <p>If you also specify a value in the Group member attribute field, the user has the same access as members of the API Gateway group and members of LDAP groups that have been mapped to an ACL.</p> <p>Note: If you do not want to select a default group, you can select <None> from the options provided.</p>
Group member attribute	<p>Specifies the name of the attribute in a group's directory entry that identifies each member of the group.</p> <p>This value is usually <code>member</code> or <code>uniqueMember</code>, but can vary depending on the schema of the LDAP directory.</p> <p>API Gateway uses this information during ACL checking to see if the user attempting to log in belongs to an LDAP group that has been mapped to an ACL.</p> <p>If no value is specified here, API Gateway does not check for membership in an LDAP group. As a result, the user's ability to access API Gateway services is controlled by the API Gateway group specified in the Default group field.</p>
Group ID property	<p>Specifies a property that identifies an LDAP group, such as CN.</p>
Group root DN	<p>Specifies the full distinguished name.</p> <p>For example, if you specify <code>ou=groups,webMethods,dc=com</code>, API Gateway issues a query that displays all the LDAP groups.</p> <p>Note: You must specify values in the Group ID property field and Group root DN fields.</p>

5. Click **Save**.

The LDAP directory is added and listed in a table under the LDAP directories section.

You can perform the following operations in the LDAP directories section where the configured LDAP directories are listed.

- You can update an LDAP directory by clicking on the **LDAP directory URL** field in the table, modify the details as required and save the changes.
- You can prioritize the LDAP directory as required by clicking in the **Prioritize** column for the corresponding LDAP directory.
- You can delete an LDAP directory by clicking the  icon in the **Delete** column for the corresponding LDAP directory.

Manage Your User Settings and Preferences

You can set the personal preferences and settings to control how you interact with API Gateway. You can specify custom values in the User settings page that are used instead of default values set by an Administrator for your user account.

The User settings page displays a summary of your current user preferences. In the User settings page, you can do the following:

- Change your account settings.
- Change your password.
- Change your display language on the user interface.
- View your roles and permissions.

Based on the logged in user, the User settings page is displayed in one of the following ways:

- As an LDAP user: Displays the roles and permissions that are assigned to your user account in the Roles and permissions section.
- As an API Gateway user: Displays the roles and permissions that are assigned to your user account in the Roles and permissions section.

Changing Your Account Settings

User account settings include the user name and email address. These settings are attributes of local users who are validated against credentials stored in API Gateway. If API Gateway uses an external authentication mechanism, such as an LDAP, you must change the equivalent settings in the external LDAP system.


When the administrator creates a user account and an email address for the user, that email address is set as the default email address for the account. If you have multiple email addresses, you can set up additional email addresses in the User settings page.

> **To change your user name and email address**

1. Expand the menu options icon , in the title bar, and select **Username**.

The User settings page appears with your user preferences.

2. Provide the following information in the  Account settings section.

Field	Description
First name	Type your first name. First name is case sensitive. A first name can contain letters, numbers, or a combination of all. You can also use special characters: . (dot), _ (underscore), and @ (at). Other special characters and spaces are not allowed.
Last name	Type your last name. Last name is case sensitive. A last name can contain letters, numbers, or a combination of all. You can also use special characters: . (dot), _ (underscore), and @ (at). Other special characters and spaces are not allowed.
Email addresses	Type a valid email address. If the email address is invalid, API Gateway prompts you with an error message. You can add multiple email addresses by clicking  .

3. Click **Save**.

The changes are applied immediately.


Changing Your Password

You can change the password that you use to log on to API Gateway.

> **To change your password**

1. Expand the menu options icon , in the title bar, and select **Username**.

The User settings page appears with your user preferences.

2. Provide the following information in the  Change password section.

Field	Description
Current password	Type the current password.
New password	Type a new password. Passwords are case sensitive. Your password must meet the complexity requirements configured in Integration Server (in the Integration Server Administrator, go to Security > User Management > Password Security Settings).
Confirm new password	Retype the new password to confirm.

3. Click **Save**.

The password is changed immediately.

Changing Your Display Language


You can select the language for the user interface of API Gateway.

API Gateway displays the user interface in English (en) by default. If you want API Gateway to use a different language other than English, you must install the intended language pack from the Software AG Installer.

> To change your display language

1. Expand the menu options icon , in the title bar, and select **Username**.

The User settings page appears with your user preferences.

2. In the  Display settings section, select the language you would like to use for the user interface.

Note:


If the language you want to use is not available by default in the **Language** list box, you need to install the intended language pack from the Software AG Installer.


3. Click **Save**.
4. Log out and log on to API Gateway for the change to take effect.

Viewing Your Roles and Permissions

The User settings page displays a list of all the roles and permissions that are assigned to your user account. The roles and permissions assigned to your user account are defined through user groups and corresponding teams (see [“Manage Users, Groups, and Teams”](#) on page 236 for more information).

> To view your roles and permissions

1. Expand the menu options icon , in the title bar.
2. Select **Username** from the menu options.

In the  Roles and permissions section, you can view the list of roles and permissions that are assigned to you.

4 APIs

■ Overview	263
■ Creating an API by Importing an API from a File	265
■ Creating an API by Importing an API from a URL	267
■ Creating an API from Scratch	268
■ API Mashups	286
■ Viewing API List and API Details	296
■ Filtering APIs	302
■ Activating an API	302
■ Deactivating an API	305
■ Publishing APIs	305
■ Unpublishing APIs	311
■ Modifying API Details	315
■ Updating APIs	316
■ API Mocking	319
■ Attaching Documents to an API	323
■ SOAP to REST Transformation	325
■ CentraSite Provided APIs	333
■ Versioning APIs	334
■ API Scopes	335
■ Exposing a SOAP API to Applications	344
■ API Grouping	345
■ API Tagging	346

- Exporting APIs 347
- Exporting Specifications 349
- Deleting APIs 350
- Example: Managing an API 351

Overview

API Gateway provides the ability to view, create, and manage APIs, and publish the APIs to API Portal for consumption. API administrators and users with the appropriate functional privileges can use API Gateway to create and manage APIs, and publish the APIs to API Portal or service registries from where they can be consumed.

API Gateway supports the following API types:

- **Representational State Transfer (REST)** defines a set of architectural principles that allow accessing and manipulating resources by using capabilities already built into HTTP, including uniform and predefined set of stateless operations, resources that are accessible using URIs, and resources that are represented by media types. This framework provides RESTful APIs based on REST architecture. There are multiple specification formats for REST APIs. In API Gateway, you would create a REST API using one of the supported formats: RAML, Swagger 2.0, OpenAPI 3.0 Specification.
 - The RAML specification is a YAML-based language for the definition of HTTP-based APIs that embody most or all of the principles of REST. The RAML specification provides all the information necessary to describe RESTful or practically-RESTful APIs.
 - The Swagger 2.0 specification defines a standard, language-agnostic interface to RESTful APIs. The Swagger specification provides a complete framework implementation for describing, producing, consuming, and visualizing RESTful APIs.
 - The OpenAPI 3.0 Specification (OAS) has a much more modular and reusable approach for describing and documenting RESTful APIs. OAS enables more power and versatility when it comes to describing the request and response messages, as well as providing details on the common components like the schemas and security definitions
- **Simple Object Access Protocol (SOAP)** defines a communication method for XML-based message exchange over different transport protocols, such as HTTP and SMTP. This framework provides SOAP APIs based on Web Services Description Language (WSDL).
- **Open Data Protocol (OData)** defines a set of best practices for the creation and consumption of RESTful APIs. It provides a uniform way to describe both data and the data model. The OData framework provides interoperable OData APIs (with a RESTful interface) based on OData standards.
- **WebSocket** protocol defines two-way (full-duplex) communications between the client and the server, over a single Transmission Control Protocol (TCP) socket. The WebSocket protocol facilitates real-time data transfer from and to the server. This framework provides WebSocket APIs (with a RESTful interface) based on W3C standards.

Asynchronous APIs

The synchronous and asynchronous nature of an API is a function of the time frame from the request to the return of data. In the case of synchronous APIs, the expectation is that there would be an immediate return of data, read from a database, from the internet, from the disk, or any other I/O source of data. You would use synchronous APIs where data or service availability,

resources and connectivity are high and low latency is a requirement. The application requests data and waits for it until a value is returned.

In the case of asynchronous APIs, the availability of a resource, service or data store may not be immediate. An asynchronous API returns a response acknowledging the receipt of the request and it continues with the processing of the data till it is done, and returns a response to the client only when the processing of the data is completed. You would use asynchronous APIs where data or service availability and connectivity are low or over-saturated with demand. These APIs may use the callback functionality to send the callback request to the requester when the requested resource is ready.

Few APIs may take a lot of time to complete their processes, for example, processes such as purging or archiving of events, and bulk processing operations. In such a scenario, a time-out may occur for these API invocations as it takes longer time in the synchronous way where there is a wait period for the return of the data.

Example: Let us consider an example of purging logs.

In the synchronous way, the client application sends a request to the native API to purge a set of logs with a filter specified. The native API in turn sends out a response with an acceptance along with a Job ID. The client uses this job ID to send out requests to the native API to check whether the job is completed. In this case the client may have to send out multiple requests to check whether the job is done.

To avoid the hassle of multiple calls to the native API and waiting for the job to get done, you can implement an API to behave asynchronously to avoid multiple checks. You can implement a REST API with a callback option that can be used to call back the requestor when the job is done. In this scenario, when the client application sends out a request to the native API to purge a set of logs with a filter specified, the native API in turn sends out a response with an acknowledgement of having received the request and makes a note of the callback request URL that it receives in the request. Now, the client does not have to send out multiple requests to the native API to check whether the job is done. Instead once the job is done, the native API uses the callback URL details to send out a response to the requestor regarding the status of the job being done.

API Gateway provides asynchronous form of API support for REST APIs. API Gateway provides the capability of defining the callback component with the supported method parameters while creating a REST API. For details on creating an API with the callback definition, see [“Creating a REST API” on page 273](#). In addition you can configure API Gateway to accept the requests from the client that contain the callback request URL and wrap it with its own URL before routing it to the native API. This lets API Gateway track the requests that the client sends to the native API and the responses that are sent by the native API to the client. For details on how to configure the callback processor settings to enable processing the callback requests, see [“Configuring API Callback Processor Settings” on page 81](#). When a client sends a request with the callback request URL to the native API, API Gateway identifies the callback request URL in the incoming request, depending on the configured callback processor settings wraps the request coming from the client with its own URL and routes it to the native API. When the requested resource is ready, the native API sends a request to the callback request URL it has received in the request it has received from API Gateway. API Gateway then routes this request to the client. You can configure API Gateway to enforce any of the response processing policies that suits your needs on the immediate responses as well as the callback requests being sent from the native API to the client.

The callback requests-related event types can be distinguished by a new field with the value set to true and displayed in the dashboard in the transaction event type. For a normal request this field is set as false. The following are the field names that are displayed for various configured destinations:

- For API Gateway destination the field name is `callbackRequest`, which is set to true.
- For Elasticsearch destination the field name is `isCallbackRequest`, which is set to true.
- For all other destinations, API Portal, Audit Log, CentraSite, Email, JDBC, and Local log, the field name is `isCallbackRequest`, which is wrapped under the `customFields` column.

You can create and manage APIs from the Manage APIs page. The page lists all the APIs, their description, and version number. You can create an API, delete an API, view API details, activate or deactivate an API, publish or unpublish an API, and view API analytics from this page.

You can create an API in one of the following ways:

- Create an API by importing a definition for an existing API (for example, in Swagger or RAML format) using an API importer
- Create an API from scratch and set its attributes manually

An API importer generates an API from a URL or an input file in one of the supported formats. For example, the RAML importer installed with API Gateway reads a RAML file and generates a REST API that the RAML definition describes. The importer also uploads the RAML file to the API Gateway repository and links the file to the REST API.

The table lists the API types and the file formats required as input to create an API using an importer.

API type	File format
REST	RESTful API Modeling Language (RAML)
	Yet Another Markup Language (YAML)
	JavaScript Object Notation (JSON)
	OpenAPI Specification (OAS)
SOAP	Web Service Definition Language (WSDL)
OData	Entity Data Model (EDMX)

Creating an API by Importing an API from a File

You must have the API Gateway's manage APIs or activate/deactivate APIs functional privilege assigned to perform this task.

➤ **To create an API by importing an API from a file**

1. Click **APIs** in the title navigation bar.

A list of available APIs appears.

2. Click **Create API**.
3. Select **Import API from file**.
4. Click **Browse** to select a file.
5. Select the required file and click **Open**.

The Swagger parser is a self-contained file with no external references and can be uploaded as is. If the RESTful API Modeling Language (RAML) file to be imported contains external references, the entire set of files must be uploaded as a ZIP file with a structure as referenced by the RAML file.

Note:


Importing an API fails for an invalid WSDL file.

6. Type a name for the API name in the **Name** field.
 - If you provide an API name, this overwrites the API name mentioned in the uploaded file and the API is displayed with the name provided.
 - If you do not provide an API name, the API name mentioned in the uploaded file is picked up and the API is displayed with that name.
 - If you do not provide an API name and the uploaded file does not have an API name mentioned, then the API is displayed as Untitled.
7. Select the required type.

The available types are **RAML**, **Swagger**, and Web Services Description Language **WSDL**.

8. Provide a version for the API in the **Version** field.
9. Select the team to which the API must be assigned in the **Team** field.

This field appears only when the Team feature is enabled. It displays only the teams that you are a part of. If you have the User management functional privilege, all teams are displayed.

You can select more than one team. To remove a team, click the  icon next to the team to be removed.

10. Click **Create**.

Note:

To avoid encountering errors while parsing large responses from the native service, you have to change the `enablesoapValidation` property by commenting out the `<parameter`

```
name="enableSoapValidation">true</parameter> in Sagroot_Installdir\IntegrationServer\instances\default\config\wss\axis2.xml and restart the server for the change to take effect.
```

Creating an API by Importing an API from a URL

You must have the API Gateway's manage APIs or activate/deactivate APIs functional privilege assigned to perform this task.

➤ To create an API by importing an API from a URL

1. Click **APIs** in the title navigation bar.

A list of available APIs appears.

2. Click **Create API**.

3. Select **Importing API from URL**.

4. Type the URL from where the API is to be imported.

5. Select **Protected** to make the API a protected API.

6. Type a name for the API name in the **Name** field.

- If you provide an API name, this overwrites the API name mentioned in the uploaded file and the API is displayed with the name provided.
- If you do not provide an API name, the API name mentioned in the uploaded file is picked up and the API is displayed with that name.
- If you do not provide an API name and the uploaded file does not have an API name mentioned, then the API is displayed as Untitled.

7. Provide a description for the API in the **Description** field.


8. Select the required type.

The available types are **RAML**, **Swagger**, **WSDL**, and **OData**.

9. Provide a version for the API in the **Version** field.

10. Select the team to which the API must be assigned in the **Team** field.

This field appears only when the Team feature is enabled. It displays only the teams that you are a part of. If you have the User management functional privilege, all teams are displayed.

You can select more than one team. To remove a team, click the  icon next to the team to be removed.

11. Click **Create**.

Note:

- Importing an API fails for an invalid WSDL file.
- Creating an API by importing swagger files from an HTTPS URL that is using self-signed certificates may fail. To workaroud this, you can set the system environment variable as: `export TRUST_ALL=true`, so that the invalid certificates are ignored. Be aware that setting this variable makes the swagger-parser ignore all invalid certificates too. So this workaroud has to be used with caution.
- To avoid encountering errors while parsing large responses from the native service, you have to change the `enableSoapValidation` property by commenting out the `<parameter name="enableSoapValidation">true</parameter>` in `Sagroot_Installdir\IntegrationServer\instances\default\config\wss\axis2.xml` and restart the server for the change to take effect.

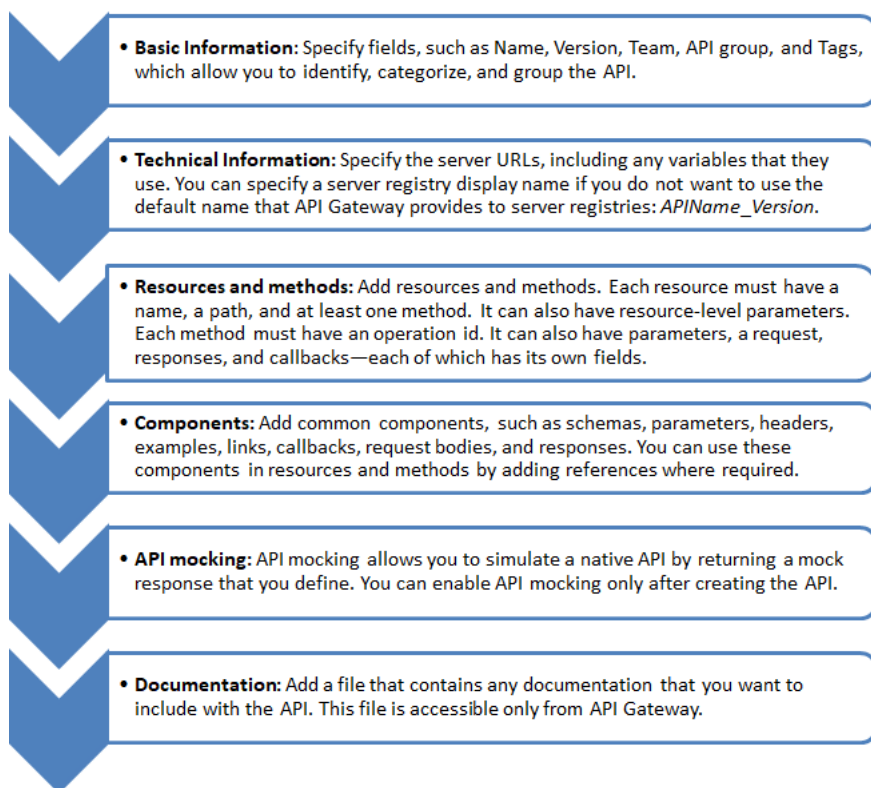
Creating an API from Scratch

You can create the following APIs from scratch, meaning that you create the asset and set its attributes manually:

- REST
- WebSocket

Overview of Creating a REST API from Scratch

The **Create REST API** wizard breaks down the task of creating a REST API from scratch into logical steps. The following figure illustrates the different pages of the wizard.



Basic Information

The **Basic Information** page includes fields that allow you to identify, categorize, and group an API.

Technical Information

The **Technical Information** page includes fields that allow you to define one or more server URLs for the API. You can also define and include variables in the URLs.

You can also specify parameters for data that must be included in every request to the API. For example, if you want a specific query parameter to be included in every request, you can add a parameter of the type **Query** and specify the value that it must include.

Resources and methods

The **Resources and methods** page includes fields that allow you to define the API resources and methods, including callback methods. In this page, you can add all the resources and their methods that are exposed by the API.

At the resource level, you add a resource by defining the following properties: name, path, and supported methods. You can additionally add parameters for data that must be included in every request to that resource. For example, if the methods in a resource are invoked using URLs that have a query string; you can add a query string parameter that captures the queries sent by the clients.

At the method level, you identify a method by adding an operation id. In addition, you can add tags that help you to categorize and search for similar methods. You can also add parameters at the method level. Similar to the parameters at the API and resource levels, method parameters enable you to capture and process the data that is sent in a particular request. In the case of method parameters, the data in the request for that method is captured and processed.

Method Requests

In the request section of a method, you can define the schema for requests that contain a JSON or XML payload. As a method can support multiple content types, you need to add a content type and then define the schema supported by that content type.

You can enter a schema or select an existing schema or global schema that you have previously added on the **Components** page, **Schemas** section. You can also add a sample for the schema that you have added or selected. These samples can be used for API mocking. They can also be used by end users to get a better understanding of the API.

Method Responses

You can define responses for different HTTP status codes. API Gateway gives you the flexibility to define responses for a status codes series (such as the 2XX series or the 4XX series) or for specific response codes, such as 201 or 400.

Note:

If you have defined the response for a series and specific numbers in that series, the more specific one is used. Example: If you have added an entry for 2XX and 201, a response with the HTTP status code 200 will be the same as 2XX. However, a response with the HTTP status code 201 will pick the one that is defined for 201.

For each status code in a method response, you define the following:

- **Response body:** you define the response body using the following fields:
 - Content Type: You can select from any of the content types.
 - Schema: You can define a schema if the response contains JSON or XML data.
 - Sample: The samples are used for API mocking. They can also be used by end users to get a better understanding of the API.
- **Header parameter:** You can add a parameter to capture and process a header in the response sent by the native API.
- **Links:** Links allow the developer of the native API to define the relationship and traversal mechanism between a response and other operations. You can include links to other methods that are related to the response. This enables an API client to dynamically navigate the methods that are exposed by the API. For example, a method that returns the temperature in Fahrenheit for a given place may also include links to methods that return: a) the temperature in Centigrade; and b) the temperature of the place on a given day of the year.

Note:

You can define the complete response, or any part of it (response body schema, header parameter, or link), in the **Components** page; and reuse it wherever required by giving a reference.

Method Callbacks

A callback is an asynchronous API request that originates from the API server and is sent to the client in response to an earlier request sent by that client. APIs can use callbacks to signal an event of interest and share data related to that event. API clients that are interested in an event or data related to that event, include a callback URL in the request they send to the API. For more information about Asynchronous APIs, see [“Asynchronous APIs” on page 263](#).

To enable API Gateway to process callback messages, you must configure the Callback processor settings, as explained in [“Configuring API Callback Processor Settings” on page 81](#).

If your API supports callbacks, you can use API Gateway to process the initial requests, the callback URLs sent by clients, and the response sent by the API—including the callback messages. Clients can provide the callback URL to API Gateway in any of the following ways:

- Request header
- Query parameter
- Request body (if the response body has JSON or XML content)

You must define the relevant parameter to capture the callback URL to process it. API Gateway can wrap the client callback URLs with its own URL to process these requests if the callback URL path defined in the following formats. Otherwise, API Gateway sends the requests received from client as it receives it.

Format	Description
<code>{request.query.<i>param-name</i>}</code>	Where <i>param-name</i> is the name of the query parameter that contains the callback URL.
<code>{request.header.<i>header-name</i>}</code>	Where <i>header-name</i> is the name of the header that contains the callback URL.
<code>{request.body#/<i>field-name</i>}</code>	Where <i>field-name</i> is a field in the request body. If the field is an array, use the syntax <code>{request.body#/<i>field-name/arrayIndex</i>}</code> , where <i>arrayIndex</i> is the index of the callback URL in the array.
<code>{response.header.<i>header-name</i>}</code> and <code>{response.headers.<i>header-name</i>}</code>	Where <i>header-name</i> is any of the valid header.
<code>{request.query.<i>param-name</i>}</code>	Where <i>param-name</i> is the name of the query parameter that contains the callback URL.
<code>{response.payload.jsonPath[<i>queryValue</i>]}</code>	Where <i>queryValue</i> is a valid JSON path expression.
<code>{response.payload.xpath[<i>queryValue</i>]}</code>	Where <i>queryValue</i> is a valid XPath path expression.

If you have enabled API Gateway to process callback messages, API Gateway wraps the callback URL provided by the client and sends an API Gateway URL to the native API. When the native API invokes the same callback URL, API Gateway processes the response and applies the policies that you have defined.

API Gateway can apply the following policies on the callback messages:

- Invoke webMethods IS
- Response Transformation
- Validate API Specification
- Data Masking
- Log Invocation

Note:

These policies are applied to the immediate responses of an API request and to all its callback requests. These policies are enforced against callback request payloads.

API mocking

API mocking allows you to simulate a native API that is not available. The mock response that you define is returned to the client that invokes the API, if the native API is not available. API mocking is not available while you are creating an API. To use API mocking, you must edit the API after creating it and enable API mocking.

You must enable API mocking for an API after creating the API. For more information about API mocking, see [“API Mocking” on page 319](#).

Components

The **Components** page allows you to add reusable elements that you can use in other pages of the wizard. You can reference these global elements using the `$ref` variable. You can add the following global elements:

- **Schemas:** The schema specified here can be reused in the resource and method specifications across multiple methods and resources.
- **Parameters:** You can define parameters that can be used as API, resource, and method parameters.
- **Headers:** You can define parameters that can be reused as header parameters at the API, method, and response levels.
- **Examples:** You can add examples that can be reused as samples across operations in the API.
- **Links:** You can define links that can be reused in responses. For more information about links, see *Links* within *Method Responses* above.

- **Callbacks:** You can define callback methods in this page and include them in the callback section of the methods that use it. For more information about callbacks, see “[Method Callbacks](#)” on page 271.
- **Request Bodies:** You can define request bodies in this page and reuse them in methods. A request body includes the content type, a schema, and a sample.
- **Responses:** You can define responses in this page and reuse them in methods. A response includes the content type, a schema, and a sample. It can also include header parameters and links.

Documentation

In the view mode, the **Documentation** page provides the following links:

- Links to the Swagger, RAML, and OpenAPI versions of the API on the Integration Server.

Note:

If Cross-Site Request Forgery (CSRF) token is enabled on the Integration Server, the links to three types of APIs will not work. You must configure Integration Server to allow these links to work.

- Links to download the API in the three different formats: Swagger, RAML, and OpenAPI.

In the edit mode, the **Documentation** page allows you to add a file that contains any documentation that you want to include with the API. This file is accessible only from API Gateway.

Creating a REST API

You must have the API Gateway's manage APIs or activate/deactivate APIs functional privilege assigned to perform this task.


You can create a REST API from scratch by providing the basic information, technical information, and defining the resources and methods as required.

➤ To create a REST API from scratch

1. Click **APIs** in the title navigation bar.
A list of all existing APIs appears.
2. Click **Create API**.
3. Select **Create from scratch**.
4. Select **REST**.
5. Click **Create**.

The **Basic information** page of the **Create REST API** wizard appears.

6. Provide the following information in the Basic information section:

Field	Description
Name	Name of the API.
Version	Version of the API being created.
Team	Team to which the API must be assigned. You can select more than one team. To remove a team, click the  icon next to the team to be removed.
Maturity state	<p>Maturity state of the API.</p> <p>Available values are: Beta, Deprecated, Experimental, Production, Test.</p> <p>The available values depend on the Maturity states configured in the <code>apiMaturityStatePossibleValues</code> property under Administration > Extended settings section.</p>
API grouping	<p>Group under which the API would be categorized.</p> <p>Available values are: Finance Banking and Insurance, Sales and Ordering, Search, and Transportation and Warehousing.</p> <p>The available values depend on the groups configured in the <code>apiGroupingPossibleValues</code> property under Administration > Extended settings section.</p>
Tags	Keywords for categorizing, identifying, and organizing APIs. You select from the list of existing tags or create new tags.
Description	Description of the API.

7. Click **Continue to provide technical information for this API >**.

Note:

Click **Save** to save the API at this stage and close the **Create REST API** wizard.

8. Enter the details of the servers that serve the API in the **Add server details** section.
- a. Click **Add server** and provide a **Server URL** and **Description**.

You can include variables in the server URL by enclosing them in curly braces. These variables are added to the list of variables. However, you have to edit these variables to add a default value, and optionally one or more values and a description.

b. Click **Add variables** and provide the following values:

- Name
- Description
- Default
- Value

Note:
Click **+** to add the value that you have entered.

c. Click **Add** to add the variable.

9. Click **Add Parameter** and provide the following information to add the API-level parameters.

Field	Description
Name	Name of the parameter.
Description	Description of the parameter.
Type	Specifies the parameter type. Available values: Query-string, Header, Cookie.
Data type	Specifies the data type. Available values: String, Date, Date time, Integer, Double, Boolean, File.
Required	Specifies the parameter is required if selected.
Repeat	Select if the input parameter is of type array.
Value	Specifies the possible values.

Note:
You need to define parameters only for data that you want API Gateway to process.

10. Type a **Service registry display name**.

By default, the API will be displayed in service registries with the name: *APIName_Version*. If you want the API to be displayed in the service registries with a different name, you can type the name here.

11. Click **Continue to provide Resource and methods for this API>**.

Note:
Click **Save** to save the API at this stage and close the **Create REST API** wizard.

12. Add resources to the API using the Resources and methods page:

- a. Click **Add Resources** and provide the following information:

Field	Description
Resource name	Name of the resource. This is the display name of the resource and resource path is used for execution.
Resource path	Specifies the path of the resource. The resource path should contain a "/".
Description	Description of the resource.
Supported methods	Select the methods that are supported by the API: GET, HEAD, POST, PUT, DELETE, PATCH.

- b. Click **Add**.

The resource is added. You can multiple resources, if required.

- c. Add **Tags**.

- d. Click **Add Resource Parameter** and provide the following information:

Field	Description
Name	Name of the parameter.
Reference	If you want to reuse a global parameter defined on the Components page, select the parameter from the drop-down list.
Description	Brief description of the parameter.
Type	Specifies the parameter type. Available values: Path, Header, Query-string, Cookie.
Data type	Specifies the data type. Available values: String, Date, Date time, Integer, Double, Boolean, File.
Required	Specifies the parameter is required if selected.
Repeat	Select if the input parameter is of type array.
Value	Specifies the possible values for the parameter.

- e. Click **+ Add** to add the resource parameter.

13. For each supported method that you have added for a resource, enter the following information:

- a. Common information:

Field	Description
Description	Type a description for the operation.
OperationId	Type an operation Id.
Tags	Type or select the keywords that you want to add to the operation.

- b. Method parameters

Field	Description
Name	Name of the parameter.
Reference	If you want to reuse a global parameter defined on the Components page, select the parameter from the drop-down list.
Description	Brief description of the parameter.
Type	Specifies the parameter type. Available values: Query-string, Header, Cookie.
Data type	Specifies the data type. Available values: String, Date, Date time, Integer, Double, Boolean, File.
Required	Specifies the parameter is required, if selected.
Repeat	Select if the input parameter is of type array.
Value	Specifies the possible values for the parameter.

- c. Requests: You can select an existing global request defined on the **Components** page or specify a new request. To enter a new request, select **New request** and provide the following information:

To add a new request that has to be processed, click **Add Request +** and provide the following information:

- **Content type:** Select one and click **Add**.
- **Schema:** Type a schema in the text box or select an existing schema from the **Select a Schema** list. You can also click **Add global schema** and create a new global schema

on the **Components** page. After creating the global schema you can select it from the **Select a Schema** list.

- **Sample:** Type a sample for selected schema. This sample can be used for API mocking, if required.

To use an existing global request to process a request, select **Global request** and provide the following information:

- **Name:**
- **Reference:** select one and click **Add**.

- d. **Responses:** First, add a status code using the **Status Code** drop-down list. Next, click on the status code to select it. For the selected status code, you can select an existing global response defined on the **Components** page or enter a new response. To enter a new response, select **New response** and define the response by adding a schema and a sample for the response body, header parameters, and links.

Note:

You can also define the response for an HTTP status code series, such as 2** or 4**.

To define a new response for the selected status code, click **Add response +** and provide the following information:

- **Content type:** select one and click **Add**.
- **Schema:** Type a schema in the text box or select an existing schema from the **Select a Schema** list. You can also click **Add global schema** and create a new global schema on the **Components** page. After creating the global schema you can select it from the **Select a Schema** list.
- **Sample:** Type a sample for selected schema. This sample can be used for API mocking, if required.

To use an existing global response, select **Global response** and provide the following information:

- **Name:** Name of the response.
- **Reference:** select one and click **Add**.

To add a header parameter, click **+ Add method parameter** and enter the following information to add a method parameter:

Field	Description
Name	Name of the parameter.
Reference	If you want to reuse a global parameter defined on the Components page, select the parameter from the drop-down list.

Field	Description
Description	Brief description of the parameter.
Type	Specifies the parameter type. Available values: Header .
Data type	Specifies the data type. Available values: String, Date, Date time, Integer, Double, Boolean, File .
Required	Specifies the parameter is required if selected.
Repeat	Select if the input parameter is of type array.
Value	Specifies the possible values for the parameter.

Click **+** in the **Value** text box to add a value to the list, and click **Add** to add the header.

To add a link, click **+ Add links** and enter the following information to add a link:

- **Name:** Name of the link.
- **Description:** Description for the link.
- **Link:** You can add a new link or select an existing global link that is defined on the **Components** page.

To add a new link, select **New link** and enter the following information:

- **Type:** Select **OperationId** for local operations only. **OperationRef** can be used for both local and external operations.
- **Value:** If **Type** is **OperationRef**, provide a reference to the target operation using the JSON Reference syntax (using by the **\$ref** keyword); and if the **Type** is **OperationId** provide the **OperationId** of the target operation.
- **Parameters:** Specify the parameters of the target operation that are required to follow the link. Enter a **Name** and **Value**, and click **Add**.
- **Request body:** Enter a request body only if the target operation has a body. Define the contents of the body of the target operation.

To include an existing global link, select **Global link** and then select an existing global link from the **Reference** drop-down list.

- e. **Callbacks:** You can add the callbacks that are supported by the method. You can add new callbacks and select existing global callbacks.

Note:

For more information about using callbacks to develop asynchronous APIs, see *Asynchronous APIs* in “[Overview](#)” on page 263. For more information on defining and using callbacks in API Gateway, see “[Overview of Creating a REST API from Scratch](#)” on page 268.

To specify a new callback, click **+ Add callbacks** and define the callback:

- **Name:** A name for the callback resource.
- Click **+ Add resources** and provide details of the API that serves as the callback API.

Note:

The user interface and procedure for defining a callback is similar to defining a resource and methods within the resource.

To include a global callback defined on the **Components** page, provide the following information:

- **Name:** Name of the callback resource.
- **Reference:** If you want to reuse a global callback defined on the **Components** page, select the callback from the drop-down list and click **Add**.

14. Click **Continue to provide Mocking information for this API>**.

Note:

Click **Save** to save the API at this stage and close the **Create REST API** wizard.

The API mocking page appears. API mocking is not enabled for a new API: You must edit the API and enable API mocking after creating the API.

15. Click **Continue to define API components for this API>**.

Alternatively, you can click **Components**.

Note:

Click **Save** to save the API at this stage and close the **Create REST API** wizard.

16. Define the reusable elements that you want to reuse in other pages of the Create REST API wizard.

An API may have several elements that are common across resources and methods, such as schemas for response bodies. You can place such common elements in the **Components** section and reference them using the *\$ref* alias.

- a. In the **Schemas** section, click **+ Add schema** and provide the following information:

Field	Description
Name	Name of the schema.

Field	Description
Schema type	Specifies the schema type. Available types: Inline schema and Upload schema .
Value	Specifies the value for the schema type selected. For an inline schema, type the request and response values. For an external schema, click Browse to upload a schema.

Click **+ Add** to add the schema component.

- b. In the **Parameters** section, click **+ Add parameter** and provide the following information:

Field	Description
Name	Name of the parameter.
Description	Description of the parameter.
Type	Specifies the parameter type. Available values: Path , Query-string , Header , and Cookie .
Data type	Specifies the data type. Available values: String , Date , Date time , Integer , Double , Boolean , and File .
Required	Specifies the parameter is required if selected.
Repeat	Select if the input parameter is of type array.
Value	Specifies the possible values for the parameter.

Click **+ Add** to add the parameter component.

- c. In the **Headers** section, click **+ Add header** and provide the following information:

Field	Description
Name	Name of the header.
Description	Description of the header.
Type	Specifies the header type. This is fixed as Header for headers.
Data type	Specifies the data type.

Field	Description
	Available values: String , Date , Date time , Integer , Double , Boolean , and File .
Required	Specifies the header is required if selected.
Repeat	Select if the input parameter is of type array.
Value	Specifies the possible values for the header.

Click **+ Add** to add the header component.

- d. In the **Examples** section, click **+ Add examples** and provide the following information:

Field	Description
Name	Name of the example.
Summary	Description of the example.
Value	The content of the example.

Click **+ Add** to add the example component.

- e. In the **Links** section, click **+ Add links** and provide the following information:

Field	Description
Name	Name of the link.
Description	Description of the link.
Type	Specifies the link type: OperationId or OperationRef .
Value	Path to the target operation or a reference to the target operation.
Parameter name	Name of the parameter that needs to be passed as a parameter to the target operation.
Parameter value	Value for the parameter. Click + Add to add the parameter. You can additional parameters if required.
Request body	Payload of the request sent to the target operation.

Click **Add** to add the link component.

- f. In the **Callbacks** section, click **+ Add callback** and provide the following information:
- Type a name for the callback.

- b. Click **+ Add resources**.
 - c. Type the **Callback path**.
 - d. Select the supported methods.
 - e. Click **Add**.
 - f. For each method that you have just added, complete the next two steps.
 - g. Click **+ Add Resource Parameter** and add the required resource parameters. The procedure for adding resource parameters is given in Step 11d.
 - h. Define the selected methods. The procedure for defining methods is given in Step 12.
- g. In the **Request Bodies** section, click **+ Add request** and provide the following information:

Field	Description
Name	Name of the request.
Content type	Select a content type from the list.
Schema	You can also select an existing schema.
Sample	Type a sample of the schema.

Click **Add** to add the request component.

- h. In the **Responses** section, click **+ Add Response** and provide the following information:

Field	Description
Name	Name of the response.
Content type	Click Add .
Schema	You can also select an existing schema.
Sample	Type a sample of the schema.
Header Parameter	Click + Add Header Parameter and provide the required information. Then, click + Add to add the header parameter.
Links	Click + Add Links and provide the required information. Then, click Add to add the link.

Click **Add** to add the response component.

17. Click **Continue to provide API documents for this API>**.

Note:

Click **Save** to save the API at this stage and close the **Create REST API** wizard.

The Documentation page appears.

18. Type a display name and click **Browse** to select a file.
19. Click **+ Add** to upload the file and add a new row.
20. Click **Save** to save your changes and create the API.

Creating a WebSocket API

You must have the API Gateway's manage APIs or activate/deactivate APIs functional privilege assigned to perform this task.


You need the WebSocket port to access the WebSocket API. Assigning global and API-specific policies is similar to assigning policies to REST or SOAP APIs.

Note:

You can not apply global policies and policy templates to a WebSocket API.

> To create a WebSocket API from scratch

1. Click **APIs** in the title navigation bar.
2. Click **Create API**.
3. Select **Create from scratch**.
4. Select **WebSocket**.
5. Click **Create**.
6. Provide the following information in the Basic information section:

Field	Description
Name	Name of the API.
Version	Version of the API.
Team	Team to which the API must be assigned. You can select more than one team. To remove a team, click the  icon next to the team to be removed.
Description	Description of the API.

7. Click **Continue to provide technical information for this API**>.

Alternatively, you can click **Technical information** to go to the Technical information section.


Click **Save** to save the API at this stage and provide the technical information for the API at a later time.

8. Provide the following information in the Technical information section:

- a. Type the WS URL in the **WS Url** field.

The format used is `ws://hostname:port/path`.

- b. Click **+ Add parameter** and provide the following information:

Field	Description
Name	Name of the parameter.
Description	Description of the parameter.
Type	Specifies the parameter type. Available values are - Query-string, Header .
Data type	Specifies the data type. Available values - String, Date, Date time, Integer, Double, Boolean .
Required	Select this to specify that the parameter is required.
Array	Select this to specify that the array is required.
Value	Type the required value and click + to add the value. Click  to include multiple values.

- c. Click **+ Add message** and provide the following information.

Field	Description
Origin	Specifies the origin of the message. Available values: Server, Client .
Type	Specifies the message type. Available types: Text, Binary

Field	Description
Sample message payload	Provide the sample message payload.
Message description	Provide the message description.

Click  to include multiple messages.

9. Click **Save**.

API Mashups

Overview

Servers that provide an API may expose a vast set of functionality. However, each individual service in the API usually provides a very specific functionality. While this is usually effective, sometimes it is useful or required to consolidate a few services and expose them as a single service. In other situations, you might want to extend a service with the functionality provided by an external API. API mashups address these requirements for grouping services and exposing them as a single service.

Note:

Currently, API Gateway supports API mashups for REST APIs only: You can define a mashup only in a REST API and only REST APIs can be included in the mashup.

The APIs that are included in an API mashup (participating APIs) can be connected to each other in the following ways:

- **API chaining:** Two or more participating APIs are connected and invoked in a sequence—one after the other.
- **API aggregation:** Two or more participating APIs are connected to a common aggregator step. The aggregator step captures the response of the aggregated APIs. The aggregator step enables you to:
 - Collate the responses and pass to the next step.
 - Process the responses and pass the processed data to the next step.

Usage scenario: API chaining

Assume an API that provides information about courses offered by different universities in a given location. This API provides a service that returns the list of universities for a given course name and postal code. This service could be:

```
GET /universities?course=medicine&postalcode=600012
```

The provider of the API wants to extend this API for use in mobile applications that have access to users' location. As mobile applications can access a user's location in terms of longitude and

latitude, this involves first retrieving the postal code for the users' current location and then passing that information to the existing API.

Suppose there is a publically available API that returns the postal code based on longitude and latitude values. This service could be:

```
GET /postalcode?lat=331&long=22324321
```

If this public API meets other requirements, such as security, performance, and usage limits, it can be utilized to deliver the required functionality.

Using an API mashup, you can create and expose a single service that calls both services: the external service that returns the postal code and the existing service that provides the list of universities. The resulting service could be:

```
GET /universities?course=medicine&lat=331&long=22324321
```

Usage scenario: API aggregation

Assume an IT services provider that provides hosting and cloud services to its customers. Users can create accounts for the different types of services that they need to use: bare metal servers, Virtual Private Servers, platforms as a service, and so on. A customer has multiple types of accounts. The statement for each type of account is returned by a different API. The API provider wants to provide a single API that consolidates the statements of a given customer and returns a single response with all the information.

Key Features of a REST API Mashup

- An API mashup allows you to orchestrate multiple resources and methods and expose the behavior as a single service. In a regular method that is not a mashup, API Gateway applies all the enforced policies and then routes the request to the native endpoint. In the case of a mashup, API Gateway still applies all the enforced policies in the request flow till routing; but thereafter, it starts the orchestration flow defined in the mashup. After the orchestration flow ends, all the policies defined for that method are applied in the response flow—in the same way as a regular method.
- API mashups are defined at the method level. You can edit any REST API and define a mashup for one or more methods within it.
- You can include any REST API defined within API Gateway in the mashup.
- The entire framework that API Gateway provides to a regular REST API method is available to an API mashup method. Therefore, you can utilize query parameters, path parameters, aliases, variables, payload transformations using XSLT transforms, transformations using webMethods IS services, and custom pipeline variables.

Considerations for Creating an API Mashup

- By default, the policies of an API that is participating in an API mashup are not enforced when it is invoked within the API mashup. However, if you select the **Should execute Outbound**

policies option, the outbound security policies of the participating API are enforced in the context of the API mashup.

- The following are specific to a mashup step and are not automatically passed from one step to another:
 - Headers
 - Query parameters
 - Path parameters
 - Payload

However, you can add parameters in a mashup step to access data from any of the previous steps or another source.

An exception to this rule is the first step (the first participating service) in a mashup, which receives the complete request sent by the client.

- A participating API cannot have reverse invoke routing.

Structure of an API Mashup

An API mashup consists of one or more mashup steps, and each step invokes an API. A mashup step defines the request for the API that it invokes. A step can use the data objects provided by API Gateway to access data in the initial request sent to the operation that has the mashup and any of the previous steps.

The following table summarizes the data objects and variables that are available in API Gateway.

Object/Variable Type	Possible values
paramStage	<ul style="list-style-type: none"> ■ request ■ response
paramType	<ul style="list-style-type: none"> ■ payload or body ■ headers ■ query ■ path ■ httpMethod ■ statusCode ■ statusMessage
queryType	<ul style="list-style-type: none"> ■ xpath ■ jsonPath

Object/Variable Type	Possible values
	<ul style="list-style-type: none"> ■ regex

The following data objects are available to a mashup step:

- `${paramStage.paramType}`

You can use this syntax to access the following string variables: `path`, `statusCode`, `statusMessage`, `httpMethod`. Examples: `${request.path}`, `${response.statusCode}`

- `${paramStage.paramType.paramName}`

You can use this syntax to access map types, such as `query`, `headers`, and `path`. Example: `${request.query.var1}`, `${response.header.Content-Type}`, `${request.path.name}`.

- `${paramStage.paramType.queryType[queryValue]}`

This syntax can be used to query a `paramType`. Examples:

- `${request.payload.xpath[//ns:emp/ns:empName]}`

Where `"//ns:emp/ns:empName"` is the XPath to be applied on the payload if `contentType` is `"application/xml"`, `"text/xml"`, or `"text/html"`.

- `${response.payload.jsonPath[$.cardDetails.number]}`

Where `$.cardDetails.number` is the `jsonPath` to be applied on payload if `contentType` is `"application/json"` or `"application/json/badgerfish"`.

- `${request.payload.regex[[0-9]+]}`

Where `"[0-9]+"` is the regular expression to be applied on the payload if `contentType` is `"text/plain"`.

Note:

While `xpath` and `jsonPath` are applicable only to payload, `regEx` can be used with both payload and path.

- `${paramStage[stepName].paramType.queryType[queryValue]}`

You can use this syntax to access data in any step. For example, you can use the following syntax to access the payload of a step named `createAPI`:

`${response[createAPI].payload.jsonPath[$.apiResponse.api.id]}`.

- You can define your own variables using the **Custom Pipeline variables** field:

Examples: **`${key}`**, **`${value}`**. The custom pipeline variables that you define are available in subsequent steps.

Note:

Data objects from any of the steps of the mashup can also be accessed by response processing policies and error processing policies of the API that contains the mashup.

Creating an API Mashup

To create a mashup you require:

- The API must include the resource and the method in which you want to add the API mashup.
- The participating APIs (that you want to include in the mashup) must exist in the API Gateway instance.

➤ To create a mashup in a REST API

1. Click **APIs** in the title navigation bar.

A list of all registered APIs appears.

2. Click the required API.

The API details page appears.

3. Click **Edit**.

4. Click **Mashups**.

The Mashups tab appears. It displays the resources in the API and their methods on the left and an empty (Default routing) routing diagram.

Note:

If the API does not have any mashup, the Mashup tab displays the list of resources only in the **Edit** mode; the tab is empty in the view mode.

5. In the **List of resources**, click the resource in which you want to include the mashup.

The resource tab expands and the methods included in the resource are displayed.

6. Click the toggle button to enable the method in which you want to create the mashup.

Note:

If you use the toggle button and disable a method that has a mashup, the mashup definition for that method is immediately cleared.

7. Click **Add invoke** to add a mashup step.

- a. Connect the step to **Start**.

The **Start** and **Stop** terminators and all steps have connection points that you can connect to the other steps and terminators.

Tip:

To select a connection point and connect it to another connection point: i) Hover the mouse over the top or bottom of the step or terminator till the connection point is highlighted; ii) Click the connection point and drag to the other step or terminator.

- b. Configure the step properties as desired.

The Mashup Routing panel that appears on the right side of the mashup canvas displays the properties for the selected step. You can configure the following properties using the Mashup Routing panel:

Section	Field	Description
Mashup step name		Provide a name for the mashup step that is unique within the mashup.
API Endpoint		The API endpoint that you want to invoke in the mashup step. The API must be published on the current API Gateway instance.
	API Gateway API	The endpoint of the API that you want to use. You can type a few letters and select from the autocomplete list.
	Resource	The resource in the API that you want to use. You can type a few letters and select a resource from the autocomplete list.
	Method	The specific method of the resource that you want to invoke.
	Execute outbound authentication policy	Select if you want the outbound security policies of the participating API to be enforced in the context of an API mashup.
Headers		
	Use incoming Headers	Select to use the headers in the incoming request.
	Custom Headers	Custom headers that you can add in addition or instead of the incoming headers. Each custom header must have the following fields: <ul style="list-style-type: none"> ■ Header Name ■ Header Value
Query Parameters		Provide the following values: <ul style="list-style-type: none"> ■ Query Parameter Name ■ Query Parameter Value

Section	Field	Description
Path Parameters		Provide the following values: <ul style="list-style-type: none"> ■ Path Parameter Name ■ Path Parameter Value
Payload		Type the Payload .
	XSLT Document	Click Add xslt document and select the XSLT file for transforming the payload. Provide the following values: <ul style="list-style-type: none"> ■ XSLT File ■ Feature Name ■ Feature value <p>For information about transforming the payload using XSLT, see “Request Transformation” on page 418.</p>
	XSLT Transformation alias	Click Add xslt transformation alias and select an existing XSLT transformation alias.
Advanced Transformation		
	webMethods IS service	Click Add webMethods IS service and provide the following values: <ul style="list-style-type: none"> ■ webMethods IS Service ■ Run As User ■ Select Comply to IS Spec <p>For information about these fields and using the webMethods IS Service, see “Invoke webMethods IS” on page 410.</p>
	webMethods IS Service Alias	For information about the webMethods IS Service Alias, see “Invoke webMethods IS” on page 410 .
Transformation Metadata		
	Namespace	Provide the following values: <ul style="list-style-type: none"> ■ Namespace Prefix

Section	Field	Description
		<ul style="list-style-type: none"> ■ Namespace URI <p>For information about transformation metadata, see “Request Transformation” on page 418.</p>
Custom Pipeline Variables		<p>You can use custom pipeline variables to hold values that need to be used in another step of the API mashup. Provide the following values:</p> <ul style="list-style-type: none"> ■ Name ■ Value <p>For more information, see “Structure of an API Mashup” on page 288.</p>

Note:

In several fields, such as **Header Value** within custom headers, **Query Parameter Value**, and **Path Parameter Value**, you can use values from previous steps and other data using the variable and alias framework provided by API Gateway. For more information, see [“Structure of an API Mashup” on page 288](#).

8. Click **Add aggregator** to add an aggregator step.

Note:

You can also add an aggregator step by connecting two invocation steps to the same previous step. An aggregator step is automatically added after the steps when you connect the second step to the same previous step.

9. If you have added the aggregator by clicking **Add aggregator**, add the following connections:

- a. Connect the steps that need to be aggregated to the aggregator step.
- b. Connect the aggregator step to the next step.

10. To add additional steps to the aggregated block, complete the following steps

- a. To add a new step to the aggregated block, click **Add invoke** and connect the new step to the same previous step.

You can configure the properties of the new step immediately or later. For details on configuring the step properties, see step 7.

- b. To add an existing step to the aggregated block, delete the connections of the step, if any and then connect the step to the previous step for the aggregated block and the aggregator step.

11. Click the mashup step and configure the properties of the mashup step as desired.

You can configure the mashup step properties using the Mashup Aggregator action panel that appears on the right side of the mashup canvas when you click the aggregator step. You can configure the following properties using the Mashup Aggregator action panel:

Section	Field	Description
Headers		
	Use incoming Headers	Select to use the headers in the incoming request.
	Custom Headers	Custom headers that you can add in addition or instead of the incoming headers. Each custom header must have the following fields: <ul style="list-style-type: none"> ■ Header Name ■ Header Value
Query Parameters		Provide the following values: <ul style="list-style-type: none"> ■ Query Parameter Name ■ Query Parameter Value
Path Parameters		Provide the following values: <ul style="list-style-type: none"> ■ Path Parameter Name ■ Path Parameter Value
Payload		Type the Payload .
	XSLT Document	Click Add xslt document and select the XSLT file for transforming the payload. Provide the following values: <ul style="list-style-type: none"> ■ XSLT File ■ Feature Name ■ Feature value
	XSLT Transformation alias	Click Add xslt transformation alias and select an existing XSLT transformation alias.
Advanced Transformation		
	webMethods IS Service	Click Add webMethods IS service and provide the following values:

Section	Field	Description
		<ul style="list-style-type: none"> ■ webMethods IS Service ■ Select a Run As User ■ Select Comply to IS Spec <p>For information about these fields and using the webMethods IS Service, see “Invoke webMethods IS” on page 410.</p>
	webMethods IS Service Alias	Select an existing webMethods IS service alias.
Transformation Metadata		
	Namespace	<p>Provide the following values:</p> <ul style="list-style-type: none"> ■ Namespace Prefix ■ Namespace URI
Custom Pipeline Variables		<p>You can use custom pipeline variables to hold values that need to be used in another step of the API mashup. Provide the following values:</p> <ul style="list-style-type: none"> ■ Name ■ Value <p>For more information, see “Structure of an API Mashup” on page 288.</p>
Mashup Response Transformation		<ul style="list-style-type: none"> ■ Select Aggregate response ■ Payload

Note:

In several fields, such as **Header Value** within custom headers, **Query Parameter Value**, and **Path Parameter Value**, you can use values from previous steps and other data using the variable and alias framework provided by API Gateway. For more information, see [“Structure of an API Mashup” on page 288](#).

12. Add, configure, and connect additional API invocation steps and API aggregator steps as desired.
13. Click **Save**.

The mashup is created for the selected method.

Note:

You must activate the API to make the mashup available to client applications. For more information about activating an API, see [“Activating an API” on page 302](#).

Viewing API List and API Details

You can view the list of registered APIs, activate, delete, or view analytics of a specific API in the Manage APIs page. In addition, you can view API details, modify API details, activate and deactivate an API in the API details page.

Note:

If you encounter any problem viewing the API details with a message that says API loading has failed, this would be because the property `watt.server.http.jsonFormat` has been set to a value that is not parsed (the default value), which API Gateway does not support.

➤ To view API list and API details

1. Click **APIs** in the title navigation bar.

A list of all registered APIs appears. You can also perform the following operations in the APIs page:

- Filter APIs by **Type**, **Activation status**, or **Team**. Select the required API type, status, or team to view the APIs based on the provided filters.

Note:

The Team filter is applicable only if you have enabled the Teams feature.

- Activate an API by clicking  that denotes an inactive state.

Once an API is activated, the Gateway endpoint is available which can be used by the consumers of this API.

- Deactivate an API by clicking the status icon that denotes an active state.
- Delete an API by clicking the **Delete** icon.
- View API analytics by clicking the **Analytics** icon.
- Publish or Unpublish an API by clicking **Publish** and the **Unpublish** icons respectively.

2. Click any API to view API details.

The API details page displays the basic information, technical information, resources and methods, and specification for the selected API.

REST API Details

The REST framework enables you to model APIs conforming to the (Resource Oriented Architecture) ROA design. For example, you might model an API that serves to expose the web service data and functionality as a collection of resources. Each resource is accessible with unique Uniform Resource Identifiers (URLs). In your API, you expose a set of HTTP operations (methods) to perform on a specific resource and capture the request and response messages and status codes that are unique to the HTTP method and linked within the specific resource of the API.

The API details view for a REST API displays the details of the API such as Basic and Technical information, Resources and methods, API mocking details, and specifications. You can also view the scopes associated, policies enforced, registered applications and the API-specific analytics.

The table lists the API details displayed for the API:

Field	Description
Basic information	Displays the information about the API, such as Name, Version, Owner of the API, the teams that the API is assigned to, status of the API whether its is Active or Inactive, the maturity state of the API, the date on which the API was created and a brief description of the API.
Technical information	Displays the native endpoints of the API.
Resources and methods	Displays a list of resources or methods available in the API sorted by resource/pathname. The list of resources are displayed in sorted order of the path names. Click each resource to view the corresponding HTTP methods, along with a summary. Below each of these methods, details such as parameters and response codes are displayed.
API mocking	Details are visible only when API mocking is enabled. Displays a list of mocked responses for the operations in the API, custom IS service list and conditions along with its mocked response.
Components	Displays the schemas defined at the API level.
Documentation	Displays the definition of the API in different formats.

Various tabs displayed in the API details page display the following details:

- The **Scopes** tab lists the scopes available for the API.
- The **Policies** tab displays the policies enforced for the API.
- The **Mashups** tab displays the mashups defined in the API.
- The **Applications** tab displays all the applications registered with the API.

- The **Analytics** tab displays the API-specific analytics for the time interval selected.

You can perform the following operations from the API details page:

- You can enable API mocking by clicking the **Enable mocking** button. If API mocking is enabled, you can disable it by clicking the **Disable mocking** button. This option is available when the API is in the deactivated state.
- You can update an API by importing from file or from URL by clicking the **Update** button. This option is available when the API is in the deactivated state.
- You can create a new version of the API by clicking the **Create new version** button.
- You can modify details of an API by clicking the **Edit** button. This option is available when the API is in the deactivated state.
- You can activate an API by clicking the **Activate** button. If the API is already activated, you can deactivate it by clicking the **Deactivate** button.

SOAP API Details

The API details view for a SOAP API displays the details of the API such as Basic and Technical information, Operations available, REST transformation details, API mocking details, and specifications. You can also view the scopes associated, policies enforced, registered applications and the API-specific analytics.

The table lists the API details displayed for the API:

Field	Description
Basic information	Displays the information about the API, such as Name, Version, Owner of the API, the teams that the API is assigned to, status of the API whether its is Active or Inactive, the maturity state of the API, the date on which the API was created and a brief description of the API.
Technical information	Displays the native endpoints of the API.
Operations	Displays a list of operations available in the API and they are sorted alphabetically. Operations are displayed along with their type of binding (SOAP 11 , SOAP 12, and other HTTP methods). Click each method to view details such as input, output, and fault messages.
REST transformation	Displays a list of operations exposed as REST resources and they are sorted alphabetically. Operations are displayed along with their type of binding. Click each method to view details such as input, output, and fault messages.
API mocking	Details are visible only when API mocking is enabled.

Field	Description
	Displays a list of mocked responses for the operations in the API, custom IS service list and conditions along with its mocked response that contains the status code and mock payload details.
Documentation	Displays a list of specifications for the API.

Various tabs displayed in the API details page display the following details:

- The **Scopes** tab lists the scopes available for the API.
- The **Policies** tab displays the policies enforced for the API.
- The **Applications** tab displays all the applications registered with the API.
- The **Analytics** tab displays the API-specific analytics for the time interval selected.

You can perform the following operations from the API details page:

- You can enable API mocking by clicking the **Enable mocking** button. If API mocking is enabled, you can disable it by clicking the **Disable mocking** button. This option is available when the API is in the deactivated state.
- You can update an API by importing from file or from URL by clicking the **Update** button. This option is available when the API is in the deactivated state.
- You can create a new version of the API by clicking the **Create new version** button.
- You can modify details of an API by clicking the **Edit** button. This option is available when the API is in the deactivated state.
- You can activate an API by clicking the **Activate** button. If the API is already activated, you can deactivate it by clicking the **Deactivate** button.

OData API Details

Open Data Protocol (OData) enables the creation of REST-based APIs, which allow resources to be exposed as endpoints and identified using the Uniform Resource Identifiers (URIs). In general, OData is represented by an abstract data model called Entity Data Model (EDM). This Entity Data Model allows Web clients to publish and edit REST services and their resources using simple HTTP messages. OData leverages the principles of HTTP, REST and ATOM, and combines the simplicity of REST and SOAP metadata definitions to describe service interfaces, data models, and semantics.

API Gateway supports OData V4 and V2 services.

The API details view for a OData API displays the details of the API such as Basic and Technical information, OData entity sets, singletons, function imports, actions imports and specifications. You can also view the policies enforced, registered applications and the API-specific analytics.

The API Gateway UI exposes only OData navigation properties to visualize the resource path structure of OData APIs. Any other OData property is not displayed.

Note:

API Gateway does not support the querying of Derived Entity Types. This includes the following operations:

- Requesting a Derived Entity
- Requesting a Derived Entity Collection
- Filter on Derived Type

Operations on Derived Types are rejected by API Gateway.

The table lists the API details displayed for the API:

Field	Description
Basic information	Displays the information about the API, such as Name, Version, the teams that the API is assigned to, status of the API whether its is Active or Inactive, the date on which the API was created, and a brief description of the API.
Technical information	Displays base URL of the API and the OData version supported.
OData entity sets	<p>Displays a list of OData entity sets. An entity set element represents a single entity or a collection of entities of a specific entity type in the data model.</p> <p>The list of entity sets is sorted alphabetically. Click each entity set to view the resource path, entity type, resource parameter details, and the corresponding HTTP methods.</p> <p>The entity types are structured records consisting of named and typed properties and key properties whose values uniquely identify one instance from another.</p>
OData singletons	<p>Displays a list of OData singletons. Singletons are single entities which are accessed as children of the entity container.</p> <p>The list of singletons is sorted alphabetically. Click each singleton to view the resource path, entity type, the corresponding HTTP methods, and the navigation properties that allow navigation from an entity to related entities.</p> <p>The OData navigation property has an impact on the resource structure. This property is represented as an OData Resource and denoted as OData Navigation properties inside the OData Resources profile. There is no restriction to the number of levels you can drill down.</p>
OData function imports	Displays a list of OData function imports. The Function Import element represents a Function in an entity model.

Field	Description
	The list of OData function imports is sorted alphabetically. Click each function import to view the resource path, entity type, and the corresponding HTTP methods.
OData action imports	<p>Displays a list of OData action imports. The Action Import element represents an Action in an entity model.</p> <p>The list of OData action imports is sorted alphabetically. Click each action import to view the resource path, entity type, and the corresponding HTTP methods.</p>
Documentation	<p>Displays a list of specifications for the API.</p> <p>The metadata document. The metadata document describes the Entity Data Model that is, the structure and organization of the OData service resources) exposed as HTTP endpoints by that particular service. This document describes the entity types, entity sets, functions and actions.</p>

Various tabs displayed in the API details page display the following details:

- The **Policies** tab displays the policies enforced for the API.
- The **Applications** tab displays all the applications registered with the API.
- The **Analytics** tab displays the API-specific analytics for the time interval selected.

You can perform the following operations from the API details page:

- You can update an API by importing from URL by clicking the **Update** button. This option is available when the API is in the deactivated state.
- You can create a new version of the API by clicking the **Create new version** button.
- You can modify details of an API by clicking the **Edit** button. This option is available when the API is in the deactivated state. Only the following properties of an OData API can be modified:
 - Name
 - Description
 - Version
 - API group
 - Maturity state

For updating the OData entity sets, singletons, function imports and action imports a new import has to be performed.

- You can activate an API by clicking the **Activate** button. If the API is already activated, you can deactivate it by clicking the **Deactivate** button.

Filtering APIs

You can filter APIs based on the API type or the activation status of the API.

> To filter APIs

1. Click **APIs** in the title navigation bar.

A list of all registered APIs appears.

2. In the Add Filter pane, select **REST**, **SOAP**, **OData** or all to filter APIs by type.
3. In the Add Filter pane, select **Active** and/or **Inactive** to filter APIs by their activation status.
4. Click **Apply filter**.

The filtered list of APIs is displayed. You can click **Reset** to reset the values to the original values.

Activating an API

You can activate an API in the Manage APIs page. Alternatively you can also activate the API from the API Details page.

You must have the Activate/Deactivate APIs functional privilege assigned to perform this task.

> To activate an API

1. Click **APIs** in the title navigation bar.

A list of available APIs appears.

2. Click the toggle button, in the corresponding column of the API to be activated, to change the status to  to activate the API.

3. Click **Yes** in the confirmation dialog box.

The API is now activated. The Gateway endpoint is now available, which can be used by the consumers of this API. You can now publish the API to the required destination and expose the API for consumption by the consumers.

You can modify API details or update the API, except the name and version of the API, when the API is in the active state. Active APIs are replaced during deployment with zero downtime

and do not break ongoing requests. The updated APIs do not become effective for ongoing requests.

Note:

If there is an error while saving after updating an active API, the API becomes inactive but the changes are saved.

WSDLs in API Gateway

When you activate a SOAP API in API-Gateway, the API exposes a link to the WSDL describing the API Gateway usage. The format of the link is as follows:

```
http://apigw-host:apigw-port/ws/<service-name>/1?wsdl
```

For example: `http://myhost:5555/ws/Hello_Service/1?wsdl`

If the WSDL imports more files, for example, sub WSDLs or XML schemas, then these files can be accessed through:

```
http://<apigw-host>:<apigw-port>/ws/<service-name>/1/<id>?xsd=<name>
```

For example: `http://myhost:5555/ws/Hello_Service/1/53fe951a-2c04-4283-8b2d-8ee2957531b1?xsd=A`

During this action, unlike all other parts of the WSDL, the `<service>` section is completely regenerated. For each activated HTTP or HTTPS port, depending on the API's transport policy, one or more endpoint entries are generated into the WSDL. By default, the following entries are present:

- Usual entry with service name and version number
- Mediator-compatible entry with service name and original port name
- One entry for each custom endpoint

Port names

The port names are numbered through the different entries to ensure uniqueness. Moreover, when the original port name has a `http` or `https` specifier at its end, then this specifier is taken over to the generated port name.

Examples

Example 1: With a single active HTTP port

```
<service name="Hello_Service">
  <port name="Hello_Port2" binding="tns:Hello_Binding">
    <soap:address location="http://myhost:5555/ws/Hello_Service/1"/>
  </port>
  <port name="Hello_Port" binding="tns:Hello_Binding">
    <soap:address location="http://myhost:5555/ws/Hello_Service.Hello_Port/1"/>
  </port>
</service>
```

You can add custom endpoints to the API, for example, per UI. The customized values (prefix, servicename, version) appear in the `<service>` section.

Example 2: With an additional custom endpoint

```
<service name="Hello_Service">
  <port name="Hello_Port3" binding="tns:Hello_Binding">
    <soap:address location="http://myhost:5555/ws/Hello_Service/1"/>
  </port>
  <port name="Hello_Port2" binding="tns:Hello_Binding">
    <soap:address location="http://myhost:5555/myprefix/myservice/5"/>
  </port>
  <port name="Hello_Port" binding="tns:Hello_Binding">
    <soap:address location="http://myhost:5555/ws/Hello_Service.Hello_Port/1"/>
  </port>
</service>
```

Example 3: With an additional HTTPS port that gets enabled and switched on in the API's transport policy

```
<service name="Hello_Service">
  <port name="Hello_Port" binding="tns:Hello_Binding">
    <soap:address location="http://myhost:5555/ws/Hello_Service.Hello_Port/1"/>
  </port>
  <port name="Hello_Port3" binding="tns:Hello_Binding">
    <soap:address location="https://myhost:5559/ws/Hello_Service.Hello_Port/1"/>
  </port>
  <port name="Hello_Port2" binding="tns:Hello_Binding">
    <soap:address location="http://myhost:5555/ws/Hello_Service/1"/>
  </port>
  <port name="Hello_Port4" binding="tns:Hello_Binding">
    <soap:address location="https://myhost:5559/ws/Hello_Service/1"/>
  </port>
  <port name="Hello_Port5" binding="tns:Hello_Binding">
    <soap:address location="http://myhost:5555/myprefix/myservice/5"/>
  </port>
  <port name="Hello_Port6" binding="tns:Hello_Binding">
    <soap:address location="https://myhost:5559/myprefix/myservice/5"/>
  </port>
</service>
```

Parameters for refining the exposed port entries

Use the parameter **wsdlPortLayout** in **Administration > Extended settings** section to refine the exposed port entries. The parameter can have the following values:

- **service-port**

All the port entries are exposed. This is the default value and the results as explained in the examples above apply.

- **service-only**

Only one port (with servicename or version) is exposed. When this value is set, only the simple endpoint with the service name is generated. Example 3 would now look as follows:

```
<service name="Hello_Service">
  <port name="Hello_Port" binding="tns:Hello_Binding">
    <soap:address location="http://myhost:5555/ws/Hello_Service/1"/>
  </port>
```



```

<port name="Hello_Port2" binding="tns:Hello_Binding">
  <soap:address location="https://myhost:5559/ws/Hello_Service/1"/>
</port>
<port name="Hello_Port3" binding="tns:Hello_Binding">
  <soap:address location="http://myhost:5555/myprefix/myservice/5"/>
</port>
<port name="Hello_Port4" binding="tns:Hello_Binding">
  <soap:address location="https://myhost:5559/myprefix/myservice/5"/>
</port>
</service>

```

■ mediator-comp

When this value is set, the entries generated are in mediator compatibility mode. Note that custom endpoints do not appear in this case. The entries look as follows:

```

<service name="Hello_Service">
  <port name="Hello_Portsoaphttp" binding="tns:Hello_Binding">
    <soap:address
location="http://myhost:5555/ws/Hello_Service.Hello_Portsoaphttp/1"/>
    </port>
  <port name="Hello_Portsoaphttps" binding="tns:Hello_Binding">
    <soap:address
location="https://myhost:5559/ws/Hello_Service.Hello_Portsoaphttps/1"/>
    </port>
</service>

```

Deactivating an API


You can deactivate an API in the Manage APIs page. Alternatively you can also deactivate the API from the API Details page.

You must have the Activate/Deactivate APIs functional privilege assigned to perform this task.

> To deactivate an API

1. Click **APIs** in the title navigation bar.

A list of available APIs appears.

2. Click the toggle button, in the corresponding column of the API to be deactivated, to change the status to  to deactivate the API
3. Click **Yes** in the confirmation dialog box.

The API is now deactivated. The API is no more available to be consumed by consumers.

Publishing APIs

You can publish an API to two types of destinations: API Portal and Service Registries.

Publishing APIs to API Portal

Publishing an API to API Portal sends the SOAP and REST APIs to API Portal on which they are exposed for testing and user consumption.

The process of publishing an API to API Portal is initiated from API Gateway and is carried out on the API Portal server.

Doing this involves the following high-level steps:

- **Step 1:** You initiate the publish process by selecting the API to be published, specify the API endpoints to be visible to the consumers, and the API Portal communities in which the API is to be published.
- **Step 2:** API Gateway publishes the API to each of the specified API Portal communities.
- **Step 3:** During bulk publishing of APIs, the process continues even if API Gateway encounters a failure with API Portal.

When publishing an API to the API Portal destination, keep the following points in mind:

- The API Portal destination must be configured in API Gateway.
- You must have the Publish to API Portal functional privilege.
- You cannot publish an API if it is in inactive state. You have to activate the API before publishing it.

Publishing a Single API to API Portal

Pre-requisites:

You must have the Publish to API Portal functional privilege assigned to perform this task.

➤ To publish an API to API Portal

1. Click **APIs** in the title navigation bar.
A list of all APIs appears.
2. Click the **Publish** icon for the API that you want to publish.
3. Select the API endpoints that need to be visible to the consumers.
At least one endpoint should be selected before publishing the API.
4. Select the API type if you want to publish a REST-enabled SOAP API.

When the REST transformation is enabled for a SOAP API in API Gateway, you can publish the REST-enabled SOAP API to API Portal in one of the following ways:

- **Publish as REST:** Default. The API is published as a REST API to API Portal. The REST resources and methods which correspond to the transformed SOAP operations are also published to API Portal.
- **Publish as SOAP:** The API is published as a SOAP API with the SOAP operations to API Portal.
- **Publish as REST and SOAP:** When both the options are selected, the API is published as a REST API as well as a SOAP API in API Portal and marked as a HYBRID API.

Note:

The **Publish as** option is available only if the REST transformation is enabled for the SOAP API.

5. Select the communities to which the API needs to be published.

By default, an API is published to the Public Community of API Portal.

Note:

If an API is already a part of the package published to a community then you cannot remove it from that community.

6. Click **Publish**.

The API along with the selected endpoints is published to API Portal and available for the consumers to consume it.

A REST-enabled SOAP API is published to API Portal based on the selected API type:

- **REST API.** The API Details view displays the published API as a REST API with the defined REST resources and methods.
- **SOAP API.** The API Details view displays the published API as a SOAP API with the defined SOAP operations.
- **HYBRID API.** The API Details view, by default, displays the published API as a REST API with the REST resources and methods. There is an option **SOAP** that can be selected to display the published API as a SOAP API with the SOAP operations.

Once an API is published, the **Publish** icon changes to **Republish** icon.

You can unpublish an API once it is published by clicking the **Unpublish** icon.

Publishing Multiple APIs to API Portal in a Single Operation

Pre-requisites:

You must have the Publish to API Portal functional privilege assigned to perform this task.

You can bulk publish APIs to API Portal.

➤ To publish multiple APIs to API Portal in a single operation

1. Click **APIs** in the title navigation bar.

A list of all APIs appears.

2. Select the APIs that you want to publish.

By default, all the respective API endpoints are internally selected to be visible to the consumers.

3. In the **Menu**  icon, click **Publish**.

4. Select the communities to which the APIs have to be published.

By default, the APIs are published to the Public Community of API Portal.

5. Click **Publish**.

The APIs along with their associated endpoints are published to API Portal and available for the consumers to consume.

If you have selected several APIs where one or more of them are REST-enabled SOAP APIs in API Gateway, then these SOAP APIs are published as REST APIs along with their specific REST endpoints in API Portal.

6. Examine the **Publish APIs report** window and check for any errors that occurred during the publishing process.

The **Publish APIs report** window displays the following information:

Field	Description
Name	The name of the published API.
Version	The version of the published API.
Status	The status of the publishing process. The available values are: <ul style="list-style-type: none">■ Success■ Failure
Description	A descriptive information if the API publishing process fails or if a warning occurs.

API Gateway writes these results to the **Audit logs** dashboard, so you can view them later.

7. Click **Download the detailed report here** to download the detailed report as an HTML file.

Publishing APIs to Service Registries

Publishing an API to a service registry enables applications to dynamically locate an API Gateway instance that can process that API.

When publishing an API to a service registry, keep the following points in mind:

- Before you publish an API to a service registry destination, you must add the service registry to the API Gateway instance from where you want to publish.
- You must have the **Publish API to service registry** functional privilege to publish APIs to a service registry.
- You can publish only active APIs. You cannot publish APIs that are in the inactivate state.
- An API that is published to a service registry:
 - Is automatically de-registered from the service registry if the API is deactivated in API Gateway. When the API is activated again, it is automatically registered on the same service registry.
 - Is automatically de-registered from the service registry if the API Gateway instance from where it was registered goes down. When the API Gateway instance comes up again, the API is registered on the same service registry.
- In a cluster of API Gateway nodes, only the API Gateway instance from where you publish an API is added to the service registry. You have to separately publish the API from each API Gateway instance that the service registry can use for an API.

Note:

Similarly, you have to separately unpublish the API from each API Gateway instance from where you want to unpublish the API.

- If a load balancer has been configured for the API Gateway cluster, APIs from all instances are registered using the load balancer URL.

Publishing a Single API to Service Registries

Pre-requisites:

You must have the **Publish API to service registry** functional privilege assigned to perform this task.

➤ **To publish an API to service registries**

1. Click **APIs** in the title navigation bar.

The list of APIs defined in API Gateway appears.

2. Click the **Publish** icon for the API that you want to publish.

3. Select **Service Registries**.

The list of service registries that have been added to API Gateway is displayed.

4. Select the service registry to which you want to publish the API.

The list of endpoints in the selected API are displayed.

5. Select the endpoints that you want to publish to the selected service registry.

6. Repeat the previous two steps to publish the API to additional service registries.

7. Click **Publish**.

Once an API is published, the **Publish** icon changes to **Republish** icon.

You can unpublish a published API by clicking the **Unpublish** icon.

Publishing Multiple APIs to Service Registries in a Single Operation

Pre-requisites:

You must have the **Publish API to service registry** functional privilege assigned to perform this task.

Note:

When you publish multiple APIs to one or more service registries in a single operation, all endpoints in the APIs are published. To selectively publish endpoints within an API, you must publish the API separately as a single API.

» To publish multiple APIs to service registries in a single operation

1. Click **APIs** in the title navigation bar.

The list of APIs defined in API Gateway appears.

2. Select the APIs that you want to publish.

3. On the  menu, click **Publish**.

4. Select **Service Registries**.

The list of service registries that have been added to API Gateway is displayed.

5. Select the service registry to which you want to publish the API and click **Publish**.

Once an API is published, the **Publish** icon changes to **Republish** icon.

You can unpublish a published API by clicking the **Unpublish** icon.

Unpublishing APIs

You can manually unpublish APIs that you had previously published to an API Portal or to one or more service registries.

Unpublishing APIs from API Portal

After you publish an API to API Portal, the API remains published and available on API Portal for consumption until you manually unpublish the API.

You can unpublish a SOAP or REST API from API Portal to suspend its interaction, testing, and user consumption in API Portal.

Unpublishing a Single API from API Portal

Pre-requisites:

You must have the Publish to API Portal functional privilege assigned to perform this task.

> To unpublish an API from API Portal

1. Click **APIs** in the title navigation bar.
A list of all APIs appears.
2. Click the **Unpublish** icon for the API that you want to unpublish.
The **Unpublish API** dialog box is displayed.
3. Select **API Portal** in **Destination**.
4. Select **Force unpublish**.
5. Click **Yes** in the confirmation dialog.

The API is unpublished from the API Portal destination. The API is no longer available on API Portal for testing and user consumption.

Once an API is unpublished, the **Republish** icon changes to **Publish** icon.

You can publish an API once it is unpublished by clicking the **Publish** icon.

Unpublishing Multiple APIs from API Portal in a Single Operation

Pre-requisites:

You must have the Publish to API Portal functional privilege assigned to perform this task.


You can bulk unpublish APIs from API Portal.

➤ To unpublish multiple APIs from API Portal in a single operation

1. Click **APIs** in the title navigation bar.

A list of all APIs appears.

2. Select the APIs that you want to unpublish.

3. In the **Menu**  icon, click **Unpublish**.

4. Click **Yes** in the confirmation dialog.

The selected APIs are unpublished from API Portal.

5. Examine the **Unpublish APIs report** window and check for any errors that occurred during the unpublishing process.

The **Unpublish APIs report** window displays the following information:

Parameter	Description
Name	The name of the unpublished API.
Status	The status of the unpublishing process. The available values are: <ul style="list-style-type: none">■ Success■ Failure
Description	A descriptive information if the API unpublishing process fails or if a warning occurs.

API Gateway writes these results to the **Audit logs** dashboard, so you can view them later.

6. Click **Download the detailed report here** to download the detailed report as an HTML file.

Unpublishing APIs from a Service Registry

You can manually unpublish APIs that you had previously published on service registries.

You must consider the following points before unpublishing an API from a service registry:

- You must have the **Publish API to service registry** functional privilege to unpublish APIs from a service registry.
- There is no option to unpublish individual endpoints. When you manually unpublish an API, all the endpoints in that API are unpublished from the selected service registries.

- As both—API publishing to service registries and API unpublishing to service registries—are specific to the current API Gateway instance, APIs are unpublished only for the API Gateway instance from where you unpublish. Therefore, if the same API was published from other instances of API Gateway, it continues to be available on the service registries from those API Gateway instances.

APIs may also get unpublished automatically from service registries, as described below.

Automatic Unpublishing of APIs

API Gateway automatically, but temporarily unpublishes an API in the following situations:

- When you deactivate an API after publishing it to a service registry.

Note:

When you reactivate the API, the temporarily unpublished endpoints are published again to the original service registries.

- When you disable or delete an API Gateway port that has endpoints that have been published to a service registry.

Note:

When you enable or add back the port again, the temporarily unpublished endpoints are published again to the original service registries.

Unpublishing a Single API from Service Registries

Pre-requisites:

You must have the **Publish API to service registry** functional privilege assigned to perform this task.

➤ To unpublish an API from Service Registries

1. Click **APIs** in the title navigation bar.

A list of all APIs appears.

2. Click the **Unpublish** icon for the API that you want to unpublish.

The **Unpublish API** dialog box is displayed.

3. Select **Service registries** in **Destination**.

The list of service registries to which the API was published is displayed.

4. Select the service registries from which you want to unpublish the API.

5. Select **Force unpublish** to mark the API as unpublished in API Gateway even if the unpublish fails on the selected service registries.

The API is unpublished from the selected service registries. The API is no longer available on selected service registries for testing and user consumption.

Once an API is unpublished, the **Republish** icon changes to **Publish** icon.

Unpublishing Multiple APIs from Service Registries in a Single Operation

Pre-requisites:

You must have the **Publish API to service registry** functional privilege assigned to perform this task.

You can bulk unpublish APIs from one or more service registries.

➤ To unpublish multiple APIs from service registries in a single operation

1. Click **APIs** in the title navigation bar.

A list of all APIs appears.

2. Select the APIs that you want to unpublish.

3. In the **Menu**  icon, click **Unpublish**.

4. Select **Service registries** in **Destination**.

The list of service registries to which the APIs were published is displayed.

5. Select the service registries from which you want to unpublish the selected APIs.
6. Select **Force unpublish** to mark the APIs as unpublished in API Gateway even if the unpublish fails on the destination service registries.
7. Examine the **Unpublish APIs report** window and check for any errors that occurred during the unpublishing process.

The **Unpublish APIs report** window displays the following information:

Parameter	Description
Name	The name of the unpublished API.
Status	The status of the unpublishing process. The available values are:

Parameter	Description
	<ul style="list-style-type: none"> ■ Success ■ Failure
Description	A descriptive information if the API unpublishing process fails or if a warning occurs.

API Gateway writes these results to the **Audit logs** dashboard, so you can view them later.

- Click **Download the detailed report here** to download the detailed report as an HTML file.

The APIs are unpublished from the selected service registries for the current API Gateway instance. Once an API is unpublished, the **Republish** icon changes to **Publish** icon.

Modifying API Details

You must have the API Gateway's manage APIs or activate/deactivate APIs functional privilege assigned to perform this task.

You can modify API details, as required, from the API details page.

➤ To modify API details

- Click **APIs** in the title navigation bar.

A list of all registered APIs appears.

- Click the required API.

The API details page appears.

- Click **Edit**.

Note:

If the API is in the active state, you cannot modify the name and version of the API. The API mocking section is unavailable for any changes.

- Modify the information as required.

- Click **Save**.

Note:

- If the API is in the active state when you modify API details, the active API is replaced with the modified API.
- The modified APIs do not become effective for ongoing requests.

Updating APIs

You can update the definition of an existing API by uploading WSDL, Swagger, or RAML file or URL. The uploaded file can also be in a ZIP format. When an API is updated, it retains the `Expose to consumers` settings, the existing scope definitions, the configured policies, and the REST-enabled path configurations for SOAP API. You can also edit an API using the **Edit** option for minor edits, whereas the update feature helps you to overwrite the complete API definition using a file or a URL at the same time.

You can update an active API. You cannot modify the name and version of an API while updating an active API.

Note:

The active APIs are replaced with the updated API. The updated APIs do not become effective for ongoing requests. Updates to an activated API are propagated across a cluster and trigger a hot deploy on each cluster node separately.

You can update an existing API in the following ways:

- By importing an API definition from a file
- By importing an API definition from a URL

Updating an API by Importing an API from a File

You must have the API Gateway's `manage APIs` or `Activate/deactivate APIs` functional privilege assigned to perform this task. You can not update an API by importing an API from a file if the API is in the active state.

> To modify API details

1. Click **APIs** in the title navigation bar.
2. Select the required API from the list of APIs.
3. Click **Update**.
4. Select **Update API by importing from file**.
5. Provide the following information:

Field	Description
Select file	Click Browse to browse to the location of file to be imported and select the required file or ZIP format file.

Field	Description
	The REST API can be updated using only the Swagger or RAML file type. The SOAP API can be updated using only the WSDL file type.
Root File Name	If you have selected a file in ZIP format, type the relative path of the main file within the ZIP file.
Name	Name for the API. Edit or delete the name of the existing API displayed. <ul style="list-style-type: none"> ■ If you provide an API name, this overwrites the API name mentioned in the uploaded file and the API is updated with the name provided. ■ If you do not provide an API name, the API name mentioned in the uploaded file is picked up and the API is updated with that name.
Type	Select the required type. The available types are RAML , Swagger , and WSDL . <ul style="list-style-type: none"> ■ For a REST API, the available options are RAML and Swagger. ■ For a SOAP API, the available option is WSDL.
Version	Version number of the API. The existing version number of the API is automatically displayed. You can edit or delete the version number. If the version number is deleted and the imported file does not have a version number, then the system automatically assigns a version number during the update. This overwrites the version of the API.
Description	Description of the API. The existing description of the API is automatically displayed. You can edit or delete the description. If you delete the description then the description from the imported file is used.

6. Click **Update**.

The API definition is updated with the latest changes from the file and is displayed in the API details page.

Updating an API by Importing an API from a URL

You must have the API Gateway's manage APIs or activate/deactivate APIs functional privilege assigned to perform this task.

> To modify API details

1. Click **APIs** in the title navigation bar.
2. Select the required API from the list of APIs.
3. Click **Update**.
4. Select **Update API by importing from URL**.
5. Provide the following information:

Field	Description
URL	Type the URL from which the API is being imported. <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: The REST API can be updated using only the Swagger or RAML type information that the URL is pointing to. The SOAP API can be updated using only the WSDL file type information that the URL is pointing to. The entity sets, singletons, function imports, and action imports of an OData API can only be updated by a re-import of the OData API definition through the URL.</p> </div>
Protected	Select this option if you want to import an API from a URL that is password protected. The user name and password fields are displayed using which you can access the provided URL.
Username	Type the user name required to access the password protected URL. If you have selected the Protected option, this field is displayed.
Password	Type the password associated with the username. If you have selected the Protected option, this field is displayed.
Name	Name for the API. The existing name of the API is automatically displayed. <ul style="list-style-type: none"> ■ If you provide an API name, this overwrites the API name mentioned in the file referred by URL and the API is updated with the name provided.

Field	Description
	<ul style="list-style-type: none"> If you do not provide an API name, the API name mentioned in the file referred to by URL is picked up and the API is updated with that name.
Type	<p>Select the required type. The available types are RAML, Swagger, WSDL, and OData.</p> <p>For a REST API, the available options are RAML and Swagger.</p> <p>For a SOAP API, the available option is WSDL.</p> <p>For a OData API, the available option is OData.</p>
Version	<p>Version number of the API. The existing version number of the API is automatically displayed. You can edit or delete the version number. If the version number is deleted and the file referred to by URL does not have a version number, then the system automatically assigns a version number during the update.</p> <p>This overwrites the version of the API.</p>
Description	<p>Description of the API. The existing description of the API is automatically displayed. You can edit or delete the description. If you delete the description then the description from the file referred to by URL is used.</p>

6. Click **Update**.

The API definition is updated with the latest changes from the URL and is displayed in the API details page.

API Mocking

Using API Gateway, you can mock an API by simulating the native API. For example, if you have an API without a native implementation, you can mock that API. The mocked response is returned to the consumer when the API is invoked.

In API Gateway, when you enable mocking for an API, a default mock response is configured for each combination of resource, operation, status code, and content-type based on the example and schema specified in that API. You can add a condition to the operation in the resource.

Note:

You cannot enable or disable mocking for an active API.

As an API Provider, you can create or modify the default mock response. You can specify conditions and associate an IS service with the mocked API. When an IS service is associated with a mocked API, the associated IS service must adhere to the *apigateway.specifications:mockService* specification.

At runtime, when the mocked API is invoked, instead of calling the native API, API Gateway returns the mocked response to the consumer based on the following priorities:

1. If any of the conditions for the invoked operation satisfies, API Gateway returns the associated mocked response.
2. If any of the conditions for the invoked operation is not satisfied, and if an IS service is configured for the API, then API Gateway invokes the IS service and returns the IS service response.
3. If any of the conditions for the invoked operation is not satisfied, and if an IS service is not configured for the API, then API Gateway returns the default mocked response.

API mocking is supported only for SOAP and REST APIs.

Note:

You must have the API Gateway's manage APIs or activate/deactivate APIs functional privilege assigned to perform API mocking.

Enabling API Mocking

You can enable or disable API mocking through the API details page.

Note:

You cannot enable or disable API mocking for active APIs.

> To enable API mocking

1. Click **APIs** in the title navigation bar.
2. Select the required API from the list of available APIs.

The API details page for the selected API appears.

3. Click **Enable mocking**.

This generates the default mock responses.

Modifying API Mocking Details

You must select **Enable mocking** from the API details page.

You can modify API mocking details, as required, from the API details page.

> To modify API mocking details

1. Click **APIs** in the title navigation bar.

A list of all registered APIs appears.

- Click the required API.

The API details page appears.

- Click **Edit**.
- Click **API mocking**.
- Specify the following information in the ESB service section:

Field	Description
Invoke service	Specifies the webMethods Integration Server service to be invoked. Note: The webMethods Integration Server service must be running on the same Integration Server as API Gateway.
Run as user	Type the user name you want API Gateway to use to invoke the IS service.

- Select the operation that you want to modify from the Mocked responses section.
- Click **Add Response** if you want to add a response and select the status code from the drop-down.

- Click .

This adds the status code created to the existing status code list.

- Select the status code you want to modify.
- Click **+ Add Response Header** and provide the following information to add the required response headers:

Field	Description
Header key	Specify the HTTP header key that would be contained in the header of HTTP response.
Header value	Specify the HTTP header value that would be contained in the header of HTTP response.

You can add more response headers by clicking .

11. Click **+ Add Content-type** to add a content-type to the status code selected and provide the following information:

Field	Description
Content type	Select the content-type to be added to the selected status code from the drop-down list.
Mock payloads	Specify a mock response payload for the content-type selected.

You can add more content-types by clicking **Add** .

12. Click **+ Add Conditions** to add a condition to the operation in the resource:

Field	Description
Condition name	Specify the name for the condition.
Condition parameter	Select the type to which the condition is to be applied. The available options are: <ul style="list-style-type: none"> ■ Body ■ Header ■ Query parameter (Applicable only for REST APIs)
Key	The key can be a string for the header and query parameter and for body it can be a JSON path or an XML path. <div style="background-color: #f0f0f0; padding: 5px; margin-top: 5px;"> <p>Note: The XML path must not contain namespace prefixes.</p> </div>
Value	The value of the condition. Additionally, you can type an * (asterisk) to ignore the value specified in this parameter and the condition is satisfied based on the value specified in the Key parameter.
Status code	Select the status code from the drop-down list. <div style="background-color: #f0f0f0; padding: 5px; margin-top: 5px;"> <p>Note:</p> <ul style="list-style-type: none"> ■ You must enable the property Send native provider fault in the Administration > General > API fault section in order to have correct mock response for status code 506. ■ While invoking an API remember to use the query parameter <code>expectedStatusCode</code> in order to have correct mock responses for status codes 100 and 202. </div>

Field	Description
+ Add Response Header	<p>Add a response header to the resource selected by providing the following information:</p> <ul style="list-style-type: none"> ■ Header key. Specify the HTTP header key that would be contained in the header of HTTP response ■ Header value. Specify the HTTP header value that would be contained in the header of HTTP response.
+ Add Content-type	<p>Add a content-type to the status code selected by providing the following information:</p> <ul style="list-style-type: none"> ■ Content type. Select the content-type to be added to the selected status code from the drop-down list. ■ Mock payload. Specify a mock response payload for the content-type selected.

You can add more conditions by clicking **Add** .

13. Click **Save**.

Custom Replacer

API Gateway allows you to send a dynamic custom response instead of a static mocked response to the consumer when the mocked API is invoked. In the mocked response, you can specify multiple custom replacers. Custom replacer is used to replace the custom variables with the values defined in the request headers, query parameters, and request body. The custom replacer is available in the `${request.<ConditionParameter>.<Key|JsonPath|XPath>}` format. The custom replacers are:

- `${request.header.<headerKey>}`: To replace the value of the `headerKey` from the request headers.
- `${request.query.<queryKey>}`: To replace the value of the `queryKey` from the query parameters in the request URL.
- `${request.body.<JsonPath|XPath>}`: To replace the value of the `<JsonPath|XPath>` from the request body.

Attaching Documents to an API

Pre-requisites:

You must have the Manage APIs functional privilege assigned to perform this task.

You can associate an input document that includes the RAML, Swagger, or WSDL specification, and additional documents such as programming guides, sample code, script files, and project plan with an API. For example, SOAP APIs can contain external documents such as Functional Requirements, Error Messages, Release Notes, and so on.

When attaching a document to an API, keep the following points in mind:

- You cannot attach or modify a document to the API if it is in active state. You have to deactivate the API before attaching or modifying it.
- API Gateway relies on file extensions to determine a file's type. When you upload a file from your local machine to the API, be sure the name of the file on your local machine includes a file extension so that API Gateway can determine the file's type and attach it correctly to the API.
- You cannot upload types of files that are restricted for attaching as the input document to the API.

API Gateway provides the ability to restrict certain kinds of files from being uploaded to the API, based on the file extension. The list of restricted files may vary depending on the file extensions configured in the `apiDocumentsRestrictedExtension` property under **Administration > Extended settings** section.

When you try to upload a file type that is restricted, API Gateway prompts you with an error message.

- By default, several standard file extensions are blocked in API Gateway, including any file extensions that are treated as executable files by Windows Explorer. The file extensions blocked by default are - `.bat`, `.bin`, `.dll`, and `.exe`.
- You cannot upload files that exceed the maximum allowed size for the API.

API Gateway provides the ability to limit the maximum file upload size to the API. The maximum file upload size is configured in the `apiDocumentsUploadSizeLimitInMB` property under **Administration > Extended settings** section.

When you try to upload a file that exceeds the maximum file upload size, API Gateway prompts you with an error message.

- You can rename an uploaded document. When you rename a document, only the display name of the document could be modified, not the document itself. If you want to modify the document as well, you must delete the file attachment, and attach the latest file.

> To attach document

1. Click **APIs** in the title navigation bar.

A list of all registered APIs appears.

2. Select the required API.

The API details page appears.



3. Click **Edit**.

4. Click **Documentation**.

5. Click **Browse** to select a file and upload it.
6. Rename the document in the **Display Name** field as required.

This is the display name of the document in the API details page.

7. Click **Add**.

The attached document is listed in a table. You can edit and delete the document by clicking the  and  icons.

8. Repeat steps 5 to 7 for each document that you want to attach to the API.
9. Click **Save**.

SOAP to REST Transformation

SOAP APIs are commonly used to expose data within enterprises. With the rapid adoption of the REST APIs, API providers must be able to provide RESTful interfaces to their existing SOAP APIs instead of creating new REST APIs. Using the API Gateway SOAP to REST transformation feature, the API provider can either expose the parts of the SOAP API or expose the complete SOAP API with RESTful interface. API Gateway allows you to customize the way the SOAP operations are exposed as REST resources. Additionally, the Swagger or RAML definitions can be generated for these REST interfaces.

Note:

You must have the API Gateway's manage APIs or activate/deactivate APIs functional privilege assigned to perform this task.

Activating SOAP to Rest Transformation

You must have the API Gateway's manage APIs functional privilege assigned to perform this task.

➤ To activate SOAP to REST transformation for a SOAP operation

1. Click **APIs** in the title navigation bar.
2. Select the required API from the list of available APIs.

The API details page for the selected API appears.

3. Click **Edit**.
4. Click **REST transformation**.

A list of SOAP operations already exposed to the consumers as well as to be transformed from SOAP to REST appears. By default, all the SOAP operations are in inactive state.

5. Click  to activate the SOAP to REST transformation for the SOAP operations.

Alternatively, you can activate the SOAP to REST transformation for multiple SOAP operations simultaneously by clicking the **Transform all operations** activation toggle button.

6. Select the operation to edit the SOAP operations.
7. Click **Save**.

The API details page for the selected API appears.

8. Click **REST transformation**.

A list of REST resources for the SOAP operations appears. Click on each resource to view the details that are already available as REST definitions.

Modifying the REST Definitions for SOAP Operations

You must have the API Gateway's manage APIs functional privilege assigned to perform this task.

> To modify the REST definitions for SOAP operation

1. Click **APIs** in the title navigation bar.
2. Select the required API from the list of available APIs.

The API details page for the selected API appears.

3. Click **Edit**.
4. Click **REST transformation**.

A list of SOAP operations already exposed to the consumers as well as to be transformed from SOAP to REST appears.

5. Click  to provide the **Resource name** and **Resource path**.


Alternatively, you can activate the SOAP to REST transformation for multiple SOAP operations simultaneously by clicking the **Transform all operations** activation toggle button.

6. Select the operation to edit the SOAP operations.
7. Provide the following information:

Field	Description
Resource name	Name of the resource. The existing name of the SOAP operation automatically appears, you can modify this name.
Resource path	Path of the resource. The existing path of the SOAP operation automatically appears, you can modify this path.

8. Click **+ Add Parameter** and provide the following information to add the required resource level parameters:

Field	Description
Name	Name of the parameter.
Description	Description of the parameter.
Type	Specifies the parameter type. Available values - Path, Query-string
Data type	Specifies the data type. Available values - String, Date, Date time, Integer, Double, Boolean.
Required	Specifies the parameter is required if selected.
Repeat	Applicable to parameters of type query. The query parameter value can take comma separated array values.
Value	Specifies the possible value.
XPath	XPath. Specifies how the request parameter must be mapped to the SOAP payload that is sent to the native SOAP service. For example, <code>/soapenv:Envelope/soapenv:Body/axis:sayHello/axis:name,</code> or <code>//axis:name</code> (If the SOAP request has only one element such as name).
Namespace prefix	Specifies the namespace prefix of the element that appears in the XPath .
Namespace URI	Specifies the namespace URI for the XPath element.

Field	Description
	You can add more namespace prefixes and namespace URIs by clicking  .

You can add more parameters by clicking .

9. Select one of the available methods: GET, POST, PUT, or DELETE. By default, POST is selected.

By default, API Gateway generates the sample JSON request and response based on the XML schema definitions of the SOAP API. Additionally, you can provide a schema and modify the generated sample.

10. Click **Add Request** and provide the schema and a sample for the content-type.
11. Click **Add Response** and select the status code from the drop-down and provide a description for the status code selected.

Additionally, to add a content-type to the status code selected, click the status code to which you want to add a content-type and select the **Content type**. Provide a schema and a sample for the content-type selected. By default, status code 200 is automatically generated by the system.

12. Click **REST transformation**.

A list of SOAP operations already exposed to the consumers as well as to be transformed from SOAP to REST appears. By default, all the SOAP operations are in inactive state.

13. Click  to activate the SOAP to REST transformation for the SOAP operations.

Alternatively, you can activate the SOAP to REST transformation for multiple SOAP operations simultaneously by clicking the **Transform all operations** activation toggle button.

14. Select the operation to edit the SOAP operations.

15. Click **Save**.

The API details page for the selected API appears.

16. Click **REST transformation**.

A list of REST resources for the SOAP operations appears. Click on each resource to view the details that are already available as REST definitions.

17. Click **Save**.

Supported Content-types and Accept Headers

The following table specifies the content-type available for the HTTP methods:

HTTP Method	Content-types	Accept Headers
GET	application/x-www-form-urlencoded	application/json
		application/xml or text/xml
		multipart/form-data or multipart/mixed
POST	application/json	application/json
	application/xml or text/xml	application/xml or text/xml
	multipart/form-data or multipart/mixed	multipart/form-data or multipart/mixed
	application/x-www-form-urlencoded	
PUT	application/json	application/json
	application/xml or text/xml	application/xml or text/xml
	multipart/form-data or multipart/mixed	multipart/form-data or multipart/mixed
	application/x-www-form-urlencoded	
DELETE	application/x-www-form-urlencoded	application/json
		application/xml or text/xml
		multipart/form-data or multipart/mixed

Note:

If a content-type is not specified, then the request verifies the value of the `Set Media Type` parameter. If the value of the `Set Media Type` parameter is not defined, then by default, for POST and PUT HTTP methods, the `application/json` content-type is used. Whereas for GET and DELETE HTTP methods, the `application/x-www-form-urlencoded` content-type is used.

REST API Endpoints

After providing the information required for the SOAP to REST transformation and activating the API, the API can be invoked as either SOAP or REST API.

The REST transformation of the SOAP API does not change the API name. The only change to the SOAP invocation is that the `<resource-path-for-the-operation>` is appended:

```
/ws/<API-NAME>/<version-number>/<resource-path-for-the-resource>
```

Note:

The REST-enabled SOAP API cannot be invoked using the `/rest` directive.

Samples for REST Request

application/json

The following table provides the samples of the REST request for the `application/json` content-type application and the equivalent SOAP request after transformation from REST to SOAP:

	Request	Equivalent SOAP Request
Consists of only one element (qualified namespaces)	<pre>{ "name": "user1" }</pre>	<pre><soapenv:Envelope xmlns:soapenv= "http://schemas.xmlsoap.org/soap/envelope/" xmlns:axis="http://ws.apache.org/axis2"> <soapenv:Body> <axis:sayHello> <axis:name>user1/axis:name> </axis:sayHello> </soapenv:Body> </soapenv:Envelope></pre>
Consists of only one element (non-qualified namespaces)	<pre>{ "name": "user1" }</pre>	<pre><soapenv:Envelope xmlns:soapenv= "http://schemas.xmlsoap.org/soap/envelope/" xmlns:axis="http://ws.apache.org/axis2"> <soapenv:Body> <axis:sayHello> <name>user1</name> </axis:sayHello> </soapenv:Body> </soapenv:Envelope></pre>
Consists of multiple elements	<pre>{ "a": "1", "b" : 2 }</pre>	<pre><soapenv:Envelope xmlns:soapenv= "http://schemas.xmlsoap.org/soap/envelope/"> <soapenv:Body> <addInts> <a>12 </addInts> </soapenv:Body> </soapenv:Envelope></pre>

application/xml and text/xml

The following table provides the samples of the REST request for the `application/xml` and `text/xml` content-type application and the equivalent SOAP request after transformation from REST to SOAP:

	Request	Equivalent SOAP Request
Consists of only one element and namespace added by the client	<pre><axis:name xmlns:axis="http://ws.apache.org/axis2">user1</axis:name></pre>	<pre><soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:axis="http://ws.apache.org/axis2"> <soapenv:Body> <axis:sayHello> <axis:name>user1</axis:name> </axis:sayHello> </soapenv:Body> </soapenv:Envelope></pre>
Consists of only one element and client does not send the Namespace	<pre><someOtherNamespace :name xmlns:toMed="http://someOtherNamespace">user1</someOtherNamespace :name></pre>	<pre><soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:axis="http://ws.apache.org/axis2"> <soapenv:Body> <axis:sayHello> <axis:name>user1</axis:name> </axis:sayHello> </soapenv:Body> </soapenv:Envelope></pre>
Consists of only one element and the client sends a different namespace to API Gateway	<pre><toMed:name xmlns:toMed="http://t0Med">user1</toMed:name></pre>	<pre><soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:axis="http://ws.apache.org/axis2"> <soapenv:Body> <axis:sayHello> <axis:name>user1</axis:name> </axis:sayHello> </soapenv:Body> </soapenv:Envelope></pre>
Multiple XML elements	<pre><addInts> <a>2 3 </addInts></pre>	<pre><soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"> <soapenv:Body> <addInts> <a>23 </addInts> </soapenv:Body> </soapenv:Envelope></pre>

Path and Query Parameters

The following table provides the samples of the REST request having path and query parameters and the equivalent SOAP request after transformation from REST to SOAP:

	Request	Equivalent SOAP Request
Simple query or path parameter	<pre>/ws/CalcService/add/{num1}</pre>	<pre><ws:addInts xmlns:ws="http://test"> <num1>10</num1></ws:addInts>s</pre>
	OR	
	<pre>/ws/CalcService/add?num1=10</pre>	

	Request	Equivalent SOAP Request
Multiple query or path parameters	<code>/ws/CalcService/add/{num1}/{num2}</code>	<pre><ws:addInts xmlns:ws="http:test"> <num1></num1><num2></num2> </ws:addInts></pre>
	or	
	<code>/ws/CalcService/add?num1=10&num2=3</code>	
	or	
	<code>/ws/CalcService/add/{num1}&num2=3</code>	
Hierarchical elements	<code>/ws/CalcService/add/{num1}/anotherNumber/{num2}</code>	<pre><ws:addInts xmlns:ws="http:test"> <num1></num1> <num2></ num2></ws:addInts></pre>
	<code>/ws/CalcService/add/{num1}/anotherNumber/{num2}</code>	

multipart/form-data

If you send the `multipart/form-data` content-type as the REST request, then you need to optimize the method to be used. This optimization is based on the value specified in the `SOAP Optimization Method` parameter available in Routing policy. The default optimization type is `Message Transmission Optimization Mechanism (MTOM)`. For example, API Gateway converts REST requests with `multipart/form-data` and `multipart/mixed` types as follows:

1. The Multipurpose Internet Mail Extensions (MIME) parts that have a content ID or name that match the elements of type `base64Binary` or `hexBinary` in the schema are added as attachments to the outbound request.
2. Parts other than the content ID or name types are converted into XML depending on the content-type of the MIME part. The `application/xml` and `application/json` content-types are converted. If API Gateway is unable to process the MIME part, it wraps the MIME part inside an XML element with the name of the content ID.

Limitations

The following limitations apply when you perform a SOAP to REST transformation:

- When the API provider defines the mapping for the SOAP operation to the REST resource or method, API Gateway allows him to specify either the path and the query parameters or the body but not both. These mappings are used when transforming the incoming REST request to the SOAP request.
- If both path and query parameters and body are sent in the incoming REST request, then the path and the query parameters are ignored.

- If your REST resource accepts the `text/xml` content-type, then you cannot modify the default resource path and resource name automatically generated by the system. This name must be same as the SOAP operation name. However, this limitation is not applicable for other content-types.
- The HTTP method filters of the global policy are not applicable to the REST transformed method of the SOAP API.
- The REST (REST transformed SOAP operations) resources do not appear as general REST resources when filtered in the **Scopes** section of the API in API Gateway.
- You cannot apply the **Inbound Authentication-Message** policy to the SOAP operation enabled as REST.
- The SOAP services that have Web Services Interoperability Organization (WS-I) non-compliant WSDLs cannot be REST-enabled.

CentraSite Provided APIs

When you want to perform governed API development with CentraSite and API Gateway, you can create an API in CentraSite defining the design-time aspects. The API can be deployed to the API Gateway. In such cases, you can also see that particular CentraSite destination is being configured in the API Gateway. The API details for the CentraSite provided APIs are set as read-only. However, you can edit the run-time aspects such as scope and policies.

Note:

When you remove the CentraSite destination from the API Gateway, this implies that the API is provided by the API Gateway and therefore the details of API are not read-only. You can edit them as required.

When you deploy

- a REST API from CentraSite, then you cannot modify the **Basic information, Technical information, Resource and methods**, and **Components** sections. The above mentioned sections are marked as read-only. However, you can modify the fields in the **API Mocking** and **Documentation** sections.
- a SOAP API from CentraSite, then you cannot modify the **Basic information, Technical information, Resource and methods**, and **Components** sections. The above mentioned sections are marked as read-only. However, you can modify the fields in the **REST transformation, API Mocking**, and **Documentation** sections.
- a OData API from CentraSite, then you cannot modify the **Basic information** and **Technical information** sections. The above mentioned sections are marked as read-only. However, you can modify the fields in the **Documentation** section.

For more information about Modifying API, see [“Modifying API Details” on page 315](#).

Versioning APIs

API Gateway supports the creation of new API versions from the existing versions. The new API has the same metadata but with an updated version. The version can either be a number or a string.

The API details page has a drop-down list that displays all the existing API versions. You can create a new version of an API and retain applications that are associated with older versions of the API. When an API is updated, it retains the `Expose to consumers` settings, the existing scope definitions, the configured policies, and the REST-enabled path configurations for SOAP API.

When you create a new version, the newer version is assigned to the teams of the older version by default. You can later change the teams, if required.

Creating New API Version

You must have the API Gateway's `manage APIs` functional privilege assigned to perform this task.

You can create a new version of an API from the latest version available for the API. For example, if the existing version is 1.1 for an API, you can create a version 1.2. If you want to create a version 1.3, you can only create it from the latest version 1.2 and not from 1.1. However, you can delete the intermediate versions. Additionally, even though the owner of the older API version is a different provider, when you create a new version of the API, you are the owner of the newly created version of the API. The new API version is in inactive state, irrespective of the state of the API from which it was versioned.

> To create a new version

1. Click **APIs** in the title navigation bar.
2. Select the required API from the list of APIs.
3. Click **Create new version**.
4. In the **Version** field, type the new version for the API.
5. Clear the **Retain applications** checkbox if you do not want to retain applications that are associated with older versions of the API.
6. Click **Create**.

The **Version** drop-down lists the newly created API version in latest to older order in the API details page. The corresponding API details page is displayed when you select any particular version.

Note:

The version is appended to the **Gateway endpoint(s)** URL once the API is activated and this can be seen in the **Technical information** section of the API details page. When a client application invokes the API without the version in the endpoint, API Gateway invokes the latest version.

API Scopes

API definitions can be complex and span across multiple REST resources and methods, or SOAP operations for an API. To reduce the complexity of an API definition, you can define scopes and impose a set of policies on each scope to suit your requirements.

A scope represents a logical grouping of REST resources, methods, or both, and SOAP operations in an API. You can then enforce a specific set of policies on each individual scope in the API.

An API can have a set of declared scopes. The available scopes for an API are listed in the **Scopes** tab of the API details page.

Creating an API Scope

Scopes enable you to group a set of REST resources, methods, or both, and SOAP operations for an API.

A scope consists of a name, description, and zero or more resources, methods, or operations. An API can have zero or more scopes.

You can define a set of policies and configure its properties for each individual scope. These policies apply to each of the resources, methods, or operations that are associated to the scope.

Instructions throughout the remainder of this guide use the term *scope-level policy* when referring to a set of policies configured for an individual scope of the API.

Note:

Ensure that you have a unique set of resources, methods, or operations in every scope in the API.

➤ To create a scope

1. Click **APIs** in the title navigation bar.

A list of APIs available in API Gateway appears.

2. Click the name of the required API.

This opens the API details page.

3. Click **Edit**.

If the API is active, API Gateway prompts you to deactivate it.

4. Click the **Scopes** tab.

This displays a list of scopes available in the API.

5. In the List of scopes section, click **Add scope**.
6. In the Basic information section, provide the required information for each data fields that appears:

Field	Description
Name	Name of the scope. A scope name must be unique within an API. Note: API Gateway automatically adds the name <code>New Scope</code> to the Name field. You can change the name of the scope to suit your needs. But you cannot leave this field empty.
Description	Description of the scope.

7. Applicable only for REST APIs. In the Resources and methods section, select the resources, methods, or both, you want to associate to this scope.

When selecting a resource or method for the scope definition, you can select whether you want some or all of the methods within that resource to be selected as well.

8. Applicable only for SOAP APIs. In the Operations section, select the operations you want to associate to this scope.
9. Click **Save**.

The scope is created and listed in the List of scopes section.

Post-requisites:

- Activate the API when you are ready to put it into effect.
- To apply and configure policies for this API scope, see “[Creating a Scope-level Policy](#)” on [page 561](#).

Viewing List of API Scopes and Scope Details

The **Scopes** tab in the API details page displays a list of all available scopes in the API.

In addition to viewing the list of scopes, you can also examine and modify the details of a scope, and delete a scope in the **Scopes** tab.

- **To view the scope list and properties of a scope**

1. Click **APIs** in the title navigation bar.

A list of APIs available in API Gateway appears.

2. Click the name of the required API.

This opens the API details page.

3. Click the **Scopes** tab.

This displays a list of scopes available in the API.

4. In the List of scopes section, click the name of the scope you want to examine.

This opens the details of the scope. The scope details appear in the following sections:

- **Basic information:** This section contains a summary of basic information such as name and description of the scope.
- **Resources and methods:** Applicable only for REST APIs. This section contains a collection of REST resources, methods, or both, that are associated to the scope.
- **Operations:** Applicable only for SOAP APIs. This section contains a collection of SOAP operations that are associated to the scope.

Modifying API Scope Details

You use the **Scopes** tab in the API details page to examine and modify the details of a scope.

➤ To modify the properties of a scope

1. Click **APIs** in the title navigation bar.

A list of APIs available in API Gateway appears.

2. Click the name of the required API.

This opens the API details page.

3. Click **Edit**.

If the API is active, API Gateway displays a warning message to let you know that the API is active.

4. Click the **Scopes** tab.

This displays a list of scopes available in the API.

5. In the List of scopes section, click the name of the scope you want to modify.

This opens the details of the scope. The scope details appears in the following sections:

- **Basic information:** This section contains a summary of basic information such as name and description of the scope.
- **Resources and methods:** Applicable only for REST APIs. This section contains a collection of REST resources, methods, or both, that are associated to the scope.
- **Operations:** Applicable only for SOAP APIs. This section contains a collection of SOAP operations that are associated to the scope.

6. Modify the basic properties, applicable resources, methods, or operations of the scope.
7. Click **Save**.

Activate the API, if it is not active, to put it into effect.

Exposing a REST API to Applications

The API Provider can restrict the exposure of specific resources and methods of a REST API to other applications.

Consider you have a native REST API created in API Gateway with resources - Resource A, Resource B, and Resource C. You might want to expose Resource A and Resource C, and restrict the visibility of Resource B to other applications. You can use the **Expose to consumers** button to switch on the visibility of Resource A and Resource C and switch off the visibility of Resource B as required. Similarly, you can restrict the visibility of one or more methods within each individual resource.

If an application attempts to invoke the Resource C in the above REST API, API Gateway returns a HTTP response code 404.

By default, the **Expose to consumers** button is switched on for all resources and methods of the REST API. Once the API is activated, all of its resources and methods are exposed to registered applications. If you do not want a particular set of resources and methods, or a set of methods in a particular resource to be hidden for registered applications, switch off the **Expose to consumers** button in the REST API definition.

Note:

Be aware that API Gateway does not allow you to activate a REST API if none of the methods in the API are selected for exposing to other applications. You must select at least one method of the REST API to enforce runtime invocations.

> To expose a set of resources and methods of the REST API

1. Click **APIs** in the title navigation bar.

A list of APIs available in API Gateway appears.

2. Click the name of the required API.

This opens the API details page.

3. Click **Edit**.

If the API is active, API Gateway displays a warning message to let you know that the API is active.

4. Click Resources and methods.

This displays a list of resources and methods available in the API.

- a. To select a resource, switch on the **Expose to consumers** button next to the resource URI.

You can select one or more resources to expose to other applications.

- b. To select a method within the resource, click on the resource path. In the expanded list of methods, switch on the **Expose to consumers** button next to the method name.

You can select one or more methods to expose to other applications.

5. Click **Save** to save the updated API.

Activate the API, if it is not active, to put it into effect.

Deleting an API Scope

You delete a scope to remove it from the API permanently.

When a scope is deleted from the API definition, API Gateway deletes the existing associations between the scope and the collection of resources, methods, or operations in the API. But, the collection of resources, methods, or operations continue to exist in the API.

> To delete a scope

1. Click **APIs** in the title navigation bar.

A list of all registered APIs appears.

2. Click the name of the required API.

This opens the API details page.

3. Click **Edit**.

If the API is active, API Gateway displays a warning message to let you know that the API is active.

4. Click the **Scopes** tab.

This tab displays a list of scopes available with the API.

5. In the List of scopes section, locate the name of the scope you want to delete.
6. Click the **Delete** (🗑️) icon next to the scope name.
7. Click **Yes** in the confirmation dialog.

The scope is removed from the List of scopes section.

8. Click **Save** to save the updated API.

Activate the API, if it is not active, to put it into effect.

Example: Usage Scenarios of API Scopes

API Provider can restrict the enforcement of policies at the resource-level or method-level for a REST API, and at the operations-level for a SOAP API. This policy enforcement on the resources, methods, or operations of the API will apply in addition to the default enforcement of policies at the global-level and the user-defined enforcement of policies at the API-level.

Consider you have a REST API, for example, *PhoneStore API*, with a collection of resources and methods.

Resource Name	Resource Path	Supported Methods
Resource A	/phones/orders	GET
		POST
Resource B	/phones/orders/{order-id}	GET
		PUT
		DELETE
Resource C	/phones/orders/{order-id}/paymentdetails	GET
		POST

This section demonstrates the application of scopes and the policy enforcement using Resource C: /phones/orders/{order-id}/paymentdetails of the PhoneStore API.

You can create scopes in the PhoneStore API, and define the individual scopes with a specific set of resources, methods, or both.

Scope Name	Applied Resource	Applied Method
PAYMENT Scope	Resource C: /phones/orders/{order-id}/paymentdetails	

Scope Name	Applied Resource	Applied Method
WRITE Scope	Resource C: /phones/orders/{order-id}/paymentdetails	POST

Assume you have an API-level policy which enforces an Identify and Authorize Application policy with HTTP Basic Authentication for the PhoneStore API. Now, you might need to have different authentication mechanisms for different methods and resources (collectively, scopes) of the PhoneStore API, depending on the level of access you need.

For example, you might want to enforce an Identify and Authorize Application policy with Require HTTP Basic Authentication for the Resource C in PAYMENT Scope to enforce secured access to the data. You might also want to apply an Identify and Authorize Application policy with API Key authentication and Throttling Traffic Optimization policy (with 5 API invocations per minute), in particular, for the POST method of the Resource C in WRITE Scope to enforce a higher-level of secured access and manipulation of the REST data.

API-level / Scope-level Policy	Applied Policies
API-level Policy	Identify and Authorize policy with HTTP Basic Authentication
Scope-level Policy for PAYMENT Scope	Identify and Authorize policy
Scope-level Policy for WRITE Scope	Identify and Authorize policy with API Key Traffic Optimization

The API Scopes definition looks like this:

API-level / Scope-level Policy	Applied Policies
API-level Policy	Identify and Authorize policy with HTTP Basic Authentication
Policy for PAYMENT Scope	Resource C: /phones/orders/{order-id}/paymentdetails Identify and Authorize policy
Policy for WRITE Scope	Resource C: /phones/orders/{order-id}/paymentdetails Method: POST Identify and Authorize policy with API Key Traffic Optimization

The precedence of the policy enforcement effective for an API at run-time is as follows:

1. Global Policy Enforcement

2. Method-level Policy Enforcement (REST APIs) -OR- Operation-level Policy Enforcement (SOAP APIs)
3. Resource-level Policy Enforcement (REST APIs)
4. API-level Policy Enforcement

The specific aspect of processing during the handling of an API invocation at run-time in API Gateway can be best understood with the following scenarios:

Scenario A: Invoke GET method on the Resource C: /phones/orders/{order-id}/paymentdetails

- Global Policy: Not applicable
- Method-level Policy: Not applicable
- Resource-level Policy(s): Identify and Authorize Application policy
- API-level Policy: Identify and Authorize Application policy with HTTP Basic Authentication

As per the precedence of policy enforcement, the Identify and Authorize Application policy at the resource-level and the Identify and Authorize Application policy with HTTP Basic Authentication at the API-level are enforced at run-time.

The effective policy set enforced on the API for the GET method at run-time includes:

- Identify and Authorize Application policy
- Identify and Authorize Application policy with HTTP Basic Authentication

Scenario B: Invoke POST method on the Resource C: /phones/orders/{order-id}/paymentdetails in WRITE Scope

- Global Policy: Not applicable
- Method-level Policy(s): (1) Identify and Authorize Application policy with API Key (2) Throttling Traffic Optimization
- Resource-level Policy(s): Identify and Authorize Application policy
- API-level Policy: Identify and Authorize Application policy with HTTP Basic Authentication

As per the precedence of policy enforcement, the Identify and Authorize Application policy with API Key at the method-level takes precedence over the Identify and Authorize Application policy with HTTP Basic Authentication at the API-level, and is enforced at run-time.

The effective policy set enforced on the API for the POST method at run-time includes:

- Identify and Authorize Application policy
- Identify and Authorize Application policy with API Key
- Throttling Traffic Optimization

Now, consider that you apply an active Global Policy that has the Identify and Authorize Application policy with Hostname Address for all REST APIs (including our PhoneStore API).

Scenario C: Invoke POST method on the Resource C: /phones/orders/{order-id}/paymentdetails in WRITE Scope

- Global Policy: Identify and Authorize Application policy with Hostname Address
- Method-level Policy(s): (1) Identify and Authorize Application policy with API Key (2) Throttling Traffic Optimization
- Resource-level Policy(s): Identify and Authorize Application policy
- API-level Policy: Identify and Authorize Application policy with HTTP Basic Authentication

As per the precedence of policy enforcement, the Identify and Authorize Application policy with Hostname Address applied through the global policy takes precedence over every other Identify and Authorize Application policy that is applied at the method-level and the API-level, and is enforced at run-time.

The effective policy set enforced on the API for the POST method at run-time includes:

- Identify and Authorize Application policy
- Identify and Authorize Application policy with Hostname Address
- Throttling Traffic Optimization

Resolving Scope Conflicts

When you save an API, API Gateway combines the scopes specified with the set of policies defined at the API-level, and on saving the API, API Gateway applies the policies to the API at various enforcement levels. API Gateway validates the scope list to ensure that it contains no conflicting or incompatible policies. If the list contains conflicts or inconsistencies, API Gateway prompts you with an error message.

Consider that you modify the existing UPDATE scope to include a POST method for Resource C. The API Scopes definition now looks like this:

API-level / Scope-level Policy	Applied Policies
API-level Policy	Identify and Authorize Application policy with HTTP Basic Authentication
Policy for PAYMENT Scope	Resource C: /phones/orders/{order-id}/paymentdetails Identify and Authorize Application policy
Policy for WRITE Scope	Resource C: /phones/orders/{order-id}/paymentdetails Method: POST Identify and Authorize Application policy with API Key Traffic Optimization
Policy for UPDATE Scope	Resource C: /phones/orders/{order-id}/paymentdetails

API-level / Scope-level Policy	Applied Policies
	Method: POST
	Identify and Authorize Application policy with API Key

Scenario D: Save the updated PhoneStore API.

- Global Policy: Not applicable
- Method-level Policy(s): (1) Identify and Authorize Application policy with API Key (2) Identify and Authorize Application policy with IP Address Range (3) Throttling Traffic Optimization
- Resource-level Policy(s): Identify and Authorize Application policy
- API-level Policy: Identify and Authorize Application policy with HTTP Basic Authentication

As per the precedence of policy enforcement, the Identify and Authorize Application policy at the method-level in WRITE and UPDATE Scopes take precedence over the Identify and Authorize Application policy at the API-level. But the Identify and Authorize Application policy with the API Key and IP Address Range authentications that are applied at the method-level results in a policy conflict.

To resolve the conflicts, you can choose one of the following workaround:

- **Option 1:** Remove the existing association between the POST method and the WRITE Scope or UPDATE Scope through the API Scope details.
- **Option 2:** Delete the WRITE Scope or UPDATE Scope.
- **Option 3:** Remove the Identify and Authorize Application policy from the WRITE Scope or UPDATE Scope.

Exposing a SOAP API to Applications

The API Provider can restrict the exposure of specific operations of a SOAP API to other applications.

Consider you have a native SOAP API created in API Gateway with operations - Operation A, Operation B, and Operation C. You might want to expose the Operation A and Operation C, and restrict the visibility of Operation B to other applications. You can use the **Expose to consumers** button to switch on the visibility of Operation A and Operation C and switch off the visibility of Operation B as required.

If an application attempts to invoke the Operation C in the above SOAP API, API Gateway returns a HTTP response code 404.

By default, the **Expose to consumers** button is switched on for all operations of the SOAP API. Once the API is activated, all of its operations are exposed to registered applications. If you do not want a particular set of operations to be hidden for registered applications, switch off the **Expose to consumers** button in the SOAP API definition.

Note:

Be aware that API Gateway will not allow you to activate a SOAP API if none of the operations in the API are selected for exposing to other applications. You must select at least one operation of the SOAP API to enforce runtime invocations.

> To expose a set of operations of the SOAP API

1. Click **APIs** in the title navigation bar.

This displays a list of APIs available in API Gateway.

2. Click the name of the required API.

This opens the API details page.

3. Click **Edit**.

If the API is active, API Gateway displays a warning message to let you know that the API is active.

4. Click **Operations**.

This displays a list of operations available in the API.

- a. To select an operation, switch on the **Expose to consumers** button next to the operation URI.

You can select one or more operations to expose to other applications.

5. Click **Save** to save the updated API.

Activate the API, if it is not active, to put it into effect.

API Grouping

You can group APIs based on various categories. Categories help consumers locate APIs easily. For example, if you are offering APIs to help your consumers manage their sales and ordering better, classifying the APIs under Sales and Ordering helps them locate these APIs easily.

The default groups available under which you can group the APIs are **Finance Banking and Insurance**, **Sales and Ordering**, **Search**, and **Transportation and Warehousing**. If you want to include more groups you can update the property `apiGroupingPossibleValues` under **Administration > Extended settings** that enables API grouping. You can modify the existing list of groups by deleting or adding new group names as comma separated values in this field. Ensure that the group name does not contain a comma as part of the name.

API grouping can be applied in one of these ways:

- While creating an API from scratch

- While editing an API

You can select one or more groups in the **API grouping** field. When an API is published to API Portal, the published APIs in API Portal are grouped as per the group assigned.

API Tagging

Tags are words or phrases that act as keywords for categorizing, identifying, and organizing APIs.

In API Gateway, you can assign tags to APIs, and their resources, methods, or operations. Tags help to logically categorize APIs in different ways, for example, by usage, owner, consuming application, or other criteria. Tags are especially useful when there are multiple APIs of the same type - it enables to quickly identify a specific API based on the tag assigned to it. For example, you can assign the tag `GET-Methods` to specific GET methods in different REST APIs, and use it to search for the list of REST APIs with the `GET-Methods` tag in API Gateway.

You can use tagging, for example, to do the following:

- Tag and untag REST APIs in API Gateway.
- Use tags to search for multiple resources and methods across the REST APIs that are available in API Gateway.
- Use tags to search for multiple operations across the SOAP APIs that are available in API Gateway.

You can assign one or more tags, remove a tag, and view the tags on the API details page. When a tagged API is published to API Portal, the published API in API Portal is tagged with the same tag defined in API Gateway.

Adding Tags to an API

Pre-requisites:

You must have the Manage APIs functional privilege assigned to perform this task.

Tags are not automatically assigned to APIs, resources, methods, or operations. You can add one or more tags, and you can remove tags from an API, resource, method, or operation at any time.

You can define a set of consistent tags that meets your needs for each API, resource, method, or operation. Using a consistent set of tags makes it easier to manage the APIs, resources, methods, or operations. You can search the APIs, resources, methods, or operations based on the tags you add.

When tagging an API, keep the following points in mind:

- You cannot tag an API if it is in active state. You have to deactivate the API before tagging it.
- You can assign tags to the following APIs and its components:
 - **SOAP API.** You can assign tags to the SOAP API itself and to its operations.
 - **REST API.** You can assign tags to the REST API itself and to its resources, and methods.

- **REST-enabled SOAP API.** You can assign tags to the REST-enabled SOAP API. Also, you can assign tags to the REST resources and methods which correspond to the transformed SOAP operations.
- **OData API.** You can assign tags to the OData API only.
- **WebSocket API.** You can assign tags to the WebSocket API only.
- You can assign tags to an API, resource, method, or operation only.
- When you delete an API, resource, method, or operation in API Gateway, any tags that were assigned to that API, resource, method, or operation are not deleted.

> To tag an API

1. Click **APIs** in the title navigation bar.

A list of all registered APIs appears.




2. Select the required API.

The API details page appears.

3. Click **Edit**.

4. Click **Tags**.

5. To add tags to the API, resource, method, or operation, do the following:

- **Tagging an API.** In the Basic information section, select an existing tag or type the new tag in the **Search and add tags** drop-down list, and then click .
- **Tagging a Resource or Method.** In the Resources and methods section, locate the required resource or method for tagging, select an existing tag or type the new tag in the **Search and add tags** drop-down list, and then click .
- **Tagging an Operation.** In the Operations section, locate the required operation for tagging, select an existing tag or type the new tag in the **Search and add tags** drop-down list, and then click .

The tag is listed next to **Added tags**. To delete a tag, click the **x** icon.

6. Click **Save**.



Exporting APIs

You must have the API Gateway's export assets privilege assigned to perform this task.

Note:

API Gateway supports backward compatibility for all the exported APIs from API Gateway 10.1 version or higher. For more information about exporting and importing APIs, see [“Overview” on page 624](#).

> To export an API

1. Click **APIs** in the title navigation bar.
2. You can export a single or multiple APIs as follows:
 - To export a single API, click  next to the required API.
 - To exported multiple APIs simultaneously, click the check-boxes adjacent to the names of the API, click  and select **Export** from the drop-down list.

The Export archive window appears.

3. Select **Include applications** if you want to export the applications associated with the APIs.
4. Select **Include application registrations** if you want to add the RegisteredApplication object lists to be added to the exported APIs.

This is not selected by default. But if you select **Include applications**, this option also gets selected. When both the options **Include applications** and **Include application registrations** are selected the export archive contains APIs with RegisteredApplication objects and applications referenced by APIs with RegisteredApplication objects.

These exported RegisteredApplication objects are merged with the RegisteredApplication objects in the API Gateway instance where it is imported. The import of these RegisteredApplication objects is based on the value of the import option **Overwrite Registered Application**. The option **Overwrite Registered Application** is not selected by default. But, if you select the option **Overwrite Application** during an import, then **Overwrite Registered Application** is also selected to support backward compatibility.

5. Select **Include groups** if you want to export the groups associated to the team that the APIs belong to.
6. Select **Include users** if you want to export the users associated to the team that the APIs belong to.

Note:

The **Include groups** and **Include users** check boxes appear, only if you have set the "enableTeamWork" extended setting to true.

7. Click **Export**.

The browser prompts you to either open or save the export archive.

8. Select the appropriate option and click **OK**.

Exporting Specifications

For a REST API, you can export specifications in Swagger and RAML formats to your local system. Similarly, for a SOAP API, you can export a specification in WSDL format to your local system. The exported WSDL is in a ZIP format consisting of the WSDL file whereas for Swagger and RAML the respective files are directly exported. API Gateway supports the following versions:

- Swagger 2.0 for a Swagger file
- RAML 0.8 for a RAML file

You can export APIs that have been created from scratch or by importing their respective definitions. The Swagger or RAML definition provides the consumer view on a REST API deployed to the API Gateway. Similarly, the WSDL definition provides the consumer view on a SOAP API. Consumer view indicates that the Swagger, RAML, or WSDL definitions contain the API Gateway endpoint and information about those resources and operations, which are exposed to customers.

Note:

In the downloaded Swagger document, the valid JSON schemas attached to a response or a request does not always appear. Only the valid JSON schemas appear correctly. For any other schema information just the generic JSON schema such as {"type": "object"} appears.

> To export the specification

1. Click **APIs** in the title navigation bar.
2. Select the required API from the list of available APIs .
The API details page for the selected API appears.
3. Click **Documentation**.
4. Based on the type of specification that you have selected to export, select any of the following:
 - **Swagger data** link to export the Swagger specification.
 - **RAML data** link to export the RAML specification.
 - **OpenAPI data** link to export the OpenAPI specification.
 - **Artifacts** link to export the WSDL specification.
 - **OData meta document** link to a zip containing the OData API and metadata document. If the OData API is active, a link to the service document and a link to the metadata document are also displayed.

When downloading a zip file the browser prompts you to either open or save the zip file.

5. Select the appropriate option and click **OK**.

Deleting APIs

Deleting an API permanently removes the API from API Gateway.

When deleting an API, keep the following points in mind:

- You cannot delete an API if it is in active state. You have to deactivate the API before deleting it.
- You must have the Manage APIs functional privilege.

Deleting a Single API

Pre-requisites:

You must have the Manage APIs functional privilege assigned to perform this task.

You delete an API to remove it from API Gateway permanently.

> To delete an API

1. Click **APIs** in the title navigation bar.
A list of all APIs appears.
2. Click the **Delete** icon for the API that you want to delete.
3. Select the **Force delete** option to delete an API forcefully. API Gateway ignores any failures even if the API is used by other applications, and clears all data from the API Gateway Data Store.
4. Click **Yes** in the confirmation dialog.
The API is deleted forcefully.

Deleting Multiple APIs in a Single Operation

Pre-requisites:


You must have the Manage APIs functional privilege assigned to perform this task.

You can bulk delete APIs in API Gateway.

> To delete multiple APIs in a single operation

1. Click **APIs** in the title navigation bar.

A list of all APIs appears.

2. Select the APIs that you want to delete.
3. In the **Menu**  icon, click **Delete**.
4. Select the **Force delete** option to delete APIs forcefully. API Gateway ignores any failures even if the selected APIs are used by other applications, and clears all data from the API Gateway Data Store.
5. Click **Yes** in the confirmation dialog.

The APIs are deleted forcefully from API Gateway.

6. Examine the **Delete APIs report** window and check for any errors that occurred during the deletion process.

The **Delete APIs report** window displays the following information:

Parameter	Description
Name	The name of the deleted API.
Status	The status of the deletion process. The available values are: <ul style="list-style-type: none"> ■ Success ■ Failure
Description	A descriptive information if the deletion fails or if a warning occurs.

API Gateway writes these results to the **Audit logs** dashboard, so you can view them later.

7. Click **Download the detailed report here** to download the detailed report as an HTML file.

Example: Managing an API

This section explains everything you would want to know about an API and how to manage it with an example API `phonestore`. You can model an API that serves to expose API data and functionality as a collection of resources. Each resource is accessible with unique Uniform Resource Identifiers (URLs). In your API, you expose a set of HTTP operations (methods) to perform on a specific resource and capture the request and response messages and status codes that are unique to the HTTP method and linked within the specific resource of the API.

The basic elements of an API are:

- The API itself (for example, `phonestore`)

- Its resource (*phones*), available on the unique base URL (*/phones*)
- The defined HTTP method (*GET*) for accessing the resource (*phones*)
- Parameters for request representations (*412456*)
- A request generated for this method (*Request 123*)
- A response with the status code received for this request (*Response ABCD*)

The example API `phonestore` that we consider here is defined to support an online phone store application. Assume, this sample `phonestore` API currently has a database that defines the various brands of phones, features in the individual phones, and the inventory of each phone. This API is used as a sample to illustrate how to model URL patterns for resources, resource methods, HTTP headers and response codes, content types, and parameters for request representations to resources.

Base URL

The base URL of an API is constructed by the domain, port, and context mappings of the API. For example, if the server name is `www.phonestore.com`, port is `8080`, and the API context is `api`. The full Base URL is:

```
http://www.phonestore.com:8080/api
```

API Parameters

Parameters defined at the higher API level are inherited by all Resources and Methods included in the individual resources.

API Resources

Resources are the basic components of an API. Examples of resources from an online `phonestore` API include a phone, an order from a store, and a collection of customers. After you identify a service to expose as an API, you define the resources for the API.

For example, for the online `phonestore` API, there are a number of ways to represent the data in the phone store database as an API. The verbs in the HTTP request maps to the operations that the database supports, such as `select`, `create`, `update`, `delete`.

Each resource has to be addressed by a unique URI. Along with the URI you're going to expose for each resource, you also need to decide what can be done to each resource. The HTTP methods passed as part of an HTTP request header directs the API on what needs to be done with the addressed resource.

Resource URLs

An URL identifies the location of a specific resource.

For example, for the online `phonestore` API, the resources have the following URLs:

URL	Description
<code>http://www.phonestore.com/api/phones</code>	Specifies the collection of phones contained in the online store.

URL	Description
http://www.phonestore.com/api/phones/412456	Accesses a phone referenced by the product code 412456.
http://www.phonestore.com/api/phones/412456/reviews	Specifies a set of reviews posted for a phone of code 412456.
http://www.phonestore.com/api/phones/412456/reviews/78	Accesses a specific review referenced by the unique ID 78 contained in the reviews of the phone of code 412456.

API Gateway supports the following patterns of resource URL: a collection of resources or a particular resource.

For example, in the online phonestore API, the patterns are as follows:

Collection URL: <http://phonestore.com/api/phones>

Unique URL: <http://phonestore.com/api/phones/412456/features> to retrieve a collection resource describing the key features of phone whose product code is 412456.

Resource Parameters

Parameters defined at the higher Resource level are inherited by all Methods in the particular resource; it does not affect the API.

Resource Methods

Individual resources can define their capabilities using supported HTTP methods. To invoke an API, the client would call an HTTP operation on the URL associated with the API's resource. For example, to retrieve the key feature information for phone whose product code is 412456, the client would make a service call HTTP GET on the following URL:

```
http://www.phonestore.com/phones/412456/features
```

Supported HTTP Methods

API Gateway supports the standard HTTP methods for modeling APIs: GET, POST, PUT, DELETE, and PATCH.

The following table describes the semantics of HTTP methods for our sample Phone Store API:

Resource URI	HTTP Method	Description
/phones/orders	GET	Asks for a representation of all of the orders.
/phones/orders	POST	Attempts to create a new order, returning the location (in the Location HTTP Header) of the newly created resource.

Resource URI	HTTP Method	Description
/phones/orders/{order-id}	GET	Asks for a representation of a specific Order resource.
/phones/orders/{order-id}	DELETE	Requests the deletion of a specified Order resource.
/phones/orders/{order-id}/status	GET	Asks for a representation of a specific Order's current status.
/phones/orders/{order-id}/paymentdetails	GET	Asks for a representation of a specific Order's payment details.
/phones/orders/{order-id}/paymentdetails	PUT	Updates a specific Order's payment details

Method Parameters

Parameters defined at the lower method level apply only to that particular method; it does not affect either the API or the resource.

API Parameters

Parameters specify additional information to a request. You use parameters as part of the URL or in the headers or as components of a message body.

Parameter Levels

A parameter can be set at different levels of an API. When you document a REST API in API Gateway, you define parameters at the API level, Resource level, or Method level to address the following scenarios:

- If you have the parameter applicable to all resources in the API, then you define this parameter at the API level. This indirectly implies that the parameter is propagated to all resources and methods under the particular API.
- If you have the parameter applicable to all methods in the API, then you define this parameter at the resource level. This indirectly implies that the parameter is propagated to all methods under the particular resource.
- If you have the parameter applicable only to a method in the API, then you define this parameter at the method level.

API-level Parameters

Setting parameters at the API level enables the automatic assignment of the parameters to all resources and methods included in the API. Any parameter value you specify at the higher API level overrides the parameter value you set at the lower resource level or the lower method level if the parameter names are the same.

For example, if you have a header parameter called API Key that is used for consuming an API.

```
x-Gateway-APIKey:a4b5d569-2450-11e3-b3fc-b5a70ab4288a
```

This parameter is specific to the entire API and to the individual components, that is resources and methods, directly below the API. Such a parameter can be defined as a parameter at the API level.

At an API level, API Gateway allows you to define the following types of parameters:

- Query-String parameter
- Header parameter

Resource-level Parameters

Setting parameters at the resource level enables the automatic assignment of the parameters to all methods within the resource. Any parameter value you specify at the higher resource level overrides the parameter value you set at the lower method level if the parameter names are the same. In contrast, the lower resource level parameters do not affect the higher API level parameters.

Consider the sample phonestore API maintains a database of reviews about different phones. Here is a request to display information about a particular user review, 78 of the phone whose product code is 412456.

```
GET /phones/412456/user_reviews/78
```

In the example, `/user_reviews/78` parameter narrows the focus of a GET request to review `/78` within a particular resource `/412456`.

This parameter is specific to the particular resource phone whose product code is 412456 and to any individual methods that are directly below the particular resource. Such a parameter can be defined as a parameter at the resource level.

At a resource level, API Gateway allows you to define the following types of parameters:

- Query-String parameter
- Header parameter
- Path parameter

Method-level Parameters

If you do not set parameters at the API level or resource level, you can set them at a method level. Parameters you set at the method level are used for the HTTP method execution. They are useful to restrict the response data returned for a HTTP request. Any parameter value you specify at the lower method level is overridden by the value set at higher API-level parameter or the higher resource-level parameter if the names are the same. In contrast, the lower method-level parameters do not affect the higher API-level or resource-level parameters.

For example, the phonestore API described might have a request to display information contributed by user Allen in 2013 about a phone whose product code is 412456.

```
GET /phones/412456/user_reviews/78?year=2013&name=Allen
```

In this example, `year=2013` and `name=Allen` narrow the focus of the GET request to entries that user Allen added to user review 78 in 2013.

At a method level, API Gateway allows you to define the following types of parameters:

- Query-String parameter
- Header parameter

Parameter Types

API Gateway supports three types of parameters in REST API: Query-String, Header, and Path.

Let's have a look at different parameter types to see how they can be used for parameterizing the resources.

Query-String Parameters

Query-String parameters are appended to the URI after a `?` with name-value pairs. The name-value pairs sequence is separated either by a semicolon or an ampersand.

For instance, if the URL is `http://phonestore.com/api/phones?itemID=itemIDValue`, the query parameter name is `itemID` and value is the `itemIDValue`. Query parameters are often used when filtering or paging through HTTP GET requests.

Now, consider the online `phonestore` API. A customer, when trying to fetch a collection of phones, might wish to add options, such as, `android v4.3 OS` and `8MP camera`. The URI for this resource would look like:

```
/phones?features=androidosv4.3&cameraresolution=8MP
```

You can also use query string to invoke the required resource of an API by appending API Key to `?` like the example seen below:

```
http://pie-3HKYMH2:5555/gateway/PetstoreAPI/1.0.3/store/  
inventory?APIKey=faab7ac6-97a4-4228-908d-f1930faba470
```

Header Parameters

Header parameters are HTTP headers. Headers often contain metadata information for the client, or server.

```
x-Gateway-APIKey:a4b5d569-2450-11e3-b3fc-b5a70ab4288a
```

You can create custom headers, as required. As a best practice, Software AG recommends that you prefix the header name with `x-`.

HTTP/1.1 defines the headers that can appear in a HTTP response in three sections of RFC 2616: 4.5, 6.2, and 7.1. Examine these codes to determine which are appropriate for the API.

Path Parameters

Path parameters are defined as part of the resource URI. For example, the URI can include `phones/item`, where `/item` is a path parameter that identifies the item in the collection of resource `/phones`. Because path parameters are part of the URI, they are essential in identifying the request.

Now, consider the online `phonestore` API. A customer wishes to fetch details about a phone `{phone-id}` whose product code is 412456. The URI for this resource would look like: `/phones/412456`

Important:

As a best practice, Software AG recommends that you adopt the following conventions when specifying a path parameter in the resource URI:

- Append a path parameter variable within curly `{}` brackets.
- Specify a path parameter variable such that it exactly matches the path parameter defined at the resource level.

Parameter Data Types

When you add a parameter to the API, you specify the parameter's data type. The data type determines what kind of information the parameter can hold.

API Gateway supports the following data types for parameters:

Data Type	Description
String	Specifies a string of text.
Date	Specifies a date stamp that represents a specific date. The date input parameters allow year, month, and day input. This data type only accepts date values in the format <code>yyyy-mm-dd</code>
Time	Specifies a timestamp that represents a specific time. The time input parameters allow hour and minute. This data type only accepts date values in the format <code>hh:mm:ss</code>
Date/Time	Specifies a timestamp that represents a specific date and/or time. The date/time input parameters allow year, month, and day input as well as hour and minute. Hour and minute default to 0. This data type only accepts date values in the format <code>yyyy-mm-dd</code> ; and time values in the format <code>hh:mm:ss</code>
Integer	Specifies an integer value for the data type. This is generally used as the default data type for integral values.
Double	Specifies the double data type value. This is a double-precision 64-bit IEEE 754 floating point and is generally used as the default data type for decimal values.
Boolean	Specifies a <code>true</code> or <code>false</code> value.

Supported HTTP Status Codes

An API response returns a HTTP status code that indicates success or failure of the requested operation.

API Gateway allows you to specify HTTP codes for each method to help clients understand the response. While responses can contain an error code in XML or other format, clients can quickly and more easily understand an HTTP response status code. The HTTP specification defines several status codes that are typically understood by clients.

API Gateway includes a set of predefined content types that are classified in the following taxonomy categories:

Category	Description
1xx	Informational.
2xx	Success.
3xx	Redirection. Need further action.
4xx	Client error. Correct the request data and retry.
5xx	Server error.

HTTP/1.1 defines all the legal status codes. Examine these codes to determine which are appropriate for your API.

Now, consider the online phonestore API. The following table describes the HTTP status codes that each of the URIs and HTTP methods combinations will respond:

Resource URI	Supported HTTP Methods	Supported HTTP Status Codes
/phones/orders	GET	200 (OK, Success)
/phones/orders	POST	201 (Created) if the Order resource is successfully created, in addition to a Location header that contains the link to the newly created Order resource; 406 (Not Acceptable) if the format of the incoming data for the new resource is not valid
/phones/orders/{order-id}	GET	200 (OK); 404 (Not Found) if Order Resource not found
/phones/orders/{order-id}	DELETE	200 (OK); 404 (Not Found) if Order Resource not found
/phones/orders/{order-id}/status	GET	200 (OK); 404 (Not Found) if Order Resource not found

Resource URI	Supported HTTP Methods	Supported HTTP Status Codes
/phones/orders/{order-id}/paymentdetails	GET	200 (OK); 404 (Not Found) if Order Resource not found
/phones/orders/{order-id}/paymentdetails	PUT	201 (Created); 406 (Not Acceptable) if there is a problem with the format of the incoming data on the new payment details; 404 (Not Found) if Order Resource not found

Sample Requests and Responses

To illustrate the usage of an API, you provide a sample request and response messages. Consider the sample phonestore API that maintains a database of phones in different brands. The phonestore API might provide the following examples to illustrate its usage:

Sample 1 - Retrieve a list of phones

Client Request

```
GET /phones HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.api.phonestore.com
Accept-Language: en-us
Accept-Encoding: text/xml
Connection: Keep-Alive
```

Server Response

```
HTTP/1.1 200 OK
Date: Mon, 29 August 11:53:27 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Mon, 18 July 2016 09:18:16 GMT
Content-Length: 356
Content-Type: text/xml
<phones>
  <phone>
    <name>Asha</name>
    <brand>Nokia</brand>
    <price currency="irs">11499</price>
    <features>
      <camera>
        <back>3</back>
      </camera>
      <memory>
        <storage scale="gb">8</storage>
        <ram scale="gb">1</ram>
      </memory>
      <network>
        <gsm>850/900/1800/1900 MHz</gsm>
      </network>
    </features>
  </phone>
```

```

<phone>
  <name>Nexus7</name>
  <brand>Google</brand>
  <price currency="irs">16499</price>
  <features>
    <camera>
      <front>1.3</front>
      <back>5</back>
    </camera>
    <memory>
      <storage scale="gb">16</storage>
      <ram scale="gb">2</ram>
    </memory>
    <network>
      <gsm>850/900/1800/1900 MHz</gsm>
      <HSPA>850/900/1900 MHz</HSPA>
    </network>
  </features>
</phone>
</phones>

```

Client Request

```

GET /phones/phone-4156 HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.api.phonestore.com
Accept-Language: en-us
Accept-Encoding: text/xml
Connection: Keep-Alive

```

Server Response

```

POST /phones/phone HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.api.phonestore.com
Accept-Language: en-us
Accept-Encoding: text/xml
Content-Length: 156
Connection: Keep-Alive
<phones>
  <phone>
    <name>iPhone5</name>
    <brand>Apple</brand>
    <price currency="irs">24500</price>
    <features>
      <camera>
        <front>1.2</front>
        <back>8</back>
      </camera>
      <memory>
        <storage scale="gb">32</storage>
        <ram scale="gb">2</ram>
      </memory>
      <network>
        <gsm>850/900/1800/1900 MHz</gsm>
        <HSPA>850/900/1900 MHz</HSPA>
      </network>
    </features>
  </phone>
</phones>

```


Sample 3 - Create a phone

```
POST /phones/phone HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.api.phonestore.com
Accept-Language: en-us
Accept-Encoding: text/xml
Content-Length: 156
Connection: Keep-Alive
<phones>
  <phone>
    <name>iPhone5</name>
    <brand>Apple</brand>
    <price currency="irs">24500</price>
    <features>
      <camera>
        <front>1.2</front>
        <back>8</back>
      </camera>
      <memory>
        <storage scale="gb">32</storage>
        <ram scale="gb">2</ram>
      </memory>
      <network>
        <gsm>850/900/1800/1900 MHz</gsm>
        <HSPA>850/900/1900 MHz</HSPA>
      </network>
    </features>
  </phone>
</phones>
```

Server Response

```
HTTP/1.1 200 OK
Date: Mon, 29 August 11:53:27 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 18 June 2014 09:18:16 GMT
Content-Type: text/xml
Content-Length: 15
<id>2122</id>
```


5 Policies

■ Overview	364
■ Policy Validation and Dependencies	366
■ Managing Threat Protection Policies	371
■ System-defined Stages and Policies	382
■ Managing Global Policies	542
■ Managing API-level Policies	557
■ Managing Scope-level Policies	560
■ Managing Policy Templates	565
■ Supported Alias and Policy Combinations	576

Overview

API Gateway provides a policy framework to manage and secure APIs.

A policy can be enforced on an API to perform specific tasks, such as transport, security, logging, routing of requests to target services, and transformation of data from one format to another. You can also define a policy to evaluate and process the various API invocations at run-time. For example, a policy could instruct API Gateway to perform any of the following tasks and prevent malicious attacks:

- Verify that the requests submitted to an API come from applications that are authenticated and authorized using the specified set of identifiers in the HTTP header to access and use the particular API.
- Validate digital signatures in the security header of request and response messages.
- Monitor a user-specified set of run-time performance conditions and limit the number of invocations during a specified time interval for a particular API and for applications, and send alerts to a specified destination when these performance conditions are violated.
- Log the request and response messages, and the run-time performance measurements for APIs and applications.

Policies are grouped into stages as per their usage. For example, the policies in a **Threat Protection** stage can be enforced for all APIs to protect against malicious attacks that could cause problems such as, large and recursive payloads, viruses, scanning with external systems, and SQL injections. The policies in the **Identify and Access** stage can be enforced on an API to specify the kind of identifiers that are used to identify the application and authorize it against all applications registered in API Gateway.

Policy enforcement mode

You can enforce policies in an API in the following ways:

- **Global Policies:** You can apply a global policy to all APIs or the selected set of APIs. You do this by configuring the filters for the API and the policy configuration in the Global Policy details page. The global policies apply globally to the selected APIs.
- **Policy Templates:** You can apply one or more policy templates to an API. You do this by applying the policy templates in the API details page. These policy templates apply at the API-level, and can be customized to suit the needs of a particular API.
- **API-specific Policies:** You can apply one or more individual policies to an API. You do this by applying the policies in the API details page. These policies apply at the API-level, and can be customized to suit the needs of a particular API.
- **API Scope-specific Policies:** You can apply one or more policies at the scope-level of an API. You do this by defining the API scopes with a collective set of resources, methods, or operations in the API details page. These policies apply at the corresponding resource-level, method-level, or operation-level, and can be customized to suit the needs of an individual API scope.

After you apply the policies both globally (through global policies) and directly (through API-level policies and scope-level policies) to an API, API Gateway determines the effective set of policies for that API by taking into account the precedence of policy enforcement at the API-level, the policy stages, the priority of policies, run-time constraints, and the status (activated or deactivated) of any applied global policy.

When you apply the Transport policy at the global level, the Transport policy applied at the API level is in the disabled state. When you try deleting the API-level Transport policy that is in the disabled state an error displays and you are not allowed to delete this policy as the API-level Transport policy is required and gets enforced when you deactivate the global policy.

Policy hierarchy

You can enforce policies on an API at the following levels:

- **Global Policy Enforcement:** This enforcement applies globally to all APIs defined in API Gateway.
- **API-level (API-specific) Policy Enforcement:** This enforcement applies to all resources and its nested methods of a REST API, or all operations of a SOAP API.
- **Resource-level (Scope-specific) Policy Enforcement:** Applicable only for REST APIs. This enforcement applies to one or more resources and its nested methods in the REST API.
- **Method-level (Scope-specific) Policy Enforcement:** Applicable only for REST APIs. This enforcement applies to one or more methods nested within a resource in the REST API.

-OR-

Operation-level (Scope-specific) Policy Enforcement: Applicable only for SOAP APIs. This enforcement applies to one or more operations in the SOAP API.

For example, if an API was given the **Identify and Authorize Application** policy at the following policy enforcement levels:

1. Global Policy Enforcement
2. API-level Policy Enforcement
3. Resource-level Policy Enforcement
4. Method-level Policy Enforcement (or) Operation-level Policy Enforcement

The precedence of the policy enforcement which are effective for the API at run-time is as follows:

1. Global Policy Enforcement
2. Method-level Policy Enforcement (or) Operation-level Policy Enforcement
3. Resource-level Policy Enforcement
4. API-level Policy Enforcement

If the API has the Identify and Authorize Application policy applied through both a global policy and at the API-level, API Gateway does not show conflict. The Identify and Authorize Application policy applied through the global policy takes precedence and is processed at run-time.

Similarly for a REST API, Identify and Authorize Application policy is applied through a scope-level policy (at the resource-level) and also at the API-level, the Identify and Authorize Application policy applied through the scope-level policy takes precedence and is processed at run-time.

Transaction logging policy

API Gateway provides a system global policy, **Transaction logging**, which is shipped with the product. By default, the policy is in the **Inactive** state. The transaction logging policy has standard filters and log invocation policy that log request or response payloads to a specified destination. You can edit this policy to include additional filters or modify the policy properties but you cannot delete this policy. You can activate this policy in the **Policies > Global policies** page or through the Global Policy details page. Activating the policy enforces it on all APIs in API Gateway based on the configured filters and logs transactions across all the APIs. If you have multiple log invocation policies assigned to an API, the policies are compiled into a single policy and the one transaction log is created per destination.

Policy Validation and Dependencies

When you enforce a policy to govern an API at run-time, API Gateway validates the policies to ensure that:

- Any policy (for example, Log Invocation) that can appear in an API multiple times is allowed to appear multiple times.
- For policies (for example, Require HTTP / HTTPS) that can appear only once in an API, API Gateway issues an error message.
- For policies (for example, Monitor Service Level Agreement) that are dependent and use another policy in conjunction (for example, Identify and Authorize Application) in an API, API Gateway prompts you with a warning message to include the dependent policy.

When you save an API, API Gateway combines the policies from all of the global and direct policies that apply to the API (that is, at the API-level) and generates what is called the *effective policy* for the API. For example, let's say your REST API is within the scope of two policies: one policy that performs a logging task and another policy that performs a security task. When you save the REST API, API Gateway automatically combines the two policies into one effective policy. The effective policy, which contains both the logging task and the security task, is the policy that API Gateway actually uses to publish the REST API.

When API Gateway generates the effective policy, it validates the resulting policy to ensure that it contains no conflicting or incompatible policies.

If the policy contains conflicts or inconsistencies, API Gateway computes the effective API policy according to policy resolution rules. For example, an effective API policy can include only one Identify and Authorize Application policy. If the resulting policy list contains multiple Identify and Authorize Application policies, API Gateway shows the conflict by including an including a Conflict (▲) icon next to the name of the conflicting policies in the effective policy.

The following table shows:

- Policy dependencies (that is, whether a policy must be used in conjunction with another particular policy).
- Conflicting or incompatible policies.
- Whether a policy can be included multiple times in a single API. If a policy cannot be included multiple times in a single API, API Gateway selects one (depending on the precedence of the policy at the enforcement level) for the effective policy and processes at run-time.

Policy Validation and Dependencies

Policy	Applicable API Type	Dependent Policy	Mutually Exclusive Policy	Can include multiple times in an API?
Authorize User	REST SOAP	Identify and Authorize Application	None.	No. API Gateway includes only one policy in the effective policy.
Conditional Error Processing	REST SOAP	None.	None.	Yes. API Gateway includes all Conditional Error Processing policies in the effective policy.
Content-based Routing	REST SOAP	None.	Straight Through Routing, Load Balancer Routing, Dynamic Routing, Context-based Routing	No. API Gateway includes only one policy in the effective policy.
Context-based Routing	REST SOAP	None.	Straight Through Routing, Load Balancer Routing, Dynamic Routing, Content-based Routing	No. API Gateway includes only one policy in the effective policy.
Custom HTTP Header	REST SOAP	None.	None.	No. API Gateway includes only one policy in the effective policy.
Data Masking (Error Handling)	REST SOAP	None.	None.	No. API Gateway includes only one policy in the effective policy.
Data Masking (Response Processing)	REST SOAP	None.	None.	No. API Gateway includes only one policy in the effective policy.

Policy	Applicable API Type	Dependent Policy	Mutually Exclusive Policy	Can include multiple times in an API?
Data Masking (Request Processing)	REST SOAP	None.	None.	No. API Gateway includes only one policy in the effective policy.
Dynamic Routing	REST SOAP	None.	Straight Through Routing, Load Balancer Routing, Content-based Routing, Context-based Routing	No. API Gateway includes only one policy in the effective policy.
Enable HTTP / HTTPS	REST SOAP	None.	None.	No. API Gateway includes only one policy in the effective policy.
Enable JMS / AMQP	REST SOAP	None	None	No. API Gateway includes only one policy in the effective policy.
Identify and Authorize Application	REST SOAP	Inbound Authentication - Message policy is required if Identification Type is configured as WS Security Username Token or WS Security X.509 Certificate or Kerberos Token for SOAP-based APIs.	None.	No. API Gateway includes only one policy in the effective policy.
Inbound Authentication - Message	SOAP	None.	None.	No. API Gateway includes only one policy in the effective policy.
Inbound Authentication - Transport	REST SOAP	None.	None.	No. API Gateway includes only one policy in the effective policy.

Policy	Applicable API Type	Dependent Policy	Mutually Exclusive Policy	Can include multiple times in an API?
Invoke webMethods IS (Response Processing)	REST SOAP	None.	None.	Yes. API Gateway includes all Invoke webMethods IS policies in the effective policy.
Invoke webMethods IS (Request Processing)	REST SOAP	None.	None.	Yes. API Gateway includes all Invoke webMethods IS policies in the effective policy.
JMS/AMQP Properties	REST SOAP	JMS/AMQP Routing	None	No. API Gateway includes only one policy in the effective policy.
JMS/AMQP Routing	REST SOAP	None	Straight Through Routing, Dynamic Routing, Content-based Routing, Context-based Routing	No. API Gateway includes only one policy in the effective policy.
Load Balancer Routing	REST SOAP	None.	Straight Through Routing, Dynamic Routing, Content-based Routing, Context-based Routing	No. API Gateway includes only one policy in the effective policy.
Log Invocation	REST SOAP	None.	None.	Yes. API Gateway includes all Log Invocation policies in the effective policy.
Monitor Service Performance	REST SOAP	None.	None.	Yes. API Gateway includes all Monitor Service Performance policies in the effective policy.
Monitor Service Level Agreement	REST SOAP	Identify and Authorize Application	None.	Yes. API Gateway includes all Monitor Service Level Agreement policies in the effective policy.

Policy	Applicable API Type	Dependent Policy	Mutually Exclusive Policy	Can include multiple times in an API?
Outbound Authentication - Message	SOAP	None.	None.	No. API Gateway includes only one policy in the effective policy.
Outbound Authentication - Transport	REST SOAP	None.	None.	No. API Gateway includes only one policy in the effective policy.
Response Transformation	REST SOAP	None.	None.	Yes. API Gateway includes all XSLT Transformation policies in the effective policy.
Request Transformation	REST SOAP	None.	None.	Yes. API Gateway includes all XSLT Transformation policies in the effective policy.
Service Result Cache	REST SOAP	None.	None.	No. API Gateway includes only one policy in the effective policy.
Set Media Type	REST	None.	None.	No. API Gateway includes only one policy in the effective policy.
Straight Through Routing	REST SOAP	None.	Load Balancer Routing, Dynamic Routing, Content-based Routing, Context-based Routing	No. API Gateway includes only one policy in the effective policy.
Throttling Traffic Optimization	REST SOAP	Identify and Authorize Application	None.	Yes. API Gateway includes all Throttling Traffic Optimization policies in the effective policy.
Validate API Specification (Response Processing)	REST SOAP	None.	None.	No. API Gateway includes only one policy in the effective policy.

Policy	Applicable API Type	Dependent Policy	Mutually Exclusive Policy	Can include multiple times in an API?
Validate API Specification (Request Processing)	REST SOAP	None.	None.	No. API Gateway includes only one policy in the effective policy.

Managing Threat Protection Policies

Threat protection policies prevent malicious attacks from client applications that typically involve large, recursive payloads, and SQL injections. You can limit the size of things, such as maximum message size, maximum number of requests, and maximum node depth and text node length, in the XML document. You can configure the global threat protection policies and rules for all the incoming requests that comes through the external port of API Gateway. These policies and rules are enforced by API Gateway based on your configuration.

You must have the API Gateway's manage threat protection functional privilege to configure the following policies and rules.

- Global Denial of Service
- Denial of Service by IP
- Rules

In addition, the API Gateway administrator can configure the necessary mobile devices and applications for which you want to deny the access, configure and customize the deny and alert rules, and manage the denied IPs.

Note:

- If the API Gateway instances used for Threat protection are clustered using TSA, and if you apply threat protection policy configuration in one of the API Gateway instances, the other API Gateway instances are updated automatically.
- If the API Gateway instances used for Threat Protection are not clustered using TSA, then you need to apply the required threat protection policy configurations in each of the API Gateway instance.

Basically, when you configure the threat protection policy in a clustered setup, you specify the limitations (such as number of requests and concurrent request) that an API Gateway instance in the cluster can handle during a specified time interval. Hence, if you add X number of API Gateway instances, the limitations set in the configuration also increases by X times.

For example, if you have two API Gateway instances and set the limitations as 100 requests per minute, then the API Gateway instances should be able to handle 200 requests per minute. When you add one more API Gateway instance, the processing capacity also increases to 300 requests per minute. Here, the API Gateway cluster used for Threat Protection does not act as a single unit.

Note:

When you have configured a load balancer, the load balancer exposes the actual client IP address using the X-Forwarded-For (XFF) headers. The `watt.server.enterprisegateway.ignoreXForwardedForHeader` property specifies whether API Gateway uses or ignores the IP address in the XFF headers. By default, API Gateway ignores the client IP address and so the `watt.server.enterprisegateway.ignoreXForwardedForHeader` property is set to true. If you want API Gateway to use the actual client IP address present in the XFF, then set the `watt.server.enterprisegateway.ignoreXForwardedForHeader` property to false.

Configuring Global Denial of Service Policy

You can configure this policy in API Gateway to prevent Denial of Service (DoS) attacks. One form of DoS attack occurs when a client floods a server with many requests in an attempt to interfere with server processing. Using API Gateway, you can limit the number of requests that API Gateway accepts within a specified time interval and the number of requests that it can process concurrently. By specifying these limits, you can protect API Gateway from DoS attacks.

You can configure API Gateway to limit the total number of incoming requests from the external ports. For example, you might want to limit the total number of requests received to 1000 requests in 10 seconds, and limit the number of concurrent requests to 100 requests in 10 seconds. When API Gateway detects that a limit has been exceeded, it blocks the exceeding requests for a specific time interval and displays an error message to the client based on your configuration. You can also configure a list of trusted IP addresses so that the requests from these IP addresses are always allowed and not blocked.

➤ To configure global denial of service policy

1. Click **Policies** in the title navigation bar.
2. Select **Threat protection > Global denial of service**.
3. Set the **Enable** button to the **On** position to enable the policy.
4. Type the maximum number of requests, in the **Maximum requests** field, that API Gateway can accept from any IP address in a given time interval.
5. Specify time in seconds, in the **In (seconds)** field, in which the maximum requests have to be processed.
6. Type the maximum number of concurrent requests, in the **Maximum requests in progress** field, that API Gateway can process concurrently.
7. Specify the time in minutes, in the **Block intervals (minutes)** field, for which you want requests to be blocked.
8. Type the alert message text, in the **Error message** field, to be displayed when the policy is breached.


9. Add IP addresses, in the **Trusted IP addresses** field, that can be trusted and are always allowed.
 - API Gateway supports IPv4 and IPv6 addresses in the trusted IP addresses lists.
 - You can specify a range of IP addresses using the classless inter-domain routing (CIDR) notation. To specify an IP address range, type the first IP address in the range followed by a forward slash (/) and a CIDR suffix.

Example IPv4 address range:

- 192.168.100.0/22 represents the IPv4 addresses from 192.168.100.0 to 192.168.103.255
- 148.20.57.0/30 represents the IPv4 addresses from 148.20.57.0 to 148.20.57.3

Example IPv6 address range:

- f000::/1 represents the IPv6 addresses from f000:: to ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff.
- 2001:db8::/48 represents the IPv6 addresses from 2001:db8:0:0:0:0:0:0 to 2001:db8:0:ffff:ffff:ffff:ffff:ffff.

Click  to add more than one IP address.

10. Click **Save**.

Configuring Denial of Service by IP Policy

You can configure this policy in API Gateway to prevent Denial of Service (DoS) attacks. One form of DoS attack occurs when a particular client floods a server with many requests in an attempt to interfere with server processing and not letting other clients in accessing the server. Using Denial of Service (DoS) by IP policy, you can limit the number of requests that API Gateway accepts from a particular IP address within a specified time interval and the number of requests that it can process concurrently from any IP address. By specifying these limits, you can protect API Gateway from DoS attacks by a particular IP address. When API Gateway detects that a limit has been exceeded, it blocks or denies the requests from that particular IP address and displays an error message to the client based on your configuration. You can also configure a list of trusted IP addresses so that the requests from these IP addresses are always allowed and not denied.

Note:

When you configure a load balancer, you need to insert the XFF headers on the load balancer to track the actual client IP address. When you use Load Balancer for high availability between the API Gateway instances, by default for all the incoming request, the source IP address will be the load balancer's IP address instead of the actual client IP address. In such scenario, when the Denial of Service by IP policy is enforced, all incoming requests will be denied irrespective of the problematic client. So, to prevent DoS attack from a problematic client, you need to consider the XFF headers that are inserted on the load balancer. This is achieved by setting `watt.server.enterprisegateway.ignoreXForwardedForHeader` property to false. When this setting

is configured, the incoming request header will have the XFF header and tracks actual client IP address, which in turn allows you to configure DoS by IP.

➤ **To configure the denial of service by IP policy**

1. Click **Policies** in the title navigation bar.
2. Select **Threat protection > Denial of service by IP**.
3. Set the **Enable** button to the **On** position to enable the policy.
4. Type the maximum number of requests, in the **Maximum requests** field, that API Gateway can accept from a specific IP address in a given time interval.
5. Specify time in seconds, in the **In (seconds)** field, in which the maximum requests have to be processed.
6. Type the maximum number of requests, in the **Maximum requests in progress** field, that API Gateway can process concurrently from any single IP address.
7. Select one of the following actions to be taken when the number of requests from a non-trusted IP address exceeds the specified limits:
 - **Add to deny list** to permanently deny future requests from the IP address.
 - **Block** temporarily block requests from this IP address.
8. Type the alert message text, in the **Error message** field, to be displayed when the policy is breached.
9. Add IP addresses, in the **Trusted IP Addresses** field, that can be trusted and not blocked.
 - API Gateway supports IPv4 and IPv6 addresses in the trusted IP addresses lists.
 - You can specify a range of IP addresses using the classless inter-domain routing (CIDR) notation. To specify an IP address range, type the first IP address in the range followed by a forward slash (/) and a CIDR suffix

Example IPv4 address range:

 - 192.168.100.0/22 represents the IPv4 addresses from 192.168.100.0 to 192.168.103.255
 - 148.20.57.0/30 represents the IPv4 addresses from 148.20.57.0 to 148.20.57.3

Example IPv6 address range:

 - f000::/1 represents the IPv6 addresses from f000:: to ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff.
 - 2001:db8::/48 represents the IPv6 addresses from 2001:db8:0:0:0:0:0:0 to 2001:db8:0:ffff:ffff:ffff:ffff:ffff.

Click  to add more than one IP address.

10. Click **Save**.


Managing Denied IP List

The Denied IPs section has a list of client IPs that were denied access due to breach of denial of service by IP policy. You can delete the IP from the denial list and make it available on client's request.

> To manage the denied IP list

1. Click **Policies** in the title navigation bar.
2. Select **Threat protection > Denied IPs**.

This displays a list of IP address that are denied from access.

3. Click  in the **Action** column so that the specified IP can be made available.

Configuring Rules

You can configure rules to filter malicious requests based on message size, authentication requests, requests from specific mobile devices and applications that could be harmful, requests that could cause an SQL injection attack, requests on anti-virus scan, XML / JSON requests, or use custom filters to avoid malicious attacks.

API Gateway applies rules in the order in which they are displayed on the **Threat Protection > Rules** screen. Because a violation of a denial rule stops API Gateway from processing a request, hence it is important to prioritize the rules based on the order in, which you want them to be executed. The API Gateway processes denial rules followed by the alert rules.

> To configure rules

1. Click **Policies** in the title navigation bar.
2. Select **Threat protection > Rules**.

This displays a list of rules that are already configured.
3. Click **Add rule**.
4. In the Rule properties section provide the following information:
 - a. Type a name for the rule in the **Rule name** field.

Valid rule names:

- Must be unique.
 - Must not be empty.
 - Must not contain spaces.
 - Must not contain the special characters - ? ~ ` ! @ # \$ % ^ & * () - + = { } | [] \ \ : \ " ; ' < > , /
- b. Type a description for the rule in the **Description** field.
- c. Select an action to be followed when the policy is violated:
- **Deny request and alert** to deny the access and send an alert when the policy is violated.
 - **Alert** to allow the request and send an alert when the policy is violated.
- d. Type the alert message text, in the **Error message** field, to be displayed when the policy is violated.
- e. Select the required **Request type** to which you want to apply the rule and provide the additional information required.

The available values are:

- **ALL**: Applies the rule to all requests.
 - **REST**: Applies the rule to all REST requests.
 - **SOAP**: Applies the rule to all SOAP requests.
 - **INVOKE**: Applies the rule to all INVOKE requests.
 - **CUSTOM**: Applies the rule to all requests specified by the custom directives. You can use this option if you want a single rule applied for multiple request types and custom directives.
- f. Provide the following information to filter the requests depending on the **Request type** selected:
- **Resource path**: Provide the **Resource path** for the REST, SOAP, INVOKE, or CUSTOM Request type selected to filter the requests based on the resource being requested. The format for the REST, SOAP, and INVOKE request types is `folder_name/service_name` and the format for a CUSTOM request type is `given_directive/service_name`. You can add multiple resource paths using the **Add** button.
 - **Custom directives**: Provide the custom directives for the CUSTOM Request type to filter the incoming requests. For example, if you provide `gateway` as the directive, the rule applies to all these requests that are received in API Gateway with the directive `gateway`. You can add multiple directives using the **Add** button.

5. Configure the required filters as follows:

- **Alert settings.** Select one of the following options:
 - **Default:** Sets the alert settings configured in the **Global Policies > Alert settings** section.
 - **Custom:** You can specify this option to use the custom alert settings and provide the required information.
 - **Alert destination:** Specify the alert destination. Values are **Email** and **Flow service**.

If you select **Email**, provide the email ids to which the alert notification has to be sent.

If you select **Flow service**, a flow service is invoked. Specify the name of the flow service. You can create a flow service using the `pub.security.enterpriseGateway:alertSpec` as the signature of the service and or use the pre-defined flow service, `pub.apigateway.threatProtection:violationListener`. When you use the pre-defined service, the alerts are saved in Internal Data Store and displayed in API Gateway Dashboard. For more information on the `pub.security.enterpriseGateway:alertSpec` specification, refer to the Integration Server Build-In Services Reference Guide.

- Provide the user, who has permissions to execute the service, as the user type. For example, `Administrator`.

Send alert: Select a condition depending on when you want the alert to be sent. Available values are **On rule violation** which sends an alert every time a request violates a rule or **Every** and specify the time interval (in minutes), which send alerts at specified intervals.

- **Message size filter**

- Set the **Enable** button to the **On** position to enable the filter.
- Type the maximum size allowed for HTTP and HTTPS requests in the **Maximum message size (MB)** field.

If the request is larger than the size specified in this limit, the request violates the rule and API Gateway performs the configured action.

- **OAuth filter**

- Set the **Enable** button to the **On** position to enable the filter.
- Set the **Require OAuth credentials** toggle button to the **On** position. This implies the request should contain the OAuth credentials else the request would be denied.

- **Mobile application protection filter**

You can configure this filter to disable access for certain mobile application versions on a predefined set of mobile platforms. By disabling access to these versions, you are ensuring that all users are using the latest versions of the applications and taking advantage of the latest security and functional updates.

- Set the **Enable** button to the **On** position to enable the filter.

- Select the device type.
- Select the mobile application.
- Select the operator condition =, >, <, >=, <= or <>.
- Type the mobile application version.

You can add multiple entries by clicking .

■ **SQL injection protection filter**

You can use the SQL injection protection filter to block requests that could possibly cause an SQL injection attack. When this filter is enabled, API Gateway checks each request message for specific patterns of characters or keywords that are associated with potential SQL injection attacks. If a match is found in the request parameters or payload, API Gateway blocks the request from further processing.

- Set the **Enable** button to the **On** position to enable the selected filter.
- Select the required filters as follows:
 - Select **Database-specific SQL injection protection** and select a database against which specific parameters needs to be checked.

When enabled, API Gateway checks the incoming payload based on the specified database and GET or POST request parameters. If no parameter is specified, all input parameters are checked for possible SQL injection attack.

- Select **Standard SQL injection protection** and specify one or more GET or POST request parameters that could be present in the incoming requests. Parameters can contain only alphanumeric characters, dollar sign (\$), and underscore (_).

You can add multiple entries by clicking .

■ **Anti virus scan filter**

You can use the antivirus scan filter to configure API Gateway to interact with an Internet Content Adaptation Protocol (ICAP)-compliant server. An ICAP server is capable of hosting multiple services that you can use to implement features such as virus scanning or content filtering. Using the antivirus scan filter, API Gateway can leverage the ICAP protocol to scan all incoming HTTP requests and payloads for viruses.

- Set the **Enable** button to the **On** position to enable the filter.
- Type the antivirus ICAP engine name in the **ICAP name** field.
- Type the host name or IP address of the machine on which the ICAP server is running in the **ICAP host name or IP address** field.
- Type the port number on which the ICAP server is listening in the **ICAP port number** field.

- Type the name of the service exposed by the ICAP server that you can use to scan your payload for viruses in the **ICAP service name** field.


■ JSON threat protection filter

You can use this filter to block attacks through JSON payload that have infinitely long strings or deeply nested payloads. Software AG recommends that this protection should be combined with message size filter to identify infinite payloads.

Set the **Enable** button to the **On** position to enable the filter.

You can specify any of these parameters as filter criteria. If you do not specify a value, the system applies a default value of -1, which means an unlimited value.

Field	Description
Container depth	Specifies the maximum allowed containment depth, where the containers are objects or arrays. For example, an array containing an object which contains an object would result in a containment depth of 3.
Object entry count	Specifies the maximum number of entries allowed in an object.
Object entry name length field	Specifies the maximum string length allowed for a property name within an object.
Array element count	Specifies the maximum number of elements allowed in an array.
String value length	Specifies the maximum length allowed for a string value.
Applicable content type	Specify any other content types to be included in the filter.

You can add more entries by clicking .


■ XML threat protection filter

You can use this filter to block attacks through XML payload that have infinitely long strings or deeply nested payloads. Software AG recommends that this protection should be combined with message size filter to identify infinite payloads.

Set the **Enable** button to the **On** position to enable the filter.

You can specify any of these parameters as filter criteria. If you do not specify a value, the system applies a default value of -1, which the system equates to no limit.

Field	Description
Namespace prefix length	Specifies a limit on the maximum number of characters permitted in the namespace prefix in the XML document.

Field	Description
Namespace URI length	Specifies a character limit for any namespace URIs present in the XML document.
Namespace count per element	Specifies the maximum number of namespace definition allowed for any element.
Child count	Specifies the maximum number of child elements allowed for any element.
Attribute name length	Specifies a limit on the maximum number of characters permitted in any attribute name in the XML document.
Attribute value length	Specifies a limit on the maximum number of characters permitted in any attribute value in the XML document.
Attribute count per element	Specifies the maximum number of attributes allowed for any element.
Element name length	Specifies a limit on the maximum number of characters permitted in any element name in the XML document.
Text length	Specifies a character limit for any text node present in the XML document.
Comment length	Specifies a character limit for any comments present in the XML document.
Processing instruction target length	Specifies a limit on the maximum number of characters permitted in the target of any processing instructions in the XML document.
Processing instruction data length	Specifies a limit on the maximum number of characters permitted in the data value of any processing instructions in the XML document.
Node depth	Specifies the maximum node depth allowed in the XML.
Applicable content types	Specify any other content types to be included in the filter. You can add multiple values by clicking  .

■ Custom filter

You can use the custom filter to invoke a service that is available on API Gateway to perform actions such as custom authentication of external clients in the DMZ, logging or auditing in the DMZ, or implementation of custom rules for processing various payloads.

- Set the **Enable** button to the **On** position to enable the filter.
- Click **Browse** and select a service to invoke it.

- Select the user name of a user you want API Gateway to run the service. The default value is Administrator.
6. Click **Save**.

The new rule is created and appears in the list of rules in the Rules page.

Note:

Rule names are truncated after thirty characters. You can view the full name of a rule by hovering the mouse pointer over the truncated name.

The rule is applied to requests only if the rule is enabled. You can enable the rule in the Rules page by selecting the enable icon for the required rule.



Registering a Mobile Device or Application

You can use API Gateway to disable access for certain mobile application versions on a predefined set of mobile platforms. By registering the required devices and applications and disabling access to these versions, you ensure that all users use the latest versions of the applications and take advantage of the latest security and functional updates.


➤ To register a mobile device or application

1. Click **Policies** in the title navigation bar.
2. Select **Global Policies > Mobile devices and apps**.

3. Provide the mobile device type name and click .

You can add more entries by clicking . You can delete the added ones by clicking .

4. Provide the mobile application name and click .

You can add more entries by clicking . You can delete the added ones by clicking .

5. Click **Save**.

Configuring Alert Settings

You can configure the alert settings to control the following aspects of alerts that API Gateway sends when a request violates a rule:

- Whether API Gateway issues an alert for a rule violation.

- How often API Gateway issues the alert.
- The method API Gateway uses to send the alert.
- Whether a rule uses the default alert options or its own customized alert options.

➤ **To configure alert settings**

1. Click **Policies** in the title navigation bar.
2. Select **Global Policies > Alert settings**.
3. Select one or both the alert destination types:
 - **Email**. This sends email alerts.
 - Type the email ids to which the email has to be sent.
 - **Flow Service**. This invokes a flow service to alert you of a rule violation. Specify the name of the flow service. You can create a flow service using the `pub.security.enterpriseGateway:alertSpec` specification as the signature of the service or use the pre-defined flow service, `pub.apigateway.threatProtection:violationListener`. When you use the predefined service, the alerts are saved in Internal Data Store and displayed API Gateway Dashboard. For more information on the `pub.security.enterpriseGateway:alertSpec` specification, refer to the Integration Server Build-In Services Reference Guide.
 - Provide the user, who has permissions to execute the service, as the user type. For example, `Administrator`.
4. Select one of the following conditions depending on when you want the alert to be sent.
 - **On rule violation** to send an alert every time a request violates a rule,
 - **Every** and specify the time interval (in minutes) to send to send alerts at specified intervals.
5. Click **Save**.

System-defined Stages and Policies

API Gateway provides system-defined policies that are grouped into stages depending on their usage:

- Transport
- Identify & Access
- Request Processing
- Routing
- Traffic Monitoring

- Response Processing
- Error Handling

Transport

The policies in this stage specify the protocol to be used for an incoming request and the content type for a REST request during communication between API Gateway and an application. The policies included in this stage are:

- Enable HTTP/HTTPS
- Set Media Type
- Enable JMS/AMQP

Enable HTTP/HTTPS

This policy specifies the protocol to use for an incoming request to the API on API Gateway. If you have a native API that requires clients to communicate with the server using the HTTP and HTTPS protocols, you can use the Enable HTTP or HTTPS policy. This policy allows you to bridge the transport protocols between the client and API Gateway.

For example, you have a native API that is exposed over HTTPS and an API that receives requests over HTTP. If you want to expose the API to the consumers of API Gateway through HTTP, then you configure the incoming protocol as HTTP.

The table lists the properties that you can specify for this policy:

Parameter	Description
Protocol	<p>Specifies the protocol (HTTP or HTTPS) and SOAP format (for a SOAP-based API) to be used to accept and process the requests.</p> <p>Select one of the following:</p> <ul style="list-style-type: none"> ■ HTTP. API Gateway accepts requests that are sent using the HTTP protocol. This is selected by default. ■ HTTPS. API Gateway accepts requests that are sent using the HTTPS protocol.
SOAP Version	<p>For SOAP-based APIs.</p> <p>Specifies the SOAP version of the requests which the API Gateway accepts from the client.</p> <p>Select one of the following:</p> <ul style="list-style-type: none"> ■ SOAP 1.1. This is selected by default. API Gateway accepts requests that are in the SOAP 1.1 format.

Parameter	Description
	<ul style="list-style-type: none"> ■ SOAP 1.2. API Gateway accepts requests that are in the SOAP 1.2 format.

Enable JMS/AMQP

Java Message Service (JMS) is a standard Java API for communicating with message oriented middleware and enables loosely coupled communication between two or more homogenous systems. It provides reliable and asynchronous form of communication.

Advanced Message Queuing Protocol (AMQP) is an open standard application layer protocol for delivering messages. AMQP can queue and route messages in a reliable and secured way. AMQP provides a standard messaging protocol that stands across all platforms and a description on how a message should be constructed. It doesn't provide an API on how the message should be sent. AMQP being language agnostic is useful in the message oriented middleware to achieve interoperability in asynchronous way among heterogenous systems.

When you want to expose a REST or SOAP API over JMS with broker native protocol or JMS with AMQP protocol add and configure the **Enable JMS/AMQP** policy in API Gateway, thereby allowing them to communicate through the messaging Queue or Topic.

For example, you can use this policy to expose your API over JMS/AMQP and hence enable your client to communicate through the messaging queue or topic.

- **JMS with Message broker native protocol support**

For example, if your Message broker is using ActiveMQ and if you are relying on the default Active MQ TCP protocol, then essentially it is JMS on open wire protocol because open wire is the native protocol of ActiveMQ Message broker.

- **JMS with AMQP protocol support**

For example, if you want to use JMS with AMQP with any message broker which supports AMQP 1.0 to achieve interoperability in asynchronous way among heterogenous systems. API Gateway supports AMQP 1.0 using Apache qpid JMS client.

Note:

The following are not supported if the **Enable JMS/AMQP** policy is added:

- **Threat protection** policies.
- API Gateway SOAP to REST transformation feature.

Use case 1: Expose a SOAP API over JMS with a message broker native protocol

This describes the high level workflow for the scenario where you want to expose a SOAP API over JMS with a message broker native protocol.

1. Create an alias to a JNDI Provider. For a detailed procedure, see [“Creating a JNDI Provider Alias” on page 83.](#)

2. Configure API Gateway to use a JMS connection alias to establish an active connection between API Gateway and the JMS provider. For a detailed procedure, see [“Creating a JMS Connection Alias” on page 87](#).
 3. Create a WS (Web Service) JMS Provider endpoint alias and configure the Alias, Description, Type (Provider), Transport Type (JMS) fields and JMS Transport Properties. For a detailed procedure, see [“Creating an Endpoint Alias for a Provider Web Service Descriptor” on page 96](#).
 4. A WS (Web Service) endpoint trigger is created when you configure WS (Web Service) JMS Provider endpoint alias. This trigger consists of the input source details like Queue name or Topic name. You can update the WS (Web Service) endpoint trigger, as required. For detailed procedures, see .
 5. Select the required API.
 6. Click **Edit**.
 7. In the API Details section click **Policies**.
 8. Enforce the **Enable JMS/AMQP** policy with the following properties configured.
 - a. Specify the name of the JMS provider endpoint alias that specifies the trigger which listens to the source queue or topic for the input message.
 - b. Specify the SOAP version of the requests which the API Gateway accepts from the client.
- For details on the **Enable JMS/AMQP** policy, see [“Using Enable JMS/AMQP for a SOAP API” on page 389](#)
9. Click **Save**.

The enforced policy **Enable JMS/AMQP** with the required configuration now allows any java client to communicate with the API asynchronously.

Use case 2: Expose a SOAP API over JMS with AMQP protocol

This describes the high level workflow for the scenario where you want to expose a SOAP API over JMS with AMQP protocol.

1. Configure API Gateway to use JMS with AMQP protocol.

Configure JNDI settings and JMS settings as per the client you are using to achieve JMS over AMQP protocol support.

Note:

For a sample procedure on configuring API Gateway to use JMS with AMQP protocol using Apache qpid libraries, see [“Configuring API Gateway for JMS with AMQP Protocol” on page 387](#)

2. Select the required API.
3. Click **Edit**.
4. In the API Details section click **Policies**.

5. Enforce the **Enable JMS/AMQP** policy with the following properties configured.
 - a. Specify the name of the JMS provider endpoint connection alias that specifies the trigger which listens to the source queue or topic for the input message.
 - b. Specify the SOAP version of the requests which the API Gateway accepts from the client.

For details on the **Enable JMS/AMQP** policy, see [“Using Enable JMS/AMQP for a SOAP API” on page 389](#)

6. Click **Save**.

The enforced policy **Enable JMS/AMQP** with the required configuration now allows all the clients such as Python, Ruby, Java, and Dotnet to communicate with the API asynchronously.

Use case 3: Expose a REST API over JMS with a message broker native protocol

This describes the high level workflow for the scenario where you want to expose a REST API over JMS with a message broker native protocol.

1. Create an alias to a JNDI Provider. For a detailed procedure, see [“Creating a JNDI Provider Alias” on page 83](#).
2. Configure API Gateway to use a JMS connection alias to establish an active connection between API Gateway and the JMS provider. For a detailed procedure, see [“Creating a JMS Connection Alias” on page 87](#).
3. Select the required API.
4. Click **Edit**.
5. In the API Details section click **Policies**.
6. Enforce the **Enable JMS/AMQP** policy with the following properties configured.
 - a. Specify the name of the JMS provider endpoint alias that contains the configuration information needed to establish a connection to a specific JMS provider.
 - b. Specify the input source name which API Gateway starts listening to when the API is activated.
 - c. Specify the type of source type Queue or Topic, which the API Gateway listens for the request message.
 - d. Specify the selector, a criteria for the API Gateway to listen to a message containing the specified criteria

For details on the **Enable JMS/AMQP** policy, see [“Using Enable JMS/AMQP for a REST API” on page 390](#)

7. Click **Save**.

The enforced policy **Enable JMS/AMQP** with the required configuration now allows any java client to communicate with the API asynchronously.

Use case 4: Expose a REST API over JMS with AMQP protocol

This describes the high level workflow for the scenario where you want to expose a REST API over JMS with AMQP protocol.

1. Configure API Gateway to use JMS with AMQP protocol.

Configure JNDI settings and JMS settings as per the client you are using to achieve JMS over AMQP protocol support.

Note:

For a sample procedure on configuring API Gateway to use JMS with AMQP protocol using Apache qpid libraries, see [“Configuring API Gateway for JMS with AMQP Protocol” on page 387](#).

2. Select the required API.
3. Click **Edit**.
4. In the API Details section, click **Policies**.
5. Enforce the **Enable JMS/AMQP** policy with the following properties configured.
 - a. Specify the name of the JMS provider endpoint alias that contains the configuration information needed to establish a connection to a specific JMS provider.
 - b. Specify the input source name which API Gateway starts listening to when the API is activated.
 - c. Specify the type of source type Queue or Topic, which the API Gateway listens for the request message.
 - d. Specify the selector, a criteria for the API Gateway to listen to a message containing the specified criteria.

For details on the **Enable JMS/AMQP** policy, see [“Using Enable JMS/AMQP for a REST API” on page 390](#).

6. Click **Save**.

The enforced policy **Enable JMS/AMQP** with the required configuration now allows all the clients such as Python, Ruby, Java, and Dotnet to communicate with the API asynchronously.

Configuring API Gateway for JMS with AMQP Protocol

Before configuring AMQP in API Gateway, ensure your message broker supports AMQP 1.0

For using JMS with AMQP protocol in API Gateway you have to configure the appropriate settings for the provider URL and the connection factory lookup name required for API Gateway to communicate using JMS with AMQP protocol.

To configure API Gateway to use JMS with AMQP protocol

1. Create a properties file that contains the information for the JNDI lookup name and connectionfactory details.
2. Configure JNDI settings as per the client you are using to achieve JMS over AMQP protocol support.

For a detailed procedure, see [“Creating a JNDI Provider Alias” on page 83](#).

3. Configure JMS settings as per the client you are using to achieve JMS over AMQP protocol support.


For a detailed procedure, see [“Creating a JMS Connection Alias” on page 87](#).

➤ To configure API Gateway for JMS with AMQP protocol using Apache qpid libraries

1. Create a properties file that contains the information for the JNDI lookup name and connectionfactory details.

A sample properties file, for example `amqp.properties`, would look like

```
# Set the InitialContextFactory class to use
java.naming.factory.initial = org.apache.qpid.jms.jndi.JmsInitialContextFactory
# Define the required ConnectionFactory instances
# connectionfactory.<JNDI-lookup-name> = <URI>
connectionfactory.qpidConnectionFactory = amqp://<hostname>:<port#>
```

2. Navigate to  > **Administration**.
3. Select **General > Messaging**.
4. Configure the JNDI provider alias as follows:
 - a. Click **Add JNDI provider alias** in the JNDI provider alias definitions section.
 - b. Provide the following information:
 - **JNDI Alias Name**. Provide a name that you want to assign to this JNDI provider.
 - **Description**. Provide a brief description for this JNDI alias.
 - **Predefined JNDI Templates**. Select the predefined JNDI template depending on the provider you may want to use.

For example, if you want to use the JMS with AMQP protocol, select **Qpid AMQP (0-x)**.
 - **Initial Context Factory**. The JNDI provider uses the initial context as the starting point for resolving names for naming and directory operations. This value gets pre-populated depending on the predefined JNDI template selected. For example, if you have selected **Qpid AMQP (0-x)** as the predefined JNDI template the Initial context factory field would display `org.apache.qpid.jms.jndi.JmsInitialContextFactory`.

- **Provider URL.** Provide the file path location of the properties file that contains the context factory details. For example, `C:\amqp.properties`
- c. Click **Add**.

The JNDI provider alias is created and listed in the JNDI Provider alias definitions table.
- 5. Configure the JMS settings as follows:
 - a. Click **Add JMS connection alias** in the JMS connection alias definitions section.
 - b. Provide the following information in the General Settings section:
 - **Connection Alias Name.** Provide a name for the connection alias. Each connection alias represents a connection factory to a specific JMS provider.
 - **Description.** Provide a brief description for the connection alias.
 - c. Provide the following information in the Connection Protocol Settings section:
 - **JNDI Provider Alias Name.** The alias to the JNDI provider that you want this JMS connection alias to use to look up administered objects. Select the JNDI Provider alias name created in the earlier step.
 - **Connection Factory Lookup Name.** The lookup name for the connection factory that you want to use to create a connection to the JMS provider specified in this JMS connection alias. Provide the value `qpidConnectionFactory`.
 - d. Click **Add**.

The JMS Connection alias is created and listed in the JMS Connection Alias Definitions table.
 - e. Enable the JMS connection alias by clicking toggle button to enable it.

The JNDI provider alias and the JMS connection alias are now set up and API Gateway is configured to use JMS with AMQP protocol.

Using Enable JMS/AMQP for a SOAP API

This policy is used to expose a SOAP API over JMS/AMQP. A SOAP API can be exposed as HTTP/HTTPS or JMS/AMQP as the policies **Enable HTTP/HTTPS** and **Enable JMS/AMQP** are mutually exclusive.

If you are using JMS with Message Broker native protocol support ensure that following actions are performed before using the **Enable JMS/AMQP** policy:

- Create an alias to a JNDI Provider. For a detailed procedure, see [“Creating a JNDI Provider Alias” on page 83](#).

- Configure API Gateway to use a JMS connection alias to establish an active connection between API Gateway and the JMS provider. For a detailed procedure, see [“Creating a JMS Connection Alias” on page 87](#).
- Create a WS (Web Service) JMS Provider endpoint alias and configure the Alias, Description, Type (Provider), Transport Type (JMS) fields and JMS Transport Properties. For a detailed procedure, see [“Creating an Endpoint Alias for a Provider Web Service Descriptor” on page 96](#).
- Configure a WS (Web Service) endpoint trigger. For detailed procedures, see [“Updating JMS triggers” on page 105](#).

If you are using JMS with AMQP protocol support, ensure the following before using the Enable JMS/AMQP policy:

- You have configured API Gateway for JMS with AMQP. For details, see [“Configuring API Gateway for JMS with AMQP Protocol” on page 387](#)

The table lists the properties that you can specify for this policy:

Parameter	Description
JMS Provider Endpoint Alias	Specifies the name of the JMS provider endpoint alias. The provider endpoint alias specifies the trigger which listens to the source queue or topic for the input message.
SOAP Version	Specifies the SOAP version of the requests which the API Gateway accepts from the client. Select one of the following: <ul style="list-style-type: none"> ■ SOAP 1.1. This is selected by default. API Gateway accepts requests that are in the SOAP 1.1 format. ■ SOAP 1.2. API Gateway accepts requests that are in the SOAP 1.2 format.

Using Enable JMS/AMQP for a REST API

This policy is used to expose a REST API over JMS/AMQP. A REST API can be exposed as both HTTP/HTTPS and JMS/AMQP at the same time.

If you are using JMS with Message broker native protocol support ensure that following actions are performed before using the Enable JMS/AMQP policy:

- Create an alias to a JNDI Provider. For a detailed procedure, see [“Creating a JNDI Provider Alias” on page 83](#).
- Configure API Gateway to use a JMS connection alias to establish an active connection between API Gateway and the JMS provider. For a detailed procedure, see [“Creating a JMS Connection Alias” on page 87](#).

If you are using JMS with AMQP protocol support, ensure the following before using the Enable JMS/AMQP policy:

- You have configured API Gateway for JMS with AMQP. For details, see [“Configuring API Gateway for JMS with AMQP Protocol” on page 387](#)

The table lists the properties that you can specify for this policy:

Parameter	Description
Connection Alias Name	<p>Specifies the name of the connection alias.</p> <p>Each connection alias contains the configuration information needed to establish a connection to a specific JMS provider.</p>
<p>Add JMS/AMQP source details. Click to add the JMS/AMQP source details and provide the required information.</p>	
Input Source Name	<p>Specifies the input source name which API Gateway starts listening to when the API is activated.</p>
Input Source Type	<p>Specifies the type of source to which the API Gateway listens for the request message.</p> <p>Select one of the following source type:</p> <ul style="list-style-type: none"> ■ QUEUE. Indicates that API Gateway listens to the specified queue for the request message. ■ TOPIC. Indicates that the API Gateway listens to the specified topic for the request message. <p>Note: Provides support only for non-durable topic.</p>
Selector	<p>Specifies the criteria for the API Gateway to listen to a message containing the specified criteria.</p> <p>For example, operation = GET</p> <p>If you have multiple selectors it follows the OR condition.</p> <p>If there are no selectors the message that comes in is listened to without any condition.</p> <p>Note: Message selectors are only applicable for headers, properties and not for payload.</p>
Resource	<p>Specifies the resource of the API.</p>
HTTP Method	<p>Specifies the routing method used.</p>

Parameter	Description
	Available routing methods: GET, POST, PUT, and DELETE.
Content Type	(Optional) Specifies the content type of the JMS/AMQP message body. Examples for content types: <code>application/json</code> , <code>application/xml</code>
	Note: Alternatively, you can use the Set media type policy to set the default content type instead of setting it here.

Set Media Type

This policy specifies the content type for a REST request. If the content type header is missing in a client request sent to an API, API Gateway adds the content type specified here before sending the request to the native API.

The table lists the properties that you can specify for this policy:

Parameter	Description
Default Content-Type	Specifies the default content type for REST request received from a client. Examples for content types: <code>application/json</code> , <code>application/xml</code> .
Default Accept Header	Specifies the default accept header for REST request received from a client.

Identify and Access

The policies in this stage provide different ways of identifying and authorizing the application, and provide the required access rights for the application. The policies included in this stage are:

- Inbound Authentication - Message
- Authorize User
- Identify and Authorize Application

The Inbound authentication policies are used to authenticate the application by specifying user-based SPN or host-based SPN for a Kerberos token, using the basic credentials for the HTTP basic authentication or through various token assertions or through the XML security actions.

The Authorize User policy authorizes the application against a list of users and a list of groups registered in API Gateway.

The Identify and Authorize Application policy is used to identify the application, authenticate the request based on policy configured and authorizes it against all applications registered in API Gateway.

Note:

From API Gateway 10.3, the Identification and Authentication policies are merged into one and you would not be able to do identification alone for Basic Authentication. You must provide the right credentials for a successful invoke.

Inbound Authentication - Message

An API Provider can use this policy to enforce authentication on the API. When this policy is configured for an API, API Gateway expects the clients to pass the authentication credentials through the payload message that will be added to the request and sent to the native API. API Gateway supports a wide range of authentication schemes, such as X.509 Certificate, WSS Username, SAML, and Kerberos, in addition to signing and encryption, at the message-level.

Note:

Message-level authentication can be used to secure inbound communication of only SOAP APIs.

The table lists the properties that you can specify for this policy:

Parameter	Description
Binding Assertion	Specifies the type of binding assertion required for the message transfer between the recipient and the initiator.

Require Encryption. Specifies that a request's XML element, which is represented by an XPath expression or by parts of a SOAP request such as the SOAP body or the SOAP headers, be encrypted.

Encrypted Parts

Click **+ Add encrypted part** to add the required properties. This allows you to encrypt parts of a SOAP request such as the SOAP body or the SOAP headers.

Provide the following information:



- **Entire SOAP Body.** Specifies encryption of the entire SOAP body.
- **All SOAP Attachments.** Specifies encryption of all the SOAP attachments.


In the SOAP Header section, provide the following information:

- **Header Name.** Specifies the name for the SOAP header field.
- **Header Namespace.** Specifies the namespace of the SOAP header to be encrypted.

You can add more SOAP headers by clicking



Parameter	Description
Encrypted Elements	<p>Click + Add encrypted element to add the required properties. Select this option to encrypt the entire element, which is represented by an XPath expression.</p> <p>Provide the following information:</p> <p>XPath. Specifies the XPath expression in the API request.</p> <p>In the Namespace section, provide the following information:</p> <ul style="list-style-type: none"> ■ Namespace Prefix. Specifies the namespace prefix of the element to be encrypted. ■ Namespace URI. Specifies the generated XPath element. <p>You can add more namespace prefixes and URIs by clicking .</p>
Require Signature.	<p>Specifies that a request's XML element, which is represented by an XPath expression or by parts of a SOAP request such as the SOAP body or the SOAP headers, be signed.</p>
Signed Elements	<p>Click + Add require signature to add the required properties. Select this option to sign the entire element represented by an XPath expression.</p> <p>Provide the following information:</p> <p>XPath. Specifies the XPath expression in the API request.</p> <p>For the Namespace section, provide the following information:</p> <ul style="list-style-type: none"> ■ Namespace Prefix. Specifies the namespace prefix of the element to be signed. ■ Namespace URI. Specifies the generated XPath element. <p>You can add more namespace prefixes and URIs by clicking .</p>
Signed Parts	<p>Click + Add signed part to add the required properties. Select this option to sign parts of a SOAP request such as the SOAP body or the SOAP headers.</p> <p>Provide the following information:</p> <ul style="list-style-type: none"> ■ Entire SOAP Body. Specifies signing of the entire SOAP body. ■ All SOAP Attachments. Specifies signing of all the SOAP attachments. <p>For the SOAP Header section, provide the following information:</p>

Parameter	Description
	<ul style="list-style-type: none"> ■ Header Name. Specifies the name for the SOAP header field. ■ Header Namespace. Specifies the Namespace of the SOAP header to be signed.
	<p>You can add more namespace prefixes and URIs by clicking </p>

Validate SAML Audience URIs. Validates the audience restriction in the conditions section of the SAML assertion. It verifies whether any of the valid audience URI within a valid condition element in SAML assertion matches with any of the configured URI. If two conditions are available, then one of the audience URIs in the first condition, and one of the audience URIs in the second condition must match with any of the configured URIs in this policy for the SOAP API.

This property is used in the following scenarios:

- When the native API is enforced with the SAML policy, and the service provider wants to delegate audience restriction validation to API Gateway.
- When **Require SAML Token** assertion is defined for the SOAP API in API Gateway.




URI	Specifies the SAML audience URI.
------------	----------------------------------

Match Criteria	<p>Select one of the following options:</p> <ul style="list-style-type: none"> ■ Allow Sublevels. Any one of the audience URI in the incoming SAML assertion either has to be an exact match or it can have sub paths to the configured URI. For example, if <code>http://yahoo.com</code> is configured as the URI and the Allow Sublevels option is selected, the audience URI has <code>http://yahoo.com/mygroup</code> and condition is matched because the main URI matches with the configured URI (<code>http://yahoo.com</code>). The extra path <code>mygroup</code> is a sublevel path. ■ Exact match. Any one of the audience URI in the incoming SAML assertion is verified for the exact match with the configured URI. For example, if <code>http://yahoo.com</code> is configured as the URI and the Exact match option is selected, the audience URI must be configured with <code>http://yahoo.com</code> in order to match the condition. This is selected by default.
-----------------------	--

For more information on audience URI, see conditions and audience restriction sections in the SAML specification available in the <https://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf> location.

Token Assertions	<p>Select the type of token assertion required to authenticate the client.</p> <p>Select any of the following:</p>
-------------------------	--

Parameter	Description
	<ul style="list-style-type: none"> ■ Require X.509 Certificate. Mandates that there should be a wss x.509 token in the incoming SOAP request. ■ Require WSS Username token. Mandates that there should be a WSS username token in the incoming SOAP request. Uses WS-Security authentication to validate user names and passwords that are transmitted in the SOAP message header for the WSS Username token. ■ Kerberos Token Authentication. Mandates that there should be a Kerberos token in the incoming SOAP request. Authenticates the client based on the Kerberos token. API Gateway extracts the Kerberos token from the SOAP body and validates the token with the KDC using SPN credentials configured by the provider for the API. If the Kerberos token sent by the client is valid, API Gateway forwards the request to the native service and the response to the client. ■ Service Principal Name. Specifies a valid SPN, which is the name type to use while authenticating an incoming client principal name. The specified value is used by the client or the server to obtain a service ticket from the KDC server. <div data-bbox="613 1010 1365 1213" style="background-color: #f0f0f0; padding: 5px; margin: 10px 0;"> <p>Note: API Gateway supports the <i>username</i> format for Service Principal Names (SPNs). This format represents the principal name as a named user defined in LDAP used for authentication to the KDC.</p> </div> <ul style="list-style-type: none"> ■ Service Principal Password. Specifies a valid password of the Service Principal Name user or the Service Principal Name host. ■ Require SAML Token. Mandates that there should be a SAML token in the incoming SOAP request. Uses a Security Assertion Markup Language (SAML) assertion token to validate applications. Provide the following information: <ul style="list-style-type: none"> ■ SAML Version. Specifies the supported SAML version. Available values are SAML 1.0, SAML 2.0 ■ SAML Subject Configuration. Select one of the following: <ul style="list-style-type: none"> ■ Bearer of Token. Select the bearer method when the client wants a security token to be issued without a proof of possession. ■ Holder of Key - Symmetric. Select the Holder of Key (Symmetric) method when either the client or the server

Parameter	Description
	<p>has to generate security tokens such as X509 tokens. A symmetric key is established using the security token. You can use this token to sign and encrypt parts and elements.</p> <ul style="list-style-type: none"> ■ Holder of Key - Public. Select the Holder of Key (Public) method when both the client and the server have security token such as X509 certificates. In this method, the client uses its private key to sign and the recipient's (API Gateway) public key to encrypt. ■ WS-Trust Version. Specifies the WS-Trust version to be used. Available values are WS-Trust 1.0, WS-Trust 1.3 ■ Encrypt Signature. Select Yes to encrypt the signature. ■ Issuer Address. Specifies the SAML issuer address. ■ Metadata Reference Address. Specifies the address from where the metadata reference document is obtained. ■ Algorithm Suite. Specifies the applicable algorithm suite. ■ Key. Specifies the Key type of the security token template. ■ Value. Specifies a value for the request token. <p>You can add more values for the key-value pair by clicking .</p> <ul style="list-style-type: none"> ■ Custom Token Assertion. Type a search string, select a custom token assertion name to authenticate the client, and click  to add. You can add more custom token assertions in a similar way. <p>Click the Custom Token Assertion arrow to see a list of all custom token assertions available in API Gateway.</p> <p>Click  to delete the custom token assertion added.</p>
Require Timestamp	<p>Specifies that the time stamps be included in the request header. API Gateway checks the time stamp value against the current time to ensure that the request is not an old message. This serves to protect your system against attempts at message tampering, such as replay attacks.</p>

Authorize User







This policy authorizes incoming requests against a list of users, a list of groups, or users who belong to LDAP groups registered in API Gateway.

Note:

LDAP groups cannot be authorized using the List of Groups configuration option. To authorize a user who belongs to an LDAP group, you must first create a team containing one or more LDAP groups and then authorize the user using List of Teams configuration option in this policy.

Use this policy in conjunction with an authentication policy (for example, Require HTTP Basic Authentication, Require WSS Username Token).

The table lists the parameters of this policy and how they are applied to authorize the incoming requests.

Parameter	Description
List of Users	<p>Authorizes applications against a list of users registered in API Gateway.</p> <p>Type a search string, select a user, and click  to add. You can add one or more users.</p> <p>Click  to delete the user added.</p>
List of Groups	<p>Authorizes applications against a list of groups registered in API Gateway.</p> <p>Type a search string, select a group, and click  to add. You can add one or more groups.</p> <p>Click  to delete the group added.</p>
List of Teams	<p>Authorizes applications against a list of teams registered in API Gateway.</p> <p>Type a search string, select a team, and click  to add. You can add one or more teams.</p> <p>Click  to delete a team.</p>

Identify and Authorize Application

This policy identifies and validates the authorization of the applications to access the APIs. The application are identified using a set of identification types such as API key, hostname address, and HTTP basic authentication and so on based on the configuration. API Gateway can identify and authorize the application based on the following **Application Lookup condition**:

- **Registered applications.** Identifies the application and validates the identified application against the registered applications. On successful validation, API Gateway allows access to the API. The application that are associated with the API are called as registered application.
- **Global applications.** Identifies the application and validates the identified application against the global applications. On successful validation, API Gateway allows access to the API. All the active applications that are available in API Gateway are called as global application.
- **Global applications and DefaultApplication.** Verifies the identity of the application against the global applications and on identification failure the API Gateway allows access to the API as default application.

Note:

If **Allow anonymous** is selected and even if the **Application Lookup condition** does not meet, API Gateway allows access to the API.

The table lists the properties that you can specify for this policy:

Property	Description
Condition	<p>Specifies the condition operator for the identification and authentication types.</p> <p>Select any of the following condition operators:</p> <ul style="list-style-type: none"> ■ AND. Applies all the identification and authentication types. ■ OR. Applies one of the selected identification and authentication types. <p>Note: Even though this policy provides the option of choosing an AND or OR operation between the different identification and authentication types, the operation across the different policies in the IAM stage is always AND.</p>
Allow anonymous	<p>Specifies whether to allow all users to access the API without restriction.</p> <p>When you add a security policy and configure Allow anonymous, all requests are allowed to pass through to the native API, but the successfully identified requests are grouped under the respective identified application, and all unidentified requests are grouped under a common application named as DefaultApplication (sys:defaultApplication). While you allow all requests to pass through you can perform all application-specific actions, such as, viewing the runtime events for a particular application, monitor the service level agreement for a few applications and send an alert email based on some criteria like request count or availability, and throttle the requests from a particular application and not allow the request</p>

Property	Description
	from that application if the number of requests reach the configured hard limit within configured period of time.

Identification Type. Specifies the identification type. You can select any of the following.

Note:

When you add an API to a package for monetization, the **API key** authentication mechanism is automatically added to the IAM policy at API level. If the API already contains an IAM policy that has two authentication mechanisms with the **AND** condition, then the condition will be switched to **OR**. This ensures the monetization is supported when certain consumers access the API by just using the API key.

API Key Specifies using the API key to identify and validate the client's API key to verify the client's identity in the registered list of applications for the specified API.

Select one of the **Application Lookup condition**:

- **Registered applications.** Identifies the client's API key against the API key of all the applications registered to the API. On successful identification, API Gateway allows access to the API.
- **Global applications.** Identifies the client's API key against the API key of all the applications available in API Gateway. On successful identification, API Gateway allows access to the API.
- **Global applications and DefaultApplication.** Identifies the client's API key against all the applications available in API Gateway. Even though, if no global application is identified, API Gateway allows access to the API as default application.

When this option is selected, you can use the API key as:

- Header parameter to consume an API. For example,

```
x-Gateway-APIKey: a4b5d569-2450-11e3-b3fc-b5a70ab4288a
```

- Query parameter to invoke a API resource. For example,

```
http://pie-3HKYMH2:5555/gateway/PetstoreAPI/1.0.3/store/inventory?APIKey=faab7ac6-97a4-4228-908d-f1930faba470
```

Hostname Address Specifies using host name address to identify the client, the hostname is resolved from the client's IP address and verify the client's identity in the specified list of applications in API Gateway.

Select one of the **Application Lookup condition**:

- **Registered applications.** Identifies the client's hostname against the *hostname* identifier of all the applications registered to the

Property	Description
	<p>API. On successful identification, API Gateway allows access to the API.</p> <ul style="list-style-type: none"> ■ Global applications. Identifies the client's hostname against the <i>hostname</i> identifier of all the applications available in API Gateway. On successful identification, API Gateway allows access to the API. ■ Global applications and DefaultApplication. Identifies the client's hostname against the <i>hostname</i> identifier of all the applications available in API Gateway. If no global application is identified, then API Gateway allows access to the API as default application. <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: If the client request has X-Forwarded-For header, then API Gateway resolves the hostname from the IP address present in the X-Forwarded-For header. Else, API Gateway resolves the hostname from the client's IP address.</p> </div>

HTTP Basic Authentication Specifies using Authorization Header in the request to identify and authorize the client application against the list of applications with the identifier *username* in API Gateway.

Provide the following information:

- Select one of the **Application Lookup condition**:
 - **Registered applications.** Authenticates the user and identifies the user against *username* identifier of all the applications registered to the API. On successful authentication and identification, API Gateway allows access to the API.
 - **Global applications.** Authenticates the user and identifies the user against *username* identifier of all the applications available in the API Gateway. On successful authentication and identification, API Gateway allows access to the API.
 - **Global applications and DefaultApplication.**
 1. Authenticates the user and identifies the user against *username* identifier of all the applications available in the API Gateway.
 2. On successful authentication and if no global application is identified, then API Gateway allows access to the API as default application.

Property	Description
	<p data-bbox="613 260 1385 323">3. In case if the authentication fails, then API Gateway does not allow access to the API.</p> <ul style="list-style-type: none"> <li data-bbox="516 352 1385 420">■ If Global applications and DefaultApplication and Allow anonymous are selected: <ol style="list-style-type: none"> <li data-bbox="565 449 1385 554">1. Authenticates the user and identifies the user against <i>username</i> identifier of all the applications available in the API Gateway. <li data-bbox="565 575 1385 680">2. On successful authentication and if no global application is identified, then API Gateway allows access to the API as default application. <li data-bbox="565 701 1385 764">3. In case if the authentication fails, then API Gateway still allows access to the API. <li data-bbox="516 798 1385 903">■ Trigger policy violation event on missing authorization header. Creates a policy violation event for basic authentication if Authorization Headers are missing. <p data-bbox="516 932 721 963">Possible values:</p> <ul style="list-style-type: none"> <li data-bbox="516 993 1385 1056">■ <code>true</code>. Requests without authorization headers are logged as a policy violation event. <li data-bbox="565 1085 1385 1148">■ <code>false</code>. Requests without authorization headers are not logged as a policy violation event.
IP Address Range	<p data-bbox="516 1180 1385 1314">Specifies using the IP address range to identify the client, extract the client's IP address from the HTTP request header, and verify the client's identity against the specified list of applications in API Gateway.</p> <p data-bbox="516 1344 1156 1375">Select one of the Application Lookup condition:</p> <ul style="list-style-type: none"> <li data-bbox="516 1404 1385 1539">■ Registered applications. Identifies the client's IP address against the <i>IP address range</i> identifier of all the applications registered to the API. On successful identification, API Gateway allows access to the API. <li data-bbox="516 1568 1385 1703">■ Global applications. Identifies the client's IP address against the <i>IP address range</i> identifier of all the applications available in API Gateway. On successful identification, API Gateway allows access to the API. <li data-bbox="516 1732 1385 1837">■ Global applications and DefaultApplication. Identifies the client's IP address against the <i>IP address range</i> identifier of all the applications available in API Gateway. If no global

Property	Description
	<p>application is identified, then API Gateway allows access to the API as default application.</p> <p>Note: If the client request has X-Forwarded-For header, then API Gateway uses the IP address present in the X-Forwarded-For header. Else, API Gateway uses the client's IP address for identification.</p>
JWT	<p>Specifies using the JSON Web Token (JWT) to identify the client, extract the claims from the JWT and validate the client's claims, and verify the client's identity against the specified list of applications in API Gateway.</p> <p>Select one of the Application Lookup condition:</p> <ul style="list-style-type: none"> ■ Registered applications. Identifies the JWT against the <i>claims</i> identifier of all the applications registered to the API. On successful identification, API Gateway allows access to the API. ■ Global applications. Identifies the JWT against the <i>claims</i> identifier of all the applications available in API Gateway. On successful identification, API Gateway allows access to the API. ■ Global applications and DefaultApplication. Identifies the JWT against the <i>claims</i> identifier of all the applications available in API Gateway. If no global application is identified, then API Gateway allows access to the API as default application. <p>Note:</p> <ul style="list-style-type: none"> ■ You can use the claims in the JWT for further processing using request transformation policy. ■ When a Policy violation event is logged in case of expired JWT tokens, the application is associated as the identified application since the identification happens before the expiry is checked.
Kerberos Token	<p>Specifies using the Kerberos token to identify the client, extract the client's credentials from the Kerberos token, and verify the client's identity against the specified list of applications in API Gateway.</p> <p>Note: You have to enforce the Inbound Auth - Message policy with the property, Kerberos Token Authentication, configured, so when Identify & Authorize policy is executed, the user details fetched are used to match with application's data to identify the application.</p>

Property	Description
	<p>Select one of the Application Lookup condition:</p> <ul style="list-style-type: none"> ■ Registered applications. Authenticates the incoming Kerberos token and identifies the user against the <i>username</i> identifier of all the applications registered to the API. On successful authentication and identification, API Gateway allows access to the API. ■ Global applications. Authenticates the incoming Kerberos token and identifies the user against the <i>username</i> identifier of all the applications available in API Gateway. On successful authentication and identification, API Gateway allows access to the API. ■ Global applications and DefaultApplication. <ol style="list-style-type: none"> 1. Authenticates the incoming Kerberos token and identifies the user against <i>username</i> identifier of all the applications available in the API Gateway. 2. On successful authentication and if no global application is identified, then API Gateway allows access to the API as default application. 3. In case if the authentication fails, then API Gateway does not allow access to the API. ■ If Global applications and DefaultApplication and Allow anonymous are selected: <ol style="list-style-type: none"> 1. Authenticates the incoming Kerberos token and identifies the user against <i>username</i> identifier of all the applications available in the API Gateway. 2. On successful authentication and if no global application is identified, then API Gateway allows access to the API as default application. 3. In case if the authentication fails, then API Gateway still allows access to the API. <p>Note: You can use the username for further processing using the request transformation policy.</p>
OAuth2 Token	Specifies using the OAuth2 token to identify the client, extract the access token from the HTTP request header, and verify the client's identity against the specified list of applications in API Gateway.

Property	Description
	<p>By default, OAuth2 token is identified against the registered applications.</p> <p>Note:</p> <ul style="list-style-type: none"> ■ You can use the client id and other parameters for further processing using the request transformation policy. ■ When a Policy violation event is logged in case of expired OAuth2 tokens, the application that is associated turn in to Unknown.
OpenID Connect	<p>Specifies using the OpenID (ID) token to identify the client, extract the client's credentials from the ID token, and verify the client's identity against the specified list of applications in API Gateway.</p> <p>Select one of the Application Lookup condition:</p> <ul style="list-style-type: none"> ■ Registered applications. Identifies the client's identity resolved as part of OpenID validation against all the applications registered to the API. On successful identification, API Gateway allows access to the API. ■ Global applications. Identifies the client's identity resolved as part of OpenID validation against all the applications available in API Gateway. On successful identification, API Gateway allows access to the API. ■ Global applications and DefaultApplication. Identifies the client's identity resolved as part of OpenID validation against all the applications available in API Gateway. If no global application is identified, then API Gateway allows access to the API as default application. <p>Note: You can use the client id and other parameters for further processing using the request transformation policy.</p>
SSL Certificate	<p>Specifies using the SSL certificate to identify the client, extract the client's identity certificate, and verify the client's identity (certificate-based authentication) against the specified list of applications in API Gateway. The client certificate that is used to identify the client is supplied by the client to API Gateway during the SSL handshake over the transport layer or is added in the header of the request.</p> <p>The certificate included in the custom header can be in the following formats:</p> <ul style="list-style-type: none"> ■ Base64 encoded PEM certificate with BEGIN CERTIFICATE and END CERTIFICATE delimiters

Property	Description
	<ul style="list-style-type: none"> <li data-bbox="516 258 1377 321">■ Non-Base64 encoded PEM certificate with BEGIN CERTIFICATE and END CERTIFICATE delimiters. <li data-bbox="516 352 1377 415">■ PEM certificate can be without BEGIN CERTIFICATE and END CERTIFICATE delimiters if a single certificate is added. <li data-bbox="516 447 1377 510">■ URL encoded PEM certificate with BEGIN CERTIFICATE and END CERTIFICATE delimiters. <li data-bbox="516 541 1377 636">■ URL encoded PEM certificate can be without the BEGIN CERTIFICATE and END CERTIFICATE delimiters if a single certificate is added.
	<p data-bbox="516 667 1377 905">If the transport protocol is HTTP then API Gateway checks for the existence of a header and fetches the certificate from the certificate header. If the certificate is coming from the custom header, then API Gateway does not check the validity of the certificate. API Gateway identifies the application using the certificate. The certificate should be validated by some external entity before sending it to API Gateway in a custom header.</p>
	<p data-bbox="516 940 1377 1104">If the transport protocol is HTTPS then API Gateway first tries to identify the application based on the certificate exposed by the client during the SSL handshake. If there is no client certificate or the identification based on the client certificate fails API Gateway tries to identify based on the certificate provided in the header.</p>
	<p data-bbox="516 1140 1377 1230">The header name is customizable and can be customized in the extended settings property, <code>customCertificateHeader</code>, the default value being <code>X-Client-Cert</code>.</p>
	<p data-bbox="516 1266 1154 1293">Select one of the Application Lookup condition:</p>
	<ul style="list-style-type: none"> <li data-bbox="516 1329 1377 1455">■ Registered applications. Identifies the client's certificate against the <i>client certificate</i> identifier of all the applications registered to the API. On successful identification, API Gateway allows access to the API.
	<ul style="list-style-type: none"> <li data-bbox="516 1493 1377 1619">■ Global applications. Identifies the client's certificate against the <i>client certificate</i> identifier of all the applications available in API Gateway. On successful identification, API Gateway allows access to the API.
	<ul style="list-style-type: none"> <li data-bbox="516 1661 1377 1818">■ Global applications and DefaultApplication. Identifies the client's certificate against the <i>client certificate</i> identifier of all the applications available in API Gateway. If no global application is identified, then API Gateway allows access to the API as default application.

Property	Description
WS Security Username Token	<p>This is applicable only for SOAP APIs.</p> <p>Specifies using the WS security username token to identify the application, extract the client's credentials (username token and password) from the WSSecurity SOAP message header, and verify the client's identity against the specified list of applications in API Gateway.</p>
	<p>Note: You have to enforce the Inbound Auth - Message policy with the property, Require WSS Username token, configured, so when Identify & Authorize policy is executed, the user details fetched are used to match with application's data to identify the application.</p>
	<p>Select one of the Application Lookup condition:</p> <ul style="list-style-type: none"> ■ Registered applications. Authenticates the client's WSS username token and identifies the user against the <i>username</i> identifier of all the applications registered to the API. On successful authentication and identification, API Gateway allows access to the API. ■ Global applications. Authenticates the client's WSS username token and identifies the user against the <i>username</i> identifier of all the applications available in API Gateway. On successful authentication and identification, API Gateway allows access to the API. ■ Global applications and DefaultApplication. <ol style="list-style-type: none"> 1. Authenticates the client's WSS username token and identifies the user against the <i>username</i> identifier of all the applications available in the API Gateway. 2. On successful authentication and if no global application is identified, then API Gateway allows access to the API as default application. 3. In case if the authentication fails, then API Gateway does not allow access to the API. ■ If Global applications and DefaultApplication and Allow anonymous are selected: <ol style="list-style-type: none"> 1. Authenticates the client's WSS username token and identifies the user against the <i>username</i> identifier of all the applications available in the API Gateway.


Property	Description
	<ol style="list-style-type: none"> <li data-bbox="565 260 1377 359">2. On successful authentication and if no global application is identified, then API Gateway allows access to the API as default application. <li data-bbox="565 386 1377 449">3. In case if the authentication fails, then API Gateway still allows access to the API. <p data-bbox="521 478 1377 590">Note: You can use the username for further processing using the request transformation policy.</p>
<p data-bbox="142 617 402 680">WS Security X.509 Certificate</p>	<p data-bbox="511 617 1015 653">This is applicable only for SOAP APIs.</p> <p data-bbox="511 680 1377 814">Specifies using the WS security X.509 certificate to identify the client, extract the client identity certificate from the WS-Security SOAP message header, and verify the client's identity against the specified list of applications in API Gateway.</p> <p data-bbox="521 844 1377 1031">Note: You have to enforce the Inbound Auth - Message policy with the property, Require X.509 Certificate, configured, so when Identify & Authorize policy is executed, the user details fetched are used to match with application's data to identify the application.</p> <p data-bbox="511 1054 1154 1089">Select one of the Application Lookup condition:</p> <ul style="list-style-type: none"> <li data-bbox="511 1115 1377 1249">■ Registered applications. Identifies the client's X.509 certificate against the <i>client certificate</i> identifier of all the applications registered to the API. On successful identification, API Gateway allows access to the API. <li data-bbox="511 1276 1377 1411">■ Global applications. Identifies the client's X.509 certificate against the <i>client certificate</i> identifier of all the applications available in API Gateway. On successful identification, API Gateway allows access to the API. <li data-bbox="511 1438 1377 1612">■ Global applications and DefaultApplication. Identifies the client's X.509 certificate against the <i>client certificate</i> identifier of all the applications available in API Gateway. If no global application is identified, then API Gateway allows access to the API as default application.
<p data-bbox="142 1640 380 1675">Payload Element</p>	<p data-bbox="511 1640 1377 1774">Specifies using the payload identifier to identify the client, extract the custom authentication credentials supplied in the request represented using the payload identifier, and verify the client's identity against the specified list of applications in API Gateway.</p> <ul style="list-style-type: none"> <li data-bbox="511 1801 1203 1837">■ Select one of the Application Lookup condition:

Property	Description
	<ul style="list-style-type: none"> ■ Registered applications. Identifies the client's payload against the <i>Payload Identifier</i> of all the applications registered to the API. On successful identification, API Gateway allows access to the API. ■ Global applications. Identifies the client's payload against the <i>Payload Identifier</i> of all the applications available in API Gateway. On successful identification, API Gateway allows access to the API. ■ Global applications and DefaultApplication. Identifies the client's payload against the <i>Payload Identifier</i> of all the applications available in API Gateway. If no global application is identified, then API Gateway allows access to the API as default application.

In the Payload identifier section, click **Add payload identifier**, provide the following information, and click **Add**.

- **Expression type:** Specifies the type of expression, which is used for identification. You can select one the following expression type:
 - **XPath.** Provide the following information:
 - **Payload Expression.** Specifies the payload expression that the specified expression type in the request has to be converted to. For example: /name/id
 - **Namespace Prefix.** The namespace prefix of the payload expression to be validated.
 - **Namespace URI.** The namespace URI of the payload expression to be validated.

Note:

You can add multiple namespace prefix and URI by clicking .

- **JSONPath.** Provide the JSONPath for the payload identification. For example, \$.name.id
- **Text.** Provide the regular expression for the payload identification. For example, any valid regular expression.

You can add multiple payload identifiers as required.

Note:

Property	Description
	Only one payload identifier of each type is allowed. For example, you can add a maximum of three payload identifiers, each being of a different type.
HTTP Headers	<p>Specifies using any header in the request to identify and authorize the client application against the list of applications with the identifier in API Gateway.</p> <p>Provide the following information:</p> <ul style="list-style-type: none"> ■ Select one of the Application Lookup condition: <ul style="list-style-type: none"> ■ Registered applications. Identifies the client's header against the <i>Header Key - Value pair</i> identifier of all the applications registered to the API. On successful identification, API Gateway allows access to the API. ■ Global applications. Identifies the client's header against the <i>Header Key - Value pair</i> identifier of all the applications available in API Gateway. On successful identification, API Gateway allows access to the API. ■ Global applications and DefaultApplication. Identifies the client's header against the <i>Header Key - Value pair</i> identifier of all the applications available in API Gateway. If no global application is identified, then API Gateway allows access to the API as default application.

Request Processing

These policies are used to specify how the request message from an application has to be transformed or pre-processed and configure the masking criteria for the data to be masked before it is submitted to the native API. This is required to protect the data and accommodate differences between the message content that an application is capable of submitting and the message content that a native API expects. The policies included in this stage are:

- Invoke webMethods IS
- Request Transformation
- Validate API Specification
- Data Masking

Invoke webMethods IS

This policy pre-processes the request messages and transforms the message into the format required by the native API or performs some custom logic, before API Gateway sends the requests to the native APIs.

For example, you might need to accommodate differences between the message content that a client is capable of submitting and the message content that a native API expects. For example, if the client submits an order record using a slightly different structure than the structure expected by the native API, you can use this action to process the record submitted by the client to the structure required by the native API.

If `Comply to IS Spec` parameter is configured as `true`, API Gateway invokes the IS Service with IS specification in the path `pub.apigateway.invokeISService.specifications:RequestSpec` for Request Processing

The following are the input and output parameters for REST, SOAP, and WebSocket APIs as specified in the above IS specification. Input parameters can be used to access the existing values of the request while output parameters can be used to modify/write the values to the request.

	Parameter name	Description
Input parameters	headers	Headers in incoming request. Data type: Document
	query	Query parameters in incoming request (this is applicable for REST API only). Data type: Document
	path	Path parameter of the incoming request (this is applicable for REST API only). Data type: String
	httpMethod	HTTP Method of the incoming request (this is applicable for REST API only). Data type: String
	payload	Payload of the incoming request. Data type: String
	payloadObject	The payload for binary content types like multi-part / form-data. Data type: Object
	MessageContext	The message context object of the request. Data type: Object
	apiName	Name of the API invoked by the request.

Parameter name	Description	
	Data type: String	
apiVersion	Version of the API invoked by the request. Data type: String	
requestUrl	URL of the request. Data type: String	
ipInfo	Contains IP information of the request. Data type: Document	
websocketInfo	Websocket related information of the request. Data type: Document	
correlationID	Correlation ID of the request/response. This is unique and same for a request and response. Data type: String	
customFieldsMap	Custom transactional fields can be added to the transactional events using this field. For more information, see Adding Custom Fields to Transactional Events section. Data type: Document	
authorization	Authorization information of the request. For more information, see Accessing authorization values hidden after IAM policy section. Data type: Document	
Output parameters	headers	Headers in incoming request. Data type: Document
	query	Query parameters in incoming request (this is applicable for REST API only). Data type: Document
	path	Path parameter of the incoming request (this is applicable for REST API only). Data type: String
	httpMethod	HTTP Method of the incoming request (this is applicable for REST API only).

Parameter name	Description
	Data type: String
payload	Payload of the incoming request. Data type: String
payloadObject	The payload for binary content types like multi-part / form-data. Data type: Object
MessageContext	The message context object of the request. Data type: Object
customFieldsMap	Custom transactional fields can be added to the transactional events using this field. For more information, see Adding Custom Fields to Transactional Events section. Data type: Document

By default the "query" pipeline variable is a key value pair, where the value is of type string. But, if the incoming request contains multiple values for the same query parameter and if you want to access those multiple values using **webMethods IS Service**, you have to ensure two things:

1. Make sure that you have checked the **Repeat** check box for query parameter in the **Add Resource Parameter** section of the API details screen.
2. To access or transform multiple values of that query parameter, you have to insert string list (instead of string) under the "query" pipeline variable in the webMethods IS Service.

Note:

- For SOAP to REST APIS, the payload contains the transformed SOAP request.
- Payload transformation does not happen automatically for content-type transformation. When you change the content type, ensure to do payload transformation also as part of IS Service. For example, if you change the content-type header from application/xml to application/json using IS service, you must also change the respective payload from application/xml to application/json
- Only Method Transformation happens when configured, but you have to take care of adding payload during transformations involving method change like GET to POST, and so on.
- When `Comply to IS spec` is true, you can change the values of headers, query, payload, and so on, programatically using Message Context, as well as using the pipeline variables given. Software AG recommends you do not change those values directly in Message Context, as the values in output pipeline variables are written to Message Context after the invocation of IS Service.

If `Comply to IS Spec` parameter is set to false, API Gateway invokes the IS Service with the same input and output parameters supported in 10.1 and the earlier versions:

- proxy.name
- JSONRESTContentString (REST only)
- SOAPEnvelope (SOAP only)
- EnvelopeString (SOAP only)

The table lists the properties that you can specify for this policy:

Parameter	Description
-----------	-------------

Invoke webMethods Integration Server Service

Add invoke webMethods Integration Server service Specifies the webMethods IS service to be invoked to pre-process the request messages and the authentication mode for the IS service.

Provide the following information:

- **webMethods IS Service.** Specify the webMethods IS service to be invoked to pre-process the request messages.

The webMethods IS service must be running on the same Integration Server as API Gateway.

Note:

If an exception occurs when invoking the webMethods IS service, by default API Gateway displays the status code as *500* and error message as *Internal Server Error*.



You can set custom status code and error message by setting the following properties in the message context of the webMethods IS service:

- service.exception.status.code
- service.exception.status.message

You must add the WmAPIGateway package as dependent to IS service to get access to the class used to set the custom status code and message.

The sample code is given below:

```
import org.apache.synapse.MessageContext
IDataCursor idc = pipeline.getCursor();
MessageContext context =
(MessageContext)IDataUtil.get(idc,"MessageContext");
if(context != null)
{
context.setProperty("service.exception.status.code",
404);
context.setProperty("service.exception.status.message",
"Object Not Found");
throw new ServiceException();
}
```

Parameter	Description
	<p>Note: If <code>ServiceException</code> or <code>FlowException</code> occurs when invoking <code>webMethods IS Service</code>, the message given in the exception is displayed to the client. If any other exception occurs, a generic error message is displayed to the client.</p> <ul style="list-style-type: none"> ■ Run as User. Specifies the authentication mode to invoke the IS service. If this field is left blank the incoming credentials of the user, identified by API Gateway, are used to authenticate and invoke the IS service. You can also specify a particular user, you want API Gateway to invoke the IS service. <p>Note: It is the responsibility of the user who activates the API to review the value configured in Run as User field to avoid misuse of this configuration.</p> <ul style="list-style-type: none"> ■ Comply to IS Spec. Mark this as <code>true</code> if you want the input and the output parameters to comply to the IS Spec present in <code>pub.apigateway.invokeISService.specifications</code> folder in <code>wmAPIGateway</code> package. <p>Note:Software AG recommends users to configure the policy with <code>Comply to IS Spec</code> as <code>true</code>, as you can read or change the values of headers, and so on, without having to read from or write to the message context.</p>
webMethods IS Service alias	<p>Specifies the <code>webMethods IS</code> service alias to be invoked to pre-process the request messages.</p> <p>Start typing the <code>webMethods</code> alias name, select the alias from the type-ahead search results displayed and click  to add one or more aliases.</p> <p>You can use the delete icon  to delete the added aliases from the list.</p>

Adding Custom Fields to Transactional Events

This section explains you how to add custom fields to the transactional events.

1. Create `webMethods IS` service by specifying the `pub.apigateway.utils:customFieldInTransactionEventSpec` as a specification reference.
2. In the `webMethods IS` service, set the required custom fields in the `customFieldsMap` output variable.

3. Once when *customFieldsMap* gets created, the custom fields will be available in the transactional events.
4. Invoke the API with the Invoke webMethods IS policy.

Accessing authorization values hidden after IAM policy

By default, API Gateway removes all the authorization related information from client request (for example authorization header) once the IAM policy is engaged. The information like authorization header can be added back to the request sent to native API using "Outbound Authentication" policy in the Routing stage. However, if you want to extract the authorization information at the request processing stage for sending the authorization values using a different header to the native API for audit purposes, or performing some business logic in IS Service based on the authorization values, then you can access the authorization values using the "authorization" pipeline variable.

The following table lists the supported authorization values:

Name	Type	Description
clientId	String	clientId identified after the OAuth / JWT / OpenID token is authenticated.
userName	String	Name of the user identified after the IAM policy.
issuer	String	Issuer identified from the JWT token.
authHeader	String	Value of the incoming "authorization" header sent by client.
		<p>Note: If the authorization header has bearer tokens (such as OAuth, OpenID, or JWT), then the "authHeader" pipeline variable will be empty. For such cases, Software AG recommends to use the "incomingToken" pipeline variable.</p>
incomingToken	String	Value of the token in case the incoming authorization header contains a bearer token.
audience	String	Audience identified from the incoming JWT token.
apiKey	String	API Key sent from client.
claims	Document (Key-value pair)	Contains the claims present in the JWT token. You can provide the claim name to access the claim value.

Name	Type	Description
certificates	Object List	Client certificates used for SSL connectivity.

Note:

All the above mentioned authorization values except certificates can be accessed using authorization pipeline variable.

Accessing client certificates used for SSL connectivity

You can now access the client certificates used for SSL Connectivity in the Invoke webMethods IS Service (comply to IS Spec = true) using pipeline authorization > certificates.

Since certificates are not string data type, you need to write JAVA code to convert the pipeline variable certificates into accessible certificate format (Java X509Certificate) and you can read the values using the methods supported by [X509Certificate](#).

The below sample code converts the pipeline variable certificates to X509Certificate:

```
import java.security.cert.X509Certificate;
IDataCursor cursor = pipeline.getCursor();
IData authIData = IDataUtil.getIData(cursor, "authorization");
IDataCursor authCursor = authIData.getCursor();
X509Certificate[] certificates = (X509Certificate[])
IDataUtil.getObjectArray(authCursor, "certificates");
```

The following watt parameters control the certification verification

- **watt.net.ssl.client.hostnameverification**

When API Gateway server acts as a HTTPS client, this parameter specifies whether API Gateway should restrict outbound HTTPS connections only when a valid hostname is found in the server's certificate. If you set this parameter to true, API Gateway verifies if the hostname is present in the server's certificate. If this verification fails, an error is logged and the connection is aborted. If you set this parameter to false, API Gateway skips the hostname verification. By default, this parameter is set to false.

- **watt.security.ssl.ignoreExpiredChains**

This parameter specifies whether API Gateway server ignores expired CA certificates in a certificate chain it receives from an Internet resource (that is, a web server, another API Gateway server). If you set this parameter to true, API Gateway, ignores the expired CA certificates. However, API Gateway allows SSL connection to be established, even if the certificate is expired. Note that this is less secure than denying connections when a certificate is expired. If you set this parameter to false, API Gateway does not ignore the expired CA certificates and a connection cannot be established, if a certificate is expired. By default, this parameter is set to false.

- **watt.security.ssl.client.ignoreEmptyAuthoritiesList**

When API Gateway acts as a client, this parameter specifies if API Gateway sends a certificate chain, after a remote SSL server returns an empty list of trusted authorities. If you set this parameter to true, API Gateway ignores the empty trusted authorities list and sends its chain anyway. If you set this parameter to false, API Gateway requires presentation of trusted certificates before sending out its certificate chain. By default, this parameter is set to false.

Request Transformation

This policy enables you to configure several transformations on the request messages from clients into a format required by the native API before it is submitted to the native API.

The transformations include Header, Query Parameter, Path Parameter transformation, HTTP Method transformation, Payload transformation, and Advanced transformation. You can configure conditions according to which the transformations are executed.

API Gateway supports the following parameter types that can be used to configure the transformation policy:

- List of variables supported for all API Types

- `request.headers.HEADER_NAME`
- `request.authorization.apiKey`
- `request.authorization.clientId`
- `request.authorization.claims.CLAIM_NAME`
- `request.authorization.userName`
- `request.authorization.issuer`
- `request.authorization.authHeader`
- `request.authorization.incomingToken`
- `request.authorization.audience`
- `request.correlationID`
- `request.application.id`
- `request.application.name`
- `request.application.version`
- `request.application.claims.claimName`

Note:

- The `request.authorization.claims.CLAIM_NAME` and `request.application.claims.claimNames` request transformation variables must be used for JWT authorized APIs.
- The `request.authorization` variables are retrieved from token introspection and not from the scope that you have defined in the external authorization server. The request authorization variable values are retrieved from the incoming JWT tokens when the

introspection is done locally and the values are retrieved from the remote introspection response received from the authorization server when the introspection is done remotely.

- List of variables supported only for REST APIs

request.query.QUERY_NAME

request.path

request.path.regex[Expression]

request.path.PATH_PARAM_NAME

request.httpMethod

jms.query.QUERY_NAME

jms.path

jms.httpMethod

jms.path.regex[Expression]

jms.headers.HEADER_NAME

The parameter types that are of the format `jms.xxx` are used when you want to use JMS/AMQP so that transformation can be applied for the `jms/amqp` values

API Gateway supports the following Query types that can be applied on payload to extract values from payload:

- `xpath`
- `jsonPath`
- `regex`

When you use these syntaxes to extract a value from the payload, the content-types applicable are:

- `${request.payload.jsonPath[Expression]}` - `application/json`, `application/json/badgerfish`
- `${request.payload.regex[Expression]}` - `text/plain`
- `${request.payload.xpath[Expression]}` - `application/xml`, `text/xml`, `text/html`. This is to be used to extract values when APIs with Transformation policy are JWT authorized.

The table lists the properties that you can specify for this policy:

Parameter	Description
Condition	<p>Conditions are used to specify when the policy has to be executed. You can add multiple conditions with logical operators.</p> <p>Available values are:</p>

Parameter	Description
-----------	-------------

- **AND.** API Gateway transforms the requests that comply with all the configured conditions.
- **OR.** This is selected by default. API Gateway transforms the requests that comply with at least one configured condition.

Click **Add Condition** and provide the following information and click

 Add

- **Variable:** Specifies the variable type with a syntax as follows:
 - `${PARAMTYPE}` : This is applicable for variables of string type - path, payload, httpMethod. For example: `${request.path}`
 - `${PARAMTYPE.paramName}` : This is applicable for map types - query and headers and also applicable for path. For example: `${request.query.var1}`, `${request.header.Content-Type}`, `${request.path.name}`
 - `${PARAMTYPE.QUERYTYPE[queryValue]}` : This syntax is applicable for payload and path. regex can be applied on path while XPath, JSONPath and regex can be applied on payload. For example:

`${request.payload.xpath[//ns:emp/ns:empName]}` where `//ns:emp/ns:empName` is the xpath to be applied on the payload if contentType is application/xml

`${request.payload.jsonPath[$.cardDetails.number]}` where `$.cardDetails.number` is the jsonPath to be applied on the payload if contentType is application/json


`${request.payload.regex[[0-9]+]}` where `[0-9]+` is the regex to be applied on the payload if contentType is text/plain
 - If you want API Gateway to apply xpath, jsonPath, regex based on Content-Type of the payload, use the following common syntax: `${PARAMTYPE.QUERYTYPE[queryValue] || PARAMTYPE.QUERYTYPE2[queryValue2] || ...}`

For example:


```
${request.payload.xpath[//ns:emp/ns:empName] ||
request.payload.jsonPath[$.cardDetails.number]} This applies
xpath for application/xml and jsonPath for application/json
```

```
${request.payload.xpath[//ns:emp/ns:empName] ||
request.payload.jsonPath[$.cardDetails.number] ||
request.payload.regex[[0-9]+]} This applies xpath for
```

Parameter	Description
	<p>application/xml, jsonPath for application/json, and regex for text/plain.</p> <ul style="list-style-type: none"> ■ Operator: Specifies the operator to use to relate variable and the value provided. You can select one of the following: <ul style="list-style-type: none"> ■ Equals ■ Equals ignore case ■ Not equals ■ Contains ■ Exists ■ Value: Specifies a value with a syntax as follows: <ul style="list-style-type: none"> ■ PLAIN VALUE, for example, application/json ■ <code>\${PARAMTYPE.paramName}</code> ■ <code>\${PARAMTYPE.QUERYTYPE[queryValue]}</code> ■ <code>\${PARAMTYPE.QUERYTYPE[queryValue] PARAMTYPE.QUERYTYPE2[queryValue2] ...}</code>
<hr/> <p>Transformation Configuration: Specifies various transformations to be configured.</p> <hr/>	
<p>Header/Query/Path Transformation for REST API</p> <p>and</p> <p>Header Transformation for SOAP API</p>	<p>Specifies the Header, Query or path transformation to be configured for incoming requests.</p> <p>You can add or modify header, query or path transformation parameters by providing the following information:</p> <ul style="list-style-type: none"> ■ Variable. Specifies the variable type with a syntax as follows: <ul style="list-style-type: none"> ■ <code>\${PARAMTYPE}</code>. This is applicable for variables of string type - path, payload, httpMethod. For example: <code>\${request.path}</code> ■ <code>\${PARAMTYPE.paramName}</code>. This is applicable for map types - query and headers and also applicable for path. For example: <code>\${request.query.var1}</code>, <code>\${request.header.Content-Type}</code>, <code>\${request.path.name}</code> ■ <code>\${PARAMTYPE.QUERYTYPE[queryValue]}</code>. This syntax is applicable for payload and path. regex can be applied on path while XPath, JSONPath and regex can be applied on payload. For example: <code>\${request.payload.xpath[//ns:emp/ns:empName]}</code> where <code>//ns:emp/ns:empName</code> is the xpath to be applied on the payload if contentType is application/xml

Parameter	Description
	<p><code>\${request.payload.jsonPath[\$.cardDetails.number]}</code> where <code>\$.cardDetails.number</code> is the <code>jsonPath</code> to be applied on the payload if <code>contentType</code> is <code>application/xml</code></p> <p><code>\${request.payload.regex[[0-9]+]}</code> where <code>[0-9]+</code> is the <code>regex</code> to be applied on the payload if <code>contentType</code> is anything</p> <ul style="list-style-type: none"> ■ If you want API Gateway to apply <code>xpath</code>, <code>jsonPath</code>, <code>regex</code> based on <code>Content-Type</code> of the payload, use the following common syntax: <code>\${PARAMTYPE.QUERYTYPE[queryValue] PARAMTYPE.QUERYTYPE2[queryValue2] ...}</code> <p>For example:</p> <p><code>\${request.payload.xpath[//ns:emp/ns:empName] request.payload.jsonPath[\$.cardDetails.number]}</code> This applies <code>xpath</code> for <code>application/xml</code> and <code>jsonPath</code> for <code>application/json</code></p> <p><code>\${request.payload.xpath[//ns:emp/ns:empName] request.payload.jsonPath[\$.cardDetails.number] request.payload.regex[[0-9]+]}</code> This applies <code>xpath</code> for <code>application/xml</code>, <code>jsonPath</code> for <code>application/json</code>, and <code>regex</code> for <code>text/xml</code>.</p> <div style="background-color: #f0f0f0; padding: 10px;"> <p>Note: The parameter types that are of the format <code>jms.xxx</code> are used when you want to use <code>JMS/AMQP</code> so that transformation can be applied for the <code>jms/amqp</code> values. For example, if you have set the path parameter as <code>jms.path.petid</code> and the corresponding value as <code>jms.header.h1</code>, then if the request contains the header value <code>h1</code>, the value <code>h1</code> is replaced by the path parameter <code>petid</code>.</p> </div> <ul style="list-style-type: none"> ■ Value. Specifies a value with a syntax as follows: <ul style="list-style-type: none"> ■ <code>PLAIN VALUE</code>, for example, <code>application/json</code> ■ <code>\${PARAMTYPE.paramName}</code> ■ <code>\${PARAMTYPE.QUERYTYPE[queryValue]}</code> ■ <code>\${PARAMTYPE.QUERYTYPE[queryValue] PARAMTYPE.QUERYTYPE2[queryValue2] ...}</code> <p>You can add multiple variables and corresponding values by clicking .</p> <p>You can remove any header, query or path transformation parameters by typing the value in the text box.</p>

Parameter	Description
	<p>Note: Software AG recommends you not to modify the headers <code>\${request.headers.Content-Length}</code> and <code>\${request.headers.Content-Encoding}</code> as API Gateway adds the right values for these headers before sending the response back to client.</p> <p>Note: Payload transformation does not happen automatically for content-type transformation. When you change the content type, ensure to do payload transformation also as part of IS Service. For example, if you change the content-type header from application/xml to application/json using IS service, you must also change the respective payload from application/xml to application/json.</p>
<p>Method transformation for REST API</p>	<p>Specifies the method transformation to be configured for incoming requests.</p> <p>Select any of the HTTP Method listed:</p> <ul style="list-style-type: none"> ■ GET ■ POST ■ PUT ■ DELETE ■ HEAD ■ CUSTOM <p>Note: When CUSTOM is selected, the HTTP method in incoming request is sent to the native service. When other methods are selected, the selected method is used in the request sent to the native service.</p> <p>Note: Only Method Transformation happens when configured, but you have to take care of adding payload during transformations involving method change like GET to POST, and so on.</p>
<p>Payload Transformation</p>	<p>Specifies the payload transformation to be configured for incoming requests.</p> <p>Provide the following information:</p> <ul style="list-style-type: none"> ■ Click + Add xslt document to add an xslt document and provide the following information:

Parameter	Description
	<ul style="list-style-type: none"> ■ XSLT file. Specifies the XSLT file used to transform the request messages as required. Click Browse to browse and select a file. ■ Feature Name. Specifies the name of the XSLT feature. ■ Feature value. Specifies the value of the XSLT feature. You can add more XSLT features and xslt documents by clicking .

Note:

API Gateway supports XSLT 1.0 and XSLT 2.0.


- Click **+ Add xslt transformation alias** and provide the following information:
 - **XSLT Transformation alias.** Specifies the XSLT transformation alias


When the incoming request is in JSON, you can use a XSLT file similar to the below sample:

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.1"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml"/>
  <xsl:template match="/" >
    <xsl:element name="fakeroot">
      <xsl:element name="fakenode">
        <!-- Apply your transformation rules based on the
request from the Client-->
      </xsl:element>
    </xsl:element>
  </xsl:template>
</xsl:stylesheet>
```

When the incoming request is in XML, you can use a XSLT file similar to the below sample:

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.1"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
  <xsl:output method="xml"/>
  <xsl:template match="/" >
    <xsl:element name="soapenv:Envelope">
      <xsl:element name="soapenv:Body">
        <!-- Apply your transformation rules based on the
request from the Client-->
      </xsl:element>
    </xsl:element>
  </xsl:template>
</xsl:stylesheet>
```


Parameter	Description
	<pre data-bbox="607 247 867 338"></xsl:element> </xsl:template> </xsl:stylesheet></pre>
Advanced Transformation	<p data-bbox="553 373 1442 436">Specifies the advanced transformation to be configured for incoming requests.</p> <p data-bbox="553 468 1003 497">Provide the following information:</p> <ul data-bbox="553 529 1474 592" style="list-style-type: none"> <li data-bbox="553 529 1474 592">■ webMethods IS Service. Specify the webMethods IS service to be invoked to process the request messages. <p data-bbox="602 657 1263 686">You can add multiple services by clicking .</p> <div data-bbox="602 709 1458 842" style="background-color: #f0f0f0; padding: 5px;"> <p data-bbox="602 720 1349 821">Note: The webMethods IS service must be running on the same Integration Server as API Gateway.</p> </div> <ul data-bbox="553 856 1474 1287" style="list-style-type: none"> <li data-bbox="553 856 1474 1031">■ Run as User. Specifies the authentication mode to invoke the IS service. If this field is left blank the incoming credentials of the user, identified by API Gateway, are used to authenticate and invoke the IS service. You can also specify a particular user, you want API Gateway to use to invoke the IS service. <li data-bbox="553 1056 1474 1192">■ Comply to IS Spec. Mark this as true if you want the input and the output parameters to comply to the IS Spec present in <code>pub.apigateway.invokeISService.specifications</code> folder in <code>wmAPIGateway</code> package. <li data-bbox="553 1218 1474 1287">■ webMethods IS Service alias. Specifies the webMethods IS service alias to be invoked to pre-process the request messages.
<p data-bbox="240 1318 1474 1423">Transformation Metadata: Specifies the metadata for transformation of the incoming requests. For example, the namespaces configured in this section can be used when you provide the syntax for XPath <code>{request.payload.xpath}</code> For example: <code>{request.payload.xpath[//ns:emp/ns:empName]}</code></p>	
Namespace	<p data-bbox="553 1444 1474 1474">Specifies the namespace information to be configured for transformation.</p> <p data-bbox="553 1505 1003 1535">Provide the following information:</p> <ul data-bbox="553 1566 1474 1724" style="list-style-type: none"> <li data-bbox="553 1566 1474 1629">■ Namespace Prefix. The namespace prefix of the payload expression to be validated. <li data-bbox="553 1654 1474 1724">■ Namespace URI. The namespace URI of the payload expression to be validated. <div data-bbox="602 1745 1458 1803" style="background-color: #f0f0f0; padding: 5px;"> <p data-bbox="602 1755 683 1785">Note:</p> </div>

Parameter	Description
	You can add multiple namespace prefix and URI by clicking 

Validate API Specification

This policy validates the incoming request against API's various specifications such as schema, query parameters, path parameters, cookie parameters, content-types, and HTTP Headers referenced in their corresponding formats as follows:

- The schema is available as part of the API definition. The schema for SOAP API are imported through WSDL and for REST APIs it can be through swagger, RAML or can be uploaded by the user when an API is created from scratch.
- The query parameters, path parameters, cookie parameters, and content- types are available as part of the API definition.
- The HTTP Headers are specified in the Validate API Specification policy page.

The request sent to the API by an application must conform with the structure or format expected by the API. The incoming requests are validated against the API specifications in this policy to conform to the structure or format expected by the API.

Various API specifications validated are:

- **Schema:**

The incoming requests are validated against the schema provided in the API definition. The schema defines the elements and attributes and specifies the data types of these elements to ensure that only appropriate data is allowed through to the API.

For a REST API, the schema can be added inline or uploaded in the Components section on the API Details page. For details on how to add the schema inline or upload, see [“Creating a REST API” on page 273](#).

The schema type for validation is selected based on:

- The Content-Type header when the policy is added in the Request processing stage.
- The Accept header when the policy is added in the Response processing stage.

If the header or payload is missing the schema validation is skipped.

The table lists the default Content type/Accept header and schema validation type mapping.

Content-type/Accept	Schema validation type
application/json	JSON schema
application/json/badgerfish	

Content-type/Accept	Schema validation type
application/xml	XML schema
text/xml	
text/html	
text/plain	Regular expression

For a SOAP API, the WSDL and the referenced schema must be provided in a zip format. The JSON schema validation is supported for the operations that are exposed as REST.

Note:

If schema mapping is not found for a content-type of the request in the API, the behavior is as follows:

- If schema mapping is not available in a REST API or SOAP-to-REST transformed API, the validation is skipped.
- If application/json is mapped to XML schema in the API definition, then the JSON content in the request is validated against XML schema to provide a backward compatibility support for APIs migrated from the 10.1 version.
- If only XML schema mappings exist for any of the content-types, the payload is converted into XML and validated against all the XML schemas. If the payload is valid against one of the schemas, the validation is successful.
- If the payload is not XML convertible, the validation is not performed and the request is allowed to reach the native API.

■ **Query Parameters:**

This is applicable only for a REST API. The incoming requests are validated against the query parameters specified in the API definition.

■ **Path Parameters:**

This is applicable only for a REST API. The incoming requests are validated against the path parameters specified in the API definition.

■ **Content-types:**

The incoming requests are validated against the content-types specified in the API definition.

Note:

When Content-type validation is selected for a SOAP API, the validation fails in case of SOAP to REST scenarios and displays an error with 500 status code instead of 400 as displayed in the other scenarios.

■ **Cookie Parameters:**

The incoming requests are validated against the cookie parameters specified in the API definition.

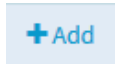
■ **HTTP Headers:**

The incoming requests are validated against the HTTP Headers specified in this policy to conform to the HTTP headers expected by the API.

The runtime invocations that fail the specification validation are considered as policy violations. You can view such policy violation events in the dashboard.

The table lists the API specification properties, you can specify for this policy, to be validated:

Parameter	Description
Schema	<p>Validates the request payload against the appropriate schema.</p> <p>Provide the following additional features for XML schema validation:</p> <ul style="list-style-type: none"> ■ Feature name. Specifies the name of the feature for XML parsing when performing XML schema validation. <p>Select the required feature name from the list:</p> <ul style="list-style-type: none"> ■ GENERATE_SYNTHETIC_ANNOTATIONS ■ ID_IDREF_CHECKING ■ IDENTITY_CONSTRAINT_CHECKING ■ IGNORE_XSL_TYPE ■ NAMESPACE_GROWTH ■ NORMALIZE_DATA ■ ROOT_ELEMENT_DECL ■ ROOT_TYPE_DEF ■ SIGMA_AUGMENT_PSVI ■ SCHEMA_DV_FACTORY ■ SCHEMA_ELEMENT_DEFAULT ■ SCHEMA_LOCATION ■ SCHEMA_NONS_LOCATION ■ SCHEMA_VALIDATOR ■ TOLERATE_DUPLICATES ■ ENPARSED_ENTITY_CHECKING ■ VALIDATE_ANNOTATIONS ■ XML_SCHEMA_FULL_CHECKING ■ XMLSCHEMA_VALIDATION

Parameter	Description
	<p>For details about XML parsing features, see http://xerces.apache.org/xerces2-j/features.html and for details about the exact constants, see https://xerces.apache.org/xerces2-j/javadocs/xerces2/org/apache/xerces/parsers/XML11Configuration.html.</p> <ul style="list-style-type: none"> ■ Feature value. Specifies whether the feature value is True or False.
Query Parameters	<i>This is applicable only for a REST API.</i> Validates the query parameters in the incoming request against the query parameters defined in that request's API Specification.
Path Parameters	<i>This is applicable only for a REST API.</i> Validates the path parameters in the incoming request against the path parameters defined in that request's API Specification.
Cookie Parameters	Validates the cookie parameters in the incoming request against the cookie parameters defined in that request's API Specification.
Content-types	Validates the content-types in the incoming request against the content-types defined in that request's API Specification.
HTTP Headers	<p>Validates the HTTP header parameters in the incoming request against the HTTP headers defined in that request's API Specification.</p> <p>Provide the following information:</p> <ul style="list-style-type: none"> ■ Condition. Specifies the logical operator to use to validate multiple HTTP headers in the incoming API requests. <p>Available values are:</p> <ul style="list-style-type: none"> ■ AND. API Gateway accepts only the requests that contain all configured HTTP headers. ■ OR. This is selected by default. API Gateway accepts requests that contain at least one configured HTTP header. ■ HTTP Header Key. Specifies a key that must be passed through the HTTP header of the incoming API requests. ■ Header Value. <i>Optional.</i> Specifies the corresponding key value that could be passed through the HTTP header of the incoming API requests. <p>The Header Value field type accepts string and regular expression (regex).</p> <p>You can add more HTTP headers by clicking  .</p>

Data Masking


Data masking is a technique whereby sensitive data is obscured in some way to render it safe and to protect the actual data while having a functional substitute for occasions when the real data is not required.



This policy is used to mask sensitive data at the application level. At the application level you must have an Identify and Access policy configured to identify the application for which the masking is applied. If no application is specified then it will be applied for all the other requests. Fields can be masked or filtered in the request messages received. You can configure the masking criteria as required for the XPath, JSONPath, and Regex expressions based on the content-type. This policy can also be applied at the API scope level.

The table lists the content-type and masking criteria mapping.

Content-type	Masking Criteria
application/xml	XPath
text/xml	
text/html	
application/json	JSONPath
application/json/badgerfish	
text/plain	Regex

The table lists the masking criteria properties that you can configure to mask the data in the request messages received:

Parameter	Description
Consumer Applications	<p><i>Optional.</i> Specifies the applications for which the masking criterion has to be applied.</p> <p>Start typing the application name, select the application from the type-ahead search results displayed, and click  to add one or more applications.</p> <p>For example: If there is a DataMasking(DM1) criteria created for application1 a second DataMasking(DM2) for application2 and a third DataMasking(DM3) with out any application, then for a request that comes from consumer1 the masking criteria DM1 is applied, for a request that comes from consumer2 DM2 is applied. If a request comes with out any application or from any other application except application1 and application2 DM3 is applied.</p>

Parameter	Description
	<p>You can use the delete icon  to delete the added applications from the list.</p>
XPath:	Specifies the masking criteria for XPath expressions in the request messages.
Masking Criteria	<p>Click Add masking criteria and provide the following information and click Add:</p> <ul style="list-style-type: none"> ■ Query expression. Specify the query expression that has to be masked or filtered. For example: /pet/details/status, /user/details/card/ccnumber. ■ Masking Type. Specifies the type of masking required. You select either Mask or Filter. Selecting Mask replaces the value with the given value (the default value being *****). Selecting Filter removes the field completely. ■ Mask Value. This is available if masking type selected is Mask. Provide a mask value. For example: sold, any mask value #####. <div style="background-color: #f0f0f0; padding: 5px; margin: 5px 0;"> <p>Note: You can add multiple masking criteria.</p> </div> <ul style="list-style-type: none"> ■ Namespace. Specifies the following Namespace information: <ul style="list-style-type: none"> ■ Namespace Prefix. The namespace prefix of the payload expression to be validated. ■ Namespace URI. The namespace URI of the payload expression to be validated <div style="background-color: #f0f0f0; padding: 5px; margin: 5px 0;"> <p>Note: You can add multiple namespace prefix and URI by clicking .</p> </div>
JSONPath:	This is applicable only for REST API. Specifies the masking criteria for JSONPath expressions in the request messages.
Masking Criteria	<p>Click Add masking criteria and provide the following information and click Add:</p> <ul style="list-style-type: none"> ■ Query expression. Specify the query expression that has to be masked or filtered. For example: \$.pet.details.status. ■ Masking Type. Specifies the type of masking required. You select either Mask or Filter. Selecting Mask replaces the value with the

Parameter	Description
	<p>given value (the default value being *****). Selecting Filter removes the field completely.</p> <ul style="list-style-type: none"> ■ Mask Value. This is available if masking type selected is Mask. Provide a mask value. For example: sold.
Regex: Specifies the masking criteria for regular expressions in the request messages.	
Masking Criteria	<p>Click Add masking criteria and provide the following information and click Add:</p> <ul style="list-style-type: none"> ■ Query expression. Specify the query expression that has to be masked or filtered. For example: [0-9]+. ■ Masking Type. Specifies the type of masking required. You select either Mask or Filter. Selecting Mask replaces the value with the given value (the default value being *****). Selecting Filter removes the field completely. ■ Mask Value. This is available if masking type selected is Mask. Provide a mask value. For example: ##### .
Apply for transaction Logging	<p>Select this option to apply masking criteria for transactional logs.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: For REST enabled SOAP services</p> <ul style="list-style-type: none"> ■ Use <i>JSONPath</i>. To mask the incoming request of application/json content-type. ■ Use <i>XPath of transformed SOAP request</i>. To mask native service request. </div>
Apply for payload	<p>Select this option to apply masking criteria for request payload in the following scenarios:</p> <ul style="list-style-type: none"> ■ incoming request from the client. ■ outgoing request to the native service.

Routing

The policies in this stage enforce routing of requests to target APIs based on the rules you can define to route the requests and manage their respective redirections according to the initial request path. The policies included in this stage are:

- Content-based Routing
- Context-based Routing
- Dynamic Routing

- Load Balancer Routing
- Straight Through Routing
- Custom HTTP Header
- Outbound Authentication - Transport
- Outbound Authentication - Message
- JMS/AMQP Routing
- JMS/AMQP Properties

In cases where the internal server is protected by a firewall, the endpoint in the routing policy that is applied should be configured as `apigateway://registrationPort-aliasname/relative path of the API`. Here the registration port alias name is the alias name configured for the external registration port to communicate with the internal port.

Content-based Routing

If you have a native API that is hosted at two or more endpoints, you can use the Content-based routing protocol to route specific types of messages to specific endpoints. You can route messages to different endpoints based on specific values that appear in the request message. You might use this capability, for example, to determine which operation the consuming application has requested, and route requests for complex operations to an endpoint on a fast machine. For example, if your entry protocol is HTTP or HTTPS, you can select the Content-based routing. The requests are routed according to the content-based routing rules you create. You may specify how to authenticate requests.

The table lists the properties that you can specify for this policy:

Parameter	Description
Default Route To:	Specifies the URLs of two or more native services in a pool to which the requests are routed.
Endpoint URI	<p>Specifies the URI of the native API endpoint to route the request to in case all routing rules evaluate to False. Service registries that have been added to the API Gateway instance are also included in the list.</p> <p>If you choose a service registry, API Gateway sends a request to the service registry to discover the IP address and port at which the native service is running. API Gateway replaces the service registry alias in the Endpoint URI with the IP address and port returned by the service registry.</p> <p>For example, if your service is hosted at the URL: <code>http://host:port/abc/</code>, you need to configure the Endpoint URI as: <code>http://\${ServiceRegistryName}/abc/</code>.</p>

Parameter	Description
HTTP Method	<p>This is applicable for REST-based APIs.</p> <p>Specifies the available routing methods: GET, POST, PUT, DELETE, and CUSTOM (default).</p> <p>When CUSTOM is selected, the HTTP method in the incoming request is sent to the native service. When other methods are selected, the selected method is used in the request sent to the native service.</p> <p>Note: Software AG recommends to use Request Transformation > Method Transformation to achieve this as other transformations can also be done under the same policy.</p>
SOAP Optimization Method	<p>This is applicable for SOAP-based APIs.</p> <p>Specifies the optimization methods that API Gateway can use to parse SOAP requests to the native API.</p> <p>Select one of the following options:</p> <ul style="list-style-type: none">■ MTOM. API Gateway uses the Message Transmission Optimization Mechanism (MTOM) to parse SOAP requests to the API.■ SwA. API Gateway uses the SOAP with Attachment (SwA) technique to parse SOAP requests to the API.■ None. API Gateway does not use any optimization method to parse the SOAP requests to the API. This is selected by default.
HTTP Connection Timeout (seconds)	<p>Specifies the time interval (in seconds) after which a connection attempt times out.</p> <p>The precedence of the Connection Timeout configuration is as follows:</p> <ol style="list-style-type: none">1. If you specify a value for the Connection timeout field in routing endpoint alias, then the Connection timeout value specified in the Endpoint alias section takes precedence over the timeout values defined at the API level and the global level.2. If you specify a value 0 for the Connection timeout field in routing endpoint alias, then API Gateway uses the value specified in the Connection timeout field in the routing protocol processing step of an API. The Read Timeout value specified at an API level takes precedence over the global configuration.

Parameter	Description
	<ol style="list-style-type: none"> 3. If you specify a value 0 or do not specify a value for the Connection timeout field in the routing protocol processing step at the API level or specify a value 0 at an alias level, then API Gateway uses the value specified in this <code>pg.endpoint.connectionTimeout</code> property. 4. If you do not specify any value for <code>pg.endpoint.connectionTimeout</code>, then API Gateway uses the default value of 30 seconds.
Read Timeout (seconds)	<p>Specifies the time interval (in seconds) after which a socket read attempt times out.</p> <p>The precedence of the Read Timeout configuration is as follows:</p> <ol style="list-style-type: none"> 1. If you specify a value for the Read timeout field in routing endpoint alias, then the Read timeout value specified in the Endpoint alias section takes precedence over the timeout values defined at the API level and the global level. 2. If you specify a value 0 for the Read timeout field in routing endpoint alias, then API Gateway uses the value specified in the Read Timeout field in the routing protocol processing step of an API. The Read Timeout value specified at an API level takes precedence over the global configuration. 3. If you specify a value 0 or do not specify a value for the Read timeout field in the routing protocol processing step at the API level or specify a value 0 at an alias level, then API Gateway uses the value specified in this <code>pg.endpoint.readTimeout</code> property. 4. If you do not specify any value for <code>pg.endpoint.readTimeout</code>, then API Gateway uses the default value of 30 seconds.
Pass WS-Security Headers	<p>This is applicable for SOAP-based APIs.</p> <p>Selecting this indicates that API Gateway should pass the WS-Security headers of the incoming requests to the native API.</p>
SSL Configuration.	<p>Configures keystore, key alias, and truststore for securing connections to native APIs.</p>
Keystore Alias	<p>Specifies the keystore alias configured in API Gateway. This value (along with the value of Client Certificate Alias) is used for performing SSL client authentication.</p> <p>Lists all available keystores. If you have not configured any keystore, the list is empty.</p>

Parameter	Description
Key Alias	Specifies the alias for the private key, which must be stored in the keystore specified by the keystore alias.
Truststore Alias	<p>Specifies the alias for the truststore that contains the list of CA certificates that API Gateway uses to validate the trust relationship with the native API.</p> <p>If you do not configure any truststore alias, it implies that API Gateway does not validate the certificates provided by native APIs.</p>

Service Registry Configuration

Service Discovery Endpoint Parameter Values required for constructing the discovery service URI.

- **Parameter:** An alias that you have included in the discovery service URI while adding the service registry to API Gateway.
- **Value:** A value for the path parameter. The alias specified in Path Parameter is substituted with this value when invoking the discovery service.


For example: if the service registry configuration of the service registry that you have selected in **Endpoint URI** has **Service discovery path** set to `"/catalog/service/{serviceName}"` (and the `{serviceName}` alias is intended for passing the service name), you must enter `{serviceName}` as **Parameter** and the name of the service as **Value**.

Rule: Defines the routing decisions based on one of the following routing options. Click **Add Rule** and provide the following information.

Payload Identifier Specifies using the payload identifier to identify the client, extract the custom authentication credentials supplied in the request represented using the payload identifier, and verify the client's identity.

In the Payload identifier section, click **Add payload identifier**, provide the following information, and click **Add**.

- **Expression type.** Specifies the type of expression, which is used for identification. You can select one the following expression type:
 - **XPath.** Provide the following information:
 - **Payload Expression.** Specifies the payload expression that the specified XPath expression type in the request has to be converted to. For example: `/name/id`

Parameter	Description
	<ul style="list-style-type: none"> ■ Namespace Prefix. The namespace prefix of the payload expression to be validated. ■ Namespace URI. The namespace URI of the payload expression to be validated.
	<p>Note: You can add multiple namespace prefix and URI by clicking .</p>
	<ul style="list-style-type: none"> ■ JSONPath. Provide the Payload Expression that specifies the payload expression that the specified JSONPath expression type in the request has to be converted to. For example: \$.name.id ■ Text. Provide the Payload Expression that specifies the payload expression that the specified Text expression type in the request has to be converted to. For example: any valid regular expression.
	<p>You can add multiple payload identifiers as required.</p>
	<p>Note: Only one payload identifier of each type is allowed. For example, you can add a maximum of three payload identifiers, each being of a different type.</p>
Route To.	Specifies the Endpoint URI of native APIs in a pool to which the requests are routed.
Endpoint URI	Specifies the URI of the native API endpoint to route the request to. You can use service registries in a similar manner as described in the main Endpoint URI above.
HTTP Method	<p>This is applicable for REST-based APIs.</p> <p>Specifies the available routing methods: GET, POST, PUT, DELETE, and CUSTOM (default).</p> <p>When CUSTOM is selected, the HTTP method in the incoming request is sent to the native service. When other methods are selected, the selected method is used in the request sent to the native service.</p>
Soap Optimization Method	<p>This is applicable for SOAP-based APIs.</p> <p>Specifies the optimization methods that API Gateway can use to parse SOAP requests to the native API.</p>

Parameter	Description
	<p>Select one of the following options:</p> <ul style="list-style-type: none"> ■ MTOM. API Gateway uses the Message Transmission Optimization Mechanism (MTOM) to parse SOAP requests to the API. ■ SwA. API Gateway uses the SOAP with Attachment (SwA) technique to parse SOAP requests to the API. ■ None. API Gateway does not use any optimization method to parse the SOAP requests to the API. This is selected by default.
<p>HTTP Connection Timeout (seconds)</p>	<p>Specifies the time interval (in seconds) after which a connection attempt times out.</p> <p>The precedence of the Connection Timeout configuration is as follows:</p> <ol style="list-style-type: none"> 1. If you specify a value for the Connection timeout field in routing endpoint alias, then the Connection timeout value specified in the Endpoint alias section takes precedence over the timeout values defined at the API level and the global level. 2. If you specify a value 0 for the Connection timeout field in routing endpoint alias, then API Gateway uses the value specified in the Connection timeout field in the routing protocol processing step of an API. The Read Timeout value specified at an API level takes precedence over the global configuration. 3. If you specify a value 0 or do not specify a value for the Connection timeout field in the routing protocol processing step at the API level or specify a value 0 at an alias level, then API Gateway uses the value specified in this <code>pg.endpoint.connectionTimeout</code> property. 4. If you do not specify any value for <code>pg.endpoint.connectionTimeout</code>, then API Gateway uses the default value of 30 seconds.
<p>Read Timeout (seconds)</p>	<p>Specifies the time interval (in seconds) after which a socket read attempt times out.</p> <p>The precedence of the Read Timeout configuration is as follows:</p> <ol style="list-style-type: none"> 1. If you specify a value for the Read timeout field in routing endpoint alias, then the Read timeout value specified in the Endpoint alias section takes precedence over the timeout values defined at the API level and the global level.

Parameter	Description
	<ol style="list-style-type: none"> <li data-bbox="623 260 1471 436">2. If you specify a value 0 for the Read timeout field in routing endpoint alias, then API Gateway uses the value specified in the Read Timeout field in the routing protocol processing step of an API. The Read Timeout value specified at an API level takes precedence over the global configuration. <li data-bbox="623 457 1471 634">3. If you specify a value 0 or do not specify a value for the Read timeout field in the routing protocol processing step at the API level or specify a value 0 at an alias level, then API Gateway uses the value specified in this <code>pg.endpoint.readTimeout</code> property. <li data-bbox="623 655 1471 730">4. If you do not specify any value for <code>pg.endpoint.readTimeout</code>, then API Gateway uses the default value of 30 seconds.
Pass WS-Security Headers	<p>This is applicable for SOAP-based APIs.</p> <p>Selecting this indicates that API Gateway should pass the WS-Security headers of the incoming requests to the native API.</p>
SSL Configuration	<p>Configures keystore, key alias, and truststore for securing connections to native APIs.</p> <p>Provide the following information:</p> <ul style="list-style-type: none"> <li data-bbox="623 1058 1471 1163">■ Keystore Alias. Specifies the keystore alias configured in API Gateway. This value (along with the value of Client Certificate Alias) is used for performing SSL client authentication. <p data-bbox="672 1184 1435 1255">Lists all available keystores. If you have not configured any keystore, the list is empty.</p> <li data-bbox="623 1289 1471 1360">■ Key Alias. Specifies the alias for the private key, which must be stored in the keystore specified by the keystore alias. <li data-bbox="623 1381 1471 1486">■ Truststore Alias. Specifies the alias for the truststore that contains the list of CA certificates that API Gateway uses to validate the trust relationship with the native API. <p data-bbox="672 1507 1471 1612">If you do not configure any truststore alias, it implies that API Gateway does not validate the certificates provided by native APIs.</p>
Service Registry Configuration	
Service Discovery Endpoint Parameter	<p>Values required for constructing the discovery service URI.</p> <ul style="list-style-type: none"> <li data-bbox="623 1759 1471 1822">■ Parameter: An alias that you have included in the discovery service URI while adding the service registry to API Gateway.

Parameter	Description
	<ul style="list-style-type: none"> ■ Value: A value for the path parameter. The alias specified in Path Parameter is substituted with this value when invoking the discovery service. <p>For example: if the service registry configuration of the service registry that you have selected in Endpoint URI has Service discovery path set to <code>"/catalog/service/{serviceName}"</code> (and the <code>{serviceName}</code> alias is intended for passing the service name), you must enter <code>{serviceName}</code> as Parameter and the name of the service as Value.</p>

Context-based Routing

If you have a native API that is hosted at two or more endpoints, you can use the Context-based protocol to route specific types of messages to specific endpoints. The requests are routed according to the context-based routing rules you create. For example, if your entry protocol is HTTP or HTTPS, you can select the Context-based routing. A routing rule specifies where requests should be routed, and the criteria by which they should be routed there. You may specify how to authenticate requests.

The table lists the properties that you can specify for this policy:

Parameter	Description
Default Route To	Specifies the URLs of two or more native services in a pool to which the requests are routed.
Endpoint URI	<p>Specifies the URI of the native API endpoint to route the request to in case all routing rules evaluate to False. Service registries that have been added to the API Gateway instance are also included in the list.</p> <p>If you choose a service registry, API Gateway sends a request to the service registry to discover the IP address and port at which the native service is running. API Gateway replaces the service registry alias in the Endpoint URI with the IP address and port returned by the service registry.</p> <p>For example, if your service is hosted at the URL: <code>http://host:port/abc/</code>, you need to configure the Endpoint URI as: <code>http://\${ServiceRegistryName}/abc/</code>.</p>
HTTP Method	<p>This is applicable to REST-based APIs.</p> <p>Specifies the available routing methods: GET, POST, PUT, DELETE, and CUSTOM (default).</p>

Parameter	Description
	<p>When CUSTOM is selected, the HTTP method in the incoming request is sent to the native service. When other methods are selected, the selected method is used in the request sent to the native service.</p> <p>Note: Software AG recommends to use Request Transformation > Method Transformation to achieve this as other transformations can also be done under the same policy.</p>

SOAP Optimization Method This is applicable for SOAP-based APIs.

Specifies the optimization methods that API Gateway can use to parse SOAP requests to the native API.

Select one of the following options:

- **MTOM.** API Gateway uses the Message Transmission Optimization Mechanism (MTOM) to parse SOAP requests to the API.
- **SwA.** API Gateway uses the SOAP with Attachment (SwA) technique to parse SOAP requests to the API.
- **None.** API Gateway does not use any optimization method to parse the SOAP requests to the API. This is selected by default.

HTTP Connection Timeout (seconds) Specifies the time interval (in seconds) after which a connection attempt times out.

The precedence of the Connection Timeout configuration is as follows:

1. If you specify a value for the **Connection timeout** field in routing endpoint alias, then the **Connection timeout** value specified in the Endpoint alias section takes precedence over the timeout values defined at the API level and the global level.
2. If you specify a value 0 for the **Connection timeout** field in routing endpoint alias, then API Gateway uses the value specified in the **Connection timeout** field in the routing protocol processing step of an API. The Read Timeout value specified at an API level takes precedence over the global configuration.
3. If you specify a value 0 or do not specify a value for the **Connection timeout** field in the routing protocol processing step at the API level or specify a value 0 at an alias level, then API Gateway uses the value specified in this `pg.endpoint.connectionTimeout` property.

Parameter	Description
	<ol style="list-style-type: none"> If you do not specify any value for <code>pg.endpoint.connectionTimeout</code>, then API Gateway uses the default value of 30 seconds.
Read Timeout (seconds)	<p>Specifies the time interval (in seconds) after which a socket read attempt times out.</p> <p>The precedence of the Read Timeout configuration is as follows:</p> <ol style="list-style-type: none"> If you specify a value for the Read timeout field in routing endpoint alias, then the Read timeout value specified in the Endpoint alias section takes precedence over the timeout values defined at the API level and the global level. If you specify a value 0 for the Read timeout field in routing endpoint alias, then API Gateway uses the value specified in the Read Timeout field in the routing protocol processing step of an API. The Read Timeout value specified at an API level takes precedence over the global configuration. If you specify a value 0 or do not specify a value for the Read timeout field in the routing protocol processing step at the API level or specify a value 0 at an alias level, then API Gateway uses the value specified in this <code>pg.endpoint.readTimeout</code> property. If you do not specify any value for <code>pg.endpoint.readTimeout</code>, then API Gateway uses the default value of 30 seconds.
SSL Configuration.	Configures keystore, key alias, and truststore for securing connections to native APIs.
Keystore Alias	<p>Specifies the keystore alias configured in API Gateway. This value (along with the value of Client Certificate Alias) is used for performing SSL client authentication.</p> <p>Lists all available keystores. If you have not configured any keystore, the list is empty.</p>
Key Alias	Specifies the alias for the private key, which must be stored in the keystore specified by the keystore alias.
Truststore Alias	<p>Specifies the alias for the truststore that contains the list of CA certificates that API Gateway uses to validate the trust relationship with the native API.</p> <p>If you do not configure any truststore alias, it implies that API Gateway does not validate the certificates provided by native APIs.</p>
Pass WS-Security Headers	This is applicable for SOAP-based APIs.

Parameter	Description
	Selecting this indicates that API Gateway should pass the WS-Security headers of the incoming requests to the native API.
Service Registry Configuration	
Service Discovery Endpoint Parameter	<p>Values required for constructing the discovery service URI.</p> <ul style="list-style-type: none"> ■ Parameter: An alias that you have included in the discovery service URI while adding the service registry to API Gateway. ■ Value: A value for the path parameter. The alias specified in Path Parameter is substituted with this value when invoking the discovery service. <p>For example: if the service registry configuration of the service registry that you have selected in Endpoint URI has Service discovery path set to <code>"/catalog/service/{serviceName}"</code> (and the <code>{serviceName}</code> alias is intended for passing the service name), you must enter <code>{serviceName}</code> as Parameter and the name of the service as Value.</p>
Rule. Defines the routing decisions based on one of the following routing options.	
Name	Provide a name for the rule.
Condition Operator	<p>Specifies the condition operator to be used.</p> <p>Select one of the following operators:</p> <ul style="list-style-type: none"> ■ OR. Specifies that one of the set conditions should be applied. ■ AND. Specifies all the set conditions should be applied.
Condition	Specify the context variables for processing client requests.
Variable	<p>Select any of the following variables:</p> <ul style="list-style-type: none"> ■ Consumer. Specifies the name of the consumer application in the text box. <ul style="list-style-type: none"> ■ Variable Value. Provide a value in the Variable Value text box. ■ Date <ul style="list-style-type: none"> ■ Operator. Select an operator: After or Before . ■ Variable Value. Type a date value in the text box. ■ IPV4. Specifies that IP version to be IPV4. <ul style="list-style-type: none"> ■ From IP.Type an IP address range.

Parameter	Description
	<ul style="list-style-type: none"> ■ To IP. Type an IP address range. ■ IPV6. Specifies that IP version to be IPV6. ■ From IP. Type an IP address range. ■ To IP. Type an IP address range. ■ Predefined Context Variable <ul style="list-style-type: none"> ■ Predefined Context. Select a predefined context. ■ Operator. Select one of the following operators: Equal To or Not Equal To. ■ Variable Value. Type a value for the selected predefined context. ■ Predefined Context Variable <ul style="list-style-type: none"> ■ Predefined Context. Select a predefined context. The available variables are: <ul style="list-style-type: none"> ■ User. API Gateway registered user. ■ Inbound HTTP method. The HTTP method used by the client to send the request (GET, POST, PUT, DELETE, PATCH) ■ Routing method. The method used by the routing policy when you select CUSTOM as the HTTP method. If you do not define this context variable, then the method used is from the Inbound HTTP method. ■ Inbound content type. Content type of the request. ■ Inbound accept. Accept header in the incoming request from the client. ■ Inbound protocol. The protocol (HTTP or HTTPS) of the request. ■ Inbound request URI. A partial reference to an API (for HTTP/HTTPS only). The protocol, host and port are not part of the value. For example, if the API is invoked: <code>http://host:port/gateway/API</code> then the expected value of this variable would be <code>/gateway/API</code>. For a REST API, the URL also includes query string parameters. For example, if the following API is invoked:

Parameter	Description
	<p data-bbox="764 260 1474 323"><code>http://host:port/gateway/cars?vin=1234</code> the expected value of this variable would be <code>/gateway/cars?vin1234</code>.</p> <ul style="list-style-type: none"> <li data-bbox="716 352 1040 382">■ Inbound IP. Client IP <li data-bbox="716 411 1360 441">■ Gateway hostname. API Gateway host name. <li data-bbox="716 470 1247 499">■ Gateway IP. API Gateway IP address <li data-bbox="716 529 1474 592">■ Operation name. Operation name for SOAP APIs. It is empty for REST API. <li data-bbox="667 621 1458 684">■ Operator. Select one of the following operators: Equal To or Not Equal To. <li data-bbox="667 714 1442 777">■ Variable Value. Type a value for the selected predefined context. <li data-bbox="618 806 1008 835">■ Custom Context Variable <ul style="list-style-type: none"> <li data-bbox="667 865 1474 1180">■ Variable Name. Type a name for the custom context variable. Use the <code>pub.apigateway.ctxvar:setContextVariable</code> API for setting the value for the custom context variable. All custom-defined context variables must be declared in a custom namespace that is identified by using the prefix <code>mx</code> (for example, <code>mx:CUSTOM_VARIABLE</code>). All parameter names are case-sensitive. For more information on <code>pub.apigateway.ctxvar:setContextVariable</code> service, see “The API for Context Variables” on page 529. <li data-bbox="667 1209 1360 1239">■ Data Type. Select the data type: String or Integer. <li data-bbox="667 1268 1458 1331">■ Operator. Select one of the following operators based on the Data Type selected: <ul style="list-style-type: none"> <li data-bbox="716 1360 1198 1390">■ String: Equal To or Not Equal To <li data-bbox="716 1419 1474 1482">■ Integer: Equal To, Not Equal To, Greater Than, or Less Than <li data-bbox="667 1512 1474 1541">■ Variable Value. Type a value for the selected custom context. <li data-bbox="618 1570 732 1600">■ Time <ul style="list-style-type: none"> <li data-bbox="667 1629 1312 1659">■ Operator. Select an operator: After or Before. <li data-bbox="667 1688 1166 1717">■ Hours. Type a time value in hours. <li data-bbox="667 1747 1219 1776">■ Minutes. Type a time value in minutes.

Route To. Specifies the endpoint URI of native services in a pool to which the requests are routed.

Parameter	Description
Endpoint URI	Specifies the URI of the native API endpoint to route the request to. You can use service registries in a similar manner as described in the main Endpoint URI above.
HTTP Method	<p>This is applicable for REST-based APIs.</p> <p>Specifies the available routing methods: GET, POST, PUT, DELETE, and CUSTOM (default).</p> <p>When CUSTOM is selected, the HTTP method in the incoming request is sent to the native service. When other methods are selected, the selected method is used in the request sent to the native service.</p>
Soap Optimization Method	<p>This is applicable for SOAP-based APIs.</p> <p>Specifies values to enable SSL authentication for SOAP APIs.</p> <p>Select one of the following options:</p> <ul style="list-style-type: none"> ■ MTOM. API Gateway uses the Message Transmission Optimization Mechanism (MTOM) to parse SOAP requests to the API. ■ SwA. API Gateway uses the SOAP with Attachment (SwA) technique to parse SOAP requests to the API. ■ None. API Gateway does not use any optimization method to parse the SOAP requests to the API. This is selected by default.
HTTP Connection Timeout (seconds)	<p>Specifies the time interval (in seconds) after which a connection attempt times out.</p> <p>The precedence of the Connection Timeout configuration is as follows:</p> <ol style="list-style-type: none"> 1. If you specify a value for the Connection timeout field in routing endpoint alias, then the Connection timeout value specified in the Endpoint alias section takes precedence over the timeout values defined at the API level and the global level. 2. If you specify a value 0 for the Connection timeout field in routing endpoint alias, then API Gateway uses the value specified in the Connection timeout field in the routing protocol processing step of an API. The Read Timeout value specified at an API level takes precedence over the global configuration. 3. If you specify a value 0 or do not specify a value for the Connection timeout field in the routing protocol processing step at the API level or specify a value 0 at an alias level, then

Parameter	Description
	<p>API Gateway uses the value specified in this <code>pg.endpoint.connectionTimeout</code> property.</p> <p>4. If you do not specify any value for <code>pg.endpoint.connectionTimeout</code>, then API Gateway uses the default value of 30 seconds.</p>
Read Timeout (seconds)	<p>Specifies the time interval (in seconds) after which a socket read attempt times out.</p> <p>The precedence of the Read Timeout configuration is as follows:</p> <ol style="list-style-type: none"> 1. If you specify a value for the Read timeout field in routing endpoint alias, then the Read timeout value specified in the Endpoint alias section takes precedence over the timeout values defined at the API level and the global level. 2. If you specify a value 0 for the Read timeout field in routing endpoint alias, then API Gateway uses the value specified in the Read Timeout field in the routing protocol processing step of an API. The Read Timeout value specified at an API level takes precedence over the global configuration. 3. If you specify a value 0 or do not specify a value for the Read timeout field in the routing protocol processing step at the API level or specify a value 0 at an alias level, then API Gateway uses the value specified in this <code>pg.endpoint.readTimeout</code> property. 4. If you do not specify any value for <code>pg.endpoint.readTimeout</code>, then API Gateway uses the default value of 30 seconds.
Pass WS-Security Headers	<p>This is applicable for SOAP-based APIs.</p> <p>Selecting this indicates that API Gateway should pass the WS-Security headers of the incoming requests to the native API.</p>
SSL Configuration	<p>Configures keystore, key alias, and truststore for securing connections to native APIs.</p> <p>Provide the following information:</p> <ul style="list-style-type: none"> ■ Keystore Alias. Specifies the keystore alias configured in API Gateway. This value (along with the value of Client Certificate Alias) is used for performing SSL client authentication. <p>Lists all available keystores. If you have not configured any keystore, the list is empty.</p>

Parameter	Description
	<ul style="list-style-type: none"> ■ Key Alias. Specifies the alias for the private key, which must be stored in the keystore specified by the keystore alias. ■ Truststore Alias. Specifies the alias for the truststore that contains the list of CA certificates that API Gateway uses to validate the trust relationship with the native API. <p>If you do not configure any truststore alias, it implies that API Gateway does not validate the certificates provided by native APIs.</p>

Service Registry Configuration

Service Discovery Endpoint Parameter	<p>Values required for constructing the discovery service URI.</p> <ul style="list-style-type: none"> ■ Parameter: An alias that you have included in the discovery service URI while adding the service registry to API Gateway. ■ Value: A value for the path parameter. The alias specified in Path Parameter is substituted with this value when invoking the discovery service. <p>For example: if the service registry configuration of the service registry that you have selected in Endpoint URI has Service discovery path set to <code>"/catalog/service/{serviceName}"</code> (and the <code>{serviceName}</code> alias is intended for passing the service name), you must enter <code>{serviceName}</code> as Parameter and the name of the service as Value.</p>
---	---

Dynamic Routing

This policy enables API Gateway to support dynamic routing of virtual aliases based on policy configuration. The configured policies are enforced on the request sent to an API and these requests are forwarded to the dynamic endpoint based on specific criteria that you specify.

The table lists the properties that you can specify for this policy:

Parameter	Description
Route To	Specifies the URLs of two or more native services in a pool to which the requests are routed.
Endpoint URI	Specifies the URI of the native API endpoint to route the request to in case all routing rules evaluate to False. Service registries that have been added to the API Gateway instance are also included in the list.

Parameter	Description
	<p>If you choose a service registry, API Gateway sends a request to the service registry to discover the IP address and port at which the native service is running. API Gateway replaces the service registry alias in the Endpoint URI with the IP address and port returned by the service registry.</p> <p>For example, if your service is hosted at the URL: <code>http://host:port/abc/</code>, you need to configure the Endpoint URI as: <code>http://\${ServiceRegistryName}/abc/</code>.</p>
HTTP Method	<p>This is applicable to REST-based APIs.</p> <p>Specifies the available routing methods: GET, POST, PUT, DELETE, and CUSTOM (default).</p> <p>When CUSTOM is selected, the HTTP method in the incoming request is sent to the native service. When other methods are selected, the selected method is used in the request sent to the native service.</p> <div data-bbox="623 898 1474 1066" style="background-color: #f0f0f0; padding: 5px;"> <p>Note: Software AG recommends to use Request Transformation > Method Transformation to achieve this as other transformations can also be done under the same policy.</p> </div>
SOAP Optimization Method	<p>This is applicable for SOAP-based APIs.</p> <p>Specifies the optimization methods that API Gateway can use to parse SOAP requests to the native API.</p> <p>Select one of the following options:</p> <ul style="list-style-type: none"> ■ MTOM. API Gateway uses the Message Transmission Optimization Mechanism (MTOM) to parse SOAP requests to the API. ■ SwA. API Gateway uses the SOAP with Attachment (SwA) technique to parse SOAP requests to the API. ■ None. API Gateway does not use any optimization method to parse the SOAP requests to the API. This is selected by default.
HTTP Connection Timeout (seconds)	<p>Specifies the time interval (in seconds) after which a connection attempt times out.</p> <p>The precedence of the Connection Timeout configuration is as follows:</p> <ol style="list-style-type: none"> 1. If you specify a value for the Connection timeout field in routing endpoint alias, then the Connection timeout value

Parameter	Description
	<p>specified in the Endpoint alias section takes precedence over the timeout values defined at the API level and the global level.</p> <ol style="list-style-type: none"> <li data-bbox="524 352 1378 527">2. If you specify a value 0 for the Connection timeout field in routing endpoint alias, then API Gateway uses the value specified in the Connection timeout field in the routing protocol processing step of an API. The Read Timeout value specified at an API level takes precedence over the global configuration. <li data-bbox="524 554 1378 728">3. If you specify a value 0 or do not specify a value for the Connection timeout field in the routing protocol processing step at the API level or specify a value 0 at an alias level, then API Gateway uses the value specified in this <code>pg.endpoint.connectionTimeout</code> property. <li data-bbox="524 751 1378 850">4. If you do not specify any value for <code>pg.endpoint.connectionTimeout</code>, then API Gateway uses the default value of 30 seconds.
Read Timeout (seconds)	<p>Specifies the time interval (in seconds) after which a socket read attempt times out.</p> <p>The precedence of the Read Timeout configuration is as follows:</p> <ol style="list-style-type: none"> <li data-bbox="524 1035 1378 1171">1. If you specify a value for the Read timeout field in routing endpoint alias, then the Read timeout value specified in the Endpoint alias section takes precedence over the timeout values defined at the API level and the global level. <li data-bbox="524 1199 1378 1373">2. If you specify a value 0 for the Read timeout field in routing endpoint alias, then API Gateway uses the value specified in the Read Timeout field in the routing protocol processing step of an API. The Read Timeout value specified at an API level takes precedence over the global configuration. <li data-bbox="524 1400 1378 1575">3. If you specify a value 0 or do not specify a value for the Read timeout field in the routing protocol processing step at the API level or specify a value 0 at an alias level, then API Gateway uses the value specified in this <code>pg.endpoint.readTimeout</code> property. <li data-bbox="524 1602 1378 1667">4. If you do not specify any value for <code>pg.endpoint.readTimeout</code>, then API Gateway uses the default value of 30 seconds.
Pass WS-Security Headers	<p>This is applicable for SOAP-based APIs.</p> <p>Selecting this indicates that API Gateway should pass the WS-Security headers of the incoming requests to the native API.</p>

Parameter	Description
SSL Configuration.	Configures keystore, key alias, and truststore for securing connections to native APIs.
Keystore Alias	<p>Specifies the keystore alias configured in API Gateway. This value (along with the value of Client Certificate Alias) is used for performing SSL client authentication.</p> <p>Lists all available keystores. If you have not configured any keystore, the list is empty.</p>
Key Alias	Specifies the alias for the private key, which must be stored in the keystore specified by the keystore alias.
Truststore Alias	<p>Specifies the alias for the truststore that contains the list of CA certificates that API Gateway uses to validate the trust relationship with the native API.</p> <p>If you do not configure any truststore alias, it implies that API Gateway does not validate the certificates provided by native APIs.</p>

Service Registry Configuration

Service Discovery Endpoint Values required for constructing the discovery service URI.

Parameter

- **Parameter:** An alias that you have included in the discovery service URI while adding the service registry to API Gateway.
- **Value:** A value for the path parameter. The alias specified in Path Parameter is substituted with this value when invoking the discovery service.

For example: if the service registry configuration of the service registry that you have selected in **Endpoint URI** has **Service discovery path** set to `"/catalog/service/{serviceName}"` (and the `{serviceName}` alias is intended for passing the service name), you must enter `{serviceName}` as **Parameter** and the name of the service as **Value**.

Rule. Defines the routing decisions based on one of the following routing options.

Route Using

Defines the dynamic URL based on the HTTP header value sent by the client or the context variable value.

Select one of the following:

- **Header:** Select and specify the **Name** required. This header name is configured by the API provider and is used to decide the routing decisions at the API level. The request message must be routed to the dynamic URL generated from the HTTP header value.

Parameter	Description
	<ul style="list-style-type: none"> ■ Context: The API providers must provide IS service in the policy, Invoke webMethods Integration Server. IS service would perform custom manipulations and set the value for the Context Variable ROUTING_ENDPOINT. API Gateway takes this ROUTING_ENDPOINT value as the native endpoint value and performs the routing. ■ Name. This field is displayed only when you select Header as the routing method. Type a name for the Routing header. API Gateway expects this header name in the incoming request that invokes the API.
Route To	<p>Specifies the endpoint URI of native services in a pool to which the requests are routed.</p> <p>Provide the following information:</p> <ul style="list-style-type: none"> ■ Endpoint URI . Specifies the URI of the native API endpoint to route the request to. You can use service registries in a similar manner as described in the main Endpoint URI above. <p>As this property supports variable framework, you can make use of the available variables. For example, you can configure the endpoint URI using hard coded URL, simple alias, endpoint alias, and variable syntax or any of these combination. If you define the endpoint URI as</p> <pre>http://\${myAliasHost}:\${request.headers.nativeport}/\${sys:resource-path},</pre> <p>where the <code>\${myAliasHost}</code> variable syntax is used to define the simple alias and the <code>\${request.headers.nativeport}</code> variable syntax is used to define the native port based on the request.</p> <p>You can also use the system-defined alias <code>\${sys:dyn-Endpoint}</code>. When you use the system-defined alias, the variables are replaced at runtime by the Header value or the Context value, selected as the Route To option.</p> <p>Consider the following URL with the system-defined alias:</p> <pre>http://HOSTNAME:5555/rest/com/ softwareag/mediator/samples/dynamicRouting/ validateDynamicURI/\${sys:dyn-Endpoint}</pre> <p>Now, if the incoming request has Header value as resource, the <code>\${sys:dyn-Endpoint}</code> alias in the URL is replaced by the Header value and the effective URL is</p> <pre>http://HOSTNAME:5555/rest/com/</pre>

Parameter	Description
	<p>softwareag/mediator/samples/dynamicRouting/validateDynamicURI/resource.</p>
	<p>Note: If you use endpoint alias, make sure the endpoint alias is created before you define it in the policy. For example, if you define <code>alias</code> syntax in the policy before creating the <code>alias</code> as endpoint alias, API Gateway considers <code>alias</code> as custom variable or simple alias and tries to resolve against those. So in that case, after creating endpoint alias you have to edit and save the API or policy to associate <code>alias</code> syntax with the endpoint alias.</p>
	<ul style="list-style-type: none"> ■ HTTP Method. This applicable to REST-based APIs. Specifies the available routing methods: GET, POST, PUT, DELETE, and CUSTOM (default). <p>When CUSTOM is selected, the HTTP method in the incoming request is sent to the native service. When other methods are selected, the selected method is used in the request sent to the native service.</p>
	<ul style="list-style-type: none"> ■ HTTP Connection Timeout (seconds). Specifies the time interval (in seconds) after which a connection attempt times out.
	<p>The precedence of the Connection Timeout configuration is as follows:</p>
	<ol style="list-style-type: none"> 1. If you specify a value for the Connection timeout field in routing endpoint alias, then the Connection timeout value specified in the Endpoint alias section takes precedence over the timeout values defined at the API level and the global level. 2. If you specify a value 0 for the Connection timeout field in routing endpoint alias, then API Gateway uses the value specified in the Connection timeout field in the routing protocol processing step of an API. The Read Timeout value specified at an API level takes precedence over the global configuration. 3. If you specify a value 0 or do not specify a value for the Connection timeout field in the routing protocol processing step at the API level or specify a value 0 at an alias level, then API Gateway uses the value specified in this <code>pg.endpoint.connectionTimeout</code> property.

Parameter	Description
	<p>4. If you do not specify any value for <code>pg.endpoint.connectionTimeout</code>, then API Gateway uses the default value of 30 seconds.</p> <ul style="list-style-type: none"> ■ Read Timeout (seconds). Specifies the time interval (in seconds) after which a socket read attempt times out. <p>The precedence of the Read Timeout configuration is as follows:</p> <ol style="list-style-type: none"> 1. If you specify a value for the Read timeout field in routing endpoint alias, then the Read timeout value specified in the Endpoint alias section takes precedence over the timeout values defined at the API level and the global level. 2. If you specify a value 0 for the Read timeout field in routing endpoint alias, then API Gateway uses the value specified in the Read Timeout field in the routing protocol processing step of an API. The Read Timeout value specified at an API level takes precedence over the global configuration. 3. If you specify a value 0 or do not specify a value for the Read timeout field in the routing protocol processing step at the API level or specify a value 0 at an alias level, then API Gateway uses the value specified in this <code>pg.endpoint.readTimeout</code> property. 4. If you do not specify any value for <code>pg.endpoint.readTimeout</code>, then API Gateway uses the default value of 30 seconds. <ul style="list-style-type: none"> ■ SSL Configuration. Configures keystore, key alias, and truststore for securing connections to native APIs. Provide the following information: <ul style="list-style-type: none"> ■ Keystore Alias. Specifies the keystore alias configured in API Gateway. This value (along with the value of Client Certificate Alias) is used for performing SSL client authentication. <p>Lists all available keystores. If you have not configured any keystore, the list is empty.</p> <ul style="list-style-type: none"> ■ Key Alias. Specifies the alias for the private key, which must be stored in the keystore specified by the keystore alias. ■ Truststore Alias. Specifies the alias for the truststore that contains the list of CA certificates that API Gateway uses to validate the trust relationship with the native API.

Parameter	Description
	If you do not configure any truststore alias, it implies that API Gateway does not validate the certificates provided by native APIs.
SOAP Optimization Method	<p>This is applicable for SOAP-based APIs.</p> <p>Specifies the optimization methods that API Gateway can use to parse SOAP requests to the native API.</p> <p>Select one of the following options:</p> <ul style="list-style-type: none"> ■ MTOM. API Gateway uses the Message Transmission Optimization Mechanism (MTOM) to parse SOAP requests to the API. ■ SwA. API Gateway uses the SOAP with Attachment (SwA) technique to parse SOAP requests to the API. ■ None. API Gateway does not use any optimization method to parse the SOAP requests to the API. This is selected by default.
Pass WS-Security Headers	<p>This is applicable for SOAP-based APIs.</p> <p>Selecting this indicates that API Gateway should pass the WS-Security headers of the incoming requests to the native API.</p>

Load Balancer Routing

If you have an API that is hosted at two or more endpoints, you can use the load balancing option to distribute requests among the endpoints. Requests are distributed across multiple endpoints. The requests are routed based on the round-robin execution strategy. The load for a service is balanced by directing requests to two or more services in a pool, until the optimum level is achieved. The application routes requests to services in the pool sequentially, starting from the first to the last service without considering the individual performance of the services. After the requests have been forwarded to all the services in the pool, the first service is chosen for the next loop of forwarding.

If the entry protocol is HTTP or HTTPS, you can select the Load Balancer routing.

The table lists the properties that you can specify for this policy:

Parameter	Description
Route To	Specifies the URLs of two or more native services in a pool to which the requests are routed.
Endpoint URI	Specifies the URI of the native API endpoint to route the request to in case all routing rules evaluate to False. Service registries that

Parameter	Description
	<p>have been added to the API Gateway instance are also included in the list.</p> <p>If you choose a service registry, API Gateway sends a request to the service registry to discover the IP address and port at which the native service is running. API Gateway replaces the service registry alias in the Endpoint URI with the IP address and port returned by the service registry.</p> <p>For example, if your service is hosted at the URL: <code>http://host:port/abc/</code>, you need to configure the Endpoint URI as: <code>http://{ServiceRegistryName}/abc/</code>.</p>
HTTP Method	<p>This is applicable to REST APIs.</p> <p>Specifies the available routing methods: GET, POST, PUT, DELETE, and CUSTOM (default).</p> <p>When CUSTOM is selected, the HTTP method in the incoming request is sent to the native service. When other methods are selected, the selected method is used in the request sent to the native service.</p>
SOAP Optimization Method	<p>This is applicable for SOAP-based APIs.</p> <p>Specifies the optimization methods that API Gateway can use to parse SOAP requests to the native API.</p> <p>Select one of the following options:</p> <ul style="list-style-type: none"> ■ MTOM. API Gateway uses the Message Transmission Optimization Mechanism (MTOM) to parse SOAP requests to the API. ■ SwA. API Gateway uses the SOAP with Attachment (SwA) technique to parse SOAP requests to the API. ■ None. API Gateway does not use any optimization method to parse the SOAP requests to the API. This is selected by default.
HTTP Connection Timeout (seconds)	<p>Specifies the time interval (in seconds) after which a connection attempt times out.</p> <p>The precedence of the Connection Timeout configuration is as follows:</p> <ol style="list-style-type: none"> 1. If you specify a value for the Connection timeout field in routing endpoint alias, then the Connection timeout value specified in the Endpoint alias section takes precedence over the timeout values defined at the API level and the global level.

Parameter	Description
	<ol style="list-style-type: none"> <li data-bbox="626 254 1468 464">2. If you specify a value 0 for the Connection timeout field in routing endpoint alias, then API Gateway uses the value specified in the Connection timeout field in the routing protocol processing step of an API. The Read Timeout value specified at an API level takes precedence over the global configuration. <li data-bbox="626 491 1468 667">3. If you specify a value 0 or do not specify a value for the Connection timeout field in the routing protocol processing step at the API level or specify a value 0 at an alias level, then API Gateway uses the value specified in this <code>pg.endpoint.connectionTimeout</code> property. <li data-bbox="626 695 1468 793">4. If you do not specify any value for <code>pg.endpoint.connectionTimeout</code>, then API Gateway uses the default value of 30 seconds.
Read Timeout (seconds)	<p data-bbox="626 821 1468 890">Specifies the time interval (in seconds) after which a socket read attempt times out.</p> <p data-bbox="626 917 1468 947">The precedence of the Read Timeout configuration is as follows:</p> <ol style="list-style-type: none"> <li data-bbox="626 974 1468 1108">1. If you specify a value for the Read timeout field in routing endpoint alias, then the Read timeout value specified in the Endpoint alias section takes precedence over the timeout values defined at the API level and the global level. <li data-bbox="626 1136 1468 1312">2. If you specify a value 0 for the Read timeout field in routing endpoint alias, then API Gateway uses the value specified in the Read Timeout field in the routing protocol processing step of an API. The Read Timeout value specified at an API level takes precedence over the global configuration. <li data-bbox="626 1339 1468 1516">3. If you specify a value 0 or do not specify a value for the Read timeout field in the routing protocol processing step at the API level or specify a value 0 at an alias level, then API Gateway uses the value specified in this <code>pg.endpoint.readTimeout</code> property. <li data-bbox="626 1543 1468 1608">4. If you do not specify any value for <code>pg.endpoint.readTimeout</code>, then API Gateway uses the default value of 30 seconds.
Suspend duration (seconds)	<p data-bbox="626 1635 1468 1665">A numeric timeout value (in seconds). The default value is 30.</p> <p data-bbox="626 1692 1468 1831">This property specifies the time, in seconds, for which API Gateway temporarily suspends an endpoint, whenever Read time-out or Connection time-out occurs for the endpoint, and routes the request to the next configured endpoint in this time interval.</p>

Parameter	Description
	<p>For example: If you have 3 endpoints configured endpoint #1, endpoint #2, and endpoint #3, the suspend duration is configured as 60 seconds for endpoint #2, and there is a Read Timeout or Connection Timeout for endpoint #2, then API Gateway temporarily suspends endpoint #2 for 60 seconds. In this time interval API Gateway skips endpoint #2 while routing the requests to the configured endpoints.</p> <p>Request 1 -> endpoint #1</p> <p>Request 2 -> endpoint #3 (endpoint #2 is suspended for 60 seconds and hence the request is sent to endpoint #3)</p> <p>Request 3 -> endpoint #1</p>
Pass WS-Security Headers	<p>This is applicable for SOAP-based APIs.</p> <p>Selecting this indicates that API Gateway should pass the WS-Security headers of the incoming requests to the native API.</p>
SSL Configuration.	<p>Configures keystore, key alias, and truststore for securing connections to native APIs.</p>
Keystore Alias	<p>Specifies the keystore alias configured in API Gateway. This value (along with the value of Client Certificate Alias) is used for performing SSL client authentication.</p> <p>Lists all available keystores. If you have not configured any keystore, the list is empty.</p>
Key Alias	<p>Specifies the alias for the private key, which must be stored in the keystore specified by the above keystore alias.</p>
Truststore Alias	<p>Specifies the alias for the truststore that contains the list of CA certificates that API Gateway uses to validate the trust relationship with the native API.</p> <p>If you do not configure any truststore alias, it implies that API Gateway does not validate the certificates provided by native APIs.</p>

Service Registry Configuration

Service Discovery Endpoint	<p>Values required for constructing the discovery service URI.</p>
Parameter	<ul style="list-style-type: none"> ■ Parameter: An alias that you have included in the discovery service URI while adding the service registry to API Gateway. ■ Value: A value for the path parameter. The alias specified in Path Parameter is substituted with this value when invoking the discovery service.

Parameter	Description
	For example: if the service registry configuration of the service registry that you have selected in Endpoint URI has Service discovery path set to <code>"/catalog/service/{serviceName}"</code> (and the <code>{serviceName}</code> alias is intended for passing the service name), you must enter <code>{serviceName}</code> as Parameter and the name of the service as Value .

Straight Through Routing

When you select the Straight Through routing protocol, the API routes the requests directly to the native service endpoint you specify. If your entry protocol is HTTP or HTTPS, you can select the Straight Through routing policy.

The table lists the properties that you can specify for this policy:

Parameter	Value
Endpoint URI	<p>Specifies the URI of the native API endpoint to route the request to in case all routing rules evaluate to False. Service registries that have been added to the API Gateway instance are also included in the list.</p> <p>If you choose a service registry, API Gateway sends a request to the service registry to discover the IP address and port at which the native service is running. API Gateway replaces the service registry alias in the Endpoint URI with the IP address and port returned by the service registry.</p> <p>For example, if your service is hosted at the URL: <code>http://host:port/abc/</code>, you need to configure the Endpoint URI as: <code>http://\${ServiceRegistryName}/abc/</code>.</p>
HTTP Method	<p>This is applicable to REST-based APIs.</p> <p>Specifies the available routing methods: GET, POST, PUT, DELETE, and CUSTOM (default).</p> <p>When CUSTOM is selected, the HTTP method in the incoming request is sent to the native service. When other methods are selected, the selected method is used in the request sent to the native service.</p> <div style="background-color: #f0f0f0; padding: 10px;"> <p>Note: Software AG recommends to use Request Transformation > Method Transformation to achieve this as other transformations can also be done under the same policy.</p> </div>

Parameter	Value
Soap Optimization Method	<p>This is applicable for SOAP-based APIs.</p> <p>Specifies the optimization methods that API Gateway can use to parse SOAP requests to the native API.</p> <p>Select one of the following options:</p> <ul style="list-style-type: none"> ■ MTOM. API Gateway uses the Message Transmission Optimization Mechanism (MTOM) to parse SOAP requests to the API. ■ SwA. API Gateway uses the SOAP with Attachment (SwA) technique to parse SOAP requests to the API. ■ None. API Gateway does not use any optimization method to parse the SOAP requests to the API. This is selected by default.
HTTP Connection Timeout (seconds)	<p>Specifies the time interval (in seconds) after which a connection attempt times out.</p> <p>The precedence of the Connection Timeout configuration is as follows:</p> <ol style="list-style-type: none"> 1. If you specify a value for the Connection timeout field in routing endpoint alias, then the Connection timeout value specified in the Endpoint alias section takes precedence over the timeout values defined at the API level and the global level. 2. If you specify a value 0 for the Connection timeout field in routing endpoint alias, then API Gateway uses the value specified in the Connection timeout field in the routing protocol processing step of an API. The Read Timeout value specified at an API level takes precedence over the global configuration. 3. If you specify a value 0 or do not specify a value for the Connection timeout field in the routing protocol processing step at the API level or specify a value 0 at an alias level, then API Gateway uses the value specified in this <code>pg.endpoint.connectionTimeout</code> property. 4. If you do not specify any value for <code>pg.endpoint.connectionTimeout</code>, then API Gateway uses the default value of 30 seconds.
Read Timeout (seconds)	<p>Specifies the time interval (in seconds) after which a socket read attempt times out.</p> <p>The precedence of the Read Timeout configuration is as follows:</p>

Parameter	Value
	<ol style="list-style-type: none"> 1. If you specify a value for the Read timeout field in routing endpoint alias, then the Read timeout value specified in the Endpoint alias section takes precedence over the timeout values defined at the API level and the global level. 2. If you specify a value 0 for the Read timeout field in routing endpoint alias, then API Gateway uses the value specified in the Read Timeout field in the routing protocol processing step of an API. The Read Timeout value specified at an API level takes precedence over the global configuration. 3. If you specify a value 0 or do not specify a value for the Read timeout field in the routing protocol processing step at the API level or specify a value 0 at an alias level, then API Gateway uses the value specified in this <code>pg.endpoint.readTimeout</code> property. 4. If you do not specify any value for <code>pg.endpoint.readTimeout</code>, then API Gateway uses the default value of 30 seconds.

Pass WS-Security Headers This is applicable for SOAP-based APIs.

Selecting this indicates that API Gateway should pass the WS-Security headers of the incoming requests to the native API.

SSL configuration. Configures keystore, key alias, and truststore for securing connections to native APIs.

Keystore Alias Specifies the keystore alias configured in API Gateway. This value (along with the value of Client Certificate Alias) is used for performing SSL client authentication.

Lists all available keystores. If you have not configured any keystore, the list is empty.

Key Alias Specifies the alias for the private key, which must be stored in the keystore specified by the keystore alias.

Truststore Alias Specifies the alias for the truststore that contains the list of CA certificates that API Gateway uses to validate the trust relationship with the native API.

If you do not configure any truststore alias, it implies that API Gateway does not validate the certificates provided by native APIs.

Service Registry Configuration

Service Discovery Endpoint Values required for constructing the discovery service URI.

Parameter

Parameter	Value
	<ul style="list-style-type: none"> ■ Parameter: An alias that you have included in the discovery service URI while adding the service registry to API Gateway. ■ Value: A value for the path parameter. The alias specified in Path Parameter is substituted with this value when invoking the discovery service. <p>For example: if the service registry configuration of the service registry that you have selected in Endpoint URI has Service discovery path set to <code>"/catalog/service/{serviceName}"</code> (and the <code>{serviceName}</code> alias is intended for passing the service name), you must enter <code>{serviceName}</code> as Parameter and the name of the service as Value.</p>

Custom HTTP Header

You can use this policy to route requests based on the custom HTTP headers specified for the outgoing message to the native service.

The table lists the properties that you can specify for this policy:

Parameter	Description
HTTP Header Key	Specifies the HTTP header key contained in the requests.
Header Value	Specifies the Header value contained in the requests.

You can add multiple entries for the Header key-value pair by clicking



Outbound Authentication - Transport

When the native API is protected and expects the authentication credentials to be passed through transport headers, you can use this policy to provide the credentials that will be added to the request and sent to the native API. API Gateway supports a wide range of authentication schemes, such as Basic Authentication, Kerberos, NTLM, and OAuth, at the transport-level.

Note:

Transport-level authentication can be used to secure inbound communication of both the SOAP APIs and the REST APIs.

The table lists the properties that you can specify for this policy:

Parameter	Description
Authentication scheme	Select one of the following schemes for outbound authentication at the transport level:

Parameter	Description
	<ul style="list-style-type: none"> ■ Basic. Uses basic HTTP authentication details to authenticate the client. ■ Kerberos. Uses Kerberos credentials for authentication. ■ NTLM. Uses NTLM configuration for authentication. ■ OAuth2. Uses OAuth token details to authenticate the client. ■ JWT. Uses JSON web token details to authenticate the client. ■ Anonymous. Authenticates the client without any credentials. ■ Alias. Uses the configured alias name for authentication.
Authenticate using	<p>Select one of the following modes to authenticate the client:</p> <ul style="list-style-type: none"> ■ Custom credentials. Uses the values specified in the policy to obtain the required token to access the native API. ■ Delegate incoming credentials. Uses the values specified in the policy by the API providers to select whether to delegate the incoming token or act as a normal client. ■ Incoming HTTP Basic Auth credentials. Uses the incoming user credentials to retrieve the authentication token to access the native API. ■ Incoming kerberos credentials. Uses the incoming kerberos credentials to access the native API. ■ Incoming OAuth token. Uses the incoming OAuth2 token to access the native API. ■ Incoming JWT. Uses the incoming JSON Web Token (JWT) to access the native API. ■ Transparent. Enables NTLM handshake between client and native API. API Gateway does not perform any authentication before passing the incoming requests to native API. It simply passes the incoming credentials to native API. The NTLM authentication happens at the native API.
Basic	<p>Uses the HTTP authentication details to authenticate the client.</p> <p>API Gateway supports the following modes of HTTP authentication:</p> <ul style="list-style-type: none"> ■ Custom credentials ■ Incoming HTTP Basic Auth credentials <p>Provide the following credentials:</p>

Parameter	Description
	<ul style="list-style-type: none">■ User Name. Specifies the user name.■ Password. Specifies the password of the user.■ Domain . Specifies the domain in which the user resides.
Kerberos	<p>Uses the Kerberos credentials to authenticate the client.</p> <p>API Gateway supports the following modes of Kerberos authentication:</p> <ul style="list-style-type: none">■ Custom credentials■ Delegate incoming credentials■ Incoming HTTP basic auth credentials■ Incoming kerberos credentials <p>Provide the following credentials:</p> <ul style="list-style-type: none">■ Client principal. Provide a valid client LDAP user name.■ Client password. Provide a valid password of the client LDAP user.■ Service principal. Provide a valid SPN. The specified value is used by the client to obtain a service ticket from the KDC server.■ Service Principal Name Form. The SPN type to use while authenticating an incoming client principal name. Select any of the following:<ul style="list-style-type: none">■ User name. Specifies the username form.■ Hostbased. Specifies the host form.
NTLM	<p>Uses the NTLM credentials to authenticate the client.</p> <p>API Gateway supports the following modes of NTLM authentication:</p> <ul style="list-style-type: none">■ Custom credentials■ Incoming HTTP basic auth credentials■ Transparent <p>Provide the following credentials:</p> <ul style="list-style-type: none">■ User Name. Specifies the user name.■ Password. Specifies the password of the user.

Parameter	Description
	<ul style="list-style-type: none"> ■ Domain . Specifies the domain in which the user resides.
OAuth2	<p>Uses the OAuth2 token to authenticate the client.</p> <p>API Gateway supports the following modes of NTLM authentication:</p> <ul style="list-style-type: none"> ■ Custom credentials ■ Incoming OAuth token <p>OAuth2 token. Specifies the client's OAuth2 token.</p>
JWT	<p>Uses the JSON Web Token (JWT) to authenticate the client.</p> <p>If the native API is enforced to use JWT for authenticating the client, then API Gateway enforces the need for a valid JWT in the outbound request while accessing the native API.</p> <p>API Gateway supports the Incoming JWT mode of JWT authentication.</p>
Alias	Name of the configured alias.

When you configure an API with an inbound authentication policy, and a client sends a request with credentials, API Gateway uses the credentials for the inbound authentication. When sending the request to native server, API Gateway removes the already authenticated credentials when no outbound authentication policy is configured.

If as an API provider you want to use the same credentials for authentication at both API Gateway and native server, you should configure the outbound authentication policy to pass the incoming credentials to the native service. If you do not configure an outbound authentication policy, API Gateway removes the incoming credentials, as it is meant for API Gateway authentication only.

However, when both the inbound authentication policy and outbound authentication policy are not configured, API Gateway just acts as a proxy and forwards the credentials to the native service. Since the credentials are not meant for API Gateway (as no inbound auth policy is configured), API Gateway forwards the credentials to native service (unless there are different settings configured in outbound authentication policy, for example, custom credentials or Anonymous).

Outbound Authentication - Message

When the native API is protected and expects the authentication credentials to be passed through payload message, you can use this policy to provide the credentials that is added to the request and sent to the native API. API Gateway supports a wide range of authentication schemes, such as WSS Username, SAML, and Kerberos, in addition to signing and encryption at the message-level.

Note:

Message-level authentication can be used to secure outbound communication of only SOAP APIs.

The table lists the properties that you can specify for this policy:

Parameter	Description
Authentication scheme	<p>Select one of the following schemes for outbound authentication at the message level:</p> <ul style="list-style-type: none"> ■ WSS username. Uses WSS credentials to authenticate the client. ■ SAML. Uses SAML issuer configuration details for authentication. ■ Kerberos. Uses Kerberos credentials for authentication. ■ None. Authenticates the client without any authentication schemes. ■ Alias. Uses the configured alias name for authentication. ■ Remove WSS headers. Uses the WSS headers for authentication.
Authenticate using	<p>Select one of the following modes to authenticate the client:</p> <ul style="list-style-type: none"> ■ Custom credentials. Uses the values specified in the policy to obtain the required token to access the native service. ■ Incoming HTTP Basic Auth credentials. Uses the incoming user credentials to retrieve the authentication token to access the native API. ■ Delegate incoming credentials. Uses the values specified in the policy by the API providers to select whether to delegate the incoming token or act as a normal client.
WSS username	<p>Uses the WSS credentials to authenticate the client.</p> <p>Provide the following credentials:</p> <ul style="list-style-type: none"> ■ User Name. Specifies the user name. ■ Password. Specifies the password of the user.
Kerberos	<p>Uses the Kerberos credentials to authenticate the client.</p> <p>Provide the following information:</p> <ul style="list-style-type: none"> ■ Client principal. Provide a valid client LDAP user name. ■ Client password. Provide a valid password of the client LDAP user. ■ Service principal. Provide a valid SPN. The specified value is used by the client to obtain a service ticket from the KDC server.

Parameter	Description
	<ul style="list-style-type: none"> ■ Service Principal Name Form. The SPN type to use while authenticating an incoming client principal name. Select any of the following: <ul style="list-style-type: none"> ■ User name. Specifies the username form. ■ Hostbased. Specifies the host form.
SAML	Provide the SAML issuer that is configured.
Signing and Encryption Configurations	<p>Uses the signing and encryption configuration details to authenticate the client.</p> <p>Provide the following information:</p> <ul style="list-style-type: none"> ■ Keystore Alias. Specifies a user-specified text identifier for an API Gateway keystore. The alias points to a repository of private keys and their associated certificates. ■ Key Alias. Specifies the alias for the private key, which must be stored in the keystore specified by the keystore alias. ■ Truststore alias. Specifies the alias for the truststore. The truststore contains the trusted root certificate for the CA that signed the API Gateway certificate associated with the key alias. ■ Certificate alias. Provide a text identifier for the certificate associated with the truststore alias. API Gateway populates the certificate alias list with the certificate aliases from the selected truststore alias.
Alias	Uses the Kerberos credentials to authenticate the client. Provide the name of the configured alias.

When you configure an API with an inbound authentication policy, and a client sends a request with credentials, API Gateway uses the credentials for the inbound authentication. When sending the request to native server, API Gateway removes the already authenticated credentials when no outbound authentication policy is configured.

If as an API provider you want to use the same credentials for authentication at both API Gateway and native server, you should configure the outbound authentication policy to pass the incoming credentials to the native service. If you do not configure an outbound authentication policy, API Gateway removes the incoming credentials, as it is meant for API Gateway authentication only.

However, when both the inbound authentication policy and outbound authentication policy are not configured, API Gateway just acts as a proxy and forwards the credentials to the native service. Since the credentials are not meant for API Gateway (as no inbound auth policy is configured), API Gateway forwards the credentials to native service (unless there are different settings configured in outbound authentication policy, for example, custom credentials or Anonymous).

JMS/AMQP Policies

To configure API Gateway for JMS with Message broker native protocol support or JMS with AMQP protocol you need to:

- Create one or more JNDI provider aliases to specify where API Gateway can look up when it needs to create a connection to JMS provider or specify a destination for sending or receiving messages.
- Create one or more connection aliases that encapsulate the properties that API Gateway needs to create a connection with the JMS provider.

JMS/AMQP Routing

You can use this policy when you want to specify a JMS queue or topic to which API Gateway submits the request, and the destination where the response should be routed to where API Gateway waits to listen to the response from the native API.

For example, you can use this policy when you have a native API that is exposed over AMQP or JMS and that requires clients to communicate with the server using other protocols. This policy allows you to bridge protocols between the client and the native API.

You can apply the JMS/AMQP routing policy to both REST and SOAP APIS. The following sections explain their usage.

Use case 1: Using the JMS/AMQP routing policy (JMS with a message broker native protocol) for a SOAP API

This describes the high level workflow for the scenario where you use the JMS/AMQP routing policy (JMS with a message broker native protocol) for a SOAP API.

1. Create an alias to a JNDI Provider For a detailed procedure, see [“Creating a JNDI Provider Alias” on page 83](#).
2. Configure API Gateway to use a JMS connection alias to establish an active connection between API Gateway and the JMS provider. For a detailed procedure, see [“Creating a JMS Connection Alias” on page 87](#).
3. Create a WS (Web Service) JMS Provider endpoint alias and configure the Alias, Description, Type (Provider), Transport Type (JMS) fields and JMS Transport Properties. For a detailed procedure, see [“Creating an Endpoint Alias for a Provider Web Service Descriptor” on page 96](#).
4. A WS (Web Service) endpoint trigger is created when you configure WS (Web Service) JMS Provider endpoint alias. This trigger consists of the input source details like Queue name or Topic name. You can update the WS (Web Service) endpoint trigger, as required. For detailed procedures, see [“Updating JMS triggers” on page 105](#).
5. Select the required API.
6. Click **Edit**.

7. In the API Details section click **Policies**.
8. Enforce the **JMS/AMQP Routing** policy with the following properties configured.
 - a. Specify the connection URL for connecting to the JMS provider.
 - b. Specify a queue name where a reply to the message must be sent.
 - c. Provide a priority of this JMS message.
 - d. Provide expiration time of the JMS message.
 - e. Specify the message delivery mode for the request message.

For details on the **JMS/AMQP Routing** policy, see [“Using JMS/AMQP Policy for a SOAP API” on page 471](#)

9. Click **Save**.

The enforced policy **JMS/AMQP Routing** with the required configuration now allows any java client to communicate with the API asynchronously.

Use case 2: Using the JMS/AMQP routing policy (JMS with AMQP protocol) for a SOAP API

This describes the high level workflow for the scenario where you use the JMS/AMQP routing policy (JMS with AMQP protocol) for a SOAP API.

1. Configure API Gateway to use JMS with AMQP protocol.

Configure JNDI settings and JMS settings as per the client you are using to achieve JMS over AMQP protocol support.

Note:

For a sample procedure on configuring API Gateway to use JMS with AMQP protocol using Apache qpid libraries, see [“Configuring API Gateway for JMS with AMQP Protocol” on page 387](#)

2. Select the required API.
3. Click **Edit**.
4. In the API Details section click **Policies**.
5. Enforce the **JMS/AMQP Routing** policy with the following properties configured.
 - a. Specify the connection URL for connecting to the JMS provider.
 - b. Specify a queue name where a reply to the message must be sent.
 - c. Provide a priority for this AMQP message.
 - d. Provide expiration time of the AMQP message.
 - e. Specify the message delivery mode for the request message.

For details on the **JMS/AMQP Routing** policy, see [“Using JMS/AMQP Policy for a SOAP API” on page 471](#)

6. Click **Save**.

The enforced policy **Enable JMS/AMQP** with the required configuration now allows all the clients such as Python, Ruby, Java, and Dotnet to communicate with the API asynchronously.

Use case 3: Using the JMS/AMQP routing policy (JMS with a message broker native protocol) for a REST API

This describes the high level workflow for the scenario where you use the JMS/AMQP routing policy (JMS with a message broker native protocol) for a REST API.

1. Create an alias to a JNDI Provider For a detailed procedure, see [“Creating a JNDI Provider Alias” on page 83](#).
2. Configure API Gateway to use a JMS connection alias to establish an active connection between API Gateway and the JMS provider. For a detailed procedure, see [“Creating a JMS Connection Alias” on page 87](#).
3. Select the required API.
4. Click **Edit**.
5. In the API Details section click **Policies**.
6. Enforce the **JMS/AMQP Routing** policy with the following properties configured.
 - a. Specify the connection alias that contains the configuration information needed to establish a connection to a specific JMS provider.
 - b. Specify the destination to which the request message is sent.
 - c. Specify the destination type to which the request message is sent.
 - d. Specify the destination to which the response message is sent.
 - e. Specify the type of destination, queue or topic, to which the response message is sent.
 - f. Provide expiration time of the JMS message.
 - g. Provide the time for which API Gateway listens for the response message.
 - h. Specify the message delivery mode for the request message.

For details on the **JMS/AMQP Routing** policy, see [“Using JMS/AMQP Policy for a REST API” on page 474](#)

7. Click **Save**.

The enforced policy **JMS/AMQP Routing** with the required configuration now allows any java client to communicate with the API asynchronously.

Use case 4: Using the JMS/AMQP routing policy (JMS with AMQP protocol) for a REST API

This describes the high level workflow for the scenario where you use the JMS/AMQP routing policy (JMS with AMQP protocol) for a REST API.

1. Configure API Gateway to use JMS with AMQP protocol.

Configure JNDI settings and JMS settings as per the client you are using to achieve JMS over AMQP protocol support.

Note:

For a sample procedure on configuring API Gateway to use JMS with AMQP protocol using Apache qpid libraries, see [“Configuring API Gateway for JMS with AMQP Protocol” on page 387](#)

2. Select the required API.
3. Click **Edit**.
4. In the API Details section click **Policies**.
5. Enforce the **JMS/AMQP Routing** policy with the following properties configured.
 - a. Specify the connection alias that contains the configuration information needed to establish a connection to a specific JMS provider.
 - b. Specify the destination to which the request message is sent.
 - c. Specify the destination type to which the request message is sent.
 - d. Specify the destination to which the response message is sent.
 - e. Specify the type of destination, queue or topic, to which the response message is sent.
 - f. Provide expiration time of the AMQP message.
 - g. Provide the time for which API Gateway listens for the response message.
 - h. Specify the message delivery mode for the request message.

For details on the **JMS/AMQP Routing** policy, see [“Using JMS/AMQP Policy for a REST API” on page 474](#).

6. Click **Save**.

The enforced policy **Enable JMS/AMQP** with the required configuration now allows all the clients such as Python, Ruby, Java, and Dotnet to communicate with the API asynchronously.

Using JMS/AMQP Policy for a SOAP API

If you are using JMS with Message broker native protocol support ensure that following actions are performed before using the Enable JMS/AMQP policy:

- Create an alias to a JNDI Provider For a detailed procedure, see [“Creating a JNDI Provider Alias” on page 83](#).
- Configure API Gateway to use a JMS connection alias to establish an active connection between API Gateway and the JMS provider. For a detailed procedure, see [“Creating a JMS Connection Alias” on page 87](#).
- Create a WS (Web Service) JMS Provider endpoint alias and configure the Alias, Description, Type (Provider), Transport Type (JMS) fields and JMS Transport Properties. For a detailed procedure, see [“Creating an Endpoint Alias for a Provider Web Service Descriptor” on page 96](#).
- Configure a WS (Web Service) endpoint trigger. For detailed procedures, see .

If you are using JMS with AMQP protocol support, ensure the following before using the Enable JMS/AMQP policy:

- You have configured API Gateway for JMS with AMQP. For details, see [“Configuring API Gateway for JMS with AMQP Protocol” on page 387](#)

The table lists the properties that you can specify for this policy:

Parameter	Description
Connection URL	<p>Provide a connection alias for connecting to the JMS provider (for example, an Integration Server alias or a JNDI URL). The connection URL contains various elements that construct the destination and other connection specific parameters. The structure of the connection URL is:</p> <pre><protocol>:<lookupVariant>:<destination>?<parameters> where</pre> <ul style="list-style-type: none"> ■ <i>protocol</i>. Specify the name of the transport protocol. The default value is JMS. ■ <i>lookupVariant</i>. Specify the destination type such as queue or topic. The default value is queue. ■ <i>destination</i>. Specify the destination name of the JMS Provider. For dynamic queue the destination name is: <code>dynamicQueues/<Queue name></code> ■ <i>Parameters</i> <ul style="list-style-type: none"> ■ <code>wm-wsendpointalias</code>. Specify the JMS consumer endpoint alias. This parameter is required for API Gateway to look up the JMS consumer alias and send the request to the specified queue. ■ <code>jndiInitialContextFactory</code>. Specify the initial context factory for the JNDI look up. For example: <code>org.apache.activemq.jndi.ActiveMQInitialContextFactory</code> for ActiveMQ ■ <code>jndiConnectionFactoryName</code>. Specify the connection factory look up name. For example: <ul style="list-style-type: none"> ■ <code>ConnectionFactory</code> for ActiveMQ if you are using the JMS with broker native protocol.

Parameter	Description
	<ul style="list-style-type: none"> ■ <code>qpidConnectionFactory</code> for ActiveMQ if you are using the JMS with AMQP protocol. ■ <code>jndiURL</code>. Specify the Provider URL for the Active MQ to connect to API Gateway. For example: <ul style="list-style-type: none"> ■ <code>tcp://vmmeddemo03:61616</code> for ActiveMQ if you are using the JMS with broker native protocol. ■ The file path location of the properties file, for example, <code>Install directory\IntegrationServer\lib\jars\amqp.properties</code> if you are using JMS with AMQP protocol. ■ <code>targetService</code>. Specify the API Gateway API name. This parameter is required if you are sending the request to another API in API Gateway that uses JMS as the entry protocol. <p>Sample: With consumer endpoint alias</p> <pre>jms:queue:dynamicQueues/MyTestQueue? wm-wsendpointalias=JMSConsumerEndpointAlias&target Service=EchoS_VS_JMS_IN</pre> <p>Sample: With JNDI lookup parameters</p> <pre>jms:queue:dynamicQueues/MyTestQueue? jndiConnectionFactoryName=ConnectionFactory &jndiInitialContextFactory=org.apache. activemq.jndi.ActiveMQInitialContextFactory &targetService=EchoS_VS_JMS_IN</pre> <p>Sample: With JNDI lookup parameters for AMQP protocol</p> <pre>jms:queue:dynamicQueues/MyTestQueue? jndiConnectionFactoryName=qpidConnectionFactory &jndiInitialContextFactory=org.apache.qpid.jms. jndi.JmsInitialContextFactory &targetService=EchoS_VS_JMS_IN</pre>
Reply To Destination	Specify a queue name where a reply to the message must be sent.
Priority	<p>Type an integer that represents the priority of this JMS or AMQP message with respect to other messages that are in the same queue. The priority value determines the order in which the messages are routed. The lowest priority value is 0 and the highest priority value is 9. The messages with this priority value are executed first.</p> <ul style="list-style-type: none"> ■ Priority values 0 through 9. ■ The default priority for a JMS or AMQP message is 0.

Parameter	Description
Time to Live (ms)	<p>Provide a numeric value that specifies the expiration time (in milliseconds) of the JMS or AMQP message.</p> <p>If the time-to-live is specified as zero, expiration is set to zero which indicates the message does not expire.</p> <p>The default value is 0.</p>
Delivery Mode	<p>The message delivery mode for the request message. This is the delivery mode that web service clients must specify in the JMS or AMQP message that serves as the request message for the web service. The available options are:</p> <ul style="list-style-type: none"> ■ Non-persistent. Indicates that the request message is not persistent. The message might be lost if the JMS provider fails. ■ Persistent. Indicates that the request message should be persistent. The message is not lost if the JMS provider fails.

Using JMS/AMQP Policy for a REST API

If you are using JMS with Message broker native protocol support ensure that following actions are performed before using the Enable JMS/AMQP policy:

- Create an alias to a JNDI Provider For a detailed procedure, see [“Creating a JNDI Provider Alias” on page 83](#).
- Configure API Gateway to use a JMS connection alias to establish an active connection between API Gateway and the JMS provider. For a detailed procedure, see [“Creating a JMS Connection Alias” on page 87](#).

If you are using JMS with AMQP protocol support, ensure the following before using the Enable JMS/AMQP policy:

- You have configured API Gateway for JMS with AMQP. For details, see [“Configuring API Gateway for JMS with AMQP Protocol” on page 387](#)

The table lists the properties that you can specify for this policy:

Parameter	Description
Connection Alias Name	<p>Specifies the name of the connection alias.</p> <p>Each connection alias contains the configuration information needed to establish a connection to a specific JMS provider.</p>
Destination Name	Specify the destination to which the request message is sent.
Destination Type	Specify the destination type to which the request message is sent.
Reply To Name	Specify the name of the destination to which the response message is sent.

Parameter	Description
Reply To Type	<p>Specifies the type of destination to which the response message is sent.</p> <p>Select one of the following source type:</p> <ul style="list-style-type: none"> ■ QUEUE. Indicates that the response message is sent to a particular queue. ■ TOPIC. Indicates that the response message is sent to a particular topic.
Time to Live (ms)	<p>Provide a numeric value that specifies the expiration time (in milliseconds) of the JMS or AMQP message. If the time-to-live is specified as zero, expiration is set to zero which indicates the message does not expire.</p> <p>The default value is 0.</p>
Time to Wait (ms)	<p>Defines the time in milliseconds for which API Gateway listens to the Reply To Queue or Topic for the response message.</p>
Delivery Mode	<p>The message delivery mode for the request message. This is the delivery mode that web service clients must specify in the JMS or AMQP message that serves as the request message for the web service. The available options are:</p> <ul style="list-style-type: none"> ■ Non-persistent. Indicates that the request message is not persistent. The message might be lost if the JMS provider fails. ■ Persistent. Indicates that the request message should be persistent. The message is not lost if the JMS provider fails.

JMS/AMQP Properties

The JMS/AMQP Properties policy can be configured to set AMQP or JMS Properties, a few standard AMQP or JMS Headers, and HTTP Transport Headers in the outgoing JMS message that is being sent from the proxy API to the native API.

AMQP or JMS headers are part of the JMS message that are used by both clients and providers. They are used to identify a message and to route the message to the applicable JMS Providers or consumers.

You can add HTTP Headers such as API Key, Authorization header, and so on. This is useful when the native API is configured with the Enable AMQP/JMS policy and the proxy API wants to pass the security headers over to that native API.

Every JMS message includes JMS/AMQP properties that are always passed from provider to client. The purpose of the properties is to convey extra information to the client outside the normal content of the message body. Additionally, JMS/AMQP property values are set exclusively by the consumer application. When a client receives a message, the properties are in read-only mode. If a client tries to modify any of the properties, a `MessageNotWriteableException` occurs.

The properties are standard Java name or value pairs. The property names must conform to the message selector syntax specifications defined in the message interface. Property fields are most often used for message selection and filtering. By using a property field, a message consumer can interrogate the property field and perform message filtering and selection. When this action is configured for a proxy API, API Gateway uses the JMS or AMQP properties to authenticate client requests before submitting to the native APIs. JMS or AMQP headers can also be set using properties, however, JMS or AMQP properties take precedence over headers.

The table lists the properties that you can specify for this policy:

Parameter	Description
JMS Property Key	Specify the JMS property key.
JMS Property Value	Specify the JMS property value for the specified key.

Predefined JMS Properties

Property categories	Property	Description	
Run-time settings	■ <code>jms.deliveryMode</code>	If the <code>jms.messageType</code> is set to <code>TextMessage</code> , the SOAP envelope in the request is sent as a text message to the JMS queue instead of byte stream.	
	■ <code>jms.priority</code>		
	■ <code>jms.timeToLive</code>		
	■ <code>jms.messageType</code>		
Standard JMS headers	■ <code>JMSType</code>	The following headers are not applicable. If they are added an error response would be sent at runtime:	
	■ <code>JMSCorrelationID</code>		
	■ <code>JMSXGroupID</code>		■ <code>JMSMessageID</code>
	■ <code>JMSXGroupSeq</code>		■ <code>JMSExpiration</code>
			■ <code>JMSRedelivered</code>
			■ <code>JMSTimestamp</code>
			■ <code>JMSDeliveryMode</code>
			■ <code>JMSPriority</code>
			■ <code>JMSReplyTo</code>
			■ <code>JMSDestination</code>
Application specific properties	■ <code>SOAPJMS_requestURI</code>		
	■ <code>SOAPJMS_bindingVersion</code>		

Property categories	Property	Description
	■ SOAPJMS_soapAction	
	■ SOAPJMS_targetService	
	■ SOAPJMS_contentType	

For more information on JMS properties and JMS headers, see *webMethods Integration Server Administrator's Guide*.

Mapping AMQP messages to JMS

Header

Field name	Description
durable	When receiving a message, the durable field of header MUST be mapped to the JMSDeliveryMode header of the Message. If the durable field of header is set to false or is not set then the JMSDeliveryMode MUST be taken to be NONPERSISTENT. When the durable field of header is set to true the JMSDeliveryMode of the Message MUST be taken to be PERSISTENT.
priority	This field is mapped to the JMSPriorityheader of the Message. JMS Priority is specified as being of type int despite the valid values only being 0-9. AMQP allows for the priority field of header to be any valid ubyte value. When receiving a message with the priority field of header greater than 9, the JMSPriority MUST be set to 9. If the priority field of header is unset then the JMSPriority MUST be taken to be DEFAULT_PRIORITY that is, the value 4).
ttl	This field defines the number of milliseconds for which a given message is considered live. There is no direct equivalent for the ttl field of header in the JMS specification. If and only if the absolute-expiry-time field of properties is not set, JMSExpiration SHOULD be based on the ttl field of header if set, by summing it with the current time in milliseconds since the Unix Epoch
first acquirer	This field does not have a direct equivalent within the JMS specification, although JMSRedelivered is related, and so vendor property JMS_AMQP_FIRST_ACQUIRER SHOULD be used.
delivery-count	This field is mapped to the JMS-defined JMSXDeliveryCount property and JMSRedelivered header of the Message as follows.

Field name	Description
	<p>AMQP uses the <code>delivery-count</code> field of header to track previously failed delivery attempts for a message, with the first delivery attempt having a value of zero, and soon.</p> <p><code>JMSXDeliveryCount</code> is defined as a Java <code>int</code> count of delivery attempts, set by the provider on receive, where the first delivery attempt has value 1, the second has value 2 and so on.</p> <p>The value of <code>JMSXDeliveryCount</code> property is thus equal to <code>delivery-count + 1</code>.</p> <p>The <code>JMSRedelivered</code> header MUST be considered to be true if and only if the <code>delivery-count</code> field of header has a value greater than 0.</p>

Properties

Field name	Description
<code>message-id</code>	<p>This field is equivalent to the <code>JMSMessageID</code> header of the <code>Message</code>.</p> <p>The <code>JMSMessageID</code> value is a Java <code>String</code> where as the <code>message-id</code> field of properties is defined as being of type providing <code>message-id</code>, that is <code>message-id-ulong</code>, <code>message-id-uuid</code>, <code>message-id-binary</code> or <code>message-id-string</code>.</p> <p>The JMS client library MUST prefix ID: to the value of the <code>message-id</code> field of properties before returning it as the <code>JMSMessageID</code> value.</p>
<code>user-id</code>	<p>This field is mapped to the JMS-defined <code>JMSXUserID</code> property of the <code>Message</code>.</p> <p><code>JMSXUserID</code> is specified as being of type <code>String</code>, while the <code>user-id</code> field of properties field is specified as type <code>binary</code>. To maintain end-to-end fidelity for this property implementations SHOULD convert between AMQP <code>binary</code> and Java <code>String</code> by using the UTF-8 Unicode[UNICODE63] character encoding.</p>
<code>to</code>	<p>This field is mapped to the <code>JMSDestination</code> header of the <code>Message</code>.</p> <p><code>JMSDestination</code> is defined as being of the <code>JMS Destination</code> type, while the <code>to</code> field of properties requires an <code>address-string</code>.</p> <p>If the <code>to</code> field of properties was not set on a received message, the <code>JMSDestination</code> header value SHOULD be derived from the <code>Destination</code> to which the receiving consumer was established.</p>
<code>subject</code>	<p>This field is mapped to the <code>JMSType</code> header of the <code>Message</code>.</p>
<code>reply-to</code>	<p>This field is mapped to the <code>JMSReplyTo</code> header of the <code>Message</code>.</p>

Field name	Description
correlation-id	<p>JMSReplyTo is defined as being of the JMSDestination type, while the reply-to field of properties requires an address-string.</p> <p>This field is mapped to the JMSCorrelationID header of the Message.</p> <p>The JMSCorrelationID value is a Java String where as the correlation-id field of properties is defined as being of type providing message-id, that is message-id-ulong, message-id-uuid, message-id-binary OR message-id-string.</p> <p>Where the correlation-id field of properties for the received message is of type message-id-string and the boolean message annotation with symbol key of x-opt-app-correlation-id is either not set or is false, then the correlation-id field of properties MUST be formatted as a JMSMessageID, that is the client library MUST prefix ID: to the value before returning it as the JMSCorrelationID value.</p>
content-type	<p>This field does not have an equivalent within the JMS specification, and so the vendor property JMSAMQPCONTENTTYPE SHOULD be used.</p>
content-encoding	<p>This field does not have an equivalent within the JMS specification, and so the vendor property JMSAMQPCONTENTENCODING SHOULD be used.</p>
absolute-expiry-time	<p>This field is mapped to the JMSExpiration head of the Message.</p> <p>If the absolute-expiry-time field of properties is set, then JMSExpiration MUST have the equivalent Java long value, representing the time at which the message expires, in milliseconds since the Unix Epoch. If the absolute-expiry-time field of properties is not set then JMSExpiration SHOULD be based on the ttl field of header instead if set.</p>
creation-time	<p>This field is mapped to the JMSTimestamp header of the Message.</p> <p>If the creation-time field of properties is not set, then JMSTimestamp MUST have the value zero. If the creation-time field of properties field is set, then JMSTimestamp MUST have the equivalent Java long value, representing the time at which the message was sent or created, in milliseconds since the Unix Epoch.</p>
group-id	<p>This field is mapped to the JMS-defined JMSXGroupID property of the Message.</p>
group-sequence	<p>This field is mapped to the JMS-defined JMSXGroupSeq property of the Message.</p> <p>As the group-sequence field of properties is an uint and JMSXGroupSeq is an int, group-sequence values in the range 2^{31} to $2^{32}-1$ inclusive</p>

Field name	Description
	MUST be mapped to JMSXGroupSeq values in the range -2^{31} to -1 inclusive.
reply-to-group-id	This field does not have an equivalent within the JMS specification, and so the vendor property JMS_AMQP_REPLY_TO_GROUP_ID MUST be used.

For more information on AMQP properties and JMS to AMQP mapping properties, see <https://www.oasis-open.org/committees/download.php/56418/amqp-bindmap-jms-v1.0-wd06.pdf>.

Traffic Monitoring

The policies in this stage provide ways to enable logging request and response payload, enable monitoring run-time performance conditions for APIs and applications, enforce limits for the number of service invocations during a specified time interval and send alerts to a specified destination when the performance conditions are violated, and enable caching of the results of API invocations depending on the caching criteria defined. The policies included in this stage are:


- Log Invocation
- Monitor Service Performance
- Monitor Service Level Agreement
- Throttling Traffic Optimization
- Service Result Cache

Log Invocation

This policy enables logging requests or responses to a specified destination. This action also logs other information about the requests or responses, such as the API name, operation name, the Integration Server user, a timestamp, and the response time.

The table lists the properties that you can specify for this policy:

Parameter	Description
Store Request Headers	Logs all request headers.
Store Request Payload	Logs all request payloads.
Store Response Headers	Logs all response headers.
Store Response Payload	Logs all response payloads.
Compress Payload Data	Compresses the logged payload data.

Parameter	Description
	<p>For details about payload compression and how to uncompress a payload, see “Uncompressing a payload” on page 482.</p>
Log Generation Frequency	<p>Specifies how frequently to log the payload.</p> <p>Select one of the following options:</p> <ul style="list-style-type: none"> ■ Always. Logs all requests and responses. ■ On Failure. Logs only the failed requests and responses. ■ On Success. Logs only the successful responses and requests.
Destination	<p>Specifies the destination where to log the payload.</p> <p>Select the required options:</p> <ul style="list-style-type: none"> ■ API Gateway ■ API Portal ■ Audit Log <p>Audit log destination can be configured as <i>DB</i> or <i>File</i> in the Administration > Destinations screen. Software AG recommends to use <i>DB</i> when you choose Audit Log as the destination to log transactions through Log Invocation policy.</p> <p>If you choose <i>File</i>, warnings appear in the log file since a few of the transaction log fields are not compatible with Audit log file destination such as BLOB types. For more information, see Configure Audit Logging section in <i>webMethods Audit Logging Guide</i>.</p> ■ CentraSite <div data-bbox="680 1360 1459 1493" style="background-color: #f0f0f0; padding: 5px;"> <p>Note: This option is applicable only for the APIs published from CentraSite to API Gateway.</p> </div> ■ Digital Events ■ Elasticsearch ■ Email (you can add multiple email addresses by clicking ). <div data-bbox="680 1696 1459 1829" style="background-color: #f0f0f0; padding: 5px;"> <p>Note: If an email alias is available, you can type the email alias in the Email Address field with the following syntax,</p> </div>

Parameter	Description
	<p><code>\${emailaliasname}</code>. For example, if test is the email alias, then type <code>\${test}</code>.</p> <ul style="list-style-type: none"> ■ JDBC ■ Local Log: You can select the severity of the messages to be logged (logging level) from the Log Level drop-down list. The available log levels are ERROR, INFO, and WARN. <p>Note:</p> <ul style="list-style-type: none"> ■ Set the Integration Server Administrator's logging level for API Gateway to match the logging levels specified for the run-time actions (go to Settings > Logging > Server Logger). For example, if a Log Invocation action is set to the logging level of Error, you must also set Integration Server Administrator's logging level for API Gateway to Error. If the action's logging level is set to a low level (Warning-level or Information level), but Integration Server Administrator's logging level for API Gateway is set to a higher level (Error-level), then only the higher-level messages are written to the log file. ■ Entries posted to the local log are identified by a product code of YAI and suffixed with the initial alphabet of the logging level selected. For example, for an error level, the entry appears as [YAI.0900.0002E]. <ul style="list-style-type: none"> ■ SNMP

Uncompressing a payload

Payload compression helps you to optimize the storage by reducing the size of the actual payload. It improves the performance while rendering the analytics information in the dashboard.

The request and response payload of the API Gateway API and native API is compressed in the encoded form.

> To generate the data and uncompress the payload.

1. Ensure you have an API enforced with a **Log invocation policy** with the property **Compress payload data** selected.


See the following example where an API is enforced with a Log invocation policy with Compress payload data selected.

- Invoke the same API using an external REST client such as Postman or SoapUI to see the API transaction.

TransactionalEvent is generated every time an API invocation happens.

- Click **Analytics** of the same API in API Gateway UI.

This displays the different types of events generated in the dashboard. For details about analytics, see [“Analytics Dashboards” on page 648](#).

- Select **Runtime events** and click  to expand your transaction.
- Click **JSON** or **Table** and copy the encoded string (value) of the request or response payload that you want to uncompress.

The screenshot shows the webMethods API Gateway interface. The top navigation bar includes 'WEBMETHODS API Gateway' and tabs for 'APIs', 'Policies', 'Applications', 'Packages', 'Microgateways', and 'AppMesh'. The breadcrumb trail is 'Home > APIs > Pet'. The main heading is 'Pet', with a subtitle 'View API details, basic and technical information, resource'. Below this are tabs for 'API details', 'Scopes', 'Policies', 'Mashups', 'Applications', and 'Analytics'. The 'Analytics' tab is active, showing a 'Default dashb' dropdown, a 'Last 12 hours' filter, and a 'From Date' field. A JSON log entry is displayed, with the 'resPayload' field highlighted in red:

```

{
  "_source": {
    "eventType": "Transactional",
    "sourceGateway": "APIGateway",
    "creationDate": 1620308120374,
    "apiName": "Pet",
    "apiVersion": "1",
    "apiId": "f78e7d42-9a74-4107-add6-73837a52fd59",
    "totalTime": 1199,
    "sessionId": "1a388b59860c4a04a299390649cab318",
    "providerTime": 1198,
    "gatewayTime": 1,
    "applicationName": "Unknown",
    "applicationIp": "172.18.112.1",
    "applicationId": "Unknown",
    "status": "SUCCESS",
    "reqPayload": "H4sIAAAAAAAAAAKtWKoksSFwYUjBKvU9XPjEqKMoMRnGNTCz1LC1rAboTdUMfAAAA",
    "resPayload": "H4sIAAAAAAAAAAKtWYkxRsrI0WjKyNLmWnDEwsbQwNTAxNjXXUSrIyC/Jdy3KKVayio7VUspJTAezagFgcFR1MwAAAA=",
    "totalDataSize": 82,
    "responseCode": "200",
    "cachedResponse": "Not-Cached"
  }
}

```

6. Pass the copied string as an *input* to the following Java program.

```

public static String uncompressString(String zippedBase64Str) throws IOException
{
    String unCompressedPayload = null;
    byte[] bytes = Base64.getDecoder().decode(zippedBase64Str);
    GZIPInputStream zi = null;
    try{
        zi = new GZIPInputStream(new ByteArrayInputStream(bytes));
        unCompressedPayload = IOUtils.toString(zi);
    }finally{
        IOUtils.closeQuietly(zi);
    }
    return unCompressedPayload;
}

```

See the following example, where an encoded string from the request payload is passed as an *input* to the Java program.

```

import org.apache.commons.io.IOUtils;

import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.util.Base64;
import java.util.zip.GZIPInputStream;

public class MyClass {
    public static void main(String[] args) throws IOException {
        String str = ["H4sIAAAAAAAAAAKtWykxRsrI0MjKyNLMwNDEwsbQwNTAxMjXXUSrIyC/JDy3KKVayio7VUSpJTAezagFgcFR1MwAAAA="];
        System.out.println(uncompressString(str));
    }

    public static String uncompressString(String zippedBase64Str) throws IOException {
        String unCompressedPayload = null;
        byte[] bytes = Base64.getDecoder().decode(zippedBase64Str);
        GZIPInputStream zi = null;
        try{
            zi = new GZIPInputStream(new ByteArrayInputStream(bytes));
            unCompressedPayload = IOUtils.toString(zi);
        }finally{
            IOUtils.closeQuietly(zi);
        }
        return unCompressedPayload;
    }
}

```

Input
↓

```

{"id":9222968140498504257,"photoUrls":[],"tags":[]}

```

↑
Output

The Java *output* contains the uncompressed payload.

Note:

This code snippet is applicable only for the payload compressed by the log invocation policy.

You can also query the data using the REST endpoints from the swagger file `APIGatewaySearch.json` and uncompress the payload with the same code snippet.

For details about the REST endpoints, see [“API Gateway Search” on page 762](#).

The screenshot shows an API client interface with the following details:

- Request:** POST to `http://localhost:5555/rest/apigateway/search`. The body is a JSON object:


```

{
  "types": ["TRANSACTION_EVENTS"],
  "condition": "and",
  "scope": {
    "attributeName": "apiName",
    "keyword": ".*"
  },
  "from": 0,
  "size": 10,
  "sortByField": "apiName",
  "sortOrder": "ASC"
}

```
- Response:** Status 200 OK, 82 ms, 28.27 KB. The body is a JSON object:


```

{
  "gatewayTime": 1,
  "applicationName": "Unknown",
  "applicationIp": "172.18.112.1",
  "applicationId": "Unknown",
  "status": "SUCCESS",
  "reqPayload": "H4sIAAAAAAAAAAKtkKqks5FlyU1BKyu9X01EqKpMbnGNTCz1LC1rAbpTdUHfAAAA",
  "resPayload": "H4sIAAAAAAAAAAKtkKqks5FlyU1BKyu9X01EqKpMbnGNTCz1LC1rAbpTdUHfAAAA",
  "totalDataSize": 82,
  "responseCode": "200",
}

```

Monitor Service Performance

This policy is similar to the Monitor Service Level Agreement policy and does monitor the same set of run-time performance conditions for an API, and sends alerts when the performance conditions are violated. However, this policy monitors run-time performance at the API level. Parameters like success count, fault count and total request count are immediate monitoring parameters and the evaluation happens immediately after the limit is breached. The rest of the parameters are Aggregated monitoring parameters whose evaluation happens once the configured interval is over. If there is a breach in any of the parameters, an event notification (Monitor event) is sent to the configured destination. In a single policy, multiple action configurations behave as AND condition. The OR condition can be achieved by configuring multiple policies.

The table lists the properties that you can specify for this policy:

Parameter	Value
-----------	-------

Action Configuration. Specifies the type of action to be configured.

Name Specifies the name of the metric to be monitored.

You can select one of the available metrics:

- **Availability.** Indicates whether the native API is available to the clients as specified in the current interval. API Gateway calculates the availability of the native API based on the alert interval specified and it is calculated from the instant the API activation takes place.

Parameter	Value
-----------	-------

The availability of the API is calculated as = (time for which the native API is up / total interval of time) x 100. This value is measured in %.


For example, if you set **Availability** as less than 90, then whenever the availability of the native API falls below 90%, in the specified time interval, API Gateway generates an alert. Suppose, the alert interval is set as 1 minute (60 seconds) and if there are 7 API invocations at various times in that 1 minute with a combination of up and down as shown in the table, the availability is calculated as follows:

Request #	Invocation time (the second at which the API is invoked)	Service status	Up time
1	5	Up	5 (from start to now)
2	15	Up	10 (between 1 and 2)
3	30	Down	15 (between 2 and 3)
4	40	Down	0 (since last state is Down)
5	45	Up	0
6	50	Down	5 (between 5 and 6)
7	55	Up	0
			5 (remaining 5 seconds considered as Up inline with last state)
	Total		40 (Availability is 67%)

As the availability of the native API calculated is 66.67% and falls below 90%, API Gateway generates an alert. The API is considered to be down for the ongoing request when API Gateway receives a connection related error from the native API in the outbound call. If the API does not respond with an HTTP response, then it is considered as down.

- **Average Response Time.** Indicates the average time taken by the service to complete all invocations in the current interval. The average is calculated from the instant the API activation takes place for the configured interval.

Parameter	Value
	<p>For example, if you set an alert for Average response time greater than 30 ms with an interval of 1 minute then on API activation, the monitoring interval starts and the average of the response time of all runtime invocations for this API in 1 minute is calculated. If this is greater than 30 ms, then a monitor event is generated. If this is configured under Monitor service performance, then all the runtime invokes are taken into account.</p> <ul style="list-style-type: none"> ■ Fault Count. Indicates the number of faults returned in the current interval. The HTTP status codes greater than or equal to 400, returned from API Gateway are considered as fault request transactions. This includes the downtime errors as well. ■ Maximum Response Time. Indicates the maximum time in milliseconds to respond to a request in the current interval. ■ Minimum Response Time. Indicates the minimum time in milliseconds to respond to a request in the current interval. ■ Success Count. Indicates the number of successful requests in the current interval. ■ Total Request Count. Indicates the total number of requests (successful and unsuccessful) in the current interval.
Operator	<p>Specifies the operator applicable to the metric selected.</p> <p>Select one of the available operator: Greater Than, Less Than, Equals To.</p>
Value	<p>Specifies the alert value for which the monitoring is applied.</p>
Destination	<p>Specifies the destination where the alert is to be logged.</p> <p>Select the required options:</p> <ul style="list-style-type: none"> ■ API Gateway ■ API Portal ■ CentraSite <div style="background-color: #f0f0f0; padding: 5px; margin: 5px 0;"> <p>Note: This option is applicable only for the APIs published from CentraSite to API Gateway.</p> </div> <ul style="list-style-type: none"> ■ Digital Events ■ Elasticsearch

Parameter	Value
	<ul style="list-style-type: none"> ■ Email (you can add multiple email addresses by clicking ). <div style="background-color: #f0f0f0; padding: 10px; margin: 10px 0;"> <p>Note: If an email alias is available, you can type the email alias in the Email Address field with the following syntax, <code>\${emailaliasname}</code>. For example, if test is the email alias, then type <code>\${test}</code>.</p> </div> <ul style="list-style-type: none"> ■ JDBC ■ Local Log: You can select the severity of the messages to be logged (logging level) from the Log Level drop-down list. The available log levels are ERROR, INFO, and WARN. <div style="background-color: #f0f0f0; padding: 10px; margin: 10px 0;"> <p>Note:</p> <ul style="list-style-type: none"> ■ Set the Integration Server Administrator's logging level for API Gateway to match the logging levels specified for the run-time actions (go to Settings > Logging > Server Logger). For example, if a Log Invocation action is set to the logging level of Error, you must also set Integration Server Administrator's logging level for API Gateway to Error. If the action's logging level is set to a low level (Warning-level or Information level), but Integration Server Administrator's logging level for API Gateway is set to a higher level (Error-level), then only the higher-level messages are written to the log file. ■ Entries posted to the local log are identified by a product code of YAI and suffixed with the initial alphabet of the logging level selected. For example, for an error level, the entry appears as [YAI.0900.0002E]. </div> <ul style="list-style-type: none"> ■ SNMP
Alert Interval	<p>Specifies the time period in which to monitor performance before sending an alert if a condition is violated.</p> <p>The timer starts once the API is activated and resets after the configured time interval. If and API is deactivated the interval gets reset and on API activation its starts afresh.</p>
Unit	<p>Specifies the unit for the time interval in minutes, hours, days, or weeks for the alert interval.</p>
Alert Frequency	<p>Specifies how frequently to issue alerts for the counter-based metrics (Total Request Count, Success Count, Fault Count).</p> <p>Select one of the options:</p>

Parameter	Value
	<ul style="list-style-type: none"> ■ Only Once. Triggers an alert only the first time one of the specified conditions is violated. ■ Every Time. Triggers an alert every time one of the specified conditions is violated.
Alert Message	Specifies the text to be included in the alert.

Monitor Service Level Agreement

This policy monitors a set of run-time performance conditions for an API, and sends alerts to a specified destination when the performance conditions are violated. This policy enables you to monitor run-time performance for one or more specified applications. You can configure this policy to define a Service Level Agreement (SLA), which is a set of conditions that defines the level of performance that an application should expect from an API. You can use this policy to identify whether the API threshold rules are met or exceeded. For example, you might define an agreement with a particular application that sends an alert to the application if responses are not sent within a certain maximum response time. You can configure SLAs for each API or application combination.

Parameters like success count, fault count and total request count are immediate monitoring parameters and the evaluation happens immediately after the limit is breached. The rest of the parameters are Aggregated monitoring parameters whose evaluation happens once the configured interval is over. If there is a breach in any of the parameters, an event notification (Monitor event) is sent to the configured destination. In a single policy, multiple action configurations behave as AND condition. The OR condition can be achieved by configuring multiple policies.

The table lists the properties that you can specify for this policy:

Parameter	Value
Action Configuration.	Specifies the type of action to be configured.
Name	<p>Specifies the name of the metric to be monitored.</p> <p>You can select one of the available metrics:</p> <ul style="list-style-type: none"> ■ Availability. Indicates whether the native API is available to the clients as specified in the current interval. API Gateway calculates the availability of the native API based on the alert interval specified and it is calculated from the instant the API activation takes place. The availability of the API is calculated as = (time for which the native API is up / total interval of time) x 100. This value is measured in %. <p>For example, if you set Availability as less than 90, then whenever the availability of the native API falls below 90%, in the specified time interval, API Gateway generates an alert. Suppose, the alert</p>

Parameter	Value
-----------	-------


interval is set as 1 minute (60 seconds) and if there are 7 API invocations at various times in that 1 minute with a combination of up and down as shown in the table, the availability is calculated as follows:

Request #	Invocation time (the second at which the API is invoked)	Service status	Up time
1	5	Up	5 (from start to now)
2	15	Up	10 (between 1 and 2)
3	30	Down	15 (between 2 and 3)
4	40	Down	0 (since last state is Down)
5	45	Up	0
6	50	Down	5 (between 5 and 6)
7	55	Up	0
			5 (remaining 5 seconds considered as Up inline with last state)
	Total		40 (Availability is 67%)



As the availability of the native API calculated is 66.67% and falls below 90%, API Gateway generates an alert. The API is considered to be down for the ongoing request when API Gateway receives a connection related error from the native API in the outbound call. If the API does not respond with an HTTP response, then it is considered as down.

- **Average Response Time.** Indicates the average time taken by the service to complete all invocations in the current interval. The average is calculated from the instant the API activation takes place for the configured interval.

For example, if you set an alert for Average response time greater than 30 ms with an interval of 1 minute then on API activation, the monitoring interval starts and the average of the response time of all runtime invocations for this API in 1 minute is calculated. If this is greater than 30 ms, then a monitor event is generated. If this is configured under Monitor service level agreement policy with an

Parameter	Value
	<p>option to configure applications so that application specific SLA monitoring can be done, then the monitoring for the average response time is done only for the specified application.</p> <ul style="list-style-type: none"> ■ Fault Count. Indicates the number of faults returned in the current interval. The HTTP status codes greater than or equal to 400, returned from API Gateway are considered as fault request transactions. This includes the downtime errors as well. ■ Maximum Response Time. Indicates the maximum time in milliseconds to respond to a request in the current interval. ■ Minimum Response Time. Indicates the minimum time in milliseconds to respond to a request in the current interval. ■ Success Count. Indicates the number of successful requests in the current interval. ■ Total Request Count. Indicates the total number of requests (successful and unsuccessful) in the current interval.
Operator	<p>Specifies the operator applicable to the metric selected.</p> <p>Select one of the available operator: Greater Than, Less Than, Equals To.</p>
Value	<p>Specifies the alert value for which the monitoring is applied.</p>
Destination	<p>Specifies the destination where the alert is to be logged.</p> <p>Select the required options:</p> <ul style="list-style-type: none"> ■ API Gateway ■ API Portal ■ CentraSite <div style="background-color: #f0f0f0; padding: 5px; margin: 5px 0;"> <p>Note: This option is applicable only for the APIs published from CentraSite to API Gateway.</p> </div> <ul style="list-style-type: none"> ■ Digital Events ■ Elasticsearch ■ Email (you can add multiple email addresses by clicking ). <div style="background-color: #f0f0f0; padding: 5px; margin: 5px 0;"> <p>Note:</p> </div>

Parameter	Value
	<p>If an email alias is available, you can type the email alias in the Email Address field with the following syntax, <code>\${emailaliasname}</code>. For example, if test is the email alias, then type <code>\${test}</code>.</p> <ul style="list-style-type: none"> ■ JDBC ■ Local Log: You can select the severity of the messages to be logged (logging level) from the Log Level drop-down list. The available log levels are ERROR, INFO, and WARN. <p>Note:</p> <ul style="list-style-type: none"> ■ Set the Integration Server Administrator's logging level for API Gateway to match the logging levels specified for the run-time actions (go to Settings > Logging > Server Logger). For example, if a Log Invocation action is set to the logging level of Error, you must also set Integration Server Administrator's logging level for API Gateway to Error. If the action's logging level is set to a low level (Warning-level or Information level), but Integration Server Administrator's logging level for API Gateway is set to a higher level (Error-level), then only the higher-level messages are written to the log file. ■ Entries posted to the local log are identified by a product code of YAI and suffixed with the initial alphabet of the logging level selected. For example, for an error level, the entry appears as <code>[YAI.0900.0002E]</code>. <ul style="list-style-type: none"> ■ SNMP
Alert Interval	<p>Specifies the time period (in minutes) in which to monitor performance before sending an alert if a condition is violated.</p> <p>The timer starts once the API is activated and resets after the configured time interval. If and API is deactivated the interval gets reset and on API activation its starts afresh.</p>
Unit	<p>Specifies the unit for the time interval in minutes, hours, days, or weeks for the alert interval.</p>
Alert Frequency	<p>Specifies how frequently to issue alerts for the counter-based metrics (Total Request Count, Success Count, Fault Count).</p> <p>Select one of the options:</p> <ul style="list-style-type: none"> ■ Only Once. Triggers an alert only the first time one of the specified conditions is violated. ■ Every Time. Triggers an alert every time one of the specified conditions is violated.

Parameter	Value
Alert Message	Specifies the text to be included in the alert.
Consumer Applications	Specifies the application to which this Service Level Agreement applies. You can type a search term to match an application and click  to add it. You can add multiple applications or delete an added application by clicking  .


Throttling Traffic Optimization



This policy limits the number of service invocations during a specified time interval, and sends alerts to a specified destination when the performance conditions are violated. You can use this policy to avoid overloading the back-end services and their infrastructure, to limit specific clients in terms of resource usage, and so on.

The Throttling Traffic Optimization policy generates two types of events when the specified limit is breached, policy violation event and monitor event. The policy violation event is for indicating the violations that occur for an API. If there are 100 violations, then 100 policy violation events are generated. The monitor event triggered by this policy is controlled by the alert frequency configuration specified in the policy.

The table lists the properties that you can specify for this policy:

Parameter	Description
Limit Configuration.	
Rule name	Specifies the name of throttling rule to be applied. For example, Total Request Count.
Operator	Specifies the operator that connects the rule to the value specified. Select the operator: Greater Than . For example, in this case the throttling rule is applied when the Total Request Count is greater than (exceeds the limit specified for) the value specified in the Value field.
Value	Specifies the value of the request count beyond which the policy is violated. When multiple requests are made at the same time, it might be possible that this limit applied to trigger an alert is not strictly adhered to. There is no loss observed in the invocation counts data, but there might be a minor delay in aggregating the count. The invocation count gets incremented, only when API Gateway receives the response from the native API. For example, if you have set the limit at 5 with an

Parameter	Description
	<p>interval alert of 1 minute and if you invoke 5 requests in parallel, out of which for 1 of the request the native API delays sending the response to API Gateway. In such cases, the invocation count would still be 4 as the native API is yet to send the response to API Gateway. There is a minor delay in aggregating the count due to native API response delay. Hence, API Gateway allows additional invocation. However, when the invocation count exceeds 5 an alert is triggered.</p>
Destination	<p>Specifies the destination to log the alerts.</p> <p>Select the required options:</p> <ul style="list-style-type: none"> ■ API Gateway ■ API Portal ■ CentraSite <div data-bbox="646 814 1458 951" style="background-color: #f0f0f0; padding: 5px;"> <p>Note: This option is applicable only for the APIs published from CentraSite to API Gateway.</p> </div> <ul style="list-style-type: none"> ■ Digital Events ■ Elasticsearch ■ Email (you can add multiple email addresses by clicking ). <div data-bbox="646 1150 1458 1360" style="background-color: #f0f0f0; padding: 5px;"> <p>Note: If an email alias is available, you can type the email alias in the Email Address field with the following syntax, <code>\${emailaliasname}</code>. For example, if test is the email alias, then type <code>\${test}</code>.</p> </div> <ul style="list-style-type: none"> ■ JDBC ■ Local Log: You can select the severity of the messages to be logged (logging level) from the Log Level drop-down list. The available log levels are ERROR, INFO, and WARN. <div data-bbox="646 1549 1458 1894" style="background-color: #f0f0f0; padding: 5px;"> <p>Note:</p> <ul style="list-style-type: none"> ■ Set the Integration Server Administrator's logging level for API Gateway to match the logging levels specified for the run-time actions (go to Settings > Logging > Server Logger). For example, if a Log Invocation action is set to the logging level of Error, you must also set Integration Server Administrator's logging level for API Gateway to Error. If the action's logging level is set to a low level (Warning-level or Information level), but Integration Server </div>

Parameter	Description
	<p>Administrator's logging level for API Gateway is set to a higher level (Error-level), then only the higher-level messages are written to the log file.</p> <ul style="list-style-type: none"> ■ Entries posted to the local log are identified by a product code of YAI and suffixed with the initial alphabet of the logging level selected. For example, for an error level, the entry appears as [YAI.0900.0002E]. <ul style="list-style-type: none"> ■ SNMP
Alert Interval	Specifies the interval of time for the limit to be reached.
Unit	Specifies the unit for the time interval in minutes, hours, days, or weeks for the alert interval.
Alert Frequency	<p>Specifies the frequency at which the alerts are issued and the monitor events are logged.</p> <p>Specify one of the options:</p> <ul style="list-style-type: none"> ■ Only Once. Triggers an alert every time the specified condition is violated and logs a monitor event for the alert interval specified. ■ Every Time. Triggers an alert every time the specified condition is violated and logs multiple monitor events based on the number of API invocations.
Alert Message	Specifies the text message to be included in the alert.
Consumer Applications	<p>Specifies the application to which this policy applies.</p> <p>You can type a search term to match an application and click  to add it.</p> <p>You can add multiple applications or delete an added application by clicking .</p>

Service Result Cache

This policy enables caching of the results of API invocations depending on the caching criteria defined. You can define the elements for which the API responses are to be cached based on the criteria such as HTTP Header, XPath, Query parameters, and so on. You can also limit the values to store in the cache using a whitelist. For the elements that are stored in the cache, you can specify other parameters such as Time to Live and Maximum Response Payload Size.



Caching the results of an API request increases the throughput of the API call and improves the scalability of the API.



The cache criteria applicable for a SOAP-based API request are HTTP Header and XPATH. The cache criteria applicable for a REST-based API request are HTTP Header and Query parameters. The caching works only for GET methods for REST APIs.

Note:

If there are no values set for any of the criteria, then, by default, all the SOAP requests and GET requests for the Rest API are based on the URL.

The table lists the properties that you can specify for this policy:

Parameter	Description
Cache Criteria.	Specifies the criteria that API Gateway uses to determine the request component, that is, the actual payload based on which the results of the API invocation are cached.
HTTP Header	<p>Specifies the header name and values to look for in the HTTP header of incoming API requests to filter the requests and cache the results based on the header names and values specified. You can use this criterion for APIs that accept payloads only in HTTP format.</p> <p>Header Name. Specifies the HTTP header name.</p> <p>Cache responses only for these values. API Gateway caches the API responses only for requests whose cache criteria match with those set for the action, and whose criteria evaluation results in any one of the values in this list. You can add multiple entries by clicking .</p> <p>Note: If you do not specify any values, all the API requests containing the specified header name are cached.</p>
Query Parameters	<p>You can use this criterion for REST-based API requests. Specifies the names and values of the query parameters to filter the incoming requests and cache the results based on the names and values specified.</p> <p>Parameter Name. Specifies the parameter name.</p> <p>Cache responses only for these values. API Gateway caches the API responses only for requests whose cache criteria match those set for the action, and whose criteria evaluation results in any one of the values in this list. You can add multiple entries by clicking .</p>
XPath	<p>You can use this criterion for SOAP-based API requests whose payload is a SOAP envelope. Specifies the XPath expression details and values to look in the HTTP header of an incoming API request to filter the requests and cache the results based on the criteria specified.</p> <ul style="list-style-type: none"> ■ Name Space. Specifies the namespace of the XPath expression.

Parameter	Description
	<ul style="list-style-type: none"> ■ Prefix. Specifies the prefix for the namespace. ■ URI. Specifies the namespace URI. <p>You can add multiple entries by clicking .</p> <p>XPath Expression. Specifies the XPath expression in the API request.</p> <p>Cache responses only for these values. API Gateway caches the API responses only for requests whose cache criteria match those set for the action, and whose criteria evaluation results in any one of the values in this list. You can add multiple entries by clicking .</p> <p>Note: If you do not specify any values, all the API requests containing the specified XPath expression are cached.</p>
Time to Live (e.g., 5d 4h 1m)	<p>Specifies the lifespan of the elements in the cache after which the elements are considered to be out-of-date.</p> <p>The time is specified in terms of days, hours, and minutes; for example, 5d 4h 1m.</p> <p>If you do not specify any value, the Time to Live is considered to be unlimited (does not expire). If you set the value to 0d 0h 0m, the API results are not cached.</p> <p>The default time format is minutes if the input is a number.</p>
Maximum Response Payload Size (in KB)	<p>Specifies the maximum payload size for the API in kilo bytes.</p> <p>The value -1 stands for unlimited payload size.</p>

Example of enforcing caching criteria:

Cache criteria	HTTP Header	Query parameters	XPATH	Values
C1	Header1			h1, h2
C2	Header2			
C3		query1		q1, q2

In the example, there are two HTTP headers and one query parameter as cache criteria. The HTTP Header **Header2** has no values specified. Hence, all the incoming requests with the HTTP Header **Header2** are cached.

When there are multiple cache criteria, the following behaviour is observed in the cache result:

- If the incoming request R1 matches criteria C1, then the result is cached. API Gateway responds to any further incoming request R1 that matches criteria C1 from the cache.
- If the incoming request R1 matches criteria C1 and C2, then the result is cached as a new request.
- If you configure multiple cache criteria, and if one or more cache criteria match, then the result is cached. The criteria are matched with the cached results while caching the request, and it follows the AND condition among the matched criteria.

Response Processing

These policies are used to specify how the response message from the API has to be transformed or pre-processed and configure the masking criteria for the data to be masked before it is submitted to the application. This is required to protect the data and accommodate differences between the message content that an API is capable of submitting and the message content that an application expects. The policies included in this stage are:

- Invoke webMethods IS
- Response Transformation
- Validate API Specification
- CORS
- Data Masking

Invoke webMethods IS

This policy processes the native API's response messages into the format required by the application, before API Gateway returns the responses to the application.

If `Comply to IS Spec` parameter is configured as `true`, API Gateway invokes the IS Service with IS specification in the path `pub.apigateway.invokeISService.specifications:ResponseSpec` (for Response Processing)

The following are the input and output parameters for REST, SOAP, and WebSocket APIs as specified in the above IS Specification. Input parameters can be used to access the existing values of the response while output parameters can be used to modify/write the values to the response.

	Parameter name	Description
Input parameters	headers	Headers in response. Data type: Document
	payload	Payload of the response. Data type: String

	Parameter name	Description
	payloadObject	The payload for binary content types like multi-part / form-data. Data type: Object
	statusCode	Status code of the response. Data type: String
	statusMessage	Status message of the response. Data type: String
	MessageContext	The message context object of the response. Data type: Object
	apiName	Name of the API invoked by the response. Data type: String
	requestUrl	URL of the response. Data type: String
	ipInfo	Contains IP information of the response. Data type: Document
	websocketInfo	Websocket related information of the response. Data type: Document
	correlationID	Correlation ID of the request/response. This is unique and same for a request and response. Data type: String
	customFieldsMap	Custom transactional fields can be added to the transactional events using this field. For more information, see Adding Custom Fields to Transactional Events section. Data type: Document
Output parameters	headers	Headers in response. Data type: Document
	payload	Payload of the response. Data type: String

Parameter name	Description
payloadObject	The payload for binary content types like multi-part / form-data. Data type: Object
statusCode	Status code of the response. Data type: String
statusMessage	Status message of the response. Data type: String
MessageContext	The message context object of the response. Data type: Object
customFieldsMap	Custom transactional fields can be added to the transactional events using this field. For more information, see Adding Custom Fields to Transactional Events section. Data type: Document

Note:

- For SOAP to REST APIS, the payload contains the transformed JSON response.
- Payload transformation does not happen automatically for content-type transformation. When you change the content type, ensure to do payload transformation also as part of IS Service.
- When `Comply to IS spec` is true, you can change the values of headers, query, payload, and so on, programatically using Message Context, as well as using the pipeline variables given. Software AG recommends you do not change those values directly in Message Context, as the values in output pipeline variables are written to Message Context after the invocation of IS Service.

If `Comply to IS Spec` parameter is set to `false`, API Gateway invokes the IS Service with the same input and output parameters supported in 10.1 and the earlier versions::

- `proxy.name`
- `JSONRESTContentString` (REST only)
- `SOAPEnvelope` (SOAP only)
- `EnvelopeString` (SOAP only)

The table lists the properties that you can specify for this policy:

Parameter	Description
-----------	-------------

Invoke webMethods Integration Server Service

Add invoke webMethods Integration Server service Specifies the webMethods IS service to be invoked to process the response messages and the authentication mode for the IS service.

Provide the following information:

- **webMethods IS Service.** Specify the webMethods IS service to be invoked to pre-process the response messages.

The webMethods IS service must be running on the same Integration Server as API Gateway

Note:

If an exception occurs when invoking the webMethods IS service, by default API Gateway displays the status code as 500 and error message as *Internal Server Error*.

You can set custom status code and error message by setting the following properties in the message context of the webMethods IS service:

- service.exception.status.code
- service.exception.status.message



The sample code is given below:

```
IDataCursor idc = pipeline.getCursor();
MessageContext context =
(MessageContext)IDataUtil.get(idc,"MessageContext");
if(context != null)
{
context.setProperty("service.exception.status.code",
404);
context.setProperty("service.exception.status.message",
"Object Not Found");
throw new ServiceException();
}
```

Note:

If ServiceException or FlowException occurs when invoking webMethods IS Service, the message given in the exception is displayed to the client. If any other exception occurs, a generic error message is displayed to the client.

- **Run as User.** Specifies the authentication mode to invoke the IS service. If this field is left blank the incoming credentials of the user, identified by API Gateway, are used to authenticate and invoke the IS service. You can also specify a particular user, you want API Gateway to invoke the IS service.

Parameter	Description
	<p>Note: It is the responsibility of the user who activates the API to review the value configured in Run as User field to avoid misuse of this configuration.</p> <ul style="list-style-type: none"> ■ Comply to IS Spec. Mark this as true if you want the input and the output parameters to comply to the IS Spec present in <code>pub.apigateway.invokeISService.specifications</code> folder in <code>wmAPIGateway</code> package. <p>Note:Software AG recommends users to configure the policy with <code>Comply to IS Spec</code> as true, as you can read or change the values of headers, and so on, without having to read from or write to the message context.</p>
webMethods IS Service alias	<p>Specifies the webMethods IS service alias used to invoke the webMethods IS service to pre-process the response messages.</p> <p>Start typing the webMethods alias name, select the alias from the type-ahead search results displayed and click  to add one or more aliases.</p> <p>You can use the delete icon  to delete the added aliases from the list.</p>

Adding Custom Fields to Transactional Events

This section explains you how to add custom fields to the transactional events.

1. Create webMethods IS service by specifying the `pub.apigateway.utils:customFieldInTransactionEventSpec` as a specification reference.
2. In the webMethods IS service, set the required custom fields in the `customFieldsMap` output variable.
3. Once when `customFieldsMap` gets created, the custom fields will be available in the transactional events.
4. Invoke the API with the Invoke webMethods IS policy.

Response Transformation

This policy specifies the properties required to transform response messages from native APIs into a format required by the client.

This policy enables you to configure several transformations on the response messages before it is sent to the client.

API Gateway supports the following parameter types that can be used to configure the transformation policy:

- response.payload
- response.headers
- response.statusCode
- response.statusMessage


API Gateway supports the following Query types that can be applied on payload to extract values from payload:

- xpath
- jsonPath
- regex

When you use these syntaxes to extract a value from the payload, the content-types applicable are:

- `${response.payload.jsonPath[Expression]}` - application/json, application/json/badgerfish
- `${response.payload.regex[Expression]}` - text/plain
- `${response.payload.xpath[Expression]}` - application/xml, text/xml, text/html.

The table lists the properties that you can specify for this policy:

Parameter	Description
Condition	<p>Conditions are used to specify when the policy has to be executed. You can add multiple conditions with logical operators.</p> <p>Available values are:</p> <ul style="list-style-type: none"> ■ AND. API Gateway transforms the responses that comply with all the configured conditions ■ OR. This is selected by default. API Gateway transforms the responses that comply at least one configured condition. <p>Click Add Condition and provide the following information and click .</p> <ul style="list-style-type: none"> ■ Variable. Specifies the variable type with a syntax as follows: <ul style="list-style-type: none"> ■ <code>\${PARAMTYPE}</code> : This is applicable for variables of string type - payload, statusCode and statusMessage. For example: <code>\${response.payload}</code>

Parameter	Description
	<ul style="list-style-type: none"> ■ <code>\${PARAMTYPE.paramName}</code> : This is applicable for map types - headers. For example: <code>, \${response.header.Content-Type}</code>, ■ <code>\${PARAMTYPE.QUERYTYPE[queryValue]}</code> : This syntax is applicable for payload. XPath, JSONPath and regex can be applied on payload. For example: <ul style="list-style-type: none"> <code>\${response.payload.xpath[//ns:emp/ns:empName]}</code> where <code>//ns:emp/ns:empName</code> is the xpath to be applied on the payload if <code>contentType</code> is <code>application/xml</code> <code>\${response.payload.jsonPath[\$.cardDetails.number]}</code> where <code>\$.cardDetails.number</code> is the jsonPath to be applied on the payload if <code>contentType</code> is <code>application/json</code> <code>\${response.payload.regex[[0-9]+]}</code> where <code>[0-9]+</code> is the regex to be applied on the payload if <code>contentType</code> is <code>text/plain</code> ■ If you want API Gateway to apply xpath, jsonPath, regex based on Content-Type of the payload, use the following common syntax: <code>\${PARAMTYPE.QUERYTYPE[queryValue] PARAMTYPE.QUERYTYPE2[queryValue2] ...}</code> <p>For example:</p> <ul style="list-style-type: none"> <code>\${response.payload.xpath[//ns:emp/ns:empName] response.payload.jsonPath[\$.cardDetails.number]}</code> This applies xpath for <code>application/xml</code> and jsonPath for <code>application/json</code> <code>\${response.payload.xpath[//ns:emp/ns:empName] response.payload.jsonPath[\$.cardDetails.number] response.payload.regex[[0-9]+]}</code> This applies xpath for <code>application/xml</code>, jsonPath for <code>application/json</code>, and regex for <code>text/plain</code>. <ul style="list-style-type: none"> ■ Operator. Specifies the operator to use to relate variable and the value provided. You can select one of the following: <ul style="list-style-type: none"> ■ Equals ■ Equals ignore case ■ Not equals ■ Contains ■ Exists ■ Value. Specifies a value with a syntax as follows: <ul style="list-style-type: none"> ■ <code>PLAIN VALUE</code>, for example, <code>application/json</code>

Parameter	Description
	<ul style="list-style-type: none"> ■ <code>\${PARAMTYPE.paramName}</code> ■ <code>\${PARAMTYPE.QUERYTYPE[queryValue]}</code> ■ <code>\${PARAMTYPE.QUERYTYPE[queryValue] PARAMTYPE.QUERYTYPE2[queryValue2] ...}</code>

Transformation Configuration. Specifies various transformations to be configured.


Header Transformation Specifies the Header, Query or path transformation to be configured for the responses received from the native API.

You can add or modify header, query or path transformation parameters by providing the following information:

- **Variable.** Specifies the variable type with a syntax as follows:
 - `${PARAMTYPE}` : This is applicable for variables of string type - payload, statusCode and statusMessage. For example: `${response.payload}`
 - `${PARAMTYPE.paramName}` : This is applicable for map types - headers. For example: `response.header.Content-Type`,
 - `${PARAMTYPE.QUERYTYPE[queryValue]}` : This syntax is applicable for payload. XPath, JSONPath and regex can be applied on payload. For example:
 - `${response.payload.xpath[//ns:emp/ns:empName]}` where `//ns:emp/ns:empName` is the xpath to be applied on the payload if contentType is application/xml
 - `${response.payload.jsonPath[$.cardDetails.number]}` where `$.cardDetails.number` is the jsonPath to be applied on the payload if contentType is application/json
 - `${response.payload.regex[[0-9]+]}` where `[0-9]+` is the regex to be applied on the payload if contentType is text/plain
- If you want API Gateway to apply xpath, jsonPath, regex based on Content-Type of the payload, use the following common syntax: `${PARAMTYPE.QUERYTYPE[queryValue] || PARAMTYPE.QUERYTYPE2[queryValue2] || ...}`

For example:

```
${response.payload.xpath[//ns:emp/ns:empName] ||
response.payload.jsonPath[$.cardDetails.number]} This applies
xpath for application/xml and jsonPath for application/json
```

Parameter	Description
	<p><code>\${response.payload.xpath[//ns:emp/ns:empName] response.payload.jsonPath[\$.cardDetails.number] response.payload.regex[[0-9]+]}</code> This applies xpath for application/xml, jsonPath for application/json, and regex for text/plain.</p> <ul style="list-style-type: none"> ■ Value. Specifies a value with a syntax as follows: <ul style="list-style-type: none"> ■ PLAIN VALUE, for example, application/json ■ <code>\${PARAMTYPE.paramName}</code> ■ <code>\${PARAMTYPE.QUERYTYPE[queryValue]}</code> ■ <code>\${PARAMTYPE.QUERYTYPE[queryValue] PARAMTYPE.QUERYTYPE2[queryValue2] ...}</code> <p>You can add multiple variables and corresponding values by clicking .</p> <p>You can remove any header, query or path transformation parameters by typing the value in the text box.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: Software AG recommends you not to modify the headers <code>\${response.headers.Content-Length}</code> and <code>\${response.headers.Content-Encoding}</code> as API Gateway adds the right values for these headers before sending the response back to client.</p> <p>Note: Payload transformation does not happen automatically for content-type transformation. When you change the content type, ensure to do payload transformation also as part of IS Service. For example, if you change the content-type header from application/xml to application/json using IS service, you must also change the respective payload from application/xml to application/json</p> </div>

Status transformation Specifies the status transformation to be configured for the responses received from the native API.

Provide the following information:

- **Code.** Specifies the status code that is sent in the response to the client.
- **Message.** Specifies the Status message that is sent in the response to the client.

Parameter	Description
-----------	-------------

Payload Transformation

Specifies the payload transformation to be configured for the responses received from the native API.

Provide the following information:

- Click **+ Add xslt document** to add an xslt document and provide the following information:
 - **XSLT file.** Specifies the XSLT file used to transform the response messages as required.

Click **Browse** to browse and select a file.

- **Feature Name.** Specifies the name of the XSLT feature.
- **Feature value.** Specifies the value of the XSLT feature.

You can add more XSLT features and xslt documents by clicking

 **Add**

Note:

API Gateway supports XSLT 1.0 and XSLT 2.0.


- Click **+ Add xslt transformation alias** and provide the following information:
 - **XSLT Transformation alias.** Specifies the XSLT transformation alias

When you receive the response in JSON, you can use a XSLT file similar to the below sample:

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.1"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml"/>
  <xsl:template match="/" >
    <xsl:element name="fakeroot">
      <xsl:element name="fakenode">
        <!-- Apply your transformation rules based on the
response received from the native API-->
      </xsl:element>
    </xsl:element>
  </xsl:template>
</xsl:stylesheet>
```

When you receive the response in XML, you can use a XSLT file similar to the below sample:

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.1"
```

Parameter	Description
	<pre> xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"> <xsl:output method="xml"/> <xsl:template match="/" > <xsl:element name="soapenv:Envelope"> <xsl:element name="soapenv:Body"> <!-- Apply your transformation rules based on the response received from the native API--> </xsl:element> </xsl:element> </xsl:template> </xsl:stylesheet> </pre>
Advanced Transformation	<p>Specifies the advanced transformation to be configured for the responses received from the native API..</p> <p>Provide the following information:</p> <ul style="list-style-type: none"> ■ webMethods IS Service. Specify the webMethods IS service to be invoked to process the response messages. <p>You can add multiple services by clicking .</p> <div data-bbox="602 968 1458 1100" style="background-color: #f0f0f0; padding: 5px;"> <p>Note: The webMethods IS service must be running on the same Integration Server as API Gateway.</p> </div> <ul style="list-style-type: none"> ■ Run as User. Specifies the authentication mode to invoke the IS service. If this field is left blank the incoming credentials of the user, identified by API Gateway, are used to authenticate and invoke the IS service. You can also specify a particular user, you want API Gateway to use to run the IS service. ■ Comply to IS Spec. Mark this as true if you want the input and the output parameters to comply to the IS Spec present in <code>pub.apigateway.invokeISService.specifications</code> folder in <code>wmAPIGateway</code> package. ■ webMethods IS Service alias. Specifies the webMethods IS service alias to be invoked to pre-process the request messages.
	<p>Transformation Metadata. Specifies the metadata for transformation of the responses received from the native API. For example, the namespaces configured in this section can be used when you provide the syntax for XPath <code>#{response.payload.xpath}</code> For example: <code>#{response.payload.xpath[//ns:emp/ns:empName]}</code></p>
Namespace	<p>Specifies the namespace information to be configured for transformation.</p> <p>Provide the following information:</p>

Parameter	Description
	<ul style="list-style-type: none"> ■ Namespace Prefix. The namespace prefix of the payload expression to be validated. ■ Namespace URI. The namespace URI of the payload expression to be validated. <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p>Note: You can add multiple namespace prefix and URI by clicking</p> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block; margin-top: 5px;">+ Add</div> </div>

Validate API Specification

This policy validates the responses against API's various specifications such as schema, content-types, and HTTP Headers referenced in their corresponding formats as follows:

- The schema is available as part of the API definition. The schema for SOAP API are imported through WSDL and for REST APIs it can be through swagger, RAML or can be uploaded by the user when an API is created from scratch.
- The content- types are available as part of the API definition. FOR SOAP APIs these are imported through WSDL and for REST APIs it can be through swagger, RAML or can be uploaded by the user.
- The HTTP Headers are specified in the Validate API Specification policy page

The response sent to the API by an application must conform with the structure or format expected by the API. The responses from the native API are validated against the API specifications in this policy to conform to the structure or format expected by the API.

Various API specifications validated are:

- **Schema:**

The responses from the native API are validated against the schema provided in the API definition. The schema defines the elements and attributes and specifies the data types of these elements to ensure that only appropriate data is allowed through to the API.

For a REST API, the schema can be added inline or uploaded in the Components section on the API Details page. For details on how to add the schema inline or upload, see [“Creating a REST API” on page 273](#).

The schema type for validation is selected based on:

- The Content-Type header when the policy is added in the Request processing stage.
- The Accept header when the policy is added in the Response processing stage.

If the header or payload is missing the schema validation is skipped.

The table lists the default Content type/Accept header and schema validation type mapping.

Content-type/Accept	Schema validation type
application/json	JSON schema
application/json/badgerfish	
application/xml	XML schema
text/xml	
text/html	
text/plain	Regular expression

For a SOAP API, the WSDL and the referenced schema must be provided in a zip format. The JSON schema validation is supported for the operations that are exposed as REST.

■ Content-types:

The responses from the native API are validated against the content-types specified in the API definition.

■ HTTP Headers:

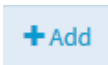
The responses from the native API are validated against the HTTP Headers specified in this policy to conform to the HTTP headers expected by the API.

The run-time invocations that fail the specification validation are considered as policy violations. Such policy violation events that are generated can be viewed in the dashboard.

The table lists the API specification properties, you can specify for this policy, to be validated:

Parameter	Description
Schema	<p>Validates the response payload against the appropriate schema.</p> <p>Provide the following additional features for XML schema validation:</p> <ul style="list-style-type: none"> ■ Feature name. Specifies the name of the feature for XML parsing when performing XML schema validation. <p>Select the required feature name from the list:</p> <ul style="list-style-type: none"> ■ GENERATE_SYNTHETIC_ANNOTATIONS ■ ID_IDREF_CHECKING ■ IDENTITY_CONSTRAINT_CHECKING ■ IGNORE_XSL_TYPE ■ NAMESPACE_GROWTH

Parameter	Description
	<ul style="list-style-type: none"> ■ NORMALIZE_DATA ■ ROOT_ELEMENT_DECL ■ ROOT_TYPE_DEF ■ SIGMA_AUGMENT_PSVI ■ SCHEMA_DV_FACTORY ■ SCHEMA_ELEMENT_DEFAULT ■ SCHEMA_LOCATION ■ SCHEMA_NONS_LOCATION ■ SCHEMA_VALIDATOR ■ TOLERATE_DUPLICATES ■ ENPARSED_ENTITY_CHECKING ■ VALIDATE_ANNOTATIONS ■ XML_SCHEMA_FULL_CHECKING ■ XMLSCHEMA_VALIDATION <p>For details about XML parsing features, see http://xerces.apache.org/xerces2-j/features.html and for details about the exact constants, see https://xerces.apache.org/xerces2-j/javadocs/xerces2/org/apache/xerces/parsers/XML11Configuration.html.</p> <ul style="list-style-type: none"> ■ Feature value. Specifies whether the feature value is True or False.
Content-types	Validates the content-types in the incoming response against the content-types defined in that response's API Specification.
HTTP Headers	<p>Validates the HTTP header parameters in the incoming response against the HTTP headers defined in that response's API Specification.</p> <p>Provide the following information:</p> <ul style="list-style-type: none"> ■ Condition: Specifies the logical operator to use to validate multiple HTTP headers in the incoming API responses. <p>Available values are:</p> <ul style="list-style-type: none"> ■ AND. API Gateway accepts only the responses that contain all configured HTTP headers. ■ OR. This is selected by default. API Gateway accepts responses that contain at least one configured HTTP header.

Parameter	Description
	<ul style="list-style-type: none"> ■ HTTP Header Key. Specifies a key that must be passed through the HTTP header of the incoming API responses. ■ Header Value. Optional. Specifies the corresponding key value that could be passed through the HTTP header of the incoming API responses. <p>The Header Value field type accepts string and regular expression (regex).</p> <p>You can add more HTTP headers by clicking .</p>

CORS

The Cross-Origin Resource Sharing (CORS) mechanism supports secure cross-domain requests and data transfers between browsers and web servers. The CORS standard works by adding new HTTP headers that allow servers to describe the set of origins that are permitted to read that information.

This policy provides CORS support that uses additional HTTP headers to let a client or an application gain permission to access selected resources. An application or a client makes a cross-origin HTTP request when it requests a resource from a different domain, protocol, or port than the one from which the current request originated.


If you want to apply this policy in API Gateway at API level, make sure you have set the `watt.server.cors.enabled` property to *false*.



Note:

Both the Integration Server CORS policy and API Gateway CORS policy cannot coexist. When you enforce the CORS policy at Integration Server level, CORS enforcement is done for all requests. The preflight requests are handled by the Integration Server before even it reaches .

This policy is applicable for REST, SOAP, and ODATA APIs.

The table lists the CORS response specifications, you can specify for this policy:

Property	Description
Allowed Origins	<p>Specifies the origin from which the responses originating are allowed.</p> <p>syntax for the origin: <code>scheme://host:port</code></p> <p>You can add multiple origins by clicking .</p> <p>You can also provide Regular expressions for allowed origins.</p>

Property	Description
	Allowed origins can also be specified in the Advanced section under Applications. Allowed origins of applications registered with this API are also allowed to access this API.
Allow Headers	Specifies the Headers that are allowed in the request. You can add multiple headers that are to be allowed by clicking  .
Expose Headers	Specifies the headers that be exposed to the user on request failure. You can add multiple headers that are to be allowed by clicking  .
Allow Credentials	Specifies whether API Gateway includes the Access-Control-Allow-Credentials header.
Allowed Methods	Specifies the methods that are allowed in the request. Specify one or more of the following: GET, POST, PUT, DELETE, and PATCH.
Max Age	Specifies the age for which the preflight response is valid.

A corresponding HTTP header is set for all the values above as per the specification. For additional information, see <https://www.w3.org/TR/cors/>.

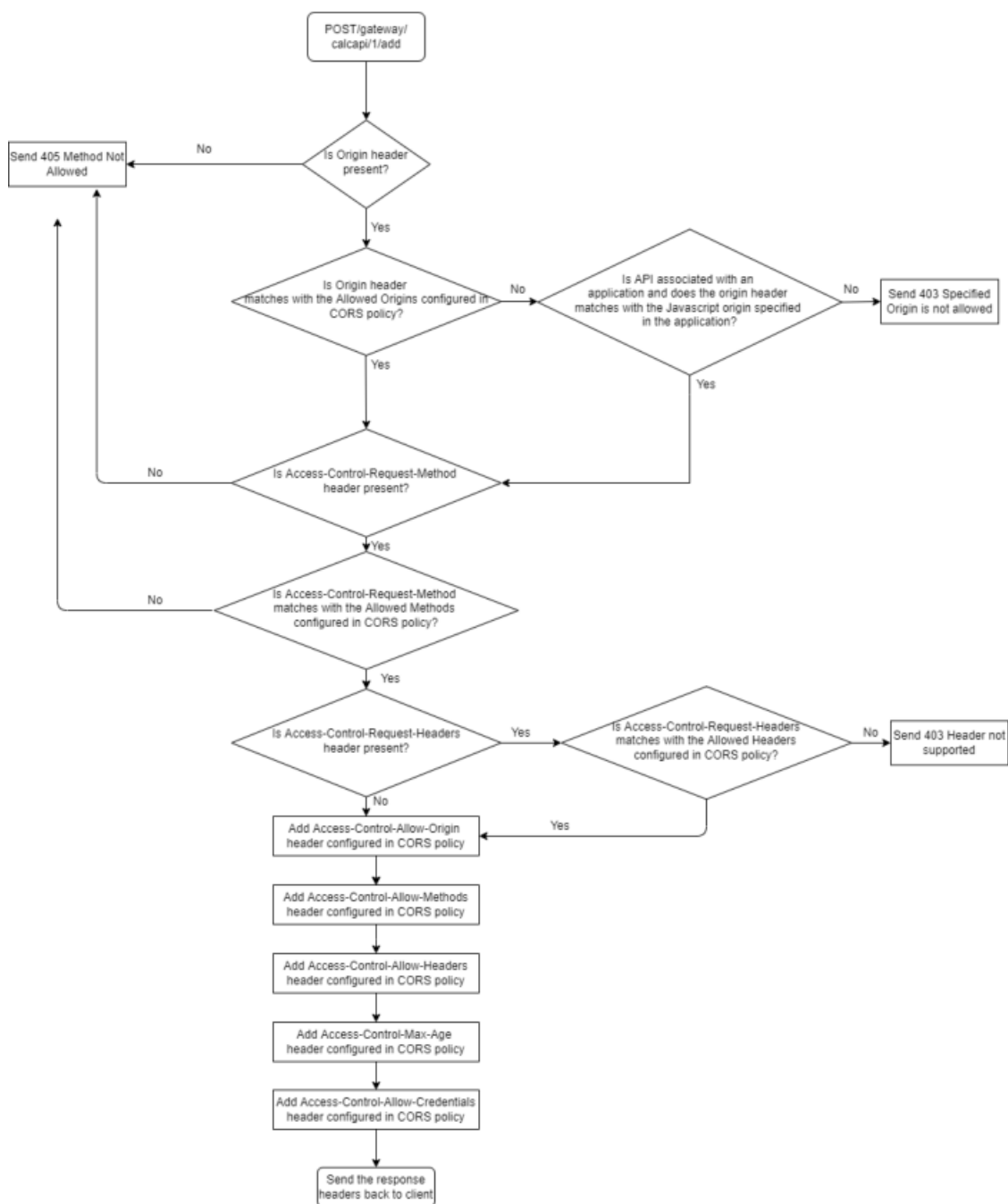
API Gateway handles CORS preflight request and CORS request differently. To know more about the work flow of CORS preflight and CORS request refer the respective flowchart.

CORS Preflight Request

A CORS preflight request is a HTTP request that a browser sends before the original CORS request to check whether the API Gateway server will permit the actual CORS request. CORS preflight request uses OPTIONS method and includes these headers as part of the request sent from the browser to API Gateway:

1. Origin
2. Access-Control-Request-Method
3. Access-Control-Request-Headers

The following flow chart explains the flow of the CORS preflight request received in API Gateway:



The following table shows the various use cases of the CORS preflight request originating from browser and how API Gateway responds to each CORS preflight requests:

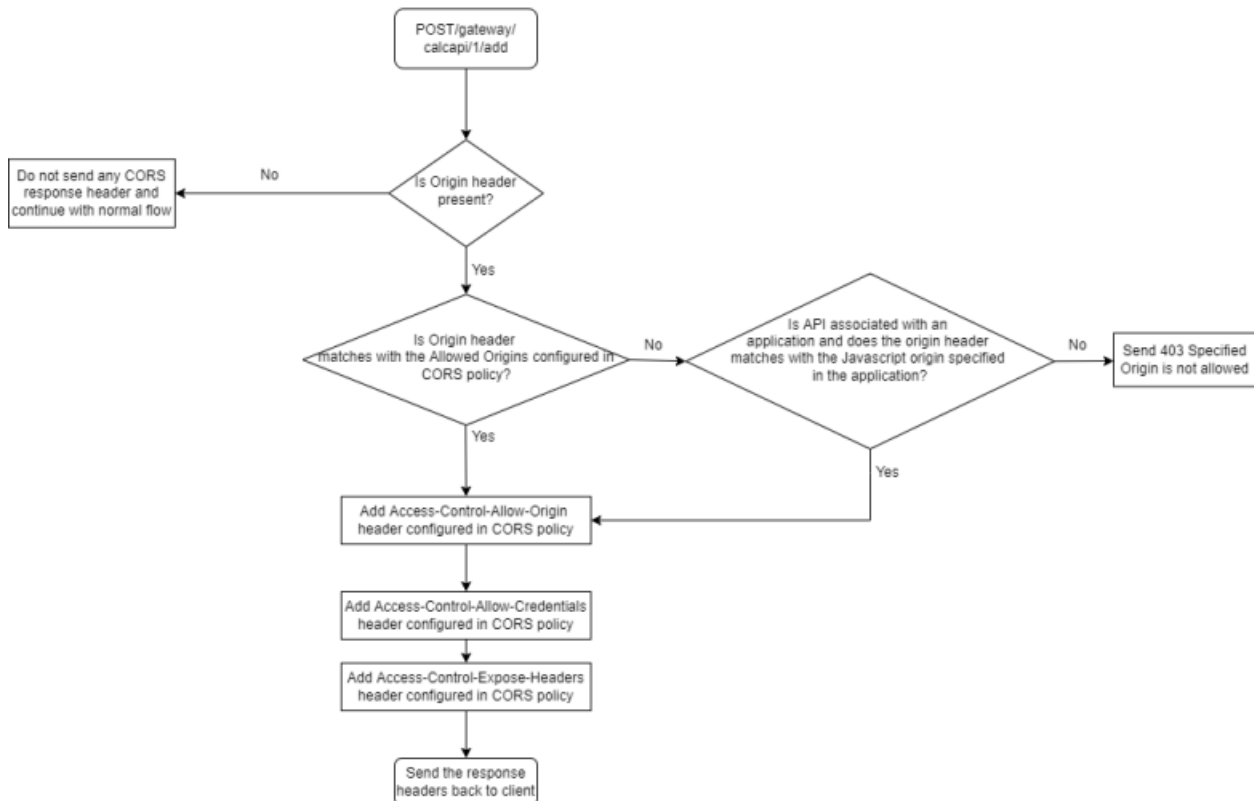
#	CORS Preflight request headers from browser	Configured CORS Policy in API Gateway	API Gateway sends the respective response to browser
1	Origin: http://test.com Access-Control-Request-Method : POST Access-Control-Request-Headers : test1,test2	Access-Control-Allow-Origin : http://test2.com Access-Control-Allow-Methods : POST,GET,PUT Access-Control-Allow-Headers : test1,test2	Sends 403 Specified Origin is not allowed status, as the Origin header (http://test.com) from the browser does not match with the Access-Control-Allow-Origin (http://test2.com) configured in the CORS policy.
2	Origin: http://test2.com Access-Control-Request-Method : DELETE Access-Control-Request-Headers : test1,test2	Access-Control-Allow-Origin : http://test2.com Access-Control-Allow-Methods : POST,GET,PUT Access-Control-Allow-Headers : test1,test2	Sends 405 Method Not Allowed status, as the Access-Control-Request-Method header (DELETE) from the browser does not match with the Access-Control-Allow-Methods (POST, GET, PUT) configured in the CORS policy.
3	Origin: http://test2.com Access-Control-Request-Method : POST Access-Control-Request-Headers : test3	Access-Control-Allow-Origin : http://test2.com Access-Control-Allow-Methods : POST,GET,PUT Access-Control-Allow-Headers : test1,test2	Sends 403 Header Not Supported , as the Access-Control-Request-Headers header (test3) from the browser does not match with the Access-Control-Allow-Headers (test1, test2) configured in the CORS policy.
4	Origin: http://test2.com Access-Control-Request-Method : POST Access-Control-Request-Headers : test1	Access-Control-Allow-Origin : http://test2.com Access-Control-Allow-Methods : POST Access-Control-Allow-Headers : test1, test2 Access-Control-Max-Age: 100 Access-Control-Allow-Credentials : true Access-Control-Expose-Headers : header1,header2	Sends 200 OK status with the following headers: <ul style="list-style-type: none"> ■ Access-Control-Allow-Origin : http://test2.com ■ Access-Control-Allow-Methods : POST,GET, PUT ■ Access-Control-Allow-Headers : test1,test2 ■ Access-Control-Max-Age: 100 ■ Access-Control-Allow-Credentials : true

#	CORS Preflight request headers from browser	Configured CORS Policy in API Gateway	API Gateway sends the respective response to browser
			Since the origin, methods, and headers from the browser matches with configured CORS policy in API Gateway.
5	Origin: http://test2.com Access-Control-Request-Method : POST Access-Control-Request-Headers : test1	Access-Control-Allow-Origin : http://test1.com Access-Control-Allow-Methods : POST Access-Control-Allow-Headers : test1, test2 Access-Control-Max-Age: 100 Access-Control-Allow-Credentials : true Access-Control-Expose-Headers : header1,header2 In addition, if you have specified the Javascript origins in the application as http://test2.com	Sends 200 OK status with the following headers: ■ Access-Control-Allow-Origin : http://test2.com ■ Access-Control-Allow-Methods : POST,GET,PUT ■ Access-Control-Allow-Headers : test1,test2 ■ Access-Control-Max-Age: 100 ■ Access-Control-Allow-Credentials : true Even though the origin header from the browser does not match with configured CORS policy, it matches with the configured javascript origins in the application.

CORS Request

A CORS request is a HTTP request that includes an *Origin* header. When API Gateway receives the CORS request, the *Origin* header in the CORS request is verified against the *Access-Control-Allow-Origin* configured in the CORS policy, if it matches then API Gateway allows to access the resources.

The following flow chart explains the flow of the CORS request received in API Gateway:



The following table shows the various use cases of the CORS request originating from browser and how API Gateway responds to each CORS requests:

#	CORS Request headers from browser	Configured CORS Policy in API Gateway	API Gateway sends the respective response to browser
1	Origin: http://test.com	Access-Control-Allow-Origin : http://test2.com Access-Control-Allow-Methods : POST,GET,PUT Access-Control-Allow-Headers : test1,test2 Access-Control-Max-Age: 100 Access-Control-Allow-Credentials : true Access-Control-Expose-Headers : header1,header2	Sends 403 Specified Origin is not allowed status, as the Origin header (http://test.com) from the browser does not match with the Access-Control-Allow-Origin (http://test2.com) configured in the CORS policy.
2	Origin: http://test2.com	Access-Control-Allow-Origin : http://test2.com	Sends 200 OK status with the following headers:

#	CORS Request headers from browser	Configured CORS Policy in API Gateway	API Gateway sends the respective response to browser
		Access-Control-Allow-Methods : POST,GET,PUT Access-Control-Allow-Headers : test1,test2 Access-Control-Max-Age: 100 Access-Control-Allow-Credentials : true Access-Control-Expose-Headers : header1,header2	<ul style="list-style-type: none"> ■ Access-Control-Allow-Origin : http://test2.com ■ Access-Control-Allow-Credentials : true ■ Access-Control-Expose-Headers : header1,header2 <p>Since the Origin header (http://test2.com) from the browser matches with the Access-Control-Allow-Origin (http://test2.com) configured CORS policy in API Gateway.</p>
3	Origin: http://test2.com Access-Control-Request-Method : POST Access-Control-Request-Headers : test1	Access-Control-Allow-Origin : http://test1.com Access-Control-Allow-Methods : POST Access-Control-Allow-Headers : test1, test2 Access-Control-Max-Age: 100 Access-Control-Allow-Credentials : true Access-Control-Expose-Headers : header1,header2 In addition, if you have specified the Javascript origins in the application as http://test2.com	Sends 200 OK status with the following headers: <ul style="list-style-type: none"> ■ Access-Control-Allow-Origin : http://test2.com ■ Access-Control-Allow-Methods : POST,GET,PUT ■ Access-Control-Allow-Headers : test1,test2 ■ Access-Control-Max-Age: 100 ■ Access-Control-Allow-Credentials : true <p>Even though the origin header from the browser does not match with configured CORS policy, it matches with the configured javascript origins in the application.</p>

Note:

- If native service supports CORS mechanism and if you have not configured the CORS policy in API Gateway, then API Gateway goes to pass-through security mode and forwards the CORS request to the native service.
- If native service supports CORS mechanism and if you have also configured the CORS policy in API Gateway, then API Gateway takes precedence in handling the CORS request.

Data Masking


Data masking is a technique whereby sensitive data is obscured in some way to render it safe and to protect the actual data while having a functional substitute for occasions when the real data is not required.

This policy is used to mask sensitive data at the application level. At the application level you must have an Identify and Access policy configured to identify the application for which the masking is applied. If no application is specified then it will be applied for all the other responses. Fields can be masked or filtered in the response messages to be sent. You can configure the masking criteria as required for the XPath, JSONPath, and Regex expressions based on the content-types. This policy can also be applied at the API scope level.

The table lists the content-type and masking criteria mapping.

Content-type	Masking Criteria
application/xml	XPath
text/xml	
text/html	
application/json	JSONPath
application/json/badgerfish	
text/plain	Regex

The table lists the masking criteria properties that you can configure to mask the data in the response messages:

Parameter	Description
Consumer Applications	<p><i>Optional.</i> Specifies the applications for which the masking criterion has to be applied.</p> <p>Start typing the application name, select the application from the type-ahead search results displayed, and click  to add one or more applications.</p> <p>For example: If there is a DataMasking(DM1) criteria created for application1 a second DataMasking(DM2) for application2 and a third DataMasking(DM3) with out any application, then for a request that comes from consumer1 the masking criteria DM1 is applied, for a request that comes from consumer2 DM2 is applied. If a request comes with out any application or from any other application except application1 and application2 DM3 is applied.</p>

Parameter	Description
-----------	-------------

You can use the delete icon  to delete the added applications from the list.

XPath: Specifies the masking criteria for XPath expressions in the response messages.

Masking Criteria Click **Add masking criteria** and provide the following information and click **Add**:

- **Query expression.** Specify the query expression that has to be masked or filtered.

For example: /pet/details/status, /user/details/card/ccnumber.

- **Masking Type.** Specifies the type of masking required. You select either **Mask** or **Filter**. Selecting **Mask** replaces the value with the given value (the default value being *****). Selecting **Filter** removes the field completely.

- **Mask Value.** This is available if masking type selected is **Mask**. Provide a mask value. For example: sold, any mask value #####.

Note:

You can add multiple masking criteria.

- **Namespace.** Specifies the following Namespace information:
 - **Namespace Prefix.** The namespace prefix of the payload expression to be validated.
 - **Namespace URI.** The namespace URI of the payload expression to be validated

Note:

You can add multiple namespace prefix and URI by clicking



JSONPath. This is applicable only for REST API. Specifies the masking criteria for JSONPath expressions in the response messages.

Masking Criteria Click **Add masking criteria** and provide the following information and click **Add**:

- **Query expression.** Specify the query expression that has to be masked or filtered. For example: \$.pet.details.status.
- **Masking Type.** Specifies the type of masking required. You select either **Mask** or **Filter**. Selecting **Mask** replaces the value with the

Parameter	Description
	<p>given value (the default value being *****). Selecting Filter removes the field completely.</p> <ul style="list-style-type: none"> ■ Mask Value. This is available if masking type selected is Mask. Provide a mask value. For example: sold
	<p>Regex. Specifies the masking criteria for regular expressions in the response messages.</p>
Masking Criteria	<p>Click Add masking criteria and provide the following information and click Add:</p> <ul style="list-style-type: none"> ■ Query expression. Specify the query expression that has to be masked or filtered. For example: [0-9]+. ■ Masking Type. Specifies the type of masking required. You select either Mask or Filter. Selecting Mask replaces the value with the given value (the default value being *****). Selecting Filter removes the field completely. ■ Mask Value. This is available if masking type selected is Mask. Provide a mask value. For example: ##### .
Apply for transaction Logging	<p>Select this option to apply masking criteria for transactional logs.</p> <p>When you select this option the transactional log for the response is masked on top of response sent to the client.</p>
Apply for payload	<p>Select this option to apply masking criteria for response payload in the following scenarios:</p> <ul style="list-style-type: none"> ■ response received from the native service. ■ response sent to the client. <p>Note: When you select this option it automatically masks the data in the transactional log.</p>

Error Handling

The policy in this stage enables you to specify the error conditions, lets you determine how these error conditions are to be processed. You can also mask the data while processing the error conditions. The policies included in this stage are:



- Conditional Error Processing
- Data Masking


Conditional Error Processing


Error Handling is the process of passing an exception message issued as a result of a run-time error to take any necessary actions. This policy returns a custom error message (and the native provider's service fault content) to the application when the native provider returns a service fault. You can configure conditional error processing and use variables to create custom error messages.


The table lists the properties that you can specify for this policy:

Parameter	Description
Error conditions.	Specifies the error conditions and how these error conditions should be processed.
Status Code Error Criteria	Specify the error status code. Provide a value for the Code .
Header Error Criteria	Provide the details of the custom HTTP header(s) included in the client requests. Provide the following information: <ul style="list-style-type: none"> ■ Header Name. Specifies the name of the HTTP header. ■ Header Value. Specifies the value of the HTTP header.
Payload Criteria	Provide the details of the payload criteria in the API request. In the Payload identifier section, click Add payload identifier , provide the following information, and click Add . <ul style="list-style-type: none"> ■ Expression type. Specifies the type of expression, which is used for identification. You can select one of the following expression types: <ul style="list-style-type: none"> ■ XPath. Provide the following information: <ul style="list-style-type: none"> ■ Payload Expression. Specifies the payload expression that the specified XPath expression type in the request or the response has to be converted to. For example: <code>/name/id</code>. The response may be a native service error or API Gateway generated error. ■ Namespace Prefix. The namespace prefix of the payload expression to be validated. ■ Namespace URI. The namespace URI of the payload expression to be validated. <p>Note:</p>

Parameter	Description
	<p>You can add multiple namespace prefix and URI by clicking .</p> <ul style="list-style-type: none"> JSONPath. Provide the Payload Expression that specifies the payload expression that the specified JSONPath expression type in the request or the response has to be converted to. For example: \$.name.id. <p>The response maybe a native service error or API Gateway generated error.</p> Text. Provide the Payload Expression that specifies the payload expression that the specified Text expression type in the request or response has to be converted to. For example: any valid regular expression. <p>The response maybe a native service error or API Gateway generated error.</p> <p>You can add multiple payload identifiers as required.</p> <p>Note: Only one payload identifier of each type is allowed. For example, you can add a maximum of three payload identifiers, each being of a different type.</p> <p>Value: Specifies a value that has to match with the value contained in the error Response.</p>
Pre-Processing.	Specifies how the native error response is to be processed before API Gateway processes it. .
Invoke webMethods Integration Server Service	<p>Specify the webMethods IS service to pre-process the error message. Provide the following information</p> <ul style="list-style-type: none"> webMethods IS Service. Specify the webMethods IS service to be invoked to pre-process the error messages. <p>You can add multiple entries for webMethods IS service by clicking .</p> Run as User. Specifies the authentication mode to invoke the IS service. If this field is left blank the incoming credentials of the user, identified by API Gateway, are used to authenticate and invoke the IS service. You can also specify a particular user, you want API Gateway to use to run the IS service.

Parameter	Description
	<ul style="list-style-type: none"> ■ Comply to IS Spec. Mark this as <code>true</code> if you want the input and the output parameters to comply to the IS Spec present in <code>pub.apigateway.invokeISService.specifications</code> folder in <code>wmAPIGateway</code> package. ■ webMethods IS Service alias. Start typing the <code>webMethods</code> alias name and select the alias from the type-ahead search results displayed to add one or more aliases.
XSLT Transformation	<p>Provide the XSLT file and feature you want to use to transform the service error response.</p> <p>Click Browse to select the XSLT file and upload it.</p> <p>Provide the following information for the XSLT feature:</p> <ul style="list-style-type: none"> ■ Feature Name. Specifies the name of the XSLT feature. ■ Feature Value. Specifies the value for the feature. <p>You can add multiple entries for feature name and value by clicking .</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: API Gateway supports XSLT 1.0 and XSLT 2.0.</p> </div>
	<p>Custom Error Variables. Specifies the error variables to be used in the custom error message.</p>
Payload Type	<p>Specify the payload type.</p> <p>Available values are:</p> <ul style="list-style-type: none"> ■ Request. Specifies the request payload type. ■ Response. Specifies the response payload type.
Name	<p>Provide a name for the payload type.</p>
Payload Identifier	<p>Provide the details of the payload criteria in the API request.</p> <p>In the Payload identifier section, click Add payload identifier, provide the following information, and click Add.</p> <ul style="list-style-type: none"> ■ Expression type. Specifies the type of expression contained in the payload request. ■ Payload Expression. Specifies the payload expression that the specified expression type in the request has to be converted to.

Parameter	Description
	<ul style="list-style-type: none"> ■ Namespace Prefix. The namespace prefix of the payload expression to be validated. ■ Namespace URI. The namespace URI of the payload expression to be validated. <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p>Note: You can add multiple namespace prefix and URI using the Add button.</p> </div> <p>You can add multiple payload identifiers as required.</p>
<p>Failure Message. Specifies the custom failure message format that API Gateway should send to the application. Specify whether the message should be in the text, json, or xml format.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p>Note: For a SOAP API, select the type text and provide the failure message to be included in the faultstring of the SOAP response.</p> <p>Failure message in type json, xml are not used for the SOAP response.</p> </div>	
<p>Send Native Provider Fault Message</p>	<p>Enable this parameter so that API Gateway sends the native SOAP or REST failure message to the application.</p> <p>When you disable this parameter, the failure message is ignored when a fault is returned by the native API provider.</p>
<p>Post-Processing. Specifies how the error response sent by the native service is to be processed before sending the same to the application.</p>	
<p>Invoke webMethods Integration Server Service</p>	<p>Specify the webMethods IS Service for post-processing the error message.</p> <p>Provide the following information</p> <ul style="list-style-type: none"> ■ webMethods IS Service. Specify the webMethods IS service to be invoked to post-process the error messages. <p>You can add multiple entries for webMethods IS service by clicking .</p> <ul style="list-style-type: none"> ■ Run as User. Specifies the authentication mode to invoke the IS service. If this field is left blank the incoming credentials of the user, identified by API Gateway, are used to authenticate and invoke the IS service. You can also specify a particular user, you want API Gateway to use to run the IS service. ■ Comply to IS Spec. Mark this as <code>true</code> if you want the input and the output parameters to comply to the IS Spec present in


Parameter	Description
	<p>pub.apigateway.invokeISService.specifications folder in wmAPIGateway package.</p> <ul style="list-style-type: none"> ■ webMethods IS Service alias. Start typing the webMethods alias name and select the alias from the type-ahead search results displayed to add one or more aliases.
XSLT Transformation	<p>Provide the XSLT file that you want to use to transform the service error response.</p> <p>Provide the following information for the XSLT feature:</p> <ul style="list-style-type: none"> ■ Feature Name. Specifies the name of the XSLT feature. ■ Feature Value. Specifies the value for the feature. <p>You can add multiple entries for feature names and values by clicking .</p> <p>Note: API Gateway supports XSLT 1.0 and XSLT 2.0.</p>

Data Masking


Data masking is a technique whereby sensitive data is obscured in some way to render it safe and to protect the actual data while having a functional substitute for occasions when the real data is not required.

This policy is used to mask sensitive data in the custom error messages being processed and sent to the application. Fields can be masked or filtered in the error messages. You can configure the masking criteria as required for the XPath, JPath, and Regex expressions. This policy can also be applied at the API scope level.

The table lists the masking criteria properties that you can configure to mask the data in the request messages received:

Parameter	Description
Consumer Applications	<p>Specifies the applications for which the masking criterion has to be applied.</p> <p>Start typing the application name, select the application from the type-ahead search results displayed, and click  to add one or more applications.</p>

Parameter	Description
-----------	-------------

You can use the delete icon  to delete the added applications from the list.

XPath. Specifies the masking criteria for XPath expressions in the error messages.

Masking Criteria Click **Add masking criteria** and provide the following information and click **Add**:

- **Query expression.** Specify the query expression that has to be masked or filtered. For example: /soapenv:Fault/faultstring.
- **Masking Type.** Specifies the type of masking required. You select either **Mask** or **Filter**.
- **Mask Value.** This is available if masking type selected is **Mask**. Provide a mask value. For example: *Error occurred while processing the request. Please check your input request* or any other meaningful message or string .

Note:

You can add multiple masking criteria.

- **Namespace.** Specifies the following Namespace information:
 - **Namespace Prefix.** The namespace prefix of the payload expression to be validated.
 - **Namespace URI.** The namespace URI of the payload expression to be validated

Note:

You can add multiple namespace prefix and URI by clicking



JPath. This is applicable only for REST API. Specifies the masking criteria for JPath expressions in the error messages.

Masking Criteria Click **Add masking criteria** and provide the following information and click **Add**:

- **Query expression.** Specify the query expression that has to be masked or filtered. For example: \$.error.reason.
- **Masking Type.** Specifies the type of masking required. You select either **Mask** or **Filter**.
- **Mask Value.** This is available if masking type selected is **Mask**. Provide a mask value. For example: *Error occurred while processing the*

Parameter	Description
	<i>request. Please check your input request or any other meaningful message or string.</i>
Regex.	Specifies the masking criteria for regular expressions in the error messages.
Masking Criteria	<p>Click Add masking criteria and provide the following information and click Add:</p> <ul style="list-style-type: none"> ■ Query expression. Specify the query expression that has to be masked or filtered. For example: (.*) . ■ Masking Type. Specifies the type of masking required. You select either Mask or Filter. ■ Mask Value. This is available if masking type selected is Mask. Provide a mask value. For example: <i>Error occurred while processing the request. Please check your input request or any other meaningful message or string.</i>
Apply for transaction Logging	<p>Select this option to apply masking criteria for transactional logs.</p> <p>When you select this option the transactional log for the response is masked on top of response sent to the client.</p>
Apply for payload	<p>Select this option to apply masking criteria for payload.</p> <p>When you select this option the payload in the response sent to the client is masked.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: When you select this option it automatically masks the data in the transactional log.</p> </div>

The API for Context Variables

API Gateway provides an API that you can use to:

- Set, get, declare, and remove custom context variables.
- Set and get the predefined context variables. (It is not allowed to declare or remove the predefined context variables.)

API Gateway provides the following JAVA services, which are defined in the class ISMediatorRuntimeFacade.java:

- pub.apigateway.ctxvar:getContextVariable
- pub.apigateway.ctxvar:setContextVariable
- pub.apigateway.ctxvar:declareContextVariable

■ pub.apigateway.ctxvar.removeContextVariable

pub.apigateway.ctxvar.getContextVariable

Use this JAVA service to retrieve a context variable's value and assign it to a pipeline variable. All parameter names are case-sensitive.

Parameter	Pipeline Type	Data Type	Description	Examples
MessageContext	in	Object ref	This object is inserted into the pipeline by API Gateway.	N/A
varName	in	String	Context variable name (system or custom).	For system context variable, use just the variable name to get its value. For example, PROTOCOL_HEADERS. For custom context variable, use the prefix "mx:" with the variable name to get its value. For example, mx:CUSTOM_VAR
serValue	out	Object ref	Java.io.serializable value. (Usually a string).	

The table lists the predefined system context variables and its syntax used to get system context variables using `pub.apigateway.ctxvar.getContextVariable`.

System Context Variable Name	ctxVar	IS Service	Syntax	Set or Get Supported
User		USER		Supports get
Inbound HTTP method		INBOUND_HTTP_METHOD		Supports get
Routing method		ROUTING_METHOD		Supports get
Inbound content type		MESSAGE_TYPE		Supports get
Inbound accept		BUILDER_TYPE		Supports get
Inbound protocol		INBOUND_PROTOCOL		Supports get
Inbound request URI		INBOUND_REQUEST_URI		Supports get
Inbound IP		INBOUND_IP		Supports get
Gateway hostname		MEDIATOR_HOSTNAME		Supports get
Gateway IP		MEDIATOR_IP		Supports get

System Context Variable Name	ctxVar IS Service Syntax	Set or Get Supported
Operation name	OPERATION	Supports get
Native Endpoint	NATIVE_ENDPOINT	Supports get
Protocol headers	PROTOCOL_HEADERS[xxx]	Supports set and get
SOAP headers	SOAP_HEADERS[xxx]	Supports set and get

Note:

This variable returns native endpoint value, only after Routing policy gets executed.

Notes on getting and setting the PROTOCOL_HEADERS

All context variable values are typed as either `string` or `int` except for the predefined context variables, `PROTOCOL_HEADERS`, which is of the type `IData`. You can set or get value for `PROTOCOL_HEADERS` in one of the following ways:

- **set or get the entire structure.**

To set the entire structure, you must:

- Set the `varName` parameter in `pub.apigateway.ctxvar:setContextVariable` to `PROTOCOL_HEADERS`.
- Use the method `ISMediatorRuntimeFacade.setContextVariableValue()`.

To get the entire structure, you must:

- Set the `varName` parameter in `pub.apigateway.ctxvar:getContextVariable` to `PROTOCOL_HEADERS`.
- Use the method `ISMediatorRuntimeFacade.getContextVariableValue()`.

If the `varName` is set to `PROTOCOL_HEADERS`, you get or set the entire `IData` structure containing all of the transport headers. The key is the transport header name (for example, `Content-Type`) and the value is a `String`. The `IData` object for `PROTOCOL_HEADERS` contains a set of string values where each `IData` string key matches the header name in the transport headers map. The set of possible keys includes the HTTP v1.1 set of headers as well as any custom key-value pairs you might have defined.

Alternatively, you can set the `varName` parameter to address a specific element in the array. For example, setting it to `PROTOCOL_HEADERS[Content-Type]` would apply to the `Content-Type` transport header.

- **set or get a nested value.**

Set a nested value in one of the following ways:

- Set the `varName` parameter in `pub.apigateway.ctxvar:setContextVariable` to `PROTOCOL_HEADERS[arrayElement]`, where `[arrayElement]` refers to a specific element. For example, `PROTOCOL_HEADERS[Content-Type]` (to indicate the first array element in the set).
- Alternatively, use the method `ISMediatorRuntimeFacade.setContextVariableValue()`. Use this method only if you are writing a JAVA service and you want to access it through the JAVA source code.

Get a nested value in one of the following ways:

- Set the `varName` parameter in `pub.apigateway.ctxvar:getContextVariable` to `PROTOCOL_HEADERS[arrayElement]`, where `[arrayElement]` refers to a specific element. For example, `PROTOCOL_HEADERS[Content-Type]` (to indicate the first array element in the set).
- Alternatively, use the method `ISMediatorRuntimeFacade.getContextVariableValue()`. Use this method only if you are writing a JAVA service and you want to access it through the JAVA source code.

You can set or get a nested value inside `PROTOCOL_HEADERS` through an additional `keyName`. In this case, the object reference is *not* an `IData` object. For `PROTOCOL_HEADERS`, the `keyName` must match the transport header name in a case-sensitive manner (for example, `PROTOCOL_HEADERS[Content-Type]` or `PROTOCOL_HEADERS[Authorization]`). In this case, the `Serializable` value will be a string.

pub.apigateway.ctxvar:setContextVariable

Use this JAVA service to set a value on a context variable. The pipeline variable containing the context variable value should be an object reference that implements `java.io.Serializable`. All parameter names are case-sensitive.

Parameter	Pipeline Data Type	Type	Description	Examples
<code>MessageContext</code>	<code>in</code>	Object ref	This object is inserted into the pipeline by API Gateway.	N/A
<code>varName</code>	<code>in</code>	String	Context variable name (predefined or custom).	<code>PROTOCOL_HEADERS</code> <code>mx: CUSTOM_VAR</code>
<code>serValue</code>	<code>in</code>	Object ref	<code>Java.io.Serializable</code> value. (Usually a string).	

pub.apigateway.ctxvar:declareContextVariable

Use this JAVA service to declare a *custom* context variable. All custom-defined context variables must be declared in a custom namespace that is identified by using the prefix `mx` (for example, `mx: CUSTOM_VARIABLE`). All parameter names are case-sensitive.

Note:

It is not legal to use this service to declare the predefined context variables; you can only declare custom variables.

Parameter	Pipeline Data Type	Data Type	Description
ctxVar	in	Object ref	The document type defining the context variable object. Use the ctxVar Document Type provided in the JAVA service <code>pub.apigateway.ctxvar:ctxVar</code> and map it to this input variable. Define the name (for example, <code>mx:CUSTOM_VARIABLE</code>), the schema_type (string or int), and isReadOnly (true or false).
ctxVar	out	Object ref	The set Context variable document type.
varNameQ	out	Object ref	<code>javax.xml.namespace.QName</code> value. The QName of the variable.

Note the following:

- After declaring the context variable, you can use the `setContext` variable to set a value on the context variable.
- You do *not* need to declare the following kinds of context variables:
 - The predefined context variables provided by API Gateway. If you attempt to declare an existing predefined context variable, an error will occur.
 - Any custom context variable that you define in a routing rule that you create in the context-based routing step.
- Any custom context variables that you explicitly declare in source code using the API will have a declaration scope of `SESSION`.
- Any custom context variable's state that is defined during the inbound request processing steps will still be available during the outbound response processing steps.
- All context variable values are typed as either `string` or `int` (excluding the `PROTOCOL_HEADERS` variables, which are of the type `IData`).
- Valid names should be upper case (by convention) and must be a valid JAVA Identifier. In general, use alpha-numeric, `$` or `_` symbols to construct these context names. Names with punctuation, whitespace or other characters will be considered invalid and will fail deployment.
- All custom context variables must be declared in a custom namespace that is identified by using an `mx` prefix (for example, `mx:CUSTOM_VARIABLE`).
- To reference a custom context variable in a flat string, you need to prepend a `$` symbol to the context variable name to indicate that variable's value should be referenced. Think of this usage as being similar to the `&` address operation for C variables.

An expression that references a custom context variable might look like this:

```
$mx:TAXID=1234 or $mx:ORDER_SYSTEM_NAME="Pluto"
```

Notice that the values of the data type “int” are not enclosed in quotation marks, while the values of the data type “string” are. The quotation marks are only needed if a context variable *expression* (as opposed to a reference) is defined.

- Referencing an undefined context variable does not result in an error.
- Once a variable has been declared it cannot be declared again.

pub.apigateway.ctxvar:removeContextVariable

Use this JAVA service to remove a *custom* context variable from a request or response session. All parameter names are case-sensitive.

Note:

Keep the following points in mind:

- It is not legal to use this service to remove any predefined context variables; you can only remove custom variables.
- Attempting to remove a non-existent context variable will *not* result in an error.

Parameter	Pipeline Data Type	Description	Examples
MessageContext in	Object ref	This object is inserted into the pipeline by API Gateway.	N/A
varName	in String	Custom context variable name.	<code>mx:CUSTOM_VAR</code>

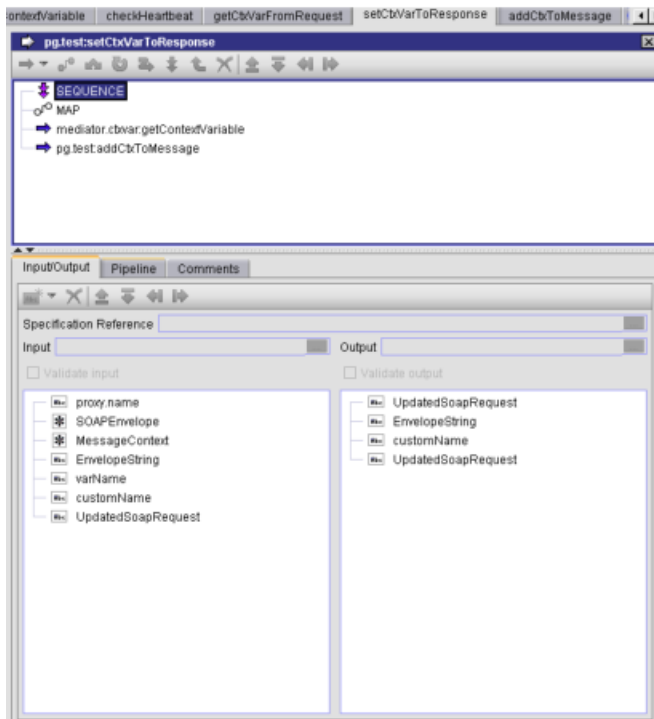
Sample Flow Service: Getting a Context Variable Value

This flow service sets the value of a custom context variable to be used in a response.

This flow service declares a pipeline variable named `customName`, which is set to the value `mx:COMP_TEST`.

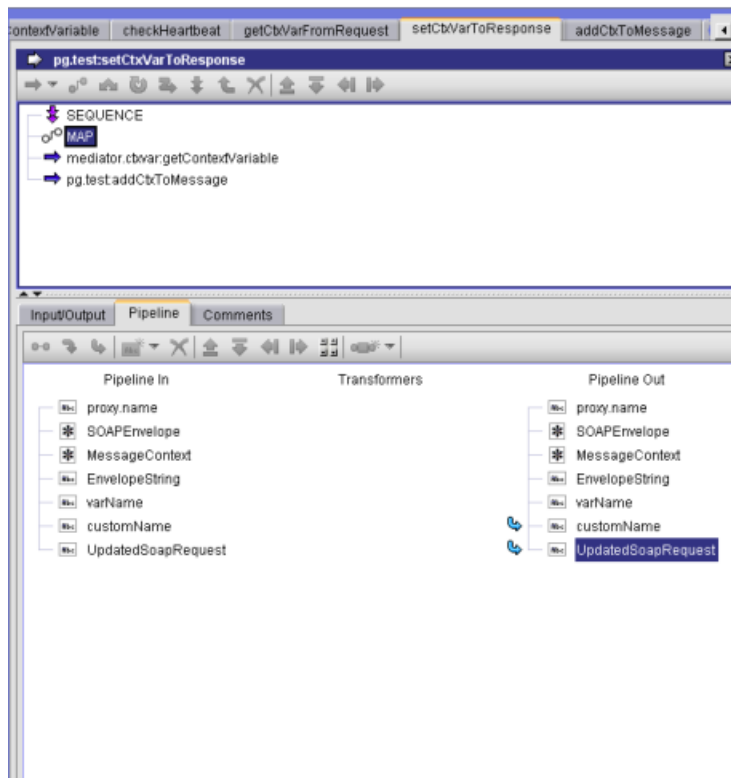
This flow service will retrieve the context variable for `customName` and create an element for its context variable value in the response message return to the consumer.

Step 1. Declaring `customName`



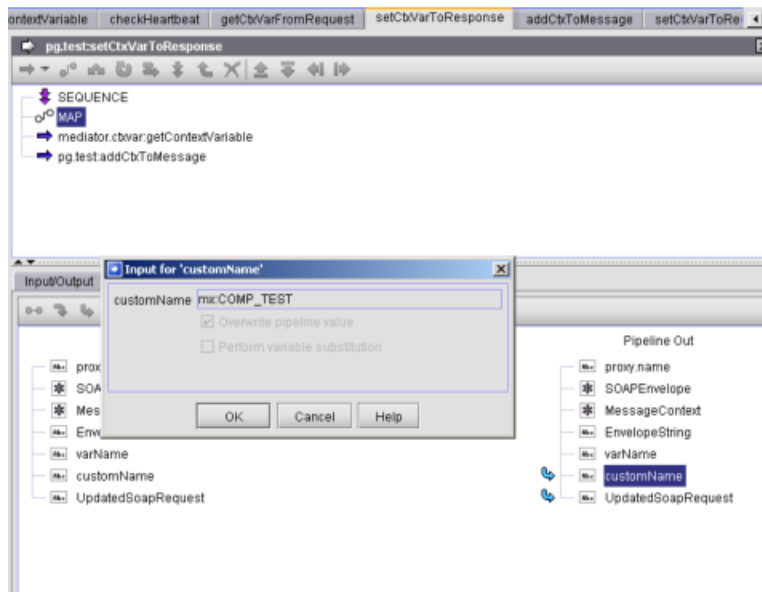
You can define the `customName` variable value to be `mx:COMP_TEST` so you can use this variable to lookup the custom variable name that was seeded in the previous example.

Step 2. Setting `customName` to `mx:COMP_TEST`



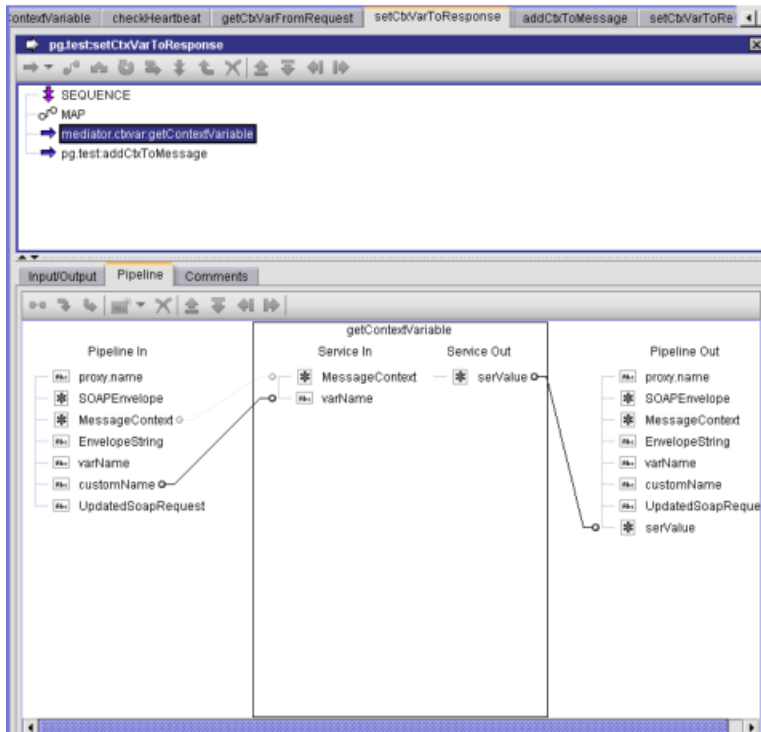
Clicking on the customName pipeline variable displays the name.

Step 3. Displaying the value of customName



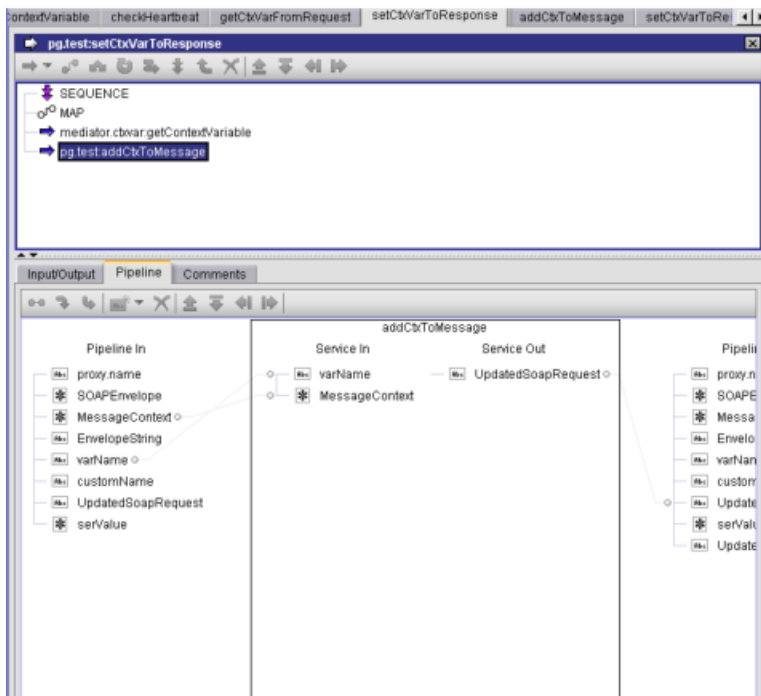
The call to `pub.mediator.ctxvar:getContextVariable` retrieves the value of the custom context variable from the context variable map.

Step 4. Calling `meditor.ctxvar:getContextVariable`



This is just a sample JAVA service that takes the context variable and creates a top-level element in the response message using the same name and value.

Step 5. Sample service using the context variable



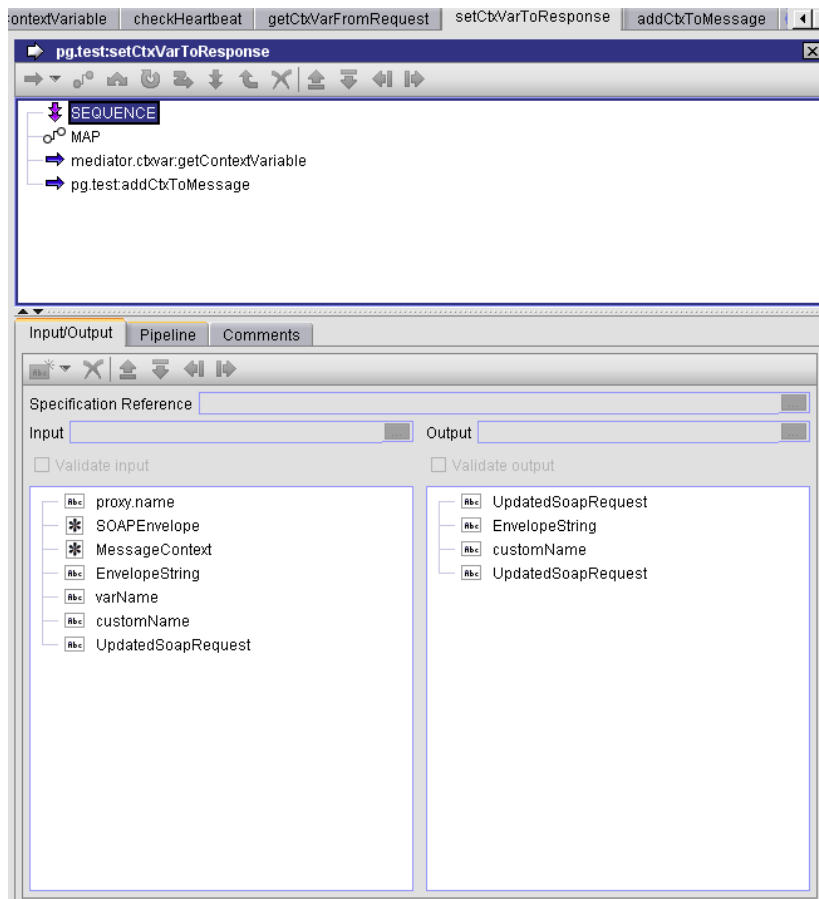
Sample Flow Service: Setting a Context Variable Value

This flow service sets the value of a custom context variable to be used in a response.

This flow service declares a pipeline variable named `customName`, which is set to the value `mx:COMP_TEST`.

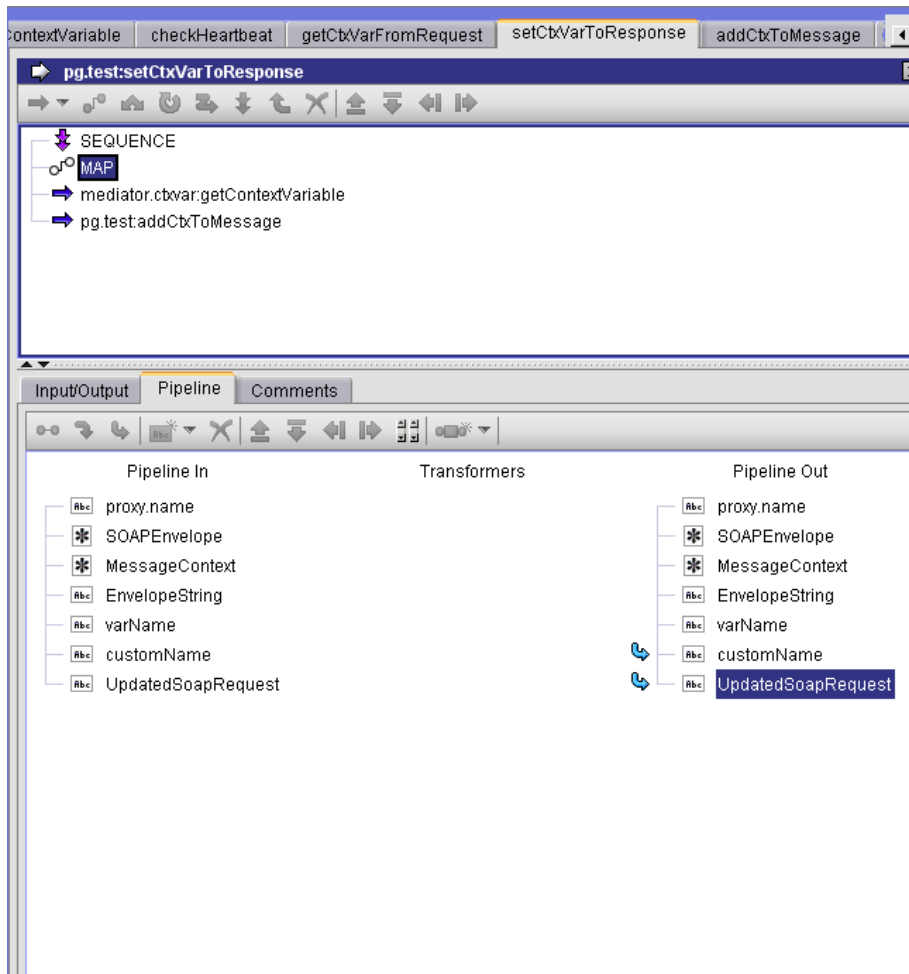
This flow service retrieves the context variable for `customName` and create an element for its context variable value in the response message return to the consumer.

Step 1. Declaring `customName`



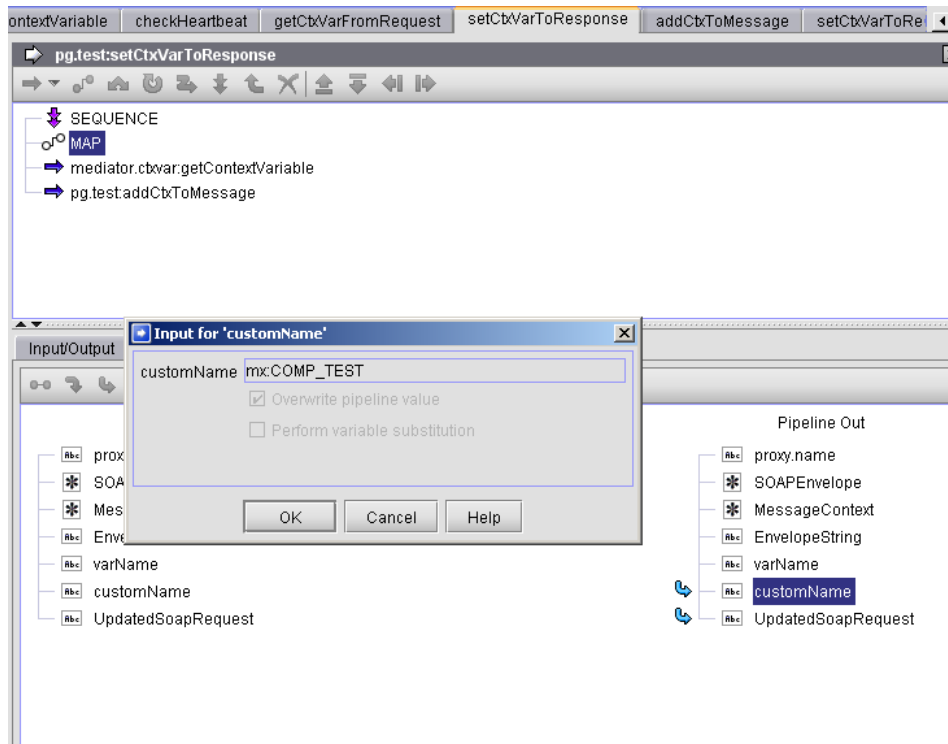
You can define the `customName` variable value to be `mx:COMP_TEST` so you can use this variable to lookup the custom variable name that was seeded in the previous example.

Step 2. Setting `customName` to `mx:COMP_TEST`



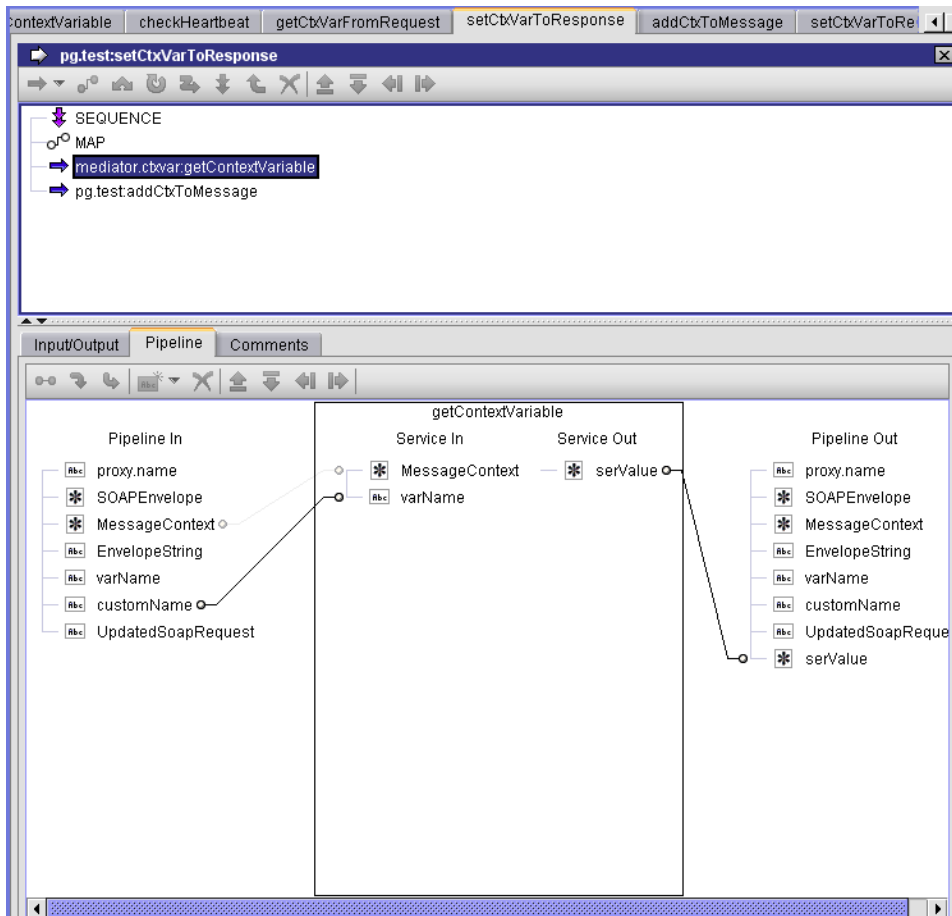
Clicking on the customName pipeline variable will display the name.

Step 3. Displaying the value of customName



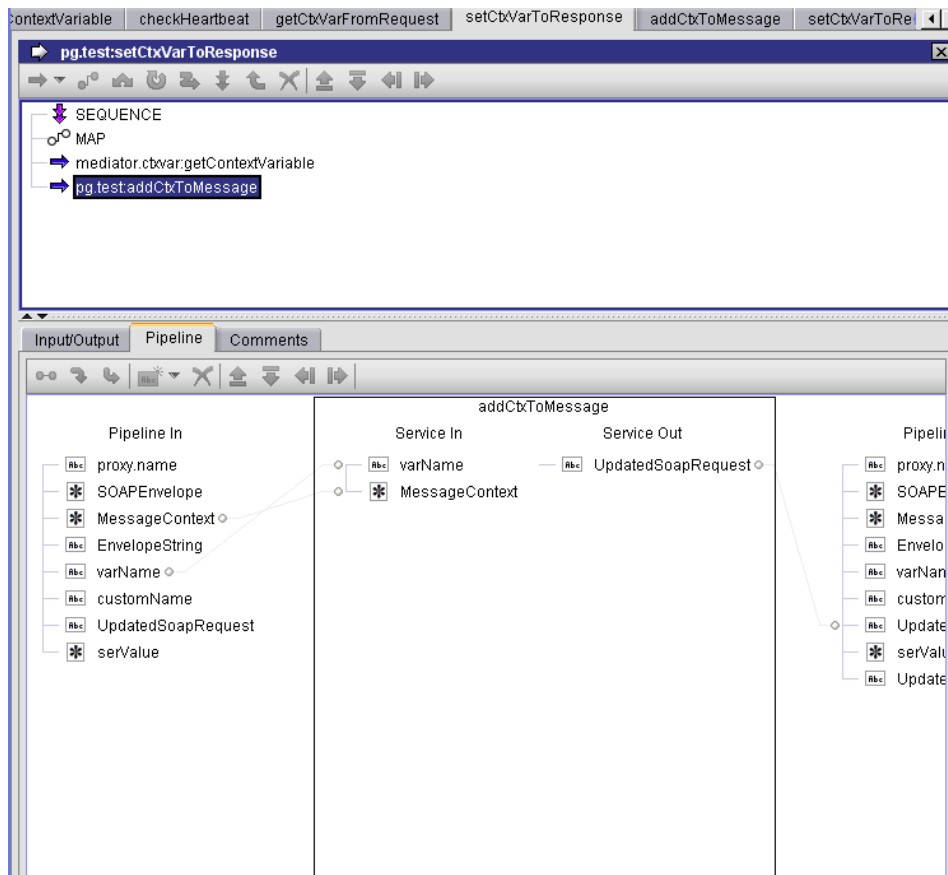
The call to `pub.mediator.ctxvar:getContextVariable` retrieves the value of the custom context variable from the context variable map.

Step 4. Calling `mediator.ctxvar:getContextVariable`



This is just a sample JAVA service that takes the context variable and creates a top-level element in the response message using the same name and value.

Step 5. Sample service using the context variable



Managing Global Policies

Important:

API Gateway's Standard Edition License does not support the functionality of Global policies. You can create and manage global policies only using the Advanced Edition License.

Global policies are a set of policies that are associated globally to all APIs or the selected set of APIs. Global policies are supported for both SOAP and REST APIs.

By associating policies globally to all APIs or the selected set of APIs, the administrator can ensure that a set of policies is applied to the selected APIs by default. The administrator can, for example, define a global policy that attaches a WS-Security (WSS) authentication to all SOAP API endpoints within a specific IP range. In this case, any client request from the specific IP range automatically inherits the security configuration defined in the global policy for SOAP APIs.

Global Policy Matrix

This table lists the stage-specific policies that can be configured as global policy for different types of APIs at the global level.

Note:

The **Policy configuration** page displays only the policies that are common to one or more API types selected in the global policy filter.

Stages	Policies
Transport	<ul style="list-style-type: none"> Require HTTP/HTTPS - This policy can be enforced for all types of API. But the SOAP versions 1.1 and 1.2 are applicable only for SOAP-based APIs. The SOAP 1.1 and SOAP 1.2 sub types are not available in UI when the REST and ODATA APIs are selected. <div style="background-color: #f0f0f0; padding: 5px; margin: 5px 0;"> <p>Note: Software AG recommends to create a separate policy for each API type.</p> </div> <ul style="list-style-type: none"> Set Media Type - This policy is applicable only for a REST request and the policy name is not listed in Policy configuration page when the SOAP and ODATA APIs are selected. Require JMS - The Require JMS policy is applicable only for SOAP APIs and the policy name is not listed in Policy configuration page when the REST and ODATA APIs are selected.
Identity & Access	<ul style="list-style-type: none"> Inbound Authentication - Transport, Authorize User, Identify and Authorize Application - These policies can be enforced to any API Type. Inbound Authentication - Message - This policy is applicable only for SOAP-based APIs and the policy name is not listed in Policy configuration page when the REST and ODATA APIs are selected.
Request Processing	<ul style="list-style-type: none"> Invoke webMethods IS, Validate API Specification, Data Masking - These policies can be enforced to any API Type. Request Transformation - This policy is applicable only for SOAP and REST APIs. and not for ODATA services. When all three API types are selected, Request Transformation policy cannot be applied at the global level.
Routing	<ul style="list-style-type: none"> Custom HTTP Header, Outbound Authentication - Transport, Outbound Authentication - Message. The Routing stage policies can be applied at a global level for all types of API.
Traffic Monitoring	<ul style="list-style-type: none"> Log Invocation, Monitor Service Performance, Monitor Service Level Agreement, Throttling Traffic Optimization, and Service Result Cache. The Traffic Monitoring stage policies can be applied at a global level for all types of API.
Response Processing	<ul style="list-style-type: none"> Invoke webMethods IS, Validate API Specification, Data Masking - These policies can be enforced to any API Type. Response Transformation - This policy can be enforced only for SOAP and REST APIs and the policy name is not listed in Policy configuration page when ODATA API type is selected. CORS - This policy can be enforced only for REST and ODATA APIs and the policy name is not listed in Policy configuration page when SOAP-based API is selected.

Stages	Policies
Error handling	Conditional Error Processing and Data Masking. The Error handling stage policies can be applied at a global level for all types of API.

Creating a Global Policy

You must have the API Gateway's manage global policies functional privilege assigned.

To create a global policy you must perform the following high-level steps:

1. **Create a new global policy:** During this step, you specify the basic details of the global policy.
2. **Optionally refine the scope of the policy:** During this step, you can specify additional criteria to narrow the set of APIs to which the global policy applies.
3. **Configure the policies:** During this step, you associate one or more policies, and assign values to each of the associated policy's properties.
4. **Activate the policy:** During this step, you put the new global policy into effect.

> To create a global policy

1. Click **Policies** in the title navigation bar.
2. Click the **Global Policies** tab.

A list of all available global policies appears.

3. In the **Policies** page, click the **Create Global Policy** button.

If you do not see the **Create Global Policy** button, it is probably because you do not have the API Gateway Administrator role to create a global policy in API Gateway.

This opens the Create Global Policy page with the default **Policy Details** tab.

4. In the Basic Information section, provide the required information for each of the displayed data fields:

Field	Description
Name	Name of the global policy.
Description	Description of the global policy.

5. Click **Save** to save the new (as yet incomplete) global policy.
6. Complete the new global policy by doing the following:

- a. On the Filters section, specify the API types, additional criteria for selecting the APIs to which you want the global policy to be applied, logical operator for the selection criteria, and the APIs to which the global policy applies. For details, see [“ Modifying the Scope of a Global Policy” on page 545](#) and [“ Refining the Scope of a Global Policy” on page 546](#).
- b. On the **Policy Configuration** tab, choose the policies and configure the properties for each policy that you want API Gateway to execute when it applies this global policy. For details, see [“ Associating Policies to a Global Policy” on page 548](#) and [“ Configuring Properties for a Global Policy” on page 550](#).
- c. Activate the global policy when you are ready to put it into effect. For details, see [“ Activating a Global Policy” on page 553](#).

Modifying the Scope of a Global Policy

You must have the API Gateway's manage global policies functional privilege assigned.

Scope refers to the set of properties that determine a selected set of APIs for the enforcement of the policy. For a global policy, scope is determined by the policy's property **API Type** in the **Policy Details** tab.

API Type	Description
REST	Global policy will be applied on all REST APIs in API Gateway.
SOAP	Global policy will be applied on all SOAP APIs in API Gateway.

➤ To modify the scope of a global policy

1. Click **Policies** in the title navigation bar.
2. Click the **Global Policies** tab.

A list of all available global policies appears.

3. Select the required policy.

The Global Policy details page appears.

4. Click **Edit**.

If you do not see the **Edit** button, it is probably because you do not have the API Gateway Administrator role to modify the scope of a global policy in API Gateway.

5. Select the policy's Filters section, and specify the following:

- a. In the **API Type** property, select the API types (**REST**, **SOAP**, **ODATA**, or all) to which the policy will be applied.
 - b. *Optional.* In the Filter using attributes section, specify additional selection criteria to narrow the set of APIs to which this policy will be applied. For details, see “[Refining the Scope of a Global Policy](#)” on page 546.
6. Click **Save**.

Refining the Scope of a Global Policy

You must have the API Gateway's manage global policies functional privilege assigned.

If you want to further restrict the set of APIs to which the global policy is applied, you can specify additional selection criteria in the Filter section of the API details page. Using the Filter section, you can filter APIs by Name, Description, Version attributes, and HTTP Methods (applicable only for REST APIs). If you specify no filter criteria, the global policy will apply to all of the selected APIs.

Filtering by Name, Description, and Version attributes

You can filter APIs based on their Name, Description, and Version attributes using any of the following comparison operators:

Comparison Operators	Description
Equals	Selects APIs whose Name, Description, or Version value matches a given string of characters. For example, use this operator to apply a policy only to REST APIs with the Name or Description value <code>4G Mobile Store</code> .
Not Equals	Selects APIs whose Name, Description, or Version value does not match a given string of characters. For example, use this operator to apply a policy only to all REST APIs except those with the Name or Description value <code>Mobile</code> .
Contains	Selects APIs whose Name or Description value includes a given string of characters anywhere within the attribute's value. For example, use this operator to apply a policy to REST APIs that had the word <code>Mobile</code> anywhere in their Name or Description attribute.
Starts with	Selects APIs whose Name or Description value begins with a given string. For example, use this operator to apply a policy only to REST APIs whose Name or Description begins with the characters <code>4G</code> .
Ends with	Selects APIs whose Name or Description value ends with a given string. For example, use this operator to apply a policy only to REST APIs whose Name or Description ends with the characters <code>Store</code> .

When specifying match strings for the comparison operators described above, keep the following points in mind:

- Match strings *are not case-sensitive*. If you define a filter for names that start with ABC it will select names starting abc and Abc.
- Wildcard characters are not supported. That is, you cannot use characters such as * or % to represent *any sequence of characters*. These characters, if present in the match string, are simply treated as literal characters that are to be matched.

Filtering by HTTP Methods (Applicable only for REST APIs)

- You can optionally restrict a policy to specific HTTP methods of the REST APIs by specifying the following options - GET, POST, PUT, DELETE, and PATCH.

HTTP Methods	Description
GET	Policy applies only to HTTP GET requests for any resource in the API. For example, use this option to apply a policy to resources of a REST API during an incoming GET request.
POST	Policy applies only to HTTP POST requests for any resource in the API. For example, use this option to apply a policy to resources of a REST API during an incoming POST request.
PUT	Policy applies only to HTTP PUT requests for any resource in the API. For example, use this option to apply a policy to resources of a REST API during an incoming PUT request.
DELETE	Policy applies only to HTTP DELETE requests for any resource in the API. For example, use this option to apply a policy to resources of a REST API during an incoming DELETE request.
PATCH	Policy applies only to HTTP PATCH requests for any resource in the API. For example, use this option to apply a policy to resources of a REST API during an incoming PATCH request.

➤ To refine the scope of a global policy

1. Click **Policies** in the title navigation bar.

2. Click the **Global Policies** tab.

A list of all available global policies appears.

3. Select the required policy.

The Global Policy details page appears.

4. Click **Edit**.

If you do not see the **Edit** button, it is probably because you do not have the API Gateway Administrator role to refine the scope of a global policy in API Gateway.

5. Click **Filters**.
6. To filter by Name, Description, or Version, take the following steps in the Filter using API Attributes section:
 - a. Select **API Name**, **API Description**, or **API Version** in the first field.
 - b. Select the comparison operator in the second field.
 - c. Specify the match string in the third field.
 - d. To specify additional criteria, click the **Add** button and repeat the above steps.
 - e. Select the logical conjunction (**AND**) or disjunction (**OR**) operation to apply when multiple criteria are specified for the global policy. The default value is AND.
7. Applicable only for REST APIs. To filter by HTTP methods, in the Filter using HTTP Methods section, select the HTTP methods by which you want to filter APIs with appropriate incoming requests.
8. Click **Save** to save the updated policy.

Associating Policies to a Global Policy

You must have the API Gateway's manage global policies functional privilege assigned.

The **Policy Configuration** tab on the Global Policy details page specifies the policy stages and the list of policies (applicable for each stage) that you want API Gateway to execute when it enforces the global policy.

When modifying the list of policies for a global policy, API Gateway validates the policies to ensure that there are no policy conflicts.

> To modify the list of policies of a global policy

1. Click **Policies** in the title navigation bar.
2. Click the **Global Policies** tab.

A list of all available global policies appears.
3. Select the required policy.

The Global Policy details page appears.

4. Click **Edit**.

If you do not see the **Edit** button, it is probably because you do not have the API Gateway Administrator role to modify the list of policies of a global policy in API Gateway.

5. Select the policy's **Policy Configuration** tab.

The policy information is provided in the following sections:

- Policy catalog - Transport, Identify and Access, Request Processing, Routing, Traffic Monitoring, Response Processing, Error Handling
- Infographic - List of applied policies
- Policy properties - Collection of policy properties

6. In the Policy catalog section, click the chevron to expand the required policy stage.

This displays a list of policies that are classified under the particular stage.

7. In the expanded list of policies, select the policies that you want API Gateway to execute when it applies this global policy. To select a policy, click the **Add (+)** icon next to the policy name. The selected policies are displayed in the Infographic section.

When you select the policies for the global policy, keep the following points in mind:

- The policies shown in the Policy catalog section are determined by the API types that you have specified on the Filters section of the Global Policy Details page.

If you do not see a policy that you need, that policy is probably not compatible with the API type that you selected in the Filters section.

- If necessary, you can click the **Policy Details** tab and change your API type selection.

Use the **x** icon in any individual policy to remove that particular policy from the Infographic section.

8. To configure the properties for any new policies that you might have added to the Infographic section in the preceding steps or to make any necessary updates to the properties for existing policies in the global policy, see “[Configuring Properties for a Global Policy](#)” on page 550.9. Click **Save**.10. Click  to view the complete list of policies in the updated policy.

The **Overview** button is located in the lower right-corner of the Infographic section.

To exit the overview, click the **Close** icon.

Configuring Properties for a Global Policy

You must have the API Gateway's manage global policies functional privilege assigned.

The **Policy Configuration** tab on the Global Policy details page specifies the list of policies that are applicable for each policy stage in the Policy catalog section. Each policy in the Infographic section has properties that you must set to configure the policy's enforcement behavior.

> To configure the properties for a global policy

1. Click **Policies** in the title navigation bar.

2. Click the **Global Policies** tab.

A list of all available global policies appears.

3. Select the required policy.

The Global Policy details page appears.

4. Click **Edit**.

If you do not see the **Edit** button, it is probably because you do not have the API Gateway Administrator role to configure the properties of a global policy in API Gateway.

5. Select the policy's **Policy Configuration** tab.

The policy information is provided in the following sections:

- Policy catalog - Transport, Identify and Access, Request Processing, Routing, Traffic Monitoring, Response Processing, Error Handling
- Infographic - List of applied policies
- Policy properties - Collection of policy properties

6. In the Policy catalog section, click the chevron to expand the required policy stage.

This displays a list of policies that are classified under the particular stage.

7. In the Infographic section, do the following for each policy in the list:

- a. Select the policy whose properties you want to examine or set.
- b. In the Policy properties section, set the values for the policy's properties as necessary.

Note:

Required properties are marked with an asterisk.

- Click **Open in full-screen** to view the policy's properties in full screen mode.

The **Open in full-screen** link is located in the upper right-hand corner of the **Policy Configuration** tab. Set the properties of the displayed policy, and then click **OK**.

To exit out of full screen mode, click the **Minimize** icon.

- Click **Save**.

- Click  to view the complete list of policies in the updated policy.

The **Overview** button is located in the lower right-corner of the Infographic section.

To exit the overview, click the **Close** icon.

Viewing List of Global Policies and Policy Details

The **Global Policies** tab displays a list of all globally available policies in API Gateway. Global policies are listed alphabetically by name.

In addition to viewing the list of policies, you can also examine the details of a policy, create a copy of the template, activate, and delete a global policy in the **Global Policies** tab.

➤ To view the policy list and properties of a global policy

- Click **Policies** in the title navigation bar.
- Click the **Global Policies** tab.

A list of all available global policies appears.

The **Global Policies** tab provides the following information about each policy:

Column	Description
Name	Name of the global policy.
Description	The description for the global policy.

You can also perform the following operations on a global policy:

- Activate a policy to begin enforcing runtime behaviors.
- Deactivate a policy to suspend enforcement of runtime behaviors.
- Create a new copy of the policy.
- Delete a policy to remove it from API Gateway.

3. Select the required policy whose details you want to examine.

The Global Policy details page appears. The policy details are displayed in the following tabs:

- **Policy Details:** This tab contains a summary of basic information such as name, description, scope of the policy as to when the policy will apply, applicable APIs, and other information.
- **Policy Configuration:** This tab contains the policy stages, applied policies, as well as the configuration details of individual policies.

Modifying Global Policy Details

You must have the API Gateway's manage global policies functional privilege assigned.

You use the Global Policy details page to examine and modify the properties of a policy.

When modifying the details of a global policy, keep the following points in mind:

- You will not be allowed to save the policy unless all of its properties have been set.
- On successful modification of the policy details for an active global policy, the policy changes apply with immediate effect in all the active APIs that are applicable for this global policy.
- You will not be allowed to remove an individual policy (for example, Identify and Authorize Application) from the active global policy, if the global policy is already applied to an active API, and if the Identify and Authorize Application is a dependent policy for another policy (for example, Throttling Traffic Optimization) that is applied for the API.
- If modification of the policy details for an active global policy fails, API Gateway issues an error message with details of the incompatible or conflicting policies.

➤ To modify the properties of a global policy

1. Click **Policies** in the title navigation bar.
2. Click the **Global Policies** tab.

A list of all available global policies appears.

3. Select the required policy.

The Global Policy details page appears. The policy details are displayed in the following tabs:

- **Policy Details:** This tab contains a summary of basic information such as name, description, scope of the policy as to when the policy will apply, applicable APIs, and other information.
- **Policy Configuration:** This tab contains the policy stages, applied policies, as well as the configuration details of individual policies.

4. Click **Edit**.

If you do not see the **Edit** button, it is probably because you do not have the API Gateway Administrator role to modify the properties of a global policy in API Gateway.

5. On the **Policy Details** tab, modify the basic properties, selection criteria, and the applicable APIs as necessary.
6. On the **Policy Configuration** tab, modify the policy list and the policy's configuration properties as necessary.
7. When you have completed the required modifications in the Global Policy details page, click **Save** to save the updated policy.
8. Click **Overview** to view the complete list of policies in the updated policy.

The **Overview** button is located in the lower right-corner of the Infographic section.

To exit the overview, click the **Close** icon.

Activating a Global Policy



You must have the API Gateway's activate global policies functional privilege assigned.

Global policies are not enforced until they are activated.

When you activate a global policy, be aware that:

- When a global policy becomes active, API Gateway begins enforcing it immediately in all the applicable APIs that are currently in the **Active** state. You can suspend enforcement of a policy by switching it to the **Inactive** state as described in [“Deactivating a Global Policy” on page 554](#).
- Activation of a global policy fails if there is a conflict in the effective policy validation in at least one of the active APIs that are applicable for this policy. API Gateway reports the conflict, and the global policy can only be activated when the conflict is resolved.

To determine whether a global policy is active or inactive, examine the policy's **Active** indicator on the **Policies > Global Policies** tab. The icon in the **Active** column indicates the policy's activation state as follows:

Icon	Description
	Policy is active.
	Policy is inactive.

The activation state of a policy is also reported in the Global Policy Details page.

> To activate a global policy

1. Click **Policies** in the title navigation bar.
2. Click the **Global Policies** tab.

A list of all available global policies appears.

3. Select the required policy.

The Global Policy details page appears.

4. Click **Activate**.

If you do not see the **Activate** button, it is probably because you do not have the API Gateway Administrator role to activate a global policy, or the policy is already in an **Active** state in API Gateway.

Deactivating a Global Policy

You must have the API Gateway's activate global policies functional privilege assigned.

Deactivating a global policy causes API Gateway to suppress enforcement of the policy.



You usually deactivate a policy to suspend enforcement of a particular policy (temporarily or permanently).

To deactivate a policy, you change the policy to the **Inactive** state. At a later time, you can begin enforcing a global policy by switching it to the **Active** state as described in “[Activating a Global Policy](#)” on page 553.

When you deactivate a global policy, be aware that:

- Deactivation of a global policy fails if there is a conflict in the effective policy validation in at least one of the active APIs that are applicable for this policy. API Gateway reports the conflict, and the global policy can only be activated when the conflict is resolved.

To determine whether a global policy is active or inactive, examine the policy's **Active** indicator on the **Policies > Global Policies** tab. The icon in the **Active** column indicates the policy's activation state as follows:

Icon	Description
	Policy is active.
	Policy is inactive.

The deactivation state of a policy is also reported in the Global Policy Details page.

> To deactivate a global policy

1. Click **Policies** in the title navigation bar.
2. Click the **Global Policies** tab.

A list of all available global policies appears.

3. Select the required policy.

The Global Policy details page appears.

4. Click **Deactivate**.

If you do not see the **Deactivate** button, it is probably because you do not have the API Gateway Administrator role to deactivate a global policy, or the policy is already in an **Inactive** state in API Gateway.

Deleting a Global Policy

You must have the API Gateway's manage global policies functional privilege assigned.

You delete a global policy to remove it from API Gateway permanently.

To delete a global policy, the following conditions must be satisfied:

- The policy must not be in-progress.
- The policy must be inactive.

> To delete a global policy

1. Click **Policies** in the title navigation bar.
2. Click the **Global Policies** tab.

A list of all available global policies appears.

3. Click the **Delete** () icon for the required policy.

If you do not see the **Delete** button, it is probably because you do not have the API Gateway Administrator role to delete a global policy, or the policy is in an **Active** state in API Gateway.

4. Click **Yes** in the confirmation dialog.

Copying a Global Policy

You must have the API Gateway's manage global policies functional privilege assigned.

A global policy can become quite complex, especially if it includes many policies. Instead of creating a new policy from scratch, it is sometimes easier to copy an existing policy that is similar to the one you need and edit the copy.

When you create a copy of a global policy, be aware that:

- When API Gateway creates a copy of a policy, the new copy of the policy is identical to the original one.
- Like all new policies, the copied policy is marked as **Inactive**.
- There is no expressed relationship between the copy and the original policy (that is, API Gateway does not establish any type of association between the two policies).

In general, a copied policy is no different from a policy that you create from scratch.

> To copy a global policy

1. Click **Policies** in the title navigation bar.
2. Click the **Global Policies** tab.

A list of all available global policies appears.

3. Click the **Copy** icon for the required policy.

If you do not see the **Copy** button, it is probably because you do not have the API Gateway Administrator role to create the copy of a global policy in API Gateway.

4. In the **Copy of Global Policy** dialog box, provide the required information for each of the displayed data fields:

Field	Description
Name	Name of the global policy. API Gateway automatically adds the name of the existing global policy to the Name field. You can change the name of the policy to suit your needs. But you cannot leave this field empty.
Description	The description for the global policy.

5. Click **Copy** to save the new policy.
6. Modify the new policy as necessary and then save it.

Activate the new policy when you are ready to put it into effect.


Exporting Global Policies

You must have the API Gateway's export assets functional privilege assigned.

Note:

For more information about exporting and importing global policies, see [“Overview” on page 624](#).

➤ **To export the global policies**

1. Click **Policies** in the title navigation bar.
2. Select **Global Policies**.
3. Click  to export a single policy.

Alternatively, you can select multiple APIs to be exported simultaneously by clicking the checkboxes adjacent to the names of the API.

Click  and select **Export** from the drop-down list.

The browser prompts you to either open or save the export archive.

4. Select the appropriate option and click **OK**.

Managing API-level Policies

The API-level policies apply to all APIs at the API level within an instance of API Gateway.

A policy at an API-level provides run-time governance capabilities to an API. The policy can be used to identify and authenticate consumers, validate digital signatures, capture performance measurements, and so on. Policies have one or more properties, which you can configure in a policy when you apply it to an API. For example, a policy that identifies consumers specifies one or more identifiers to identify the consumers who are trying to access the API.

The API level policies are categorised in the following stages:

- Threat protection - These policies can be viewed on the API details page of an API but can be managed only through the Policies > Global threat protection section and cannot be managed from the API details page.
- Transport
- Identify & Access
- Request Processing
- Routing
- Traffic Monitoring

- Response Processing
- Error Handling

Assigning a Policy to an API

Ensure that the API is in **Edit** mode before you start assigning a policy to the API.

> To assign a policy to an API

1. Click **APIs** in the title navigation bar.
2. Select the required API.
3. Click the **Policies** tab.
4. Select the policy stage and the required policy.

The policy is displayed in the infographic with its properties displayed in properties section.

5. Provide the properties for the selected policy.
6. Click **Save**.

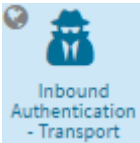
The policy is assigned to the API.



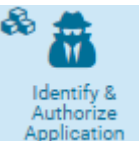
Viewing API Policy Details

The **Policies** tab on the API details page specifies the set of policies that are applied for that particular API.

The API can have a set of policies that are configured globally through a policy, or directly through a policy template or a scope-level policy.

The global policy in an API details page has each of its policies differentiated using a specific icon from the rest of the policies that are defined at the API-level and scope-level. The icon in the policy indicates the Inbound Authentication - Transport policy's enforcement level within an API:

Icon	Description
	Policy is applied from a global policy. This policy is applicable across all resources / methods / operations of all APIs.


Icon	Description
	<p>Policy is applied from a policy template or at the API definition. This policy is applicable across all resources / methods / operations of that particular API.</p>
	<p>Policy is applied for the API's scope. This policy is applicable across a set of resources / methods / operations of that particular API.</p>
	<p>Policy is applied through an active package definition. This policy is applicable across all resources / methods / operations of that particular API.</p>

Unlike the policy defined at API-level or scope-level, the policy defined as part of a global policy cannot be edited or deleted through the details page of an API.

➤ To view the policy details of an API

1. Click **APIs** in the title navigation bar.
A list of all registered APIs appears.
2. Select the required API.
3. Click the **Policies** tab.

The Infographic view displays policies configured for the API.

When this API is associated with one or more plans through active packages, a list of the **Identify and Authorize Application** policies and **Threat Protection** policies that are inherited from the corresponding plans and enforced on the API also appears. The inherited policies are differentiated using the package  icon. The **Identify and Authorize Application** policy, always, has the **Identification Type** set to API Key.

4. Click  .

A list of all available policies enforced on the API appears.

Modifying API Policy Details

Ensure that the API is in **Edit** mode before you modify a policy that is assigned to the API.

> To modify the policy details of an API

1. Click **APIs** in the title navigation bar.
2. Select the required API.
3. Click the **Policies** tab.
4. Select the policy stage, and the required policy.

The Infographic view displays policies configured for the API.

5. You can do one of the following:
 - Add more policies to the API. Select the policy stage and add the required policy. Configure the properties for the newly added policy as required.
 - Modify the already configured policy. Select the required policy and modify the properties as required.
 - Delete policies from the API. To remove a policy, click the **x** icon.
6. Click **Save**.

Managing Scope-level Policies

You can define policies at the API-level or scope-level for an API. API-level policies are processed for all incoming requests to the API. Scope-level policies are processed only for incoming requests that apply to a specific scope in the API. Any policy you specify at the API-level is overridden by the policy defined at the scope-level if the policies are the same. In contrast, the API-level policies will not affect the scope-level policies. But if there are policies applied at the global-level (through a global policy) for the API, then those policies will override every other policy configured at the API-level.

The scope-level policies for an API provide a granular enforcement of policies at the resource-level, method-level, or both for the REST API, or at the operation-level for the SOAP API.

Note:

Scope-level policies are not supported for OData APIs.

An API can have zero or more scope-level policies. When you define the scope-level policies for an API, keep the following points in mind:

- For a policy (for example, Identify and Authorize Application) that can appear only once in an API, if the same policy is already applied through the API details page, API Gateway prompts you with a warning message that the scope-level policy takes precedence over the API-level policy, and is enforced on the API at run-time.

- For a policy (for example, Monitor Service Level Agreement) that can appear multiple times in an API, if the same policy is already applied to the API through a global policy, API Gateway prompts you with a warning message that the global policy takes precedence over the scope-level policy, and is enforced on the API at run-time.
- If a resource or method or operation has the same policy (for example, Require HTTP / HTTPs) applied through different scopes, API Gateway prompts you with an error message and sets the focus to the conflicting policies. You must remove the required policy from the individual scope(s) to resolve the conflicts.

API Gateway supports scope-level policies only for the following stages:

- Identify and Access: All policies in this stage are supported.
- Request Processing: Only Data Masking policy in this stage is supported.
- Traffic Monitoring: All policies in this stage are supported.
- Response Processing: Only Data Masking policy in this stage is supported.
- Error Handling: Only Data Masking policy in this stage is supported.

For information on the usage scenarios of policies configured for the scopes of an API, see [“Example: Usage Scenarios of API Scopes”](#) on page 340.

Creating a Scope-level Policy

You create a policy for the API scope, to enforce the specific set of policies on a collection of resources, methods, or both, or operations that are associated to the scope. An API can have zero or more scope-level policies.

➤ To create a scope-level policy

1. Click **APIs** in the title navigation bar.

This displays a list of APIs available in API Gateway.

2. Click the name of the required API.

This opens the API details page.

3. Click **Edit**.

If the API is active, API Gateway displays a warning message to let you know that the API is active.

4. Click the **Policies** tab.

This displays a list of scopes and policies available in the API.

5. In the **API Scope** box, select the scope for that you want to create a policy.

6. In the Policy catalog section, click the chevron to expand the required policy stage.

This displays a list of policies that are classified under the particular stage.

7. In the expanded list of policies, select the policies that you want to associate with this scope. To select a policy, click the **Add (+)** icon next to the policy name. The selected policies are displayed in the **Infographic** section.

When you select the policies for the scope-level policy, keep in mind that the policies shown in the **Policy catalog** section are determined by the type of the displayed API. If you do not see a policy that you need, that policy is probably not compatible with this API.

Use the **Delete (X)** icon in any individual policy to remove that particular policy from the **Infographic** section.

8. In the Infographic section, do the following for each policy in the list:
 - a. Select the policy whose properties you want to examine or set.
 - b. In the Policy properties section, set the values for the policy's properties as necessary.

Note:

Required properties are marked with an asterisk.

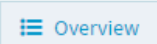
9. Click **Open in full-screen** to view the policy's properties in full-screen mode.

The **Open in full-screen** link is located in the upper right-corner of the **Policies** tab.

10. Set the properties of the displayed policy, and then click **OK**.

To exit out of full-screen mode, click the **Minimize** icon.

11. Click **Save** to create the new scope-level policy.

Click  to view the complete list of policies in the updated API. Activate the API, if it is not active, to put it into effect.

Viewing List of Scope-level Policies and Policy Details

The Infographic section displays the list of policies that are associated to a selected scope in the API's **Policies** tab.

> To view the scope-level policies and properties of a policy

1. Click **APIs** in the title navigation bar.

This displays a list of APIs available in API Gateway.

2. Click the name of the required API.

This opens the API details page.

3. Click the **Policies** tab.

This displays a list of scopes and policies available in the API.

4. In the **API Scope** box, select the scope whose policy details you want to examine.

5. In the Infographic section, do the following for each policy in the list:

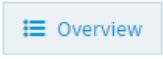
- a. Select the policy whose properties you want to examine.

- b. In the Policy properties section, examine the values for the policy's properties as required.

6. Click **Open in full-screen** to view the policy's properties in full-screen mode.

The **Open in full-screen** link is located in the upper right-corner of the **Policies** tab. Examine the properties of the displayed policy, and then click **OK**.

To exit out of full-screen mode, click the **Minimize** icon.

7. Click  **Overview** to view the complete list of policies in the updated API.

The **Overview** button is located in the lower right-corner of the Infographic section.

To exit the overview, click the **Close** icon.

Modifying Scope-level Policy Details

The API can have a set of policies that are configured globally through a global policy, or directly through a policy template, or a set of individual policies at the API-level or scope-level.

To customize the policy configurations at the scope-level, you need to apply the policies that are available for the API's scope, and then configure the properties of the individual policies to suit the needs of runtime behavior of that particular API.

You use the **Policies** tab to examine and modify the properties of a policy at the scope-level.

➤ To modify the properties of a scope-level policy

1. Click **APIs** in the title navigation bar.

This displays a list of APIs available in API Gateway.

2. Click the name of the required API.

This opens the API details page.

3. Click **Edit**.

If the API is active, API Gateway displays a warning message to let you know that the API is active.

4. Click the **Policies** tab.

This displays a list of scopes and policies available in the API.

5. In the **API Scope** box, select the scope whose policy details you want to modify.
6. On the Infographic section, modify the policy list and the policy's configuration properties as necessary.

Use the **Delete** (X) icon in any individual policy to remove that particular policy from the **Infographic** section.

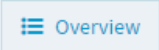
7. Click **Open in full-screen** to view the policy's properties in full-screen mode.

The **Open in full-screen** link is located in the upper right-corner of the **Policies** tab.

8. Modify the properties of the displayed policy, and then click **OK**.

To exit full-screen mode, click the **Minimize** icon.

9. When you have completed the required modifications for the scope-level policy, click **Save** to save the updated scope-level policy.

Click  to view the complete list of policies in the updated API. Activate the API, if it is not active, to put it into effect.

Deleting a Scope-level Policy

You delete a policy at the scope-level to remove the association between the policy and a scope.

When deleting a scope-level policy in the API details page, keep the following points in mind:

- When a scope is deleted from the API details, API Gateway removes the policies that were associated with the deleted scope.

> To delete a scope-level policy

1. Click **APIs** in the title navigation bar.

A list of all registered APIs appears.

2. Select the required API.

This opens the API details page.

3. Click **Edit**.

If the API is active, API Gateway displays a warning message to let you know that the API is active.

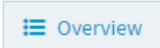
4. Click the **Policies** tab.

A list of scopes and policies available with the API appears.

5. In the **API Scope** box, select the scope whose policy you want to remove.

6. On the Infographic section, click the **x** icon in any individual policy to remove that particular policy from the scope.

7. When you have removed the policy, click **Save** to save the updated scope-level policy.

Click  to view the complete list of policies in the updated API. Activate the API, if it is not active, to put it into effect.

Managing Policy Templates

Important:

API Gateway's Standard Edition License does not support policy templates. You can create and manage policy templates only using the Advanced Edition License.

Policy templates are a set of policies that can be associated directly with an individual API. The direct association of the policy template with an API provides the flexibility to alter the policy's configurations to suit the individual API requirements.

To apply a policy template to an API, modify the details page of the API, and apply the selected policy template.

Creating a Policy Template

You must have the API Gateway's manage policy templates functional privilege assigned.

To create a policy template you must perform the following high-level steps:

1. **Create a new policy template:** During this step, you specify the basic details of the policy template.
2. **Configure the policies:** During this step, you associate one or more policies with the template, and assign values to each of the associated policy's properties.

> To create a policy template

1. Click **Policies** in the title navigation bar.

2. Click the **Policy Templates** tab.

A list of all available policy templates appears.

3. In the **Policies** page, click the **Create Policy Template** button.

This opens the Create Policy Template page with the default **Policy Details** tab.

4. In the Basic Information section, provide the required information for each of the displayed data fields:

Field	Description
Name	Name of the policy template.
Description	Description of the policy template.

5. Click **Continue to policy configuration**.

6. In the **Policy Configuration** tab, select the policies and configure the properties for each policy that you want API Gateway to execute when it applies this policy template. For details, see [“Associating Policies with a Policy Template” on page 566](#) and [“Configuring Properties for a Policy Template” on page 567](#).

7. Click **Save**.

Associating Policies with a Policy Template

You must have the API Gateway's manage policy templates functional privilege assigned.

The **Policy Configuration** tab on the Policy Template details page specifies the set of policy stages and the list of policies (applicable for each stage).

> To modify the list of policies of a policy template

1. Click **Policies** in the title navigation bar.

2. Click the **Policy Templates** tab.

A list of all available policy templates appears.

3. Select the required template.

The Policy Template details page appears.

4. Click **Edit**.
5. Click the **Policy Configuration** tab.

The policy template information is provided in the following sections:

- Policy catalog - Transport, Identify and Access, Request Processing, Routing, Traffic Monitoring, Response Processing, Error Handling
- Infographic - List of applied policies
- Policy properties - Collection of policy properties

6. In the Policy catalog section, click the chevron to expand the required policy stage.

This displays a list of policies that are classified under the particular stage.

7. In the expanded list of policies, select the policies that you want API Gateway to execute when it applies this policy template. To select a policy, click the **Add (+)** icon next to the policy name. The selected policies are displayed in the Infographic section.

Use the **Delete (X)** icon in any individual policy to remove that particular policy from the Infographic section.

8. To configure the properties for any new policies that you might have added to the Infographic section in the preceding steps or to make any necessary updates to the properties for existing policies in the policy template, see “[Configuring Properties for a Policy Template](#)” on page 567.
9. When the list of policies is complete and you have configured all of the properties for the policies correctly, click **Save** to save the updated policy template.

10. Click  to view the complete list of policies in the updated policy template.

The **Overview** button is located in the lower right-corner of the Infographic section. In addition, you can view the configured properties for the individual policies.

To exit the overview, click the **Close** icon.

Configuring Properties for a Policy Template

You must have the API Gateway's manage policy templates functional privilege assigned.

The **Policy Configuration** tab on the Policy Template details page specifies the list of policies that are applicable for each policy stage in the Policy catalog section. Each policy in the Infographic section has properties that you must set to configure the policy's enforcement behavior.

➤ **To configure the properties for a policy template**

1. Click **Policies** in the title navigation bar.

2. Click the **Policy Templates** tab.

A list of all available policy templates appears.

3. Select the required template.

The Policy Template details page appears.

4. Click **Edit**.

5. Click the **Policy Configuration** tab.

The policy template information is provided in the following sections:

- Policy catalog - Transport, Identify and Access, Request Processing, Routing, Traffic Monitoring, Response Processing, Error Handling
- Infographic - List of applied policies
- Policy properties - Collection of policy properties

6. In the Policy catalog section, click the chevron to expand the required policy stage.

This displays a list of policies that are classified under the particular stage.

7. In the Infographic section, do the following for each policy in the list:

a. Select the policy whose properties you want to examine or set.

b. In the Policy catalog section, set the properties as necessary.

Note:

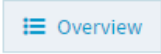
Required properties are marked with an asterisk.

8. Click **Open in full-screen** to view the policy's properties in full screen mode.

The **Open in full-screen** link is located in the upper right-hand corner of the **Policy Configuration** tab. Set the properties of the displayed policy, and then click **OK**.

To exit full screen mode, click the **Minimize** icon.

9. After you configure the properties for all of the policies in the Infographic section, click **Save** to save the updated policy template.

10. Click  to view the complete list of policies in the updated policy template.

The **Overview** button is located in the lower right-corner of the Infographic section.

To exit the overview, click the **Close** icon.

Viewing List of Policy Templates and Template Details

The **Policy Templates** tab displays a list of all available policy templates in API Gateway. Policy templates are listed alphabetically by name.

In addition to viewing the list of policy templates, you can also examine the details of a template, create a copy of the template, and delete a policy template in the **Policy Templates** tab.

➤ To view the policy template list and properties of a policy template

1. Click **Policies** in the title navigation bar.
2. Click the **Policy Templates** tab.

A list of all available policy templates appears. This tab provides the following information about each template:

Column	Description
Name	Name of the policy template.
Description	The description for the policy template.

You can also perform the following operations on a policy template:

- Create a new copy of the policy template.
 - Delete a policy template to remove it from API Gateway.
3. Select the required policy template.

The Policy Template details page appears. The policy template details are displayed in the following tabs:

- **Policy Details:** This tab contains a summary of basic information such as the name and description of the policy template.
- **Policy Configuration:** This tab contains the policy stages, applied policies, as well as the configuration details of individual policies.

Modifying Policy Template Details

You must have the API Gateway's manage policy templates functional privilege assigned.

You use the Policy Template details page to examine and modify the properties of a policy template.

» To modify the properties of a policy template

1. Click **Policies** in the title navigation bar.

2. Click the **Policy Templates** tab.

A list of all available policy templates appears.

3. Select the required template.

The Policy Template details page appears. The policy template details are displayed in the following tabs:

- **Policy Details:** This tab contains a summary of basic information such as name and description of the policy template.
- **Policy Configuration:** This tab contains the policy stages, applied policies, as well as the configuration details of individual policies.

4. Click **Edit**.

5. On the **Policy Details** tab, modify the basic properties of the policy as necessary.

6. On the **Policy Configuration** tab, modify the policy list and the policy's configuration properties as necessary.

7. When you have completed the required modifications on the Policy Template details page, click **Save** to save the updated policy template.

If update of a policy template fails, API Gateway displays a pop-up style error message.

8. Click **Overview** to view the complete list of policies in the updated policy template.

The **Overview** button is located in the lower right-corner of the Infographic section.

To exit the overview, click the **Close** icon.

Deleting a Policy Template

You must have the API Gateway's manage policy templates functional privilege assigned.

You delete a policy template to remove it from API Gateway permanently.

> To delete a policy template

1. Click **Policies** in the title navigation bar.
2. Click the **Policy Templates** tab.
A list of all available policy templates appears.
3. Click the **Delete** (🗑️) icon for the required template.
4. Click **Yes** in the confirmation dialog.

Copying a Policy Template

You must have the API Gateway's manage policy templates functional privilege assigned.

A policy template can become quite complex, especially if it includes many policies. Instead of creating a new policy template from scratch, it is sometimes easier to copy an existing template that is similar to the one you need and edit the copy.

When you create a copy of a policy template, be aware that:

- When API Gateway creates a copy of a policy template, the new copy of the policy template is identical to the original one.
- There is no expressed relationship between the copy and the original policy (that is, API Gateway does not establish any type of association between the two policy templates).

In general, a copied policy template is no different from a policy template that you create from scratch.

> To copy a policy template

1. Click **Policies** in the title navigation bar.
2. Click the **Policy Templates** tab.
A list of all available policy templates appears.
3. Click the **Copy** icon for the required template.
4. In the **Copy of Policy Template** dialog box, provide the required information for each of the displayed data fields:

Field	Description
Name	Name of the policy template.

Field	Description
	API Gateway automatically adds the name of the existing policy template to the Name field. You can change the name of the template to suit your needs. But you cannot leave this field empty.
Description	The description for the policy template.

5. Click **Copy** to save the new policy template.
6. Modify the new policy template as necessary and then save it.

Applying a Policy Template on the API Details Page

You must have the API Gateway's manage APIs functional privilege assigned.

The **Policies** tab on the API details page specifies the set of policies that API Gateway will execute when an application requests access to that particular API.

The API can have a set of policies that are applied through a global policy, through a policy template, through a scope-level policy, and through API-level policies.

To customize the policy configurations for an API using a policy template, you need to apply the template (containing a set of policies), and then configure the properties of the individual policies to suit the runtime requirements for that API.

➤ To apply a policy template on the API details page

1. Click **APIs** in the title navigation bar.

A list of all registered APIs appears.

2. Select the required API.
3. Click **Edit**.
4. Click the **Policies** tab.

The API's policy information is provided in the following sections:

- Policy stages - Threat Protection, Transport, Identify and Access, Request Processing, Routing, Traffic Monitoring, Response Processing, Error Handling
- Infographic - List of applied policies
- Policy properties - Collection of policy properties

5. Click **Apply template** located in the lower right-corner of the **Infographic** section.

This opens the **Apply template** dialog box.

6. In the **Template chooser**, select one or more policy templates that you want to apply to the API.

You can choose to display the details of an individual policy template by clicking the **Info** icon. This option populates the list of policies that are defined in the particular template.

7. Select one or more policy templates that you want API Gateway to execute at run-time.
8. Click **Next**.

You must have at least one template selected to use the **Next** button.

9. In the **Apply Templates to API** wizard, review the list of policies and the configuration details of the associated policies.
 - If necessary, you can click **Previous** to return to the **Template chooser** wizard and change your template selections.
 - If at any time you wish to abandon all your changes and return to the **Policies** tab, click **Cancel**.
10. Click **Apply**.

If you have one or more policy conflicts, API Gateway displays the conflicting/incompatible policies with a **Conflict** icon. You can choose to resolve the policy conflicts and do a **Apply**, or simply continue to **Apply with conflicts**.

If you choose the continue with conflicts, API Gateway sets the focus on the conflicting policies with Conflict (▲) icon displayed next to the policy names in the Infographic section and the corresponding policy stages.

API Gateway will redirect you to the **Policies** tab. The newly applied policy template comprising a set of policies and the policy properties is displayed in the Infographic section.

11. After you apply the required policy templates, click **Save** to save the updated API.

Post-requisites:

Activate the API when you are ready to put it into effect.

Modifying a Policy Template on the API Details Page

You must have the API Gateway's manage policy templates functional privilege assigned.

The **Policies** tab on the API details page specifies the set of policies that API Gateway executes when an application requests access to that particular API.

The API can have a set of policies that are applied through a global policy, through a policy template, through a scope-level policy, and through API-level policies.

➤ **To modify the details of a policy template on the API details page**

1. Click **APIs** in the title navigation bar.

A list of all registered APIs appears.

2. Select the required API.

3. Click **Edit**.

If the API is active, API Gateway displays a warning message to let you know that the API is active.

4. Click the **Policies** tab.

The API's policy information is provided in the following sections:

- Policy catalog - Threat Protection, Transport, Identify and Access, Request Processing, Routing, Traffic Monitoring, Response Processing, Error Handling
- Infographic - List of applied policies
- Policy properties - Collection of policy properties

5. In the Infographic section, do the following for each policy in the list:

- a. Select the policy whose properties you want to examine or set.
- b. In the Policy properties section, set the properties as necessary.

Note:

Required properties are marked with an asterisk.

- c. Use the **Delete** (X) icon in any individual policy to remove that particular policy from the Infographic section.

6. Click **Open in full-screen** to view policy properties in full screen mode.

The **Open in full-screen** button is located in the upper right-hand corner of the **Policy Configuration** tab.

7. Set the properties of the displayed policy, and then click **OK**.

To exit full screen mode, click the **Minimize** icon.

8. Click **Save** to save the updated API.

Activate the API, if it is not active, to put it into effect.

Saving Policy Definition of an API as Policy Template

You must have the API Gateway's manage policy templates functional privilege assigned.

The **Policies** tab on the API details page specifies the set of policies that API Gateway will execute when an application requests access to that particular API.

The API can have a set of policies that are applied through a global policy, through a policy template, through a scope-level policy, and through API-level policies.

You can save the current policy definition of an API as a new policy template. At a later time, you can reuse this policy template in other APIs. For more information, see [“Applying a Policy Template on the API Details Page” on page 572](#).

➤ To save policy definition as policy template

1. Click **APIs** in the title navigation bar.

A list of all registered APIs appears.

2. Select the required API.
3. Click the **Policies** tab.

The API's policy information is provided in the following sections:

- Policy catalog - Threat Protection, Transport, Identify and Access, Request Processing, Routing, Traffic Monitoring, Response Processing, Error Handling
- Infographic - List of applied policies
- Policy properties - Collection of policy properties

4. Click **Save as template** located in the lower right-hand corner of the Infographic section.
5. In the **Save as template** dialog box, provide the required information for each of the displayed data fields:

Field	Description
Name	Name of the policy template.
Description	Description of the policy template.

6. Click **Save**.

Supported Alias and Policy Combinations

API Gateway provides a set of aliases whose runtime-specific environment variables can be used in configuring the policy routing endpoints, routing rules, endpoint connection properties, and outbound authentication tokens. The types of aliases whose properties you can use for the policy configurations are:

- Simple alias
- Endpoint alias
- HTTP transport security alias
- SOAP message security alias
- webMethods IS Service alias
- XSLT Transformation alias

Not all policies support the full set of aliases that are available in API Gateway. Some aliases are applicable only with certain policies and for certain policy parameters. For example, a *Simple* alias applies to the routing and traffic monitoring policies, whereas an *Endpoint* alias applies only to the routing policies. When you define a Straight Through Routing policy with a simple alias, the alias property is defined using the Endpoint URI field. When you define the same Straight Through Routing policy with an endpoint alias, the alias property is defined using a set of fields - Endpoint URI, SOAP Optimization Method, HTTP Connection Timeout, Read Timeout, Pass WS-Security Headers, and Keystore Alias.

The following table identifies the policies and policy parameters that each alias type supports:

Simple Alias

Policy Name	Policy Parameter Name
Straight Through Routing	In the Straight Through Routing definition: <ul style="list-style-type: none"> ■ Endpoint URI
Content-based Routing	In the default and custom Route To rule definitions: <ul style="list-style-type: none"> ■ Endpoint URI
Context-based Routing	In the default and custom Route To rule definitions: <ul style="list-style-type: none"> ■ Endpoint URI
Load Balancer Routing	In the Route To rule definition: <ul style="list-style-type: none"> ■ Endpoint URI
Dynamic Routing	In the default and custom Route To rule definitions: <ul style="list-style-type: none"> ■ Endpoint URI

Policy Name	Policy Parameter Name
Log Invocation	In the Email Destination section: <ul style="list-style-type: none"> ■ Email Address
Monitor Service Performance	In the Email Destination section: <ul style="list-style-type: none"> ■ Email Address
Monitor Service Level Agreement	In the Email Destination section: <ul style="list-style-type: none"> ■ Email Address
Throttling Traffic Optimization	In the Email Destination section: <ul style="list-style-type: none"> ■ Email Address

Endpoint Alias

Policy Name	Policy Parameter Name
Straight Through Routing	In the Straight Through Routing definition: <ul style="list-style-type: none"> ■ Endpoint URI ■ SOAP Optimization Method (Applicable only for SOAP APIs) ■ HTTP Connection Timeout ■ Read Timeout ■ Pass WS-Security Headers (Applicable only for SOAP APIs) ■ Keystore Alias ■ Key Alias
Content-based Routing	In the default and custom Route To rule definitions: <ul style="list-style-type: none"> ■ Endpoint URI ■ SOAP Optimization Method (Applicable only for SOAP APIs) ■ HTTP Connection Timeout ■ Read Timeout ■ Pass WS-Security Headers (Applicable only for SOAP APIs) ■ Keystore Alias ■ Key Alias
Context-based Routing	In the default and custom Route To rule definitions:

Policy Name	Policy Parameter Name
	<ul style="list-style-type: none">■ Endpoint URI■ SOAP Optimization Method (Applicable only for SOAP APIs)■ HTTP Connection Timeout■ Read Timeout■ Pass WS-Security Headers (Applicable only for SOAP APIs)■ Keystore Alias■ Key Alias

Load Balancer Routing In the **Route To** rule definition:

- Endpoint URI
 - SOAP Optimization Method (Applicable only for SOAP APIs)
 - HTTP Connection Timeout
 - Read Timeout
 - Pass WS-Security Headers (Applicable only for SOAP APIs)
 - Keystore Alias
 - Key Alias
-

Dynamic Routing In the default and custom **Route To** rule definitions:

- Endpoint URI
 - SOAP Optimization Method (Applicable only for SOAP APIs)
 - HTTP Connection Timeout
 - Read Timeout
 - Pass WS-Security Headers (Applicable only for SOAP APIs)
 - Keystore Alias
 - Key Alias
-

HTTP Transport Security Alias

Policy Name	Policy Parameter Name
Outbound Authentication - Transport	In the Authentication scheme: <ul style="list-style-type: none">■ Alias

SOAP Message Security Alias (Applicable only for SOAP APIs)

Policy Name	Policy Parameter Name
Outbound Authentication - Message	In the Authentication scheme: <ul style="list-style-type: none"> ■ Alias

webMethods IS Service Alias

Policy Name	Policy Parameter Name
Invoke webMethods IS (Request Processing)	webMethods IS Service Alias
Invoke webMethods IS (Response Processing)	webMethods IS Service Alias

XSLT Transformation Alias

Policy Name	Policy Parameter Name
Request Transformation (Request Processing)	Transformation Configuration <ul style="list-style-type: none"> ■ Payload Transformation ■ XSLT Transformation alias
Response Transformation (Response Processing)	Transformation Configuration <ul style="list-style-type: none"> ■ Payload Transformation ■ XSLT Transformation alias

6 Aliases

■ Overview	582
■ Creating a Simple Alias	582
■ Creating an Endpoint Alias	583
■ Creating an HTTP Transport Security Alias	586
■ Creating a SOAP Message Security Alias	589
■ Creating a webMethods Integration Server Service Alias	592
■ Creating an XSLT Transformation Alias	593

Overview

An alias in API Gateway holds environment-specific property values that can be used in policy routing configuration. The aliases can be referred to in routing endpoints, routing rules, endpoint connection properties, and outbound authentication tokens instead of providing a real value. The corresponding alias value is substituted in place of an alias name during run-time. Thus the same alias can be referred to in multiple policies and the change in a particular alias would affect all the policy properties in which it is being referred. When an API is exported and imported to a different environment, you can update the alias values specific to the environment instead of updating the policy with environment specific values.

Not all policies support the full set of aliases that are available in API Gateway. Some aliases are applicable only with certain policies and for certain policy parameters. For details, see [“Supported Alias and Policy Combinations” on page 576](#).

You can create six types of alias:


- Simple alias
- Endpoint alias
- HTTP transport security alias
- SOAP message security alias
- webMethods IS Service alias
- XSLT Transformation alias

Creating a Simple Alias

You must have the API Gateway's manage aliases functional privilege assigned to perform this task.

A simple alias holds simple key property values. The name of the alias can be used in the configuration of the properties of a routing policy or an email destination for the Log Invocation, Monitor Service Level Agreement, Monitor Service Performance, and Throttling Traffic Optimization policies.

> To create a simple alias

1. Expand the menu options icon , in the title bar, and select **Aliases**.
2. Click **Create alias**.
3. In the Basic information section, provide the following information:

Field	Description
Name	Name of the alias.
Type	Select Simple alias .
Description	Description of the alias.

4. Click **Technical information** and specify a value in the **Default value** field.
5. Specify a stage, if you want the alias to be applicable to a specific stage.
6. Click **Save**.


For example, if you create a simple alias called `hostname` with value `dev.com` and in the routing policy you can specify the `https://dev.com/v2endpoint` as `https://{hostname}/v2`. At runtime invocation the simple alias `{hostname}` will be replaced with its value `dev.com`.

Creating an Endpoint Alias

You must have the API Gateway's `manage aliases` functional privilege assigned to perform this task.

An endpoint alias stores the endpoint value along with additional properties such as connection timeout, read timeout, whether to pass security headers or not, keystore alias, key alias, and so on.

➤ To create an endpoint alias

1. Expand the menu options icon , in the title bar, and select **Aliases**.
2. Click **Create alias**.
3. In the Basic information section, provide the following information:

Field	Description
Name	Name of the alias.
Type	Select Endpoint alias .
Description	Description of the alias.

4. Click **Technical information** and provide the following information:

Field	Description
Optimization technique	<p>This is applicable only for a SOAP API.</p> <p>Specify the optimization technique for the SOAP request received. Select any one of the following:</p> <ul style="list-style-type: none"> ■ None. This is the default value. API Gateway does not use any optimization method to parse the SOAP requests to the API. ■ MTOM. Indicates that API Gateway expects to receive a request with a Message Transmission Optimization Mechanism (MTOM) attachment and forwards the attachment to the native service. ■ SWA. Indicates that API Gateway expects to receive a SOAP with Attachment (SWA) request and forwards the attachment to the native service.
Pass WS-Security Headers	Passes the security header.
Endpoint URI	Specify the default URI or components of the URI such as service name.
Connection timeout	<p>Specify the time interval (in seconds) after which a connection attempt times out.</p> <p>The precedence of the Connection Timeout configuration is as follows:</p> <ol style="list-style-type: none"> a. If you specify a value for the Connection timeout field in routing endpoint alias, then the Connection timeout value specified in the Endpoint alias section takes precedence over the timeout values defined at the API level and the global level. b. If you specify a value 0 for the Connection timeout field in routing endpoint alias, then API Gateway uses the value specified in the Connection timeout field in the routing protocol processing step of an API. The Read Timeout value specified at an API level takes precedence over the global configuration. c. If you specify a value 0 or do not specify a value for the Connection timeout field in the routing protocol processing step at the API level or specify a value 0 at an alias level, then API Gateway uses the value specified in this <code>pg.endpoint.connectionTimeout</code> property.

Field	Description
	<p>d. If you do not specify any value for <code>pg.endpoint.connectionTimeout</code>, then API Gateway uses the default value of 30 seconds.</p>
Read timeout	<p>Specify the time interval (in seconds) after which a socket read attempt times out.</p> <p>The precedence of the Read Timeout configuration is as follows:</p> <ol style="list-style-type: none"> If you specify a value for the Read timeout field in routing endpoint alias, then the Read timeout value specified in the Endpoint alias section takes precedence over the timeout values defined at the API level and the global level. If you specify a value 0 for the Read timeout field in routing endpoint alias, then API Gateway uses the value specified in the Read Timeout field in the routing protocol processing step of an API. The Read Timeout value specified at an API level takes precedence over the global configuration. If you specify a value 0 or do not specify a value for the Read timeout field in the routing protocol processing step at the API level or specify a value 0 at an alias level, then API Gateway uses the value specified in this <code>pg.endpoint.readTimeout</code> property. If you do not specify any value for <code>pg.endpoint.readTimeout</code>, then API Gateway uses the default value of 30 seconds.
Keystore alias	<p>Specifies the keystore alias configured in API Gateway. This value (along with the value of Client Certificate Alias) is used for performing SSL client authentication.</p> <p>Lists all available keystores. If you have not configured any keystore, the list is empty.</p>
Key alias	<p>Specifies the alias for the private key, which must be stored in the keystore specified by the keystore alias.</p>
Truststore alias	<p>Specifies the alias for the truststore that contains the list of CA certificates that API Gateway uses to validate the trust relationship with the native API.</p> <p>If you do not configure any truststore alias, it implies that API Gateway does not validate the certificates provided by native APIs.</p>
Stage	<p>Specify a stage, if you want the alias to be applicable to a specific stage.</p>

5. Click **Save**.


Creating an HTTP Transport Security Alias

You must have the API Gateway's manage aliases functional privilege assigned to perform this task.

An HTTP Transport security alias contains transport level security information required while accessing the native API. Transport level security that are supported in API Gateway outbound are as follows:

- HTTP Basic authentication
- OAuth2 authentication
- NTLM authentication
- Kerberos authentication
- JWT authentication

> To create an HTTP transport secure alias

1. Expand the menu options icon , in the title bar, and select **Aliases**.
2. Click **Create alias**.
3. In the Basic information section, provide the following information:

Field	Description
Name	Name of the alias.
Type	Select HTTP transport security alias .
Description	Description of the alias.

4. Click **Technical information** and provide the following information:

Field	Description
Authentication scheme	Specify the type of authentication you want to use while communicating with the native API. Select one of the following: <ul style="list-style-type: none"> ■ Basic. Uses basic authentication (user name and password). ■ Kerberos. Uses Kerberos authentication.

Field	Description
	<ul style="list-style-type: none"> ■ NTLM. Uses NTLM authentication. ■ OAuth2. Uses OAuth2 authentication. ■ JWT. Uses JWT authentication.
For the Authentication type Basic , authenticate using the following:	
Custom credentials	<p>Specifies the values provided in the policy required to access the native API.</p> <p>Provide the following information:</p> <ul style="list-style-type: none"> ■ Username. Specify a username to access the native API. ■ Password. Specify a password to access the native API. ■ Domain. Specify a domain to access the native API.
Incoming HTTP basic auth credentials	No properties required. Considers the incoming HTTP basic authentication credentials.
For Authentication type Kerberos , authenticate using any of the following:	
Custom credentials	<p>Specifies the values provided in the policy required to obtain the Kerberos token to access the native API.</p> <p>Provide the following information:</p> <ul style="list-style-type: none"> ■ Client principal. A valid client LDAP user name. ■ Client password. A valid password of the client LDAP user. ■ Service principal. A valid Service Principal Name (SPN). The specified value is used by the client to obtain a service ticket from the KDC server. ■ Service principal nameform. Specifies the format in which you want to specify the principal name of the service that is registered with the principal database. Select one of the following: <ul style="list-style-type: none"> ■ Username. Represents the principal name as a named user defined in LDAP used for authentication to the KDC. ■ Hostbased. Represents the principal name using the service name and the host name, where host name is the host computer.
Delegate incoming credentials	Specifies the values provided in the policy required by the API providers to select whether to delegate the incoming Kerberos token or act as a normal client.

Field	Description
	<p>Provide the following information:</p> <ul style="list-style-type: none"> ■ Client principal. A valid client LDAP user name. ■ Client password. A valid password of the client LDAP user. ■ Service principal. A valid Service Principal Name (SPN). The specified value is used by the client to obtain a service ticket from the KDC server. ■ Service principal nameform. Specifies the format in which you want to specify the principal name of the service that is registered with the principal database. Select one of the following: <ul style="list-style-type: none"> ■ Username. Represents the principal name as a named user defined in LDAP used for authentication to the KDC. ■ Hostbased. Represents the principal name using the service name and the host name, where host name is the host computer.
Incoming HTTP basic auth credentials	<p>Specifies the incoming HTTP basic authentication credentials in the transport header of the incoming request for client principal and client password.</p> <p>Provide the following information:</p> <ul style="list-style-type: none"> ■ Service principal. A valid Service Principal Name (SPN). The specified value is used by the client to obtain a service ticket from the KDC server. ■ Service principal nameform. Specifies the format in which you want to specify the principal name of the service that is registered with the principal database. Available values are: <ul style="list-style-type: none"> ■ Username. Represents the principal name as a named user defined in LDAP used for authentication to the KDC. ■ Hostbased. Represents the principal name using the service name and the host name, where host name is the host computer.
Incoming kerberos credentials	<p>No properties required. Considers the incoming kerberos credentials.</p>
<p>For Authentication type NTLM, authenticate using any of the following:</p>	
Custom credentials	<p>Specifies the credentials that are required for the NTLM handshake.</p>

Field	Description
	Provide the following information: <ul style="list-style-type: none"> ■ Username. Name of a consumer who is available in the Integration Server on which API Gateway is running. ■ Password. A valid password of the consumer. ■ Domain. The domain used by the server to authenticate the consumer.
Incoming HTTP basic auth credentials	No properties required. Considers the incoming HTTP basic authentication credentials.
Transparent	No properties required.
For the Authentication type OAuth2 , authenticate using any of the following:	
Custom credentials	Specifies the OAuth2 token value that would be added as bearer token in the transport header while accessing the native API.
Incoming OAuth token	Considers the incoming OAuth token to access the native API.
For Authentication type JWT , authenticate using any of the following:	
Incoming JWT	Considers the incoming JSON web token to access the native API.

5. Specify a stage, if you want the alias to be applicable to a specific stage.
6. Click **Save**.

Creating a SOAP Message Security Alias

You must have the API Gateway's manage aliases functional privilege assigned to perform this task.

A SOAP message security alias contains message level security information that is requires to access the native API. If the native service is enforced with any WS security policy, API Gateway enforces those policies in the outbound request while accessing the native API using the configuration parameters specified in the alias.

➤ To create SOAP message secure alias

1. Expand the menu options icon , in the title bar, and select **Aliases**.
2. Click **Create alias**.

3. In the Basic information section, provide the following information:

Field	Description
Name	Name of the alias.
Type	Select SOAP message secure alias .
Description	Description of the alias.

4. Click **Technical information** and provide the following information:

Field	Description
Authentication scheme	<p>Specify the type of authentication scheme you want to use to authenticate the client.</p> <p>Available values are:</p> <ul style="list-style-type: none">■ None. Does not use any authentication types to authenticate the client.■ WSS Username. Generates a WSS username token and sends it in the soap header to the native API.■ Kerberos. Fetches a Kerberos token and sends it to the native API.■ SAML. Fetches a SAML token and sends it to the native API.

For Authentication scheme **None**. Does not require any properties.

For Authentication type **WSS Username**, authenticate using any of the following:

Custom credentials	<p>Specifies the values provided in the policy to be used to obtain the WSS username token to access the native API.</p> <p>Provide the following information:</p> <ul style="list-style-type: none">■ Username. Specifies a username used to generate the WSS username token.■ Password. Specifies the password used to generate the WSS username token.
---------------------------	--

For Authentication type **Kerberos**, authenticate using any of the following:

Custom Credentials	<p>Uses the Basic authentication credentials coming in the transport header of the incoming request for client principal and client password.</p>
---------------------------	---

Provide the following information:

Field	Description
	<ul style="list-style-type: none"> ■ Client principal. A valid client LDAP user name. ■ Client password. A valid password of the client LDAP user. ■ Service principal. A valid Service Principal Name (SPN). The specified value is used by the client to obtain a service ticket from the KDC server. ■ Service principal nameform. Specifies the format in which you want to specify the principal name of the service that is registered with the principal database. Select one of the following: <ul style="list-style-type: none"> ■ Username. Represents the principal name as a named user defined in LDAP used for authentication to the KDC. ■ Hostbased. Represents the principal name using the service name and the host name, where host name is the host computer.
Delegate incoming credentials	<p>Specifies the values provided in the policy to be used by the API providers to select whether to delegate the incoming Kerberos token or act as a normal client.</p> <p>Provide the following information:</p> <ul style="list-style-type: none"> ■ Client principal. A valid client LDAP user name. ■ Client password. A valid password of the client LDAP user. ■ Service principal. A valid Service Principal Name (SPN). The specified value is used by the client to obtain a service ticket from the KDC server. ■ Service principal nameform. Specifies the format in which you want to specify the principal name of the service that is registered with the principal database. Available values are: <ul style="list-style-type: none"> ■ Username. Represents the principal name as a named user defined in LDAP used for authentication to the KDC. ■ Hostbased. Represents the principal name using the service name and the host name, where host name is the host computer.
Incoming HTTP basic auth credentials	<p>Specifies the incoming HTTP basic authentication credentials to access the native API.</p> <p>Provide the following information:</p>

Field	Description
	<ul style="list-style-type: none"> ■ Service principal nameform. Specifies the format in which you want to specify the principal name of the service that is registered with the principal database. Select one of the following: <ul style="list-style-type: none"> ■ Username. Represents the principal name as a named user defined in LDAP used for authentication to the KDC. ■ Hostbased. Represents the principal name using the service name and the host name, where host name is the host computer.
For Authentication type SAML	
SAML issuer configuration	<p>Specifies the SAML issuer configuration that is used by the API Gateway to fetch the SAML token which is then added in the SOAP header and sent to the native API.</p> <p>This field is visible and required only if you have configured a SAML issuer in Administration > Security > SAML issuer section.</p>
Signing configurations	
Keystore alias	Specify the keystore that needs to be used by API Gateway while sending the request to the native API. A keystore is a repository of private key and its corresponding public certificate.
Key alias	The key alias is the private key that is used sign the request sent to the native API.
Encryption configurations	
Truststore alias	Select the truststore to be used by API Gateway when sending the request to the native API. Truststore is a repository that holds all the trusted public certificates.
Certificate alias	Select the certificate from the truststore that is used to encrypt the request that is sent to the native API.
Stage	Specify a stage, if you want the alias to be applicable to a specific stage.


5. Click **Save**.

Creating a webMethods Integration Server Service Alias

You must have the API Gateway's manage aliases functional privilege assigned to perform this task.

A webMethods Integration Server service alias holds the IS service value. The name of the alias can be used to invoke the Invoke webMethods IS policy for request and response processing.

➤ **To create a webMethods IS service alias**

1. Expand the menu options icon , in the title bar, and select **User management**.
2. Click **Create alias**.
3. In the Basic information section, provide the following information:

Field	Description
Name	Name of the alias.
Type	Select webMethods IS Service alias .
Description	Description of the alias.

4. Click **Technical information** and provide the following information:

Field	Description
Service name	Specify the IS service name. Note: The IS service must be available in the Integration Server, to which the aliases are deployed.
Comply to IS Spec (pubapiGatewayInvokeISServiceSpecifications)	Select Comply to IS Spec , if you want the input and the output parameters to comply to the IS Spec specified.
Stage	Specify a stage, if you want the alias to be applicable to a specific stage.


5. Click **Save**.

Creating an XSLT Transformation Alias

You must have the API Gateway's manage aliases functional privilege assigned to perform this task.

An XSLT transformation alias holds a list of XSLT style sheets. The name of the alias can be used in the XSLT Transformation policies for request and response processing.

➤ **To create a transformation alias**

1. Expand the menu options icon , in the title bar, and select **User management**.
2. Click **Create alias**.
3. In the Basic information section, provide the following information:

Field	Description
Name	Name of the alias.
Type	Select XSLT Transformation alias .
Description	Description of the alias.

4. Click **Technical information** and browse and select an XSLT style sheet in the **Select transformation file** field.
5. Specify a stage, if you want the alias to be applicable to a specific stage.
6. Click **Save**.

7 Applications

■ Overview	596
■ Creating an Application	597
■ Viewing List of Applications and Application Details	605
■ Regenerating API Access Key	606
■ Modifying Application Details	606
■ Registering an API with Consumer Applications from API Details Page	607
■ Registering APIs with Consumer Applications from Application Details Page	607
■ Suspending an Application	608
■ Activating a Suspended Application	608

Overview

An application defines the precise identifiers by which messages from a particular application is recognized at run time. The identifiers can be, for example, user name in HTTP headers, a range of IP addresses, such that API Gateway can identify or authenticate the applications that are requesting an API.

The ability of API Gateway to relate a message to a specific application enables it to:

- Control access to an API at run time (that is, allow only authorized applications to invoke an API).
- Monitor an API for violations of a Service-Level Agreement (SLA) for a specified application.
- Indicate the application to which a logged transaction event belongs.

An application has the following attributes for specifying the identifiers:

- IP address, which specifies one or more IP addresses that identify requests from a particular application. Example: `192.168.0.10`

This attribute is queried when the Identify and Authorize Application policy is configured to identify applications using IP address.

- Claims set, which specifies one or more claims that identify requests from a particular application. The claims are a set of name-value pairs that provide sufficient information about the application. Example: `sub = Administrator`.

This attribute is queried when the Identify and Authorize Application policy is configured to identify applications using a JWT token or an OpenID token.

- Client certificate, which specifies the X.509 certificates that identify requests from a particular application.

This attribute is queried when the Identify and Authorize Application policy is configured to identify the applications by a client certificate.

- Identification token, which specifies the host names, user names or other distinguishing strings that identify requests from a particular application.

This attribute is queried when the Identify and Authorize Application policy action is configured to identify applications by host name, token, HTTP user name, and WSS user name.

You can configure various authentication strategies to authenticate an incoming request to the application. You can create multiple strategies authorized by an API for an application. These strategies provide multiple authentication mechanisms or multiple authorization servers for a single authentication scheme. For example, in case of OAuth authentication scheme, you want the application to support both OKTA and PINGFederate or OKTA with multiple tenants. This can be configured as OAuth strategy for the application.

If you have the **Manage Application** functional privilege assigned, you can create and manage applications, and register applications with the APIs.

These are the high level stages of managing and using an application:

1. API developers request the API Gateway administrators to create an application for access as per the required identification criteria.
2. API Gateway provider or administrator validates the request and creates a new application, there by provisioning the application specific access tokens (API access key and OAuth credentials).
3. API Developer, upon finding a suitable API, sends a request to API Gateway for consumption by providing the application details.
4. After validating the request, API Gateway provider or administrator associates the application with the API. Keys are generated for applications and not for every API that the application consumes.

Note:

The approval process, if any, is handled by the requesting application and not handled by API Gateway.

5. The API developer can then use the application with the proper identifier (such as the access key or identifier) to access the API.

API key expiration date

An API Gateway application has an optional expiration date for its API key. When the API access key expires, the application cannot be identified. The API Gateway Administrator can configure the **apiKeyExpirationPeriod** parameter from the **General > Extended settings** page. If the expiration date is not specified, then the API key never expires.

Suspended Applications

You can suspend applications so as to disable the identification of requests temporarily. If a suspended application is identified while processing a request the request is rejected with HTTP 403 (Forbidden) error. The response body has the following content:

```
Application has been identified but it is currently suspended. Please contact the API Gateway administrator for further details.
```

You can resume the suspended applications to enable the identification again.


Creating an Application

You must have the API Gateway's manage applications functional privilege assigned to perform this task.

You can create an application from the Applications page.

> To create an application

1. Click **Applications** in the title navigation bar.

2. Click **Create application**.
3. Provide the following information in the Basic information section:
 - **Name.** Type a name for the application.
 - **Version.** Version of the application. By default it is 1.0 but can be modified to a required value.
 - **Team.** *This field is visible when the enableTeamWork is set to true in the Administration > General > Extended Settings section.* Team to which the application must be assigned to. You can select more than one team. To remove a team, click the  icon next to the team.
 - **Description.** Type a description of the application.
 - **Requestor comment.** *This field is visible when Approval configuration for Create application is enabled in the Administration > General > Approval Configuration > Create application section.*

4. Click **Continue to Identifiers >**.

Alternatively, you can click **Identifiers** in the left navigation panel.

You can save the application by clicking **Save** at this stage and add the Identifiers and APIs at a later time.

5. Provide the following information in the Identifiers section:

Field	Description
IP address range	<p>Provide the IP address range or range of trusted IPv4 or IPv6 addresses that identify requests from a particular application.</p> <p>You can add more range options by clicking +Add and adding the required information.</p>
Partner identifier	<p>Specifies the third-party partner's identity.</p> <p>The specified partner can access the APIs if business-to-business communication between trading partners is enabled and where partners can invoke the exposed APIs to exchange information.</p> <p>For example, if you have enabled business-to-business communication between trading partners using APIs, partners can invoke the exposed APIs to exchange information. These APIs are available by associating Trading Networks with API Gateway. A partner can access the APIs that appear in the Partner Profiles and associated Partner Groups page. Once APIs are added as part of Partner, respective application is created in API Gateway with name partnerName Application and appropriate Partner ID.</p>

Field	Description
	<p>For more details on information on enabling business-to-business communication between trading partners and required configuration, see <i>webMethods Trading Networks Administrator's Guide</i></p> <p>Note: No identification or enforcement of application happens in API Gateway using this identifier.</p>
Client certificates	<p>Click Browse and select the client certificate or certificate chain to be uploaded. The client certificate specifies the X.509 certificates that requests from a particular application.</p> <p>Note: API Gateway supports .cer and .pem certificates for identifying consumer applications.</p> <p>You can add multiple certificates by clicking +Add</p>
Claims	<p>Provide a set of claims for the JWT and OpenID clients.</p> <p>A claim is a unique identifying information that identify requests from a particular consumer application. The claim set is identified by a unique Name and is defined as a name-value pair that consists of a Claim name and a Claim value.</p> <p>You can add more claims and claims sets by clicking +Add and adding the required information.</p>
Header key	<p>Specify the HTTP header key to identify the requests from an application.</p>
Header value	<p>Specify the HTTP header value to identify the requests from an application.</p> <p>You can add multiple header key and value by clicking +Add</p>
Other identifiers	<p>Select one of the options to identify requests from a particular application and provide the required value:</p> <ul style="list-style-type: none"> ■ Hostname. The host name to identify requests from an application. ■ Payload identifier. The payload identifier to identify requests from an application. ■ Team. The team to identify requests from an application. A team can contain one or more groups or LDAP groups. The application can be identified against a user belonging to any of these groups by the specified team.

Field	Description
	<ul style="list-style-type: none"> ■ Token. The token to identify requests from an application. ■ Username. The username credential to identify requests from an application. ■ WS-Security username. The WSS username to identify requests from an application.

6. Click **Continue to APIs >**.

Alternatively you can click **APIs** in the left navigation panel.

You can save the application by clicking **Save** at this stage and add the APIs at a later time.

7. Type a keyword to find the required API and click **+** to add the API.

Adding an API to the application enables the application to access the API. An API developer while invoking the API at runtime, has to provide the access token or identification token for API Gateway to identify the application.

8. Type the required Requestor comment.

9. Click **Continue to Advanced >**.

Alternatively you can click **Advanced** in the left navigation panel.


You can save the application by clicking **Save** at this stage and add the APIs at a later time.

10. Specify the origin from which the responses originating are allowed during response processing for the application.

Note:

You cannot provide Regular expressions for allowed origins.

11. Click **+Add** to add the origin.

You can add multiple origins using  .

12. Click **Continue to Authentication >**.

Alternatively you can click **Authentication** in the left navigation panel.

You can save the application by clicking **Save** at this stage and add the Authentication strategy at a later time.

13. Click **Create strategy**.

A strategy is a way to authenticate the incoming request and provides multiple authentication mechanisms or multiple authorization servers for a single authentication scheme. You can create multiple strategies authorized by an API for an application.

14. Select one of the **Authentication schemes**:

- **OAuth2.** Provide the following information:

Field	Description
Name	Provide the name for the strategy.
Description	Provide a description to describe the strategy.
Authentication server	Specify the authentication server. The available values are local , which is the default server or any other configured external authorization server.
Generate Credentials	Enable the toggle button to generate the client dynamically in the authorization server and provide the following information: <ul style="list-style-type: none"> ■ Type. Select one of the client types: <ul style="list-style-type: none"> ■ Confidential. A confidential client is an application that is capable of keeping a client password confidential to the world. This client password is assigned to the client app by the authorization server. This password is used to identify the client to the authorization server, to avoid fraud. An example of a confidential client could be a web app, where no one but the administrator can get access to the server, and see the client password. ■ Public. A public client is an application that is not capable of keeping a client password confidential. For instance, a mobile phone application or a desktop application that has the client password embedded inside it. Such an application could get cracked, and this could reveal the password. The same is true for a JavaScript application running in the users browser. The user could use a JavaScript debugger to look into the application, and see the client password. ■ Application type. Specify the application type. <ul style="list-style-type: none"> ■ WEB. A web application is an application running on a web server. In reality, a web application typically consists of both a browser part and a server part. The client password could be stored on the server. The password would thus be confidential.

Field	Description
	<ul style="list-style-type: none"> <li data-bbox="638 258 1385 432">■ USER_AGENT. A user agent application is for instance a JavaScript application running in a browser. The browser is the user agent. A user agent application may be stored on a web server, but the application is only running in the user agent once downloaded. <li data-bbox="638 457 1385 663">■ NATIVE. A native application is for instance a desktop application or a mobile phone application. Native applications are typically installed on the users computer or device (phone, tablet etc.). Thus, the client password will be stored on the users computer or device too. <li data-bbox="589 688 1385 758">■ Token lifetime. Specify the token lifetime in seconds for which the token is active <li data-bbox="589 783 1385 852">■ Token refresh limit. Specify the number of times you can use the refresh token to get a new access token. <li data-bbox="589 877 1385 1020">■ Redirect URIs. Specify the URIs that the authorization server can use to redirect the resource owner's browser during the grant process. You can add multiple URIs by clicking +Add. <li data-bbox="589 1045 1385 1293">■ Grant type. Specify the grant type to be used to generate the credentials. Available options can be authorization_code, password, client_credentials, refresh_token, and implicit, which are dynamically populated from the authorization server. For example, if the authorization server does not support client credentials, the option is not available in the options list. <li data-bbox="589 1318 1385 1388">■ Scopes. Select the scopes that are to mapped for the authentication strategy. <div data-bbox="638 1409 1365 1646" style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: in API Gateway 10.2, the scopes are automatically created when you associate an API to an application. From API Gateway 10.3 onwards you have to select scopes from the authorization server that have to be associated with the strategy.</p> </div>
Client id	Specify the Client identifier for a client application available in the authorization server that identifies the client application in the authorization server to map the client to the API Gateway application.

Field	Description
	<i>This is required if you have a client application available in the authorization server and do not want to dynamically create a client.</i>

- **JWT.** Provide the following information:

Field	Description
Name	Provide the name for the strategy.
Description	Provide a description to describe the strategy.
Authentication server	Specify the authentication server. The possible values are local , which is the default server or any other configured external authorization server.
HMAC algorithm	Select if the authorization server is returning a JWT with HMAC algorithm and provide the shared secret value to validate the JWT.

- **OPENID.** Provide the following information:

Field	Description
Name	Provide the name for the strategy.
Description	Provide a description to describe the strategy.
Authentication server	Specify the authentication server. The available values are local , which is the default server or any other configured external authorization server.
Generate Credentials	Enable the toggle button to generate the credentials required to identify the client application and provide the following information:

- **Type.** Select the client type, Public or Confidential
 - **Confidential.** A confidential client is an application that is capable of keeping a client password confidential to the world. This client password is assigned to the client app by the authorization server. This password is used to identify the client to the authorization server, to avoid fraud. An example of a confidential client could be a web app, where no one but the administrator can get access to the server, and see the client password.

Field	Description
	<ul style="list-style-type: none"><li data-bbox="634 260 1385 573">■ Public. A public client is an application that is not capable of keeping a client password confidential. For instance, a mobile phone application or a desktop application that has the client password embedded inside it. Such an application could get cracked, and this could reveal the password. The same is true for a JavaScript application running in the users browser. The user could use a JavaScript debugger to look into the application, and see the client password.<li data-bbox="589 600 1385 632">■ Application type. Specify the application type.<ul style="list-style-type: none"><li data-bbox="634 659 1385 831">■ WEB. A web application is an application running on a web server. In reality, a web application typically consists of both a browser part and a server part. The client password could be stored on the server. The password would thus be confidential.<li data-bbox="634 858 1385 1031">■ USER_AGENT. A user agent application is for instance a JavaScript application running in a browser. The browser is the user agent. A user agent application may be stored on a web server, but the application is only running in the user agent once downloaded.<li data-bbox="634 1058 1385 1262">■ NATIVE. A native application is for instance a desktop application or a mobile phone application. Native applications are typically installed on the users computer or device (phone, tablet etc.). Thus, the client password will be stored on the users computer or device too.<li data-bbox="589 1289 1385 1356">■ Token lifetime. Specify the token lifetime in seconds for which the token is active<li data-bbox="589 1383 1385 1451">■ Token refresh limit. Specify the time in seconds for which the token refresh is applicable<li data-bbox="589 1478 1385 1619">■ Redirect URIs. Specify the URIs that the authorization server can use to redirect the resource owner's browser during the grant process. You can add multiple URIs by clicking +Add.<li data-bbox="589 1646 1385 1749">■ Grant type. Specify the grant type to be used to generate the credentials. Available options are Authorization code, Implicit, Resource owner, Client credentials.<li data-bbox="589 1776 1385 1843">■ Scopes. Select the scopes that are to be associated to the generated client.

Field	Description
	<p>Note: in API Gateway 10.2, the scopes are automatically created when you associate an API to an application. From API Gateway 10.3 onwards you have to select scopes from the authorization server that have to be associated with the strategy.</p>
Client id	<p>Specify the Client identifier that identifies the client application in the authorization server to map the client to the API Gateway application.</p> <p><i>This is required if you do not choose to generate credentials to identify the client application.</i></p>

15. Click **Add**.

The strategy is configured and listed in the Strategies table.

16. Click **Save**.

The application is created and listed in the list of applications in the Manage applications page, once approved.

Viewing List of Applications and Application Details

You can view the list of applications in the Manage applications page from where you can create, delete, and select an application to view its details.

➤ To view the application list and application details

1. Click **Applications** in the title navigation bar.

A list of all registered applications is displayed.

2. Select an application.

The application details page displays the following information: basic information that contains details such as name, description, owner, and creation time, identifiers, access tokens, APIs registered for the application, advanced configurations, and authentication strategies configured for the application.

Application credentials, such as, API Keys or OAuth client secrets are visible only to the application owner. All other users can only see an encrypted value. Since API Portal and API Gateway do not support a central user management, API Gateway users cannot see the application credentials of the application requested through API Portal.

Regenerating API Access Key

You must have the API Gateway's manage applications functional privilege assigned to perform this task.

You can regenerate an API access key in the Application details page from where you can view application details.

➤ To regenerate an API key

1. Click **Applications** in the title navigation bar.

A list of all registered applications is displayed.

2. Select an application.

The application details page displays the basic information, identifiers, access tokens, API key, APIs registered and strategies configured for that application.

3. Click .

The API access key is regenerated and the new API access key appears in the **API access key** field.

Modifying Application Details

You can modify the details of an application as required from the application details page.

➤ To modify application details

1. Click **Applications** in the title navigation bar.

A list of registered applications is displayed.

2. Select an application.

3. Click **Edit** in the application details page.

4. Modify the required fields in the Basic information section.

5. Click **Identifiers**.

6. Modify the required fields in the Identifiers section.

7. Click **APIs**.

8. Add or delete the APIs that are registered.
9. Modify the strategies or create a new strategy.
10. Modify the required values.
11. Click **Save**.

Registering an API with Consumer Applications from API Details Page

Consumer applications created in API Gateway can be associated with APIs from the API details page.

➤ To register APIs with consumer applications

1. Click **APIs** in the title navigation bar.
A list of APIs is displayed.
2. Select an API.
3. Click **Edit** in the API details page.
4. Click **Application** tab in the API details page.
5. Type characters in the search field and click the **Search** icon.

This field displays the only list of applications that are assigned to the teams that you are a part of.

6. Select the required applications and click **+**.
You can add more applications in a similar way.
7. Click **Save**.

Registering APIs with Consumer Applications from Application Details Page

Consumer applications created in API Gateway can be associated with the APIs from the application details page.

➤ To register APIs with consumer applications

1. Click **Applications** in the title navigation bar.

A list of registered applications is displayed.

2. Select an application.

3. Click **Edit** in the application details page.

4. Click **APIs** in the left navigation panel.

5. Type characters in the search field and click the **Search** icon.

This displays the APIs that are assigned to the team(s) that you are a part of.

6. Select the required API and click **+**.

You can add more APIs in a similar way.

7. Click **Save**.

Suspending an Application


You must have the API Gateway's manage applications functional privilege assigned or you must be the owner of the application to perform this task.

You can suspend an application from the Applications details page.

> To suspend an application


1. Click **Applications** in the title navigation bar.

A list of all the available applications are displayed.

2. Click the toggle button  (Active state), in the action column for the respective application, to suspend the application.

Alternatively, you can click **Suspend** in the application details page.

3. Click **Yes** in the confirmation dialog box.

The application is suspended. The toggle button in the Applications page changes to  (suspended state) and the option in the application details page changes to **Suspend**.

Activating a Suspended Application


You must have the API Gateway's manage applications functional privilege assigned or you must be the owner of the application to perform this task.

You can activate a suspended application, from the Applications details page, which enables the identification again.

➤ **To activate a suspended application**


1. Click **Applications** in the title navigation bar.

A list of all the available applications are displayed.

2. Click the toggle button  (suspended state), in the action column for the respective application, to activate the application.

Alternatively, you can click **Activate** in the application details page.

3. Click **Yes** in the confirmation dialog box.

The application resumes. The toggle button in the Applications page changes to  (active state) and the option in the application details page changes to **Suspend**.

8 API Packages and Plans

■ Overview	612
■ Creating a Package	612
■ Creating a Plan	614
■ Viewing List of Packages and Package Details	617
■ Modifying a Package	617
■ Deleting a Package	618
■ Activating a Package	618
■ Publishing a Package	619
■ Viewing List of Plans and Plan Details	620
■ Modifying a Plan	620
■ Deleting a Plan	621

Overview

An API Package refers to a logical grouping of multiple APIs from a single API provider. A package can contain one or more APIs and an API can belong to more than one package. You must have the API Gateway's manage packages and plans functional privilege assigned to manage API packages and plans.

An API Plan is the contract proposal presented to consumers who are about to subscribe to APIs. Plans are offered as tiered offerings with varying availability guarantees, SLAs or cost structures associated with them. An API package can be associated with multiple plans at a time. This helps the API providers in providing tiered access to their APIs to allow different service levels and pricing plans. Though you can edit or delete a plan that has subscribers, Software AG recommends you not to do so.

Note:

Package and plan subscriptions can be done only through API Portal.

You can create packages and plans, associate a plan with a package, associate APIs with a package, view the list of packages, package details, and APIs and plans associated with the package in the API Gateway user interface.

When you add an API to a package for monetization, the **API key** authentication mechanism is automatically added to the IAM policy at API level. If the API already contains an IAM policy that has two authentication mechanisms with the **AND** condition, then the condition will be switched to **OR**. This ensures the monetization is supported when certain consumers access the API by just using the API key.

Note:

If you are configuring API consumption rate limits for any purpose other than monitoring, apply the Traffic optimization policy, instead of packaging. For information about the policy, see [“Throttling Traffic Optimization” on page 494](#).

Creating a Package


You must have the API Gateway's manage packages and plans functional privilege assigned to perform this task.

You can create an API Package from the Manage packages and plans page.

> To create an API Package

1. Click **Packages** in the title navigation bar.
2. Click **Create** in the Manage packages and plans section.
3. Select **Package**.
4. Click **Create**.

5. Provide the following information in the Basic information section:

Field	Description
Name	Name of the API package.
Version	Version assigned for the API package.
Team	Team to which the application must be assigned to. You can select more than one team. To remove a team, click the  icon next to the team.
Description	A brief description for the API package.
Icon	An icon that is displayed for the API package. Click Browse and select the required image to be displayed as the icon for the API package. The icon size should not be more than 100 KB.

You can save the API package at this point and add the plans at a later time.

6. Click **Continue to add plans**.

Alternatively, click **Plans** in the left navigation pane.

7. Select the plans that are to be associated with the API package.

You can save the API package at this point and add APIs at a later time.

8. Click **Continue to add APIs**.

Alternatively, click **APIs** in the left navigation pane.

When you add an API to a package for monetization, the **API key** authentication mechanism is automatically added to the IAM policy at API level. If the API already contains an IAM policy that has two authentication mechanisms with the **AND** condition, then the condition will be switched to **OR**. This ensures the monetization is supported when certain consumers access the API by just using the API key.

9. Type characters in the search box and click the search icon to search for the required APIs.

A list of APIs that contain the characters specified in the search box appears.

10. Select the required APIs to be associated with the Package and click **+** to add them.

You can delete the APIs from the package by clicking the **Delete** icon adjacent to the API in the API list.

11. Click **Save**.


Creating a Plan

You must have the API Gateway's manage packages and plans functional privilege assigned to perform this task.

You can create a Plan from the Manage packages and plans page.

> To create a plan

1. Click **Packages** in the title navigation bar.
2. Click **Create** in the Manage packages and plans section.
3. Select **Plan**.
4. Click **Create**.
5. Provide the following information in the Basic information section:

Field	Description
Name	Name of the plan.
Version	Version assigned for the plan.
Team	Team to which the application must be assigned to. You can select more than one team. To remove a team, click the  icon next to the team.
Description	A brief description for the plan.
Icon	An icon that is displayed for the plan. Click Browse and select the required image to be displayed as the icon for the plan. The icon size should not be more than 100 KB.

You can save the plan at this point and add the pricing and traffic optimization configurations at a later time.

6. Click **Continue to Pricing**.

Alternatively, click **Pricing** in the left navigation pane.

7. Provide the following information in the Pricing section:

Field	Description
Cost	Specifies the cost for the plan.
Terms	Specifies the terms of conditions for the pricing.
License	Specifies the license information.

You can save the plan at this point and provide traffic optimization configurations at a later time.

- Click **Continue to Quality of Service**.

Alternatively, click **Rate limits** in the left navigation pane.

Note:

If you are configuring API consumption rate limits for any purpose other than monetization, apply the Traffic optimization policy, instead of packaging. For information about the policy, see [“Throttling Traffic Optimization” on page 494](#).

- Click **+ Add Rule**.
- Provide the following information in the Create Rule section:


Field	Description
Maximum request count	Specifies the maximum number of requests handled. Value provided should be an integer.
Interval	Specifies the value for the interval for which the maximum request count is handled. Value provided should be an integer.
Interval unit	Specifies the unit of measurement of the time interval. For example: minutes, seconds, and so on
Violation message	Specifies the text that displays when the rule is violated.

- Click **Ok**.

This creates the rule and displays it in the Configured rules table. Click **+ Add rule** to add more rules. You can edit or delete the rules by clicking the **Edit** and the **Delete** icons respectively.

At a later time, when this plan is applied to an API through a package, the rules that you configured for this plan are enforced on the applied API.

- Click **Quota** and provide the following information in the Quota settings section.

Field	Description
Maximum request quota	Specifies the maximum number of requests handled. Value provided should be an integer.
Block on breach	When selected it specifies that the access to the API is blocked when there is a rule violation. By default, this option is not selected.
Interval	Specifies the value for the interval for which the maximum request quota is handled. Value provided should be an integer.
Interval unit	Specifies the unit of measurement of the interval. Example: minutes, seconds, and so on
Violation message	Specifies the text that displays when the policy is violated.
Notification settings	Specifies whether notifications are to be sent on rule violations. Enable the toggle button to enable the notifications and provide the following information: <ul style="list-style-type: none"> ■ Notify after (in %): Provide a value which is a number. A notification is sent to the configured email IDs once the total request count reaches the % value as provided in the maximum quota value. ■ Violation message: Provide the content of the mail that is sent to the configured email Ids once the quota request count reaches the limit specified. ■ Email Ids: Provide email Id of the recipient to which notifications have to be sent of notification once the quota request count reaches the limit specified. Click . You can add multiple recipients. <p>Note: The SMTP settings under Administrator settings > Destinations has to be provided for email to be sent.</p>

Note:

The number of requests within a Quota interval is retained in case of server crash or restart. For example, if the number of requests against a Quota is 10 and if the server restarts or crashes, the number of requests for that Quota is retained when the server is up.

13. Click **Save**.

The plan is created and listed in the list of plans.

Viewing List of Packages and Package Details

You can view the list of packages in the Packages section of the Manage packages and plans page from where you can create, delete, and select a package to view its details.

> To view the package list and package details

1. Click **Packages** in the title navigation bar.

A list of all packages appears. You can perform various operations like activating a package, publishing or unpublishing a package, and deleting a package.

2. Select a package.

The basic information, and the associated plans and APIs for the selected package appears in the package details page.

Modifying a Package

You must have the API Gateway's manage packages and plans assigned to perform this task.

You can modify the basic information, include or exclude plans and APIs of the package. You can modify a package only if it is in inactive state.

> To modify a package

1. Click **Packages** in the title navigation bar.

A list of all packages appears.

2. Select a package.

The basic information, and the associated plans and APIs for the selected package appear on the package details page.

3. Click **Edit**.

The package details appear.

Note:

The **Edit** option is available only if the package is in inactive state.

4. You can modify the information related to the package, as required, in the Basic information section.
5. Click **Plans** in case you want to modify the plans associated with the package.

A list of plans associated with the package and list of available plans appears.

6. You can do the following:

- Add more plans to the package by selecting plans listed in the available plans list.
- Delete the plans from the package by clearing the check box of the plan associated with the package.

7. Click **APIs** in case you want to modify the APIs associated with the package.

A list of APIs associated with the package and a search box to search for APIs that need to be added to the package appear.

8. You can do one of the following:

- Add more APIs to the package. You can search for APIs using the search box and click **+** adjacent to the API to add it
- Delete the APIs from the package by clicking the **Delete** icon adjacent to the API in the APIs list.

9. Click **Save**.

This saves the modified package.

Deleting a Package

You must have the API Gateway's manage packages and plans assigned to perform this task.

You can delete a package from the Package list that appears on the Manage packages and plans page. You can not delete a package if it is in active state. You have to deactivate it before deleting it.

> To delete a package

1. Click **Packages** in the title navigation bar.

A list of all packages appears.

2. Click the **Delete** icon for the package that has to be deleted.

3. Click **Yes** in the confirmation dialog.

Activating a Package

You must have the API Gateway's activate/deactivate packages assigned to perform this task.

You can activate a package so that a consumer can try out APIs in the package with the package level token. When the consumer requests a token from API Portal, the request is processed in API Gateway and a token is sent back to API Portal. This token is visible to the consumer on the Access Token page. The consumer can test the APIs in the package with this token on the API Try out page.

> To activate a package

1. Click **Packages** in the title navigation bar.

A list of all packages appears with their status as **Inactive** or **Active**.

2. Click the activation toggle button for the package.

The package is now activated.

Alternatively you can click **Activate** on the Packages details page to activate the package.

Publishing a Package

You must have the API Gateway's publish to API Portal functional privilege assigned to perform this task.

You can publish a package to the configured destination, for example API Portal. Once the package is published, the APIs associated with the package are available to consumers. The package level token is applicable to all APIs associated with the package. The consumers do not have to request an access token for individual APIs to consume them.

Ensure the following before publishing a package:

- A destination is configured.
- The package is active.
- The package has at least one plan and API associated with it.
- The APIs associated with the package is published to the destination.

> To publish a package

1. Click **Packages** in the title navigation bar.

A list of all packages appears.

2. Click the **Publish** icon for the package that has to be published.

3. Select the communities to which the package needs to be published.

By default, a package is published to the Public Community of API Portal.

Note:

The list of communities displayed are those that are common to which the APIs associated with this package are already published to.

4. Click **Publish**.

A success messages is displayed when the package is successfully published. The package is now published to the destination, for example API Portal, that is configured and is available on API Portal to consumers.

You can unpublish a package once it is published by clicking the **Unpublish** icon for the required package.

Viewing List of Plans and Plan Details

You can view the list of plans in the Plans section of the Manage packages and plans page from where you can create, delete, and select a plan to view its details.

> To view the plan list and plan details

1. Click **Packages** in the title navigation bar.
2. Click **Plans**.

A list of all plans appears. You can deleting a plan by clicking the **Delete** icon for the respective plan.

3. Select a plan.

The basic information, the pricing, and Quality of service associated with the selected plan appears in the plan details page.

Modifying a Plan

You must have the API Gateway's manage packages and plans functional privilege assigned to perform this task.

You can modify a plan to change the pricing details and Quality of service associated with the plan.

> To modify a plan

1. Click **Packages** in the title navigation bar.
A list of all packages appears.
2. Click **Plans**.

A list of all plans appears.

3. Select a plan.

The plan details page displays the basic information, pricing details, and the Quality of service associated with the plan.

4. Click **Edit**.

The plan details appear with fields that you can edit.

5. You can modify the information related to the plan, as required, in the Basic information section.

6. Click **Pricing** in case you want to modify the pricing model associated with the plan.

7. Modify the pricing plan as required.

8. Click **Rate limits** if you want to modify the rules associated with the plan.

A list of rules associated with the plan appears.

9. You can do one of the following:

- Add more rules to the plan. Click **Add rule** to create and add rules to the plan.
- Modify the already configured rule. Click the **Edit** icon for the rule listed in the **Configured rules** list and modify the details as required.
- Delete rules from the plan. Click the **Delete** icon adjacent to the rule in the **Configured rules** list.

10. Click **Quota settings** if you want to modify the quota settings for the plan.

11. Modify the quota settings as required.

12. Click **Save**.

This saves the modified plan.

Deleting a Plan

You must have the API Gateway's manage packages and plans functional privilege assigned to perform this task.

You can delete a plan from the Plans list that appears in the Plans section of the Manage packages and plans page. You can delete a plan only if it is not associated with a package. You have to disassociate the plan with the package before deleting it.

> **To delete a plan**

1. Click **Packages** in the title navigation bar.

2. Click **Plans**.

A list of plans appears.

3. Click the **Delete** icon for the plan that has to be deleted.

4. Click **Yes** in the confirmation dialog.

9 Export and Import Assets and Configurations

■ Overview	624
■ Importing Asset and Configuration Archives	629

Overview

API Gateway supports the import and export of the assets that you create or configure in API Gateway. You can import archives of APIs, global policies, and other related assets that you have exported and re-create them in API Gateway. This enables you to easily export and archive the assets; and when required, import them to a different instance of API Gateway or redeploy them on the same instance.

Each artifact in an archive is associated with a universally unique identifier (UUID) that is unique across all API Gateway installations. When importing an archive, the UUID helps in determining whether the corresponding artifact is already available in API Gateway. You can configure whether you want to overwrite the existing artifact or keep the available artifact during the import process.

Note:

During export or import of assets, ensure that the master password is identical across stages and on different instances of API Gateway.

Considerations while importing assets:

- The APIs, applications, policies, and aliases you import become visible in API Gateway immediately.
- Active APIs are replaced during import with the updated API and the API level policies.
- The updated APIs and updated API level policies do not become effective for ongoing requests.
- Active APIs are replaced during deployment with zero downtime without breaking ongoing requests.
- Imported applications become effective immediately, even the ongoing requests are affected.
- Imported aliases and global policies do not affect the ongoing requests.
- You can not define multiple aliases with the same name in API Gateway as overwriting of aliases based on their names during import is not supported. Aliases, like other assets, are identified based on their UUID. Hence, if you want to overwrite an alias by importing, then ensure that the alias being imported has the same UUID as the one in the target instance.

Note:

- Do not attempt to modify and import an archive file because import of modified archive files is not supported.
- You can export archives from an earlier version to a later version of API Gateway. However, you can not import from a later version to an earlier version. For example, you can not import an 10.5 asset into a 10.3 API Gateway.

You can also export and import assets using the API Gateway REST APIs. For more information, see [“API Gateway Archive” on page 756](#).

When you export an asset, the dependent assets are also exported. If any of the exported assets contain secure strings, the user credential information (passwords) associated with the assets is also exported. When you import this exported asset, API Gateway enforces conditions to check the order of import and the dependency evaluation between assets, and the dependent assets

along with the user credential data are imported. For example, if you import an API, API Gateway checks and ensures that all associated policies and aliases are imported along with any passwords, if present, before importing the API.


The **Overwrite** option available for all the assets allows you to decide whether the asset should be imported if an existing version of the asset already exists in the target instance. In scenarios where you select to overwrite the asset in the target instance, API Gateway also checks for any associated passwords and applies the overwrite accordingly. There is no separate overwrite option for the passwords during import. The password uses the overwrite option of the asset it is associated with. For example, if you are importing an alias with a password, the overwrite option provided for the alias is applied for the password as well. If set to true, the password is overwritten if it is already exists in the system.

Functional Privileges

The **Export or Import assets and Purge and Archive events** category on the **Functional privileges** page has the available import and export privileges. You must assign the following functional privileges for the required permissions:

- **Import assets:** To import assets previously exported assets from a local system.
- **Export assets:** To export assets and save them on a local system.





Accessing the Export and Import commands in the API Gateway user interface





The export command is either a button with the label **Export**, or the  icon. You can export multiple items within lists, such as APIs in the API page, by using the export command in the list menu.

You can import assets using the user menu () > **Import** command.

Assets that can be exported and imported

Path to Page/Tab	Assets that can be exported and imported
APIs	APIs
Policies > Threat protection	Global denial of service
	Denial of service by IP
	Rules
	Mobile device and apps
	Alert settings
Policies > Global policies	Global policies
Policies > Policy templates	Policy templates

Path to Page/Tab	Assets that can be exported and imported
Applications	Applications
Packages > Packages	Packages
Packages > Plans	Plans
User menu () > Administration > General	<ul style="list-style-type: none"> ■ Load balancer ■ Extended settings ■ API fault ■ Approval configuration ■ Outbound proxy ■ URL Aliases ■ Custom content-types ■ Cache configuration ■ Log level configuration ■ Callback processor settings ■ Messaging ■ Web services
User menu () > Administration > General > Messaging	<ul style="list-style-type: none"> ■ JNDI Provider Alias ■ JMS Connection Alias
User menu () > Administration > Security	<ul style="list-style-type: none"> ■ Keystore/Truststore ■ Ports ■ SAML issuer ■ Custom assertions ■ Kerberos ■ JWT/OAuth/OpenID ■ Providers
User menu () > Administration > Destinations	<ul style="list-style-type: none"> ■ API Gateway ■ API Portal (both the Event configurations and communication configurations are exported)

Path to Page/Tab	Assets that can be exported and imported
	<ul style="list-style-type: none"> ■ Transaction logger ■ Elasticsearch (properties on both tabs—Elasticsearch communication and Events—are exported) ■ Email (properties on both tabs—Email configuration and Templates—are exported) ■ SNMP (properties on both tabs—SNMP communication and Events—are exported)
User menu () > Administration > System settings	<ul style="list-style-type: none"> ■ Configurations ■ SAML SSO <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note: A change in the SAML SSO configuration from the API Gateway user interface forces the logged in user to log out. However, importing an SAML SSO does not.</p> </div>
User menu () > Administration > Service registries	<ul style="list-style-type: none"> ■ Service registry
User menu () > Administration > Aliases	<ul style="list-style-type: none"> ■ Aliases
User menu () > User management	<ul style="list-style-type: none"> ■ Users ■ Groups ■ Teams ■ Global team assignments ■ Account settings <ul style="list-style-type: none"> ■ Password restrictions ■ Password expiry settings ■ Account locking settings ■ LDAP configuration

For more information about how to export APIs and Global policies, see the following:

- [“Exporting APIs” on page 347](#)
- [“Exporting Global Policies” on page 557](#)

Dependencies

Some API Gateway assets use other assets. For example, APIs uses policies, aliases, and other assets. As the configuration of an asset is incomplete without the assets it uses, the export features includes the assets that are used by the asset that you export.

Note:

The association of a user to a group is not exported. After importing a user archive, you must manually link the new users to the required groups.

The following table shows the asset dependencies of each type of asset:

Asset	Dependencies (Required)	Dependencies (Optional)
APIs	Policies, Aliases	Applications, Applications registrations
Applications	APIs, Application Registrations	—
Registered Applications	APIs	Applications
Packages	APIs, Plans, Policies, Subscriptions	—
Plans	Policies	—
Subscriptions	Packages, Plans	Applications
Teams	—	Group
Approval configurations	Teams	—
Configuration > Keystore	Keystore, Truststore	—
Email destination	—	Trust store
Group	—	User
JMS connection alias	JNDI provider alias	—
LDAP configuration	Group	—
Password expiry settings	—	User
Port (https)	Keystore, Truststore	—
Service Registry	Keystore, Truststore	—
Web service endpoint alias	Teams, JMS, JNDI, JMS Trigger, Keystore, Truststore	—


Importing Asset and Configuration Archives

You can import archives of assets that you have exported and re-create them in API Gateway.

Note:

You need to import the authorization servers before you import the assets (such as applications) that depend on these authorization servers.

➤ **To import the exported files**

1. Expand the menu options icon , in the title bar, and select **Import**.
2. Provide the following information:

Parameter	Description
Select archive file	Click Browse to select a file or ZIP format file.
Overwrite	Select an overwrite option: <ul style="list-style-type: none"> ■ None: If you do not want to overwrite matching objects that exist on the server. Import will fail for the object in the archive if a matching object/asset already exists on the server. ■ All: If you want to overwrite any matching asset that exists on the server. If a match is not found, then a new asset is created. ■ Custom: If you want to select the specific types of assets that will be overwritten on the server if a match is found. If a matching asset exists on the server for an asset type that is not selected in the Custom overwrite list, the import operation will fail.

Note:

If a duplicate asset is found for any asset type that is not selected in the **Custom** overwrite list, the import will fail.

Note:

Some types of assets have dependencies on other asset types. For example, APIs have a dependency on policies, aliases, and applications. Some of the dependencies are required, while others are optional. The required dependencies are always included in the archive when you export the asset. You should consider your requirements and select the assets that need to be overwritten in the **Custom** list. For more information, see [“Overview” on page 624](#).

Parameter	Description
API version history	Select the option Fix missing versions to fix the API version history. On selecting this option, the API versions are newly linked according to the system version of the APIs.

Note:

API Gateway supports backward compatibility for API Gateway 10.1 version or higher when importing the archives of APIs. In addition, the compatibility of archives across API Gateway fix levels is also maintained. For example, you can import the archives created from lower fix levels of API Gateway into higher fix levels.

The import of an archive created from a higher fix level of API Gateway into a lower fix level can be rejected if the higher fix level's configurations are not supported by the lower fix level.

3. Click **Import**.

The **Import report** displays the following information:

Parameter	Description
Type	The asset type.
Successful	The number of successful imports for each artifact type.
Unsuccessful	The number of unsuccessful imports for each artifact type.
Replaced	The number of instances replaced for each artifact type.
Warning	The number of warnings displayed during the import of each artifact type. API Gateway displays warning messages when the import is successful but some additional information is required.

4. Click **Download the detail report here >** to download the detail report.

The detail report displays the following information about the imported artifact:

Parameter	Description
Name	The name of the artifact imported.
Type	The artifact type.

Parameter	Description
Status	The status of the imported artifact. The available values are: <ul style="list-style-type: none">■ Success■ Replaced■ Warning■ Failure
Explanation	The reason if the import fails or if a warning occurs.

If you want to take a backup of an API that you want to overwrite during import, you can set the parameter `enableImportBackup` as `true` under **Administration > General > Extended Settings** section. For more information about this extended setting, see [“Configuring Extended Settings” on page 22](#).

If an API import fails, one of the reasons might be that a configuration that is required by the API is not set up correctly on API Gateway. If something happens unexpectedly while the import is in progress, API Gateway discontinues the import and restores the existing API. This is necessary as parts of the existing API such as policies may already have been overwritten.

10 Asset Promotions

■ Manage Stages, Promotions, and Rollbacks	634
--	-----

Manage Stages, Promotions, and Rollbacks

API Gateway supports staging and promotion of assets. In a typical enterprise-level, solutions are separated according to the different stages of Software Development Lifecycle (SDLC) such as development, quality assurance (QA), and production stages. As each organization builds APIs for easy consumption and monetization, continuous integration (CI) and continuous delivery (CD) is an integral part of the solution. CI is a development practice that requires developers to integrate code into a shared repository several times a day, and CD is a software engineering approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time.

API Gateway provides the staging and promotion feature to automate the CI and CD practices. Modifications made to the APIs, policies, and other assets can be efficiently delivered to the application developers with speed and agility.

Staging and promotion allows you to:

- Promote all the run time assets across different stages. For the list of assets and configuration settings that can be promoted, see [“Promotions” on page 638](#).
- Select and promote a subset of assets from one stage to another stage. For example, you can promote a single API and its dependencies from one stage to another.
- Optionally select the asset's dependencies during the promotion.
- Create stage-specific aliases.

When promoting an alias from source stage to target stage, API Gateway checks if the target stage already has an alias. If so, then API Gateway replaces the existing alias in target stage with the alias that is promoted from source stage.

- Roll back assets in case of incomplete information.
- Repromote assets, if there are any recent changes, to the already promoted stages.

Stages

The fundamental phases of assets starts with requirements or needs at the development stage and once the assets are developed, they are promoted to the QA stage for testing, after testing of the assets is complete, the assets are promoted to the production stage.

An asset's overall lifecycle can be split across two or more API Gateway instances. For example, assets that are in the development and test phases of their lifecycle can be maintained in one instance and assets that are deployed (that is, in production) can be maintained in a separate instance.

When the asset's lifecycle is split across multiple API Gateway instances, each participating instance is referred to as a stage. A stage definition in API Gateway describes an API Gateway instance by its name and configuration information.

Note:

Software AG recommends you to have API Gateway instances across stages to be completely independent. For example, the API Gateway instances from the Development stage and the API Gateway instances from the QA stage must not share any resources in common such as databases.


Adding a Stage

Pre-requisites:

You must have the Manage promotions functional privilege assigned to perform this task.

API Gateway can contain one or more stages. You define a stage to represent the asset's lifecycle such as development, test, and production.

> To add a stage

1. Expand the menu options icon , in the title bar, and select **Promotion management**.
2. Click **Add Stage**.
3. Provide the following information in the Basic information section:

Field	Description
Name	Mandatory. Name of the stage. Note: A stage name must be unique within API Gateway.
Description	Description of the stage.

4. Provide the following information in the Technical information section:

Field	Description
Stage URL	Mandatory. The URL of the host machine where the stage is deployed on an API Gateway installation. The Stage URL value is specified in the format, <code><scheme>://<host>:<port></code> . The scheme is http or https. The host is the host name of the machine on which the target API Gateway instance is running. The port is the port number of the target API Gateway instance.
Username	Mandatory. The username of a registered API Gateway user who has the Manage promotions or Manage assets functional privilege in the target API Gateway instance.

Field	Description
Password	Mandatory. A valid password of the API Gateway user identified by the attribute Username .
Keystore alias	<p>The alias of the keystore containing the private key that is used for performing asset promotion from one (source) stage to another (target) stage.</p> <p>The Keystore alias field contains a list of the available keystore aliases in API Gateway. If there are no configured keystore aliases, this field lists the default Integration Server keystore, <code>DEFAULT_IS_KEYSTORE</code>.</p> <p>Note: This field is required when the Stage URL scheme is set to <code>https</code> with Two-way SSL.</p>
Key alias (signing)	<p>The alias of the private key that is stored in the keystore specified by the keystore alias.</p> <p>The Key alias field contains a list of the available aliases in the selected keystore. If there are no configured keystores, this field is empty.</p> <p>Note: This field is required when the Stage URL scheme is set to <code>https</code> with Two-way SSL.</p>


5. Click **Save**.

Viewing Stage List and Stage Details

The **Stages** tab displays a list of the available stages in API Gateway.

In addition to viewing the list of stages, you can also examine the details of a stage and delete a stage in the **Stages** tab.

> To view the stage list and stage details

1. Expand the menu options icon , in the title bar, and select **Promotion management**.

The **Stages** tab displays a list of available stages.

This tab provides the following information about each stage:

Column	Description
Name	Name of the stage.

Column	Description
Description	The description for the stage.

- Click the required stage whose details you want to examine.

The Stage details page appears. The stage details are displayed in the following sections:

- **Basic information:** This section contains the stage name and description.
- **Technical information:** This section contains the stage specific URL, authentication credentials to access the stage, and the keystore configurations.


Modifying Stage Details

Pre-requisites:

You must have the Manage promotions functional privilege assigned to perform this task.

You can modify the basic or the technical information of a stage at any time.

➤ To modify the stage details

- Expand the menu options icon , in the title bar, and select **Promotion management**.

The **Stages** tab displays a list of available stages.

- Click the required stage whose details you want to modify.

The Stage details page appears.

- Click **Edit**.

This opens the basic information of the stage.

- Modify the basic information as required.

- Click **Technical information**.

This opens the technical information of the stage.

- Modify the technical information as required.

- Click **Save**.


Deleting a Stage

Pre-requisites:

You must have the Manage promotions functional privilege assigned to perform this task.

You delete a stage to remove it from API Gateway permanently. You can remove a stage at any time.

➤ To delete a stage

1. Expand the menu options icon , in the title bar, and select **Promotion management**.

The **Stages** tab displays a list of available stages.

2. Click the **Delete** icon for the stage that you want to delete.
3. Click **Yes** in the confirmation dialog.

Promotions

Promotion refers to moving API Gateway assets from the source stage to one or more target stages. For example, you might want to promote assets you have developed on servers in a Development stage (the source API Gateway instance) to servers in a QA or Production stage (the target API Gateway instance). When you promote an asset from one stage to another, the asset's metadata is copied from the source instance to the target instance.

Note:

For the list of assets and configuration settings that can be promoted, see [“Assets that can be exported and imported” on page 625](#).

Assets that are dependencies for the asset are also promoted. The required dependencies of an asset are always promoted with the asset. However, you can choose which optional dependencies are promoted at the time of promoting the asset. For example, at the time of promoting an API, the policies and aliases used by the API are always promoted. However, you can choose which dependent applications are promoted as applications an optional dependency for APIs. You can choose to promote all or no application, or select specific applications that you want to included with the API in the promotion.

The following table lists the required and optional dependencies of assets that are supported by promotion management.

Asset	Dependencies (Required)	Dependencies (Optional)
APIs	Policies, Aliases	Applications, Applications registrations
Applications	APIs, Application Registrations	—
Registered Applications	APIs	Applications
Packages	APIs, Plans, Policies, Subscriptions	—

Asset	Dependencies (Required)	Dependencies (Optional)
Plans	Policies	—
Subscriptions	Packages, Plans	Applications
Teams	—	Group
Approval configurations	Teams	—
Configuration > Keystore	Keystore, Truststore	—
Email destination	—	Trust store
Group	—	User
JMS connection alias	JNDI provider alias	—
LDAP configuration	Group	—
Password expiry settings	—	User
Port (https)	Keystore, Truststore	—
Service Registry	Keystore, Truststore	—
Web service endpoint alias	Teams, JMS, JNDI, JMS Trigger, Keystore, Truststore	—

Note:

Till API Gateway version 10.3, promotion management supported only the simple and endpoint types of aliases. Starting from API Gateway version 10.4, all types of aliases are supported.

Promoting Assets

Pre-requisites:

You must have the Manage promotions functional privilege assigned to perform this task.

Promoting assets from one (source) stage to another (target) stage includes the following high-level steps:

1. **Authorization server promotion:** Promote Authorization server in one of the following ways:
 - Export the authorization server from one environment and import it into another, if you want to use the same authorization server across multiple environments.
 - Create the authorization server with the same name as in the current environment, if you want to use a different authorization server instance.

Proceed to the next step once the authorization server is ready.

2. **Select assets for promotion:** During this step, you search for assets by using a keyword and by performing a type search that sorts and filters the results.

- **Search using a keyword:** You can search for all assets whose string attributes (asset name, description, and so on) contain a certain keyword (character string).
- **Search using a Type:** You can search for assets on the basis of types.

You may use the **Search by type** filter to restrict the types on which the search is conducted. In the **Search by type** panel, API Gateway shows you a list of supported asset types.

- **Search by team:** You can search for assets assigned to team(s). Click the team(s) to view the assets that are assigned to the selected team(s).
 - **Search using two keywords.** You can provide more than one keyword by separating the keywords using a pipe symbol (|). For example, to search for assets whose names contain *pet* and *test*, you can provide `pet|test` in the search field.
 - **Search using wild card characters.** You can use wild card characters in your search keyword. For example, to list all assets that start with *graph*, you can provide `graph*`.
3. **Optionally select assets' dependencies for promotion:** During this step, you specify whether the dependencies (for example, a list of applications that are registered for an API, subscriptions for a package, and so on) of the selected assets will be included for the promotion.

Note:

As groups and users are part of the team, if you want to promote the Groups and Users (optional dependencies) of the team that the selected asset belongs to, you have to set the "enableTeamWork" extended setting to true. Only then the **Groups** and **Users** sections appear.

4. **Select the stages to promote:** During this step, you specify one or more target stages to which you want to promote the selected assets and their dependencies.
5. **Configure the promotion details:** During this step, you provide the promotion-specific information.
6. Ensure that you are promoting the assets from an lower version of API Gateway to a higher version of API Gateway. You cannot promote assets from a higher version to a lower version of API Gateway. For example, if you are using API Gateway v10.11, you can promote the assets from this version to v10.15 and other higher versions. You cannot promote assets back to the v10.7, v10.5, and so on.

Important:


If you plan to perform bulk promotion of assets from a source stage to a target stage, Software AG recommends you to perform the following:

- Increase the value of **maxRegexLengthInSearchQuery** setting on the **Extended Settings** page under *Administration > General* in both the source and target stages, considering that the generated RegEx search query length may become longer during bulk promotion of assets. The default value of this property is 37000. When you increase the value, it can lead to higher memory and CPU utilization, particularly when dealing with complex regular expressions. Therefore, Software AG recommends to closely monitor any spike in memory and CPU usage while promoting assets. For example, if your RegEx search query's length is more than 37000, asset promotion fails indicating that the length of RegEx used in RegEx Query request has exceeded the allowed maximum. In such case, you must increase the

value of `maxRegexLengthInSearchQuery` setting to more than the value specified in the error message.

- Increase the value of the **`pg.gateway.elasticsearch.http.socketTimeout`** property, located at `SAG_Install_Directory/IntegrationServer/instances/IS_Instance_Name/packages/WmAPIGateway/config/resources/elasticsearch/config.properties`. The default value of this property is 3000. To bulk promote assets, for example, 50 numbers, you would set the value of this property to 60000. Depending on the number of assets you want to promote, increment the value as 70000, 80000, and so on. You must restart API Gateway for changes to this property to take effect.

> To promote assets

1. Expand the menu options icon , in the title bar, and select **Promotion management**.

2. Select **Promotions**.

The Search page appears.

3. Click **Promote**.

4. In the **Search by keyword** text box, type the keyword to search for assets. You can use one or more wildcards to specify the keywords.

API Gateway returns the assets that match the specified keyword.

5. In the **Search by type** drop-down list, select the required type(s). Select the **All** check box to search across all asset types.

API Gateway returns the assets that match the selected type(s). The number of search results is displayed in the results area, for example, *Showing 10 results*. If no results are found, the results area displays a blank page.

6. Optional. Select the **Include admin configurations** check box if you want to include administrative configurations in the promotion.

API Gateway displays the administrative configurations that are supported for promotion. For the list of assets and configuration settings that can be promoted, see [“Promotions” on page 638](#).

Important:

Before you configure API Gateway to promote an administrative configuration, make sure that the corresponding administrative settings are already configured in API Gateway. For example, to include the Load balancer configuration in the asset promotion set, the required Load balancer URLs should be configured.

7. Select the assets and the administrative configurations that you want to promote.

8. Click **Next**.

The Assets and dependencies page appears.

9. Select the referenced assets that you want to promote.

10. Click **Next**.

The Stages page appears.

11. Select the target stages to that you want to promote the assets, dependencies, and administrative configurations.

12. Click **Next**.

The Promote page appears.

13. Provide the following information:

Field	Description
Name	Mandatory. Name of the promotion.
Description	Description of the promotion.
Overwrites assets except alias that already exist on the selected target stages	Select this check box to overwrite assets without overwriting their aliases when promoting assets. If the asset being promoted has its aliases already existing in the target stage, then you can select this check box to overwrite the assets without overwriting their aliases.
Overwrites aliases that already exist on the selected target stages.	Select this check box to overwrite assets along with their aliases in the target stage.
Fix missing version	Select the check box to fix the API version history. The discrepancy between the asset versions in the source and target stages, if any, are fixed.

Note:

The **Overwrite assets if exists on selected stages** option allows you to overwrite assets that exist on the target stages. If you do not want to overwrite assets in the target stages, clear this check box.

14. Click **Promote**.

The selected assets, dependencies, and administrative configurations are promoted to the selected stages.

The Stage-specific promotion status page displays the status of the asset promotion in each of the selected stages. The available values are:

- Success
- Failure


The page also displays the reason if the promotion fails.

Viewing Promotion List and Promotion Details

The **Promotions** tab displays a list of the available asset promotions in API Gateway. The asset promotions are listed alphabetically by name.

In addition, you can examine the details of an asset promotion history, repromote assets to the configured stages, and delete the promotion history from API Gateway Data Store in the **Promotions** tab.

➤ To view the promotion list and promotion details

1. Expand the menu options icon , in the title bar, and select **Promotion management**.
2. Select **Promotions**.

The **Promotions** tab displays a list of available asset promotions.

This tab provides the following information about each promotion:

Column	Description
Name	Name of the stage.
Promoted by	Name of the user who initiated the asset promotion.
Promotion time	The time at which the asset promotion occurred.
Status	The status report on the asset promotion.

3. Examine the **Report** link in the **Status** column and check for any errors that occurred during the asset promotion.

The Promotion status report displays the following information:

Column	Description
Asset type	Name of the asset type that was promoted from the source instance.

Column	Description
Name	Name of the asset that was promoted from the source instance.
Status	The status of the asset promotion.
Comments	A descriptive comment on the asset promotion status.

4. Click the required promotion whose details you want to examine.

The Promote page appears.

Repromoting Assets


Pre-requisites:

You must have the Manage promotions functional privilege assigned to perform this task.

If you have made recent changes to an asset's information, you can repromote the updated asset to the already promoted target stages.

If you need to update an asset, for example, you need to correct an attribute setting, modify the asset's description, that has already been promoted to a target stage, you can simply make the change directly to the asset in the source target and then repromote the updated asset to the target stage just as you did with the previous version of the asset. In a multi-stage environment, you will need to repromote the updated assets in each of the participating API Gateway instances.

> To repromote assets

1. Expand the menu options icon , in the title bar, and select **Promotion management**.
2. Select **Promotions**.

The **Promotions** tab displays a list of available promotion histories in API Gateway.

3. Click the **Repromote** icon for the required assets' promotion history.

The Stages page appears.

4. Select the target stages to that you want to repromote the assets, dependencies, and administrative configurations.
5. Click **Next**.

The Promote page appears.

6. Provide the following information:

Field	Description
Name	Mandatory. Name of the promotion.
Description	Description of the promotion.

Note:

The **Overwrite assets if exists on selected stages** option allows you to overwrite assets that exist on the target stages. If you do not want to overwrite assets in the target stages, clear this check box.

7. Click **Promote**.

The selected assets, dependencies, and administrative configurations are repromoted to the selected stages.

The Stage-specific promotion status page displays the status of the asset repromotion in each of the selected stages. The available values are:

- Success
- Failure

The page also displays the reason if the repromotion fails.

Rollbacks

Rollback is the process of restoring the asset's metadata in the target API Gateway instance to a previous state.

You might need to rollback assets to revert any promotional data changes being made in the target API Gateway instance.


Rollback Asset Promotions

Pre-requisites:

You must have the Manage promotions functional privilege assigned to perform this task.

You can rollback an asset promotion that is already available in the target stage at any time.

> To rollback asset promotion

1. Expand the menu options icon , in the title bar, and select **Promotion management**.
2. Select **Rollbacks**.

The **Rollbacks** tab displays a list of available promotion histories.


3. Click the **Rollback** icon for the promotion history that you want to rollback.
4. Click **Yes** in the confirmation dialog.

Viewing Rollback List and Rollback Details

The **Rollbacks** tab displays a list of the available asset promotion rollbacks in API Gateway. The rollbacks are listed alphabetically by name.

In addition, you can also examine the details of a rollback, and delete the rollback history in the **Rollbacks** tab.

> To view the rollback list and rollback details

1. Expand the menu options icon , in the title bar, and select **Promotion management**.
2. Select **Rollbacks**.

The **Rollbacks** tab displays a list of available asset promotion rollbacks.

This tab provides the following information about each rollback:

Column	Description
Name	Name of the rollback.
Promotion time	The time at which the asset promotion was rolled back.
Status	The status report on the rollback operation.

3. Examine the **Report** link in the **Status** column and check for any errors that occurred during the rollback process.

The Promotion status report displays the following information:


Column	Description
Asset Type	Name of the asset type that was promoted from the source instance.
Asset Name	Name of the asset that was promoted from the source instance.

11 API Gateway Analytics

■ Analytics Dashboards	648
■ Runtime Events and Metrics Data Model	656

Analytics Dashboards

The analytics dashboards display a variety of charts to provide an overview of API Gateway performance and its API usage. The data for these dashboards come from the API Gateway destination store. In API Gateway there are two types of dashboards. Each of these dashboards has various filters that can be applied as per the required metrics to be monitored.

- **API Gateway dashboard.** Displays API Gateway-wide analytics such as Summary of APIs, API usage, API trends, the top performing API and the non-performing API analytics, audit logs, applications and package related event information. This can be accessed by expanding the menu options icon , in the title bar, and selecting **Analytics**.
- **API-specific dashboard.** Displays API specific analytics such as API invocation trends by response time, success and failure rates, API performance, consumer or application traffic for a specific API. This can be accessed from the API details page.

The dashboard displays depend on the events and metrics generated in API Gateway and their types. An event is a kind of notification or alert generated by the API Gateway Metrics and Event Notification module. Various types of event are generated based on the behavior of the transactions in the system. Events generated by API Gateway are real time events made persistent in the store and sent to configured destinations.

These are the types of events generated in API Gateway:

- **Transactional event :** Provides a summary of each runtime transaction in the system. It is generated when a Log Invocation policy is included for the API. For example, if an API has the policy attached to it, then for every invoke the system generates a transaction event. API Gateway provides a system global policy, Transaction logging, which are pre-configured in the product. This policy is, by default, deactivated. The transaction logging policy has standard filters and a log invocation policy that logs request or response payloads to a specified destination.
- **Error event:** Provides details of an error that occurred during an API invoke. This event is generated whenever there is an error in the system during a runtime service invocation. This is configured as part of destination configuration.
- **Monitoring event:** Provides a summary of event details along with the breach information when there is a threshold breach in any of the configured parameters. Monitoring could be done based on various parameters such as Total Request Count, Total Success Count, Response Time, and Availability. Monitoring can be done at the consumer application level too so that each consumer can be tracked individually. These events are generated when a Monitor Service performance and Monitor SLA Policy is included for the API. In addition, the Traffic Optimization policy generates these events for every API invocation only when there is a breach in the parameters configured in the policy or once per alert interval based on the alert frequency configured in the policy.
- **Policy violation event:** Provides a summary of the policy violations that occurred in the system. When a policy attached to an API is violated, the system generates the policy violation event for alerting the provider. The Identity and Access, Authorization, and Schema Validation policies generate these events. This is configured as part of destination configuration. In


addition, the Traffic Optimization policy generates these events for every API invocation only when there is a breach in the parameters configured in the policy.

- **Lifecycle event:** Provides a summary of the life cycle of the API Gateway instance. Whenever the instance is started or stopped, a life cycle notification is generated. This is configured as part of destination configuration.
- **Threat protection event:** Provides a summary of the threat protection filter and rule violations. When a filter or rule is violated, the system generates the threat protection violation event. This is configured a part of destination configuration.

Note:

Internalization is not supported in API Gateway dashboards.

API Gateway Dashboard

You can view the API Gateway dashboard by expanding the menu options icon  in the title bar, and selecting **Analytics**. The dashboard displays the API Gateway-wide analytics based on the metrics monitored.

You can select the time interval from the drop-down options, and click **Apply filter** to filter the analytics based on the time interval chosen.

If you select **Custom**, you can type the **From Date** and **To Date** to specify the time interval in which you want to view the API Gateway-wide analytics.

You can click on the specific event in the list under Legend to view the specific event in any of the widgets. You can view additional details for an event by hovering the cursor over a particular color in the graphical representations.

In the **Applications** dashboard, you can filter the data using the filter for Applications in the specified time interval. The Applications drop-down list displays all the applications. When you select an application, its data is displayed. By default, the data displayed is for all the applications.

In the **Packages** dashboard, you can filter the data using the filter for Packages in the specified time interval. The Packages drop-down list displays all the packages. When you select a package, its data are displayed. By default, the data displayed is for all the packages.

In the **Audit logs** dashboard, you can filter the data using the filter for Audit logs in the specified time interval. It displays the data of all the auditable events.

In the **Cache statistics** dashboard, you can filter the cache statistics data depending on the Node name and Application type specified in the specified time interval.

In the **Application logs** dashboard, you can filter the application logs depending on the node, origin of log and so on in the specified time interval. Click **Download** to download the aggregated logs, the logs collected from different sources such as API Gateway server logs, API Gateway UI logs, Internal Data Store logs, dashboard logs and platform logs. The downloaded logs would contain the logs filtered as per the time interval filter applied.

In the **API usage details** dashboard, you can filter the data using the filter for the API invocations in the specified time interval (in years). By default, the data displayed is for all the API invocations. This dashboard is visible only when API Gateway uses a transaction-based licensing model when each API invocation is considered as a transaction and API Gateway keeps a track of these transactions.

Note:

The Summary, Trends, and Application analytics are visible only in API Gateway Full Edition. Threat protection analytics is the only data visible in API Gateway Firewall Edition. The threat protection analytics information is visible only if you select the Alert destination as flow service in **Policies > Threat protection > Alert settings** section.

Category	Metric	Description
Summary	Overall events	Displays a pie chart that lists different events being monitored and each of these event categories is depicted with different colors.
	Application activity	Displays the application activity in API Gateway during the specified time.
	Runtime events	Displays the run time event details such as time when the event was generated, API Name, the application that generated the event, event type, description of the alert generated due to the event, status, and the source of event.
	Payload size	Displays the payload size of the request and responses during data transfer in the specified time. This data is picked up from the transactional event that is triggered when a log invocation policy is applied to the API.
	Package performance	Displays a pie chart depicting package performance during the specified time. The different colors in the pie chart depict different packages this API belongs to.
Trends	Events over time	Displays the trending of events generated by the APIs across API Gateway over time.
	API trend by success	Displays the trending of APIs based on their success rate in the performance metrics.
	API trend by failure	Displays the trending of APIs based on their failure rate in the performance metrics.
	Overall error trends	Displays a graph depicting the performance of all the APIs in the system based on the error event

Category	Metric	Description
		generated. Each of these event categories is depicted with different colors.
Applications	Events per application	Displays a pie chart that depicts the activity of events per application being monitored and each of these categories is depicted with different colors.
	Violations per application	Displays the number of violations per application based on the events generated such as monitoring, SLA violation, and policy violations.
	Activity rate of consumed packages	<p>This bar chart displays the package that the selected application has consumed (when an application is chosen in the filter).</p> <p>Hover the cursor over the bar chart to see the number of invocations to the package using the specified application.</p>
	Activity rate for consumed APIs	Displays the activity rate for all the APIs that are consumed by the application during the specified time.
	Runtime events	Displays the run time event details such as API Name, event type, date when the event was created, the agent on which the event was generated, description of the alert generated due to the event, the source of event, and the application that generated the event.
Packages	Package invocations	Displays the number of package invocations during the specified time.
	Trending subscription for package	<p>Displays the trending subscriptions for the package based on the number of invocations.</p> <p>The different colors in the donut pie chart depict the trending behavior of the different applications in the package.</p>
	Trending APIs in the package	Displays the number of invocations for an API for an application for the selected package over the specified time interval.
Threat protection	Threat protection filters	Displays the graphical representation of the events based on the filter violations during the specified time.

Category	Metric	Description
	Threat protection rules	Displays the graphical representation of the events based on the rule violations during the specified time.
	Threat protection events	Displays the threat protection event details such as Time, filter name, rule name, resource path, server host, and request time.
Audit logs	Time	Displays the time the event occurred.
	User	Displays the name of the user who caused the event.
	Status	Displays the current status of the transaction. The available values are: <ul style="list-style-type: none"> ■ SUCCESS ■ FAILURE
	Source machine	Displays the host name of the machine on which the event occurred.
	Object type	Displays the type of API Gateway object on which the event occurred. The available values are: <ul style="list-style-type: none"> ■ ACCESS_PROFILE_MANAGEMENT ■ ALIAS_MANAGEMENT ■ ANALYTICS_MANAGEMENT ■ API_MANAGEMENT ■ APPLICATION_MANAGEMENT ■ APPROVALS_MANAGEMENT ■ GROUPS_MANAGEMENT ■ PACKAGE_MANAGEMENT ■ PLAN_MANAGEMENT ■ PROMOTION_MANAGEMENT ■ POLICY_MANAGEMENT ■ USER_MANAGEMENT
	Object	Displays the UUID that uniquely identifies the object in the database.

Category	Metric	Description
	Message	Displays the success message or error message as a result of the event.
	Client IP address	Displays the IP address of the machine on which the event occurred.
	Action	<p>Displays the type of action for the event. The available values are:</p> <ul style="list-style-type: none"> ■ LOGIN ■ LOGOUT ■ CREATE ■ UPDATE ■ DELETE ■ ACTIVATE ■ DEACTIVATE
	Payload	Displays the content of data payload for the event.
Cache statistics:	Cache counts	Displays the hit, miss, and eviction count for API invocations across API Gateway.
	Cache usage statistics	Displays the cache usage size and the free size as a bar chart.
Application logs	Application logs saved search	<p>Displays a table that lists the cumulative logs collected across sources with details of each log that is collected in the time interval specified in the filter.</p> <p>These are the details displayed in the form of a table:</p> <ul style="list-style-type: none"> ■ Time. Specifies the date and time when the log was collected. ■ node. Specifies the node from which the log is generated. ■ fileType. Specifies the file type to which the logs belong. The following are the file types to which a log can belong: <ul style="list-style-type: none"> ■ <code>APIGatewayServerLogs</code> ■ <code>APIGatewayUILogs</code>

Category	Metric	Description
		<ul style="list-style-type: none"> ■ PlatformLogs ■ OSGILogs ■ WrapperLogs ■ InternalDataStoreLogs ■ DashboardLogs <ul style="list-style-type: none"> ■ logLevel. Specifies the log level. ■ message. Displays the actual message for the event for which the log was saved. ■ correlationId. Specifies the correlation id that applies to the API Gateway server logs with which you can identify a particular request. <p>You can expand each entry to view details of the actual log in the tabular or a JSON format.</p> <p>In addition you can create a filter to display the logs based on their id, index, type, correlation id, and so on. This helps in analyzing the events effectively.</p>
	Source vs log level	<p>Displays the log data per source per log level for the specified time interval.</p> <p>The data is displayed in the form of a pie chart.</p> <p>Hover the cursor over the piechart to view the following details.</p> <p>The inner section of the pie chart displays the number of logs collected per file type. The corresponding outer section displays the log levels for the logs collected for that file type.</p>
	Log level tag cloud	<p>Displays the log levels available and shortcut filters to filter the logs by log levels.</p> <p>Click on one of the log levels. You now see the logs for the specified log level in the table under Application logs saved search and the distribution of the selected logs per sources that produced them in the pie chart under Source vs log level.</p>

Category	Metric	Description
API usage details	API Gateway invocation usage	This bar chart displays the trending of API invocations across API Gateway. Hover the cursor over the bar chart to see the number of API invocations for the current month.
	API Gateway invocation usage details	Displays the details of the number of API invocations for each month.
	API usage details	Displays the API invocation details for each API such as API Name, API usage for each month and year.

API-specific Dashboard

You can view the API-specific dashboard by navigating to the API details page and clicking the **Analytics** icon for the specific API. The dashboard displays the following analytics based on the metrics monitored.

You can select the time interval and click **Apply filter** to filter the analytics based on the time interval chosen. You can select the time interval from the drop-down options.

If you select custom, you can type the **From date** and **To date** to specify the time interval for which you want to view the API-specific analytics.

For the specified time interval you can also filter based on an API. The API drop-down list displays all the APIs. On selecting an API, the data displayed is for the selected API.

You can click on the specific event in the list under Legend to view the specific event in any of the widgets. You can view additional details for an event by hovering the cursor over a particular color in the graphical representations.

Metric	Description
Events over time	Displays the trending of events generated by the selected API over time.
API invocations	Displays the number of time the API was invoked during the specified time.
API invocation pattern	Displays API invocation over period of time during the specified time interval in the form of a line graph.
Native service performance	Displays information on how fast the native service responds to the request received in the specified time based on the data in the transactional event.

Metric	Description
Response code trend	Displays the trend based on the response codes received from various events for the API during the specified time.
API trend by response	Displays the trending of the selected API based on the response time from the performance metrics for that API.
Success vs Failure	Displays the trending of API based on its success rate as compared to its failure rate in the performance metrics for the specified time.
Runtime events	Displays the run time event details for the selected API. Displays information on the event type, date when the event was created, the agent on which the event was generated, description of the alert generated, the source of event, and the application that generated the event.
Service result cache	Displays a bar graph showing the number of responses served from cache and the number of responses fetched from the native service at the operation level for the selected API during the specified time.
Method level invocations	<p>Displays the method level invocations per operation for the API during the specified time.</p> <p>You can hover the cursor over the stacked bar chart to view the various methods invoked per operation or resource and also the operations or resources for the selected API during the specified time.</p>

Runtime Events and Metrics Data Model

API Gateway generates runtime events and Key Performance Indicator (KPI) metrics for the currently active APIs. The types of runtime events that API Gateway can generate are:

Events	Description
Lifecycle	A Lifecycle event is generated each time the API Gateway instance is started or shut down.
Error	An Error event is generated each time an invocation of API results in an error.
Policy Violation	A Policy Violation event is generated each time an invocation of API violates a policy that was configured for the API.
Transaction	A Transaction event is generated each time an API is invoked (successfully or unsuccessfully).

Events	Description
Monitor	A Monitor event is generated when a configured SLA parameter, such as the average response time, fault count, availability, and so on, is breached for the API.

KPI metrics are used to monitor the run-time execution of APIs. Metrics include the maximum response time, average response time, fault count, availability of APIs, and so on. If you include runtime monitoring policies, the policies will monitor the KPI metrics for APIs, and can send alerts to various destinations when user-specified performance conditions for an API are violated. The KPI metrics that API Gateway can generate are:

Metric	Reports on...
Availability	The percentage of time that an API was available during the current interval. A value of 100 indicates that the API was always available. Only the time when the API is unavailable counts against this metric. If invocations fail due to policy violations, this parameter could still be as high as 100.
Average Response Time	The average amount of time it took the API to complete all invocations in the current interval. This is measured from the moment API Gateway receives the request until the moment it returns the response to the client.
Fault Count	The number of failed invocations in the current interval.
Maximum Response Time	The maximum amount of time it took the API to complete an invocation in the current interval.
Minimum Response Time	The minimum amount of time it took the API to complete an invocation in the current interval.
Successful Request Count	The number of successful API invocations in the current interval.
Total Request Count	The total number of requests for each API running in API Gateway in the current interval.

You can configure API Gateway to publish the runtime events and metrics data to different destinations. The following sections describe the runtime events and metrics data model for each of these destinations:

- API Gateway
- API Portal
- Audit Log
- CentraSite
- Elasticsearch

- Email
- JDBC
- Local Log

API Gateway

The runtime events and metrics payload is generated by API Gateway at run-time. The columns that make up the events and metrics data model for API Gateway are listed below:

Transactional Events

Column	Description
apild	The unique identifier for the API. Example: c0f84954-9732-11e5-b9f4-f159eafe47b1
apiName	Name of the API in which the event occurred. Example: pet1
apiVersion	The system-assigned version identifier for the API. Example: 1.0
applicationId	The unique identifier for the application associated with the API invocation. Example: c0f84954-9732-11e5-b9f4-f159eafe47b2
applicationIp	IP address of the application associated with the API invocation. Example: 10.20.248.33
applicationName	Name of the application associated with the API invocation. An application name is populated as unknown when API Gateway is unable to identify the application using a security policy that is configured for the API. Example: SampleApplication
callbackRequest	Indicates whether the event is generated for a callback request. Possible values are: <ul style="list-style-type: none">■ true. This denotes that the event is generated for a callback request.■ false. This denotes that the event is generated for a normal response.

Column	Description
correlationID	<p>The unique identifier that is automatically generated for every request coming to API Gateway and can be used to query the log.</p> <p>Example: MED38e9cfa4-2348-408b-9462-124b2181c1a6:656</p>
creationDate	<p>Date and time when the event was generated in API Gateway.</p> <p>Example: 1501671101509</p>
customFields	<p>The custom fields an API Provider can provide to log a new field and value for a transaction event.</p> <p>Example: {"customfield":"customvalue"}</p>
errorOrigin	<p>The origin of error.</p> <p>Example: Nativeservice</p>
eventType	<p>The type of event that occurred.</p> <p>Example: Transactional</p>
externalCalls	<p>List the external calls from API Gateway. These external calls can be to a native service or service registry.</p> <p>Example:</p> <pre>[{ "externalCallType": "SERVICE_REGISTRY_CALL", "externalURL": "http://service.registry.com", "callDuration": 49, "callStartTime": 1562244570486, "callEndTime": 1562244570535, "responseCode": "200" }, { "externalCallType": "NATIVE_SERVICE_CALL", "externalURL": "https://petstore.swagger.io/v2/store/inventory", "callDuration": 1285, "callStartTime": 1562244569252, "callEndTime": 1562244570537, "responseCode": "200" }]</pre>
gatewayTime	<p>Duration in milliseconds, to process a request by API Gateway. This does not include native service processing duration.</p> <p>gatewayTime = totalTime - providerTime</p> <p>Example: 20</p>
httpMethod	<p>The HTTP method used to invoke the API.</p>

Column	Description
	Example: GET
nativeHttpMethod	The HTTP method used to invoke the native service. Example: GET
nativeReqPayload	The native service request data. Example: <pre>{ "param1" : "value1", "param2" : 10 }</pre>
nativeRequestHeaders	Request header in the incoming request from the API Gateway to native service. Example: <pre>{ "Authorization":"*****", "Accept": "*/*", "Authorization": "*****", "Accept":"*/*", "Cache-Control": "no-cache", "User-Agent": "PostmanRuntime/7.13.0", "Postman-Token": "381424fa-e3b3-4058-8df9-4abf9d72c899", "postmanHeader": "hello", "accept-encoding": "gzip, deflate", "Content-Type": "application/x-www-form-urlencoded" }</pre>
nativeResPayload	The native service response data. Example: <pre>{ "id":2, "category": { "id":2, "name":"string" }, "name":"pysen", "photoUrls":["string"], "tags": [{ "id":0, "name":"string" }], "status":"available" }</pre>

Column	Description
nativeResponseHeaders	<p>Response header in the outgoing response from the native service to API Gateway.</p> <p>Example:</p> <pre>{ "Server": "Jetty(9.2.9.v20150224)", "Access-Control-Allow-Origin": "*", "Access-Control-Allow-Methods": "GET, POST, DELETE, PUT", "Connection": "close", "Date": "Fri, 07 Jun 2019 12:44:13 GMT", "Access-Control-Allow-Headers": "Content-Type, api_key, Authorization", "Content-Type": "application/json" }</pre>
nativeURL	<p>URL of the native service.</p> <p>Example: http://petstore.swagger.io/v2/pet/2</p>
operationName	<p>Name of the API operation that is invoked.</p> <p>Example: Using a Calculator API, you can perform various operations such as addition, subtraction, multiplication, and division. When an addition operation is invoked in API Gateway, then the operation field name is populated as addInts.</p>
queryParameters	<p>This is applicable only for REST APIs. Query parameters present in the incoming REST request.</p> <p>Example: {"status": "available"}</p>
reqPayload	<p>The API request payload data.</p> <p>Example: RequestPayload</p>
requestHeaders	<p>Request header in the incoming request from the client.</p> <p>Example:</p> <pre>{ "Cache-Control": "max-age=0", "Accept": "text/plain,application/json;q=0.9,image/webp,image/apng,*/*;q=0.8", "Upgrade-Insecure-Requests": "1", "Connection": "keep-alive", "User-Agent": "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36", "Host": "mcdaso02:5555", "Accept-Encoding": "gzip, deflate", "Accept-Language": "en-US,en;q=0.9,ta;q=0.8", }</pre>

Column	Description
	<pre>"Content-Type":"application/x-www-form-urlencoded" }</pre>
responseCode	<p>The HTTP response status code that indicates success or failure of the requested operation.</p> <p>Example: 200</p>
responseHeaders	<p>Response header in the outgoing response.</p> <p>Example:</p> <pre>{ "Server":"Jetty(9.2.9.v20150224)", "Access-Control-Allow-Origin":"*", "Access-Control-Allow-Methods":"GET,POST, DELETE, PUT", "Connection":"close", "Date":"Fri, 30 Mar 2018 08:25:45 GMT", "Access-Control-Allow-Headers":"Content-Type, api_key,Authorization", "Content-Type":"application/xml" }</pre>
serverID	<p>The API Gateway server on which the transaction event occurred. This column is currently not used. It appears as NULL or as an empty string.</p> <p>Example: SampleHost:80</p>
sessionId	<p>A string the API Gateway server generates to uniquely identify each session. This is either the IS session token or the automatically generated GUID if the token is missing from the message context.</p> <p>Example: c0f84954-9732-11e5-b9f4-f159eafe47b2</p>
sourceGateway	<p>Source of event generation.</p> <p>Possible values: APIGateway or Microgateway.</p>
sourceGatewayNode	<p>Source API Gateway's IP address.</p> <p>Example:10.0.75.1</p>
status	<p>Status of the API request.</p> <p>Possible values are: SUCCESS, FAILURE</p>
totalDataSize	<p>The total combined size of request and response payloads in bytes.</p> <p>Example: 100</p>
totalTime	<p>Time in milliseconds required to invoke the API provider. This time includes the overhead incurred by API Gateway. Overhead includes</p>

Column	Description
	security overhead for encryption, decryption, and load-balance retries. Example: 120

Error Events

Column	Description
apild	The unique identifier for the API. Example: c0f84954-9732-11e5-b9f4-f159eafe47b1
apiName	Name of the API in which the event occurred. Example: SampleAPI
apiVersion	The system-assigned version identifier for the API. Example: 1.0
applicationId	The unique identifier for the application associated with the API invocation. Example: c0f84954-9732-11e5-b9f4-f159eafe47b2
applicationIp	IP address of the application associated with the API invocation. Example: 10.20.248.33
applicationName	Name of the application associated with the API invocation. An application name is populated as unknown when API Gateway is unable to identify the application using a security policy that is configured for the API. Example: SampleApplication
correlationID	The unique identifier that is automatically generated for every request coming to API Gateway and can be used to query the log. Example: MED38e9cfa4-2348-408b-9462-124b2181c1a6:656
creationDate	Date and time when the event was generated in API Gateway. Example: 1501671101509
errorDesc	Message that describes the error that occurred. Example: Invocation for SampleAPI was rejected based on policy violation, response code: 503

Column	Description
eventType	The type of event that occurred. Example: Error Event
operationName	Name of the API operation that is invoked. Example: Using a Calculator API, you can perform various operations such as addition, subtraction, multiplication, and division. When an addition operation is invoked in API Gateway, then the operation field name is populated as addInts .
responseCode	The HTTP response status code that indicates success or failure of the requested operation. Example: 200
sessionId	A string the API Gateway server generates to uniquely identify each session. This is either the IS session token or the automatically generated GUID if the token is missing from the message context. Example: c0f84954-9732-11e5-b9f4-f159eafe47b2
sourceGateway	Source of event generation. Possible values: APIGateway or Microgateway .
sourceGatewayNode	Source API Gateway's IP address. Example: 10.0.75.1
userAgent	Name of the client used to invoke the API. Example: Postman

Monitoring Events

Column	Description
alertDesc	Text of the alert message sent to a configured destination when the performance conditions are violated. The alert message is specified in the policy definition of an API. Example: EnforcePolicy-HardLimit
alertType	The type of alert generated for the event. Possible values are: Monitor , sla
apid	The unique identifier for the API. Example: c0f84954-9732-11e5-b9f4-f159eafe47b1

Column	Description
apiName	Name of the API in which the event occurred. Example: SampleAPI
apiVersion	The system-assigned version identifier for the API. Example: 1.0
applicationId	The unique identifier for the application associated with the API invocation. Example: c0f84954-9732-11e5-b9f4-f159eafe47b2
applicationIp	IP address of the application associated with the API invocation. Example: 10.20.248.33
applicationName	Name of the application associated with the API invocation. An application name is populated as unknown when API Gateway is unable to identify the application using a security policy that is configured for the API. Example: SampleApplication
creationDate	Date and time when the event was generated in API Gateway. Example: 1501671101509
eventSource	The source where the event occurred. Example: API_Gateway_Instance
eventType	The type of event that occurred. Example: Monitor Event
httpMethod	The HTTP method used to invoke the API. Example: GET
operationName	Name of the API operation that is invoked. Example: Using a Calculator API, you can perform various operations such as addition, subtraction, multiplication, and division. When an addition operation is invoked in API Gateway, then the operation field name is populated as addInts .
sessionId	A string the API Gateway server generates to uniquely identify each session. This is either the IS session token or the automatically generated GUID if the token is missing from the message context. Example: c0f84954-9732-11e5-b9f4-f159eafe47b2

Column	Description
sourceGateway	Source of event generation. Possible values: APIGateway or Microgateway .
sourceGatewayDetails	Details of events generated only from Microgateway. The details include Microgateway Id, Source Gateway Host, Source Gateway Port, Source Gateway Version, Microgateway pool.

Lifecycle Events

Column	Description
creationDate	Date and time when the event was generated in API Gateway. Example: 1501671101509
eventType	The type of event that occurred. Example: LifeCycle
gatewayStatus	Status of the API Gateway instance. Possible values are: STARTED or STOPPED
sourceGateway	Source of event generation. Possible values: APIGateway or Microgateway .
sourceGatewayDetails	Details of events generated only from Microgateway. The details include Microgateway Id, Source Gateway Host, Source Gateway Port, Source Gateway Version, Microgateway pool.

Policy Violation Events

Column	Description
alertDesc	Text of the alert message sent to a configured destination when the performance conditions are violated. The alert message is specified in the policy definition of an API. Example: A violation was detected for policy (Unknown-Policyuser): application could not be identified. Anonymous access is not allowed for this service!
alertSource	Name of the API Gateway policy that generated the alert message. Example: Unknown-Policy

Column	Description
alertType	The type of alert generated for the event. Example: PolicyViolation
apild	The unique identifier for the API. Example: c0f84954-9732-11e5-b9f4-f159eafe47b1
apiName	Name of the API in which the event occurred. Example: SampleAPI
apiVersion	The system-assigned version identifier for the API. Example: 1.0
applicationId	The unique identifier for the application associated with the API invocation. Example: c0f84954-9732-11e5-b9f4-f159eafe47b2
applicationIp	IP address of the application associated with the API invocation. Example: 10.20.248.33
applicationName	Name of the application associated with the API invocation. An application name is populated as unknown when API Gateway is unable to identify the application using a security policy that is configured for the API. Example: SampleApplication
creationDate	Date and time when the event was generated in API Gateway. Example: 1501671101509
eventType	The type of event that occurred. Example: Policy Violation Event
httpMethod	The HTTP method used to request the API access. Example: GET
operationName	Name of the API operation that is invoked. Example: Using a Calculator API, you can perform various operations such as addition, subtraction, multiplication, and division. When an addition operation is invoked in API Gateway, then the operation field name is populated as addInts .

Column	Description
responseCode	The HTTP response status code that indicates success or failure of the requested operation. Example: 200
sessionId	A string the API Gateway server generates to uniquely identify each session. This is either the IS session token or the automatically generated GUID if the token is missing from the message context. Example: c0f84954-9732-11e5-b9f4-f159eafe47b2
sourceGateway	Source of event generation. Possible values: APIGateway or Microgateway .
sourceGatewayNode	Source API Gateway's IP address. Example: 10.0.75.1
userAgent	Name of the client used to invoke the API. Example: Postman

Performance Metrics

Column	Description
apild	The unique identifier for the API. Example: c0f84954-9732-11e5-b9f4-f159eafe47b1
apiName	Name of the API in which the event occurred. Example: SampleAPI
apiVersion	The system-assigned version identifier for the API. Example: 1.0
availability	The percentage of time that an API was available during the current interval. A value of 100 indicates that the API was always available. If invocations fail due to policy violations, this parameter could still be as high as 100. Example: 100
avgResponseTime	The average amount of time it took the API to complete each invocation in the current interval. Response time is measured from the moment API Gateway receives the request until the moment it returns the response to the caller.

Column	Description
	Example: 135
creationDate	Date and time when the event was generated in API Gateway. Example: 1501671101509
eventType	The type of event that occurred. Example: Performance Metrics Event
faultCount	The number of failed API invocations in the current interval. Example: 10
intervalStart	The starting date and time from which you want to examine metrics. Example: 1526294632172
intervalStop	The ending date and time until which you want to examine metrics. Example: 1526294632182
maxResponseTime	The maximum amount of time (in milliseconds) it took for the API to complete an invocation in the current interval. Example: 343
minResponseTime	The minimum amount of time (in milliseconds) it took for the API to complete an invocation in the current interval. Example: 10
sourceGateway	Source of event generation. Possible values: APIGateway or Microgateway .
successCount	The number of successful API invocations in the current interval. Example: 100
totalCount	The total number of API invocations (successful and unsuccessful) in the current interval. Example: 110

Threat Protection Events

Column	Description
id	The unique identifier for an event.

Column	Description
	Example: 8e05267a-45c9-45f0-a3dd-8b2ee1e98ca2
alertAction	A helpful action taken on the API for the alert. Example: DENY
creationDate	Date and time when the event was generated in API Gateway. Example: 1501671101509
eventType	The type of event that occurred. Example: Transactional
filterName	Name of the threat protection filter. Example: DoSFilter
message	If the API invocation failed, message that describes the error that occurred. Example: Global Denial of Service limits were reached: Maximum requests limit of 2 in 120 seconds has been exceeded.
requestHost	Hostname of the machine from which the API access request was submitted. Example: 10.60.34.152
requestTime	Date and time the request was submitted. Example: 1501671101509
requestType	The type of request that was received for the API. Example: ALL
resourcePath	The relative URI path of a resource that was used for API invocation. Example: invoke/pub.date/getCurrentDate
ruleName	The API Gateway rule that triggered the event. Example: GlobalDoSRule
serverHost	The name or IP address of the machine on which the thread protection server is running. Example: 10.60.34.83
serverPort	The port number on which the thread protection server is configured to listen for incoming requests.

Column	Description
	Example: 8911

API Portal

The runtime events and metrics payload generated by API Gateway at run-time is published to the configured API Portal destination. The columns that make up the events and metrics data model for API Portal are listed below:

Transactional Events

Column	Description
apild	The unique identifier for the API. Example: c0f84954-9732-11e5-b9f4-f159eafe47b1
apiName	Name of the API in which the event occurred. Example: SampleAPI
apiVersion	The system-assigned version identifier for the API. Example: 1.0
consumerId	The unique identifier for the consumer associated with the API invocation. Example: c0f84954-9732-11e5-b9f4-f159eafe47b2
consumerIp	IP address of the consumer associated with the API invocation. Example: 10.1.1.211
consumerName	Name of the consumer associated with the API invocation. A consumer name is populated as unknown when API Gateway is unable to identify the consumer using a security policy that is configured for the API. Example: SampleApplication
correlationID	The unique identifier that is automatically generated for every request coming to API Gateway and can be used to query the log. Example: MED38e9cfa4-2348-408b-9462-124b2181c1a6:656
creationDate	Date and time when the event was generated in API Gateway. Example: 1501671101509

Column	Description
customFieldsrequest	<p>The custom fields an API Provider can provide to log a new field and value for a transaction event.</p> <p>Example: {"customfield":"customvalue"}</p>
errorOrigin	<p>The origin of error.</p> <p>Example: Nativeservice</p>
eventType	<p>The type of event that occurred.</p> <p>Example: Transactional</p>
externalCalls	<p>List the external calls from API Gateway. These external calls can be to a native service or service registry.</p> <p>Example:</p> <pre>[{ "externalCallType":"SERVICE_REGISTRY_CALL", "externalURL":"http://service.registry.com", "callDuration":49, "callStartTime":1562244570486, "callEndTime":1562244570535, "responseCode": "200" }, { "externalCallType":"NATIVE_SERVICE_CALL", "externalURL": "https://petstore.swagger.io/v2/store/inventory", "callDuration":1285, "callStartTime":1562244569252, "callEndTime":1562244570537, "responseCode":"200" }]</pre>
gatewayTime	<p>Duration in milliseconds, to process a request by API Gateway. This does not include native service processing duration.</p> <p>$gatewayTime = totalTime - providerTime$</p> <p>Example: 20</p>
operationName	<p>Name of the API operation that is invoked.</p> <p>Example: Using a Calculator API, you can perform various operations such as addition, subtraction, multiplication, and division. When an addition operation is invoked in API Gateway, then the operation field name is populated as addInts.</p>
providerTime	<p>Time in milliseconds required for API Gateway to invoke a native provider and receive a response. This time includes the overhead incurred by API Gateway. Overhead includes the time it takes for a</p>

Column	Description
	<p>provider to process a request and return a response, plus any network latency to or from the provider. Subtracting total time from provider time must give a rough indicator of the API Gateway overhead.</p> <p>Example: 20</p>
queryParameters	<p>This is applicable only for REST APIs. Query parameters present in the incoming REST request.</p> <p>Example: {"status":"available"}</p>
requestHeaders	<p>Request header in the incoming request from the client.</p> <p>Example:</p> <pre>{ "Cache-Control": "max-age=0", "Accept": "text/plain,application/json;q=0.9,image/webp,image/apng,*/*;q=0.8", "Upgrade-Insecure-Requests": "1", "Connection": "keep-alive", "User-Agent": "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36", "Host": "mcdaso02:5555", "Accept-Encoding": "gzip,deflate", "Accept-Language": "en-US,en;q=0.9,ta;q=0.8", "Content-Type": "application/x-www-form-urlencoded" }</pre>
response	<p>The API response payload data.</p> <p>Example: <ResponsePayload></p>
responseCode	<p>The HTTP response status code that indicates success or failure of the requested operation.</p> <p>Example: 200</p>
responseHeaders	<p>Response header in the outgoing response.</p> <p>Example:</p> <pre>{ "Server": "Jetty(9.2.9.v20150224)", "Access-Control-Allow-Origin": "*", "Access-Control-Allow-Methods": "GET,POST,DELETE,PUT", "Connection": "close", "Date": "Fri, 30 Mar 2018 08:25:45 GMT", "Access-Control-Allow-Headers": "Content-Type,api_key,Authorization", "Content-Type": "application/xml" }</pre>

Column	Description
nativeHttpMethod	The HTTP method used to invoke the native service. Example: GET
nativeRequestHeaders	Request header in the incoming request from the API Gateway to native service. Example: <pre>{ "Authorization": "*****", "Accept": "*/*", "Authorization": "*****", "Accept": "*/*", "Cache-Control": "no-cache", "User-Agent": "PostmanRuntime/7.13.0", "Postman-Token": "381424fa-e3b3-4058-8df9-4abf9d72c899", "postmanHeader": "hello", "accept-encoding": "gzip, deflate", "Content-Type": "application/x-www-form-urlencoded" }</pre>
nativeRequestPayload	The native service request data. Example: <pre>{ "param1" : "value1", "param2" : 10 }</pre>
nativeResonsePayload	The native service response data. Example: <pre>{ "id":2, "category": { "id":2, "name":"string" }, "name":"pysen", "photoUrls":["string"], "tags": [{ "id":0, "name":"string" }], "status":"available" }</pre>
nativeURL	URL of the native service. Example: http://petstore.swagger.io/v2/pet/2

Column	Description
sessionId	A string the API Gateway server generates to uniquely identify each session. This is either the IS session token or the automatically generated GUID if the token is missing from the message context. Example: c0f84954-9732-11e5-b9f4-f159eafe47b2
sourceGatewayNode	Source API Gateway's IP address. Example: 10.0.75.1
status	Status of the API request. Possible values are: SUCCESS, FAILURE
targetName	Name of the API Gateway instance reporting the event. Example: API_Gateway_Instance
totalDataSize	The total combined size of request and response payloads in bytes. Example: 100
totalTime	Time in milliseconds required to invoke the API provider. This time includes the overhead incurred by API Gateway. Overhead includes security overhead for encryption, decryption, and load-balance retries. Example: 120

Error Events

Column	Description
apild	The unique identifier for the API. Example: c0f84954-9732-11e5-b9f4-f159eafe47b1
apiName	Name of the API in which the event occurred. Example: SampleAPI
apiVersion	The system-assigned version identifier for the API. Example: 1.0
consumerId	The unique identifier for the consumer associated with the API invocation. Example: c0f84954-9732-11e5-b9f4-f159eafe47b2
consumerIp	IP address of the consumer associated with the API invocation.

Column	Description
	Example: 10.20.248.33
consumerName	<p>Name of the consumer associated with the API invocation.</p> <p>A consumer name is populated as unknown when API Gateway is unable to identify the consumer using a security policy that is configured for the API.</p> <p>Example: SampleApplication</p>
correlationID	<p>The unique identifier that is automatically generated for every request coming to API Gateway and can be used to query the log.</p> <p>Example: MED38e9cfa4-2348-408b-9462-124b2181c1a6:656</p>
creationDate	<p>Date and time when the event was generated in API Gateway.</p> <p>Example: 1501671101509</p>
errorDesc	<p>Message that describes the error that occurred.</p> <p>Service invocation for SampleAPI was rejected based on policy violation, response code: 503</p>
eventType	<p>The type of event that occurred.</p> <p>Example: Error Event</p>
operationName	<p>Name of the API operation that is invoked.</p> <p>Example: Using a Calculator API, you can perform various operations such as addition, subtraction, multiplication, and division. When an addition operation is invoked in API Gateway, then the operation field name is populated as addInts.</p>
responseCode	<p>The HTTP response status code that indicates success or failure of the requested operation.</p> <p>Example: 503</p>
sessionId	<p>A string the API Gateway server generates to uniquely identify each session. This is either the IS session token or the automatically generated GUID if the token is missing from the message context.</p> <p>Example: c0f84954-9732-11e5-b9f4-f159eafe47b2</p>
targetName	<p>Name of the API Gateway instance reporting the event.</p> <p>Example: API_Gateway_Instance</p>

Monitoring Events

Column	Description
alertDesc	Text of the alert message sent to a configured destination when the performance conditions are violated. The alert message is specified in the policy definition of an API. Example: EnforcePolicy-HardLimit
alertSource	Name of the API Gateway policy that generated the alert message. Example: Unknown-Policy
alertType	The type of alert generated for the event. Example: sla
apild	The unique identifier for the API. Example: c0f84954-9732-11e5-b9f4-f159eafe47b1
apiName	Name of the API in which the event occurred. Example: SampleAPI
apiVersion	The system-assigned version identifier for the API. Example: 1.0
applicationId	The unique identifier for the application associated with the API invocation. Example: c0f84954-9732-11e5-b9f4-f159eafe47b2
applicationIp	IP address of the application associated with the API invocation. Example: 10.20.248.33
applicationName	Name of the application associated with the API invocation. An application name is populated as unknown when API Gateway is unable to identify the application using a security policy that is configured for the API. Example: SampleApplication
creationDate	Date and time when the event was generated in API Gateway. Example: 1501671101509
eventType	The type of event that occurred. Example: Monitor Event

Column	Description
monitorAttr	The monitored attribute which has breached the configured SLA. Example: AVGRESPONSETIME GT 1.0, SUCCESSCOUNT EQ 3, REQUESTCOUNT GT 10
operationName	Name of the API operation that is invoked. Example: Using a Calculator API, you can perform various operations such as addition, subtraction, multiplication, and division. When an addition operation is invoked in API Gateway, then the operation field name is populated as addInts .
sessionId	A string the API Gateway server generates to uniquely identify each session. This is either the IS session token or the automatically generated GUID if the token is missing from the message context. Example: c0f84954-9732-11e5-b9f4-f159eafe47b2
targetName	Name of the API Gateway instance reporting the event. Example: API_Gateway_Instance

Lifecycle Events

Column	Description
creationDate	Date and time when the event was generated in API Gateway. Example: 1501671101509
eventStatus	Status of the API Gateway instance. Possible values are: STARTED or STOPPED
eventDesc	
eventType	The type of event that occurred. Example: LifeCycle
targetName	Name of the API Gateway instance reporting the event. Example: API_Gateway_Instance

Policy Violation Events

Column	Description
alertDesc	Text of the alert message sent to a configured destination when the performance conditions are violated. The alert message is specified in the policy definition of an API. Example: A violation was detected for policy (Unknown-Policyuser): application could not be identified. Anonymous access is not allowed for this service!
alertSource	Name of the API Gateway policy that generated the alert message. Example: Unknown-Policy
alertType	The type of alert generated for the event. Example: PolicyViolation
apild	The unique identifier for the API. Example: c0f84954-9732-11e5-b9f4-f159eafe47b1
apiName	Name of the API in which the event occurred. Example: SampleAPI
apiVersion	The system-assigned version identifier for the API. Example: 1.0
consumerId	The unique identifier for the consumer associated with the API invocation. Example: c0f84954-9732-11e5-b9f4-f159eafe47b2
consumerIp	IP address of the consumer associated with the API invocation. Example: 10.20.248.33
consumerName	Name of the consumer associated with the API invocation. A consumer name is populated as unknown when API Gateway is unable to identify the consumer using a security policy that is configured for the API. Example: SampleApplication
creationDate	Date and time when the event was generated in API Gateway. Example: 1501671101509
eventType	The type of event that occurred.

Column	Description
	Example: Policy Violation Event
operationName	Name of the API operation that is invoked. Example: Using a Calculator API, you can perform various operations such as addition, subtraction, multiplication, and division. When an addition operation is invoked in API Gateway, then the operation field name is populated as addInts .
responseCode	The HTTP response status code that indicates success or failure of the requested operation. Example: 200
sessionId	A string the API Gateway server generates to uniquely identify each session. This is either the IS session token or the automatically generated GUID if the token is missing from the message context. Example: c0f84954-9732-11e5-b9f4-f159eafe47b2
targetName	Name of the API Gateway instance reporting the event. Example: API_Gateway_Instance

Performance Metrics

Column	Description
apild	The unique identifier for the API. Example: c0f84954-9732-11e5-b9f4-f159eafe47b1
apiName	Name of the API in which the event occurred. Example: SampleAPI
apiVersion	The system-assigned version identifier for the API. Example: 1.0
availability	The percentage of time that an API was available during the current interval. A value of 100 indicates that the API was always available. If invocations fail due to policy violations, this parameter could still be as high as 100. Example: 100
avgResponseTime	The average amount of time it took the API to complete each invocation in the current interval. Response time is measured from

Column	Description
	<p>the moment API Gateway receives the request until the moment it returns the response to the caller.</p> <p>Example: 135</p>
creationDate	<p>Date and time when the event was generated in API Gateway.</p> <p>Example: 1501671101509</p>
eventType	<p>The type of event that occurred.</p> <p>Example: Performance Metrics Event</p>
faultCount	<p>The number of failed API invocations in the current interval.</p> <p>Example: 10</p>
includeFaults	<p>Includes failed API invocations.</p> <p>Possible values are: true, false</p>
intervalStart	<p>The starting date and time from which you want to examine metrics.</p> <p>Example: 2015-08-26 04:13:35 PM</p>
intervalStop	<p>The ending date and time until which you want to examine metrics.</p> <p>Example: 2015-08-26 04:13:45 PM</p>
maxResponseTime	<p>The maximum amount of time (in milliseconds) it took for the API to complete an invocation in the current interval.</p> <p>Example: 343</p>
minResponseTime	<p>The minimum amount of time (in milliseconds) it took for the API to complete an invocation in the current interval.</p> <p>Example: 10</p>
operationName	<p>Name of the API operation that is invoked.</p> <p>Example: Using a Calculator API, you can perform various operations such as addition, subtraction, multiplication, and division. When an addition operation is invoked in API Gateway, then the operation field name is populated as addInts.</p>
successCount	<p>The number of successful API invocations in the current interval.</p> <p>Example: 100</p>
targetName	<p>Name of the API Gateway instance reporting the event.</p> <p>Example: API_Gateway_Instance</p>

Column	Description
totalCount	The total number of API invocations (successful and unsuccessful) in the current interval. Example: 110

Audit Log

The runtime events and metrics payload generated by API Gateway at run-time is published to the configured Audit Log destination. The columns that make up the events and metrics data model for Audit Log are listed below:

Transactional Events

Column	Description
API_ID	The unique identifier for the API. Example: ec1473cc-40a0-479e-9126-474a917c3c89
API_NAME	Name of the API in which the event occurred. Example: SampleAPI
API_VERSION	The system-assigned version identifier for the API. Example: 1.0
AUDITTIMESTAMP	Date and time when the event was written to the log. Example: 2017-08-07 07:22:22
CONSUMER_IP	IP address of the consumer associated with the API invocation. Example: 10.60.37.42
CONSUMER_NAME	Name of the consumer associated with the API invocation. A consumer name is populated as unknown when API Gateway is unable to identify the consumer using a policy that is configured for the API. Example: SampleApplication
CONTEXTID	The unique identifier for the current context information API Gateway uses to connect related entries from different logs. This column is currently not used. It appears as NULL or as an empty string.

Column	Description
	Example: 81546147-41a8-4998-8150-02ba67bb08c2
CORRELATIONID	The unique identifier that is automatically generated for every request coming to API Gateway and can be used to query the log. Example: MED38e9cfa4-2348-408b-9462-124b2181c1a6:656
CUSTOMFIELDS	The custom fields an API Provider can provide to log a new field and value for a transaction event. Example: {"customfield":"customvalue"}
ERROR_ORIGIN	The origin of error. Example: Nativeservice
EVENT_PK	The primary key (PK) that uniquely identifies the event that occurred. Example: 1
EXTERNAL_CALLS	List the external calls from API Gateway. These external calls can be to a native service or service registry. Example: <pre>[{ "externalCallType":"SERVICE_REGISTRY_CALL", "externalURL":"http://service.registry.com", "callDuration":49, "callStartTime":1562244570486, "callEndTime":1562244570535, "responseCode": "200" }, { "externalCallType":"NATIVE_SERVICE_CALL", "externalURL": "https://petstore.swagger.io/v2/store/inventory", "callDuration":1285, "callStartTime":1562244569252, "callEndTime":1562244570537, "responseCode":"200" }]</pre>
INSERTTIMESTAMP	Date and time when the event was generated in API Gateway. Example: 2017-08-07 07:22:22
MSGID	The ID assigned to the message by the API provider. This column is currently not used. Example: 361dc2f8-a60b-fc21-8545-9b07fce1a479

Column	Description
NATIVE_ENDPOINT	The endpoint URL of the native API that is invoked. Example: http://petstore.swagger.io/v2/pet/55
NATIVE_HTTP_METHOD	The HTTP method used to invoke the native service. Example: GET
NATIVE_REQUEST_HEADERS	Request header in the incoming request from the API Gateway to native service. Example: <pre>{ "Authorization": "*****", "Accept": "*/*", "Authorization": "*****", "Accept": "*/*", "Cache-Control": "no-cache", "User-Agent": "PostmanRuntime/7.13.0", "Postman-Token": "381424fa-e3b3-4058-8df9-4abf9d72c899", "postmanHeader": "hello", "accept-encoding": "gzip, deflate", "Content-Type": "application/x-www-form-urlencoded" }</pre>
NATIVE_REQ_PAYLOAD	The native service request data. Example: <pre>{ "param1" : "value1", "param2" : 10 }</pre>
NATIVE_RESPONSE_HEADERS	Response header in the outgoing response from the native service to API Gateway. Example: <pre>{ "Server": "Jetty(9.2.9.v20150224)", "Access-Control-Allow-Origin": "*", "Access-Control-Allow-Methods": "GET, POST, DELETE, PUT", "Connection": "close", "Date": "Fri, 07 Jun 2019 12:44:13 GMT", "Access-Control-Allow-Headers": "Content-Type, api_key, Authorization", "Content-Type": "application/json" }</pre>
NATIVE_RES_PAYLOAD	The native service response data.

Column	Description
	<p>Example:</p> <pre>{ "id":2, "category": { "id":2, "name":"string" }, "name":"pysen", "photoUrls":["string"], "tags": [{ "id":0, "name":"string" }], "status":"available" }</pre>
NATIVE_URL	<p>URL of the native service.</p> <p>Example: http://petstore.swagger.io/v2/pet/2</p>
OPERATION_NAME	<p>Name of the API operation or resource that is invoked.</p> <p>Example: /pet/{petId}</p>
PROVIDER_TIME	<p>Time in milliseconds required for API Gateway to invoke a native provider and receive a response. This time includes the overhead incurred by API Gateway. Overhead includes the time it takes for a provider to process a request and return a response, plus any network latency to or from the provider. Subtracting total time from provider time must give a rough indicator of the API Gateway overhead.</p> <p>Example: 1336</p>
QUERY_PARAMETERS	<p>This is applicable only for REST APIs. Query parameters present in the incoming REST request.</p> <p>Example: {"status":"available"}</p>
REQUEST_HEADERS	<p>Request header in the incoming request from the client.</p> <p>Example:</p> <pre>{ "Cache-Control":"max-age=0", "Accept":"text/plain,application/json; q=0.9,image/webp,image/apng,*/*; q=0.8", "Upgrade-Insecure-Requests":"1", "Connection":"keep-alive", "User-Agent":"Mozilla/5.0(Windows NT 6.1; Win64; x64)" }</pre>

Column	Description
	<pre>AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181Safari/537.36", "Host":"mcdaso02:5555", "Accept-Encoding":"gzip,deflate", "Accept-Language":"en-US,en;q=0.9,ta;q=0.8", "Content-Type":"application/x-www-form-urlencoded" }</pre>
RESPONSE_HEADERS	<p>Response header in the outgoing response.</p> <p>Example:</p> <pre>{ "Server":"Jetty(9.2.9.v20150224)", "Access-Control-Allow-Origin":"*", "Access-Control-Allow-Methods":"GET,POST,DELETE, PUT", "Connection":"close", "Date":"Fri, 30 Mar 2018 08:25:45 GMT", "Access-Control-Allow-Headers":"Content-Type,api_key, Authorization", "Content-Type":"application/xml" }</pre>
ROOTCONTEXTID	<p>The unique identifier for the root context information API Gateway uses to connect related entries from different logs.</p> <p>This column is currently not used. It appears as NULL or as an empty string.</p> <p>Example: 81546147-41a8-4998-8150-02ba67bb08c2</p>
SERVERID	<p>The API Gateway server on which the transaction event occurred.</p> <p>This column is currently not used. It appears as NULL or as an empty string.</p> <p>Example: SampleHost:80</p>
SERVICE_NAME	<p>Name of the service in which the event occurred.</p> <p>Example: Swagger_Petstore</p>
SESSION_ID	<p>A string the API Gateway server generates to uniquely identify each session. This is either the IS session token or the automatically generated GUID if the token is missing from the message context.</p> <p>Example: 6dfcd849198c4a7e96b4ff89bc2deaf5</p>
SOURCE_GATEWAY_NODE	<p>Source API Gateway's IP address.</p> <p>Example: 10.0.75.1</p>
STATUS	<p>Status of the API request.</p>

Column	Description
	Possible values are: SUCCESS, FAILURE
TOTAL_TIME	Time in milliseconds required to invoke the API provider. This time includes the overhead incurred by API Gateway. Overhead includes security overhead for encryption, decryption, and load-balance retries. Example: 1042

CentraSite

The runtime events and metrics payload generated by API Gateway at run-time is published to the configured CentraSite destination. The columns that make up the events and metrics data model for CentraSite are listed below:

Transactional Events

Column	Description
ApiUserVersion	The system-assigned version identifier for the API. Example: 1.0
Consumer	Name of the consumer associated with the API invocation. A consumer name is populated as unknown when API Gateway is unable to identify the consumer using a policy that is configured for the API. Example: SampleApplication
ConsumerId	The unique identifier for the consumer associated with the API invocation. Example: be8b27d6-8f79-4c6e-b06c-a628d2ba30c3
Consumer IP Address	IP address of the consumer associated with the API invocation. Example: 10.60.20.169
CorrelationID	The unique identifier that is automatically generated for every request coming to API Gateway and can be used to query the log. Example: MED38e9cfa4-2348-408b-9462-124b2181c1a6:656
Created Time	Date and time when the event was generated in API Gateway. Example: 2017-08-09 01:27:45 AM

Column	Description
CustomFields	<p>The custom fields an API Provider can provide to log a new field and value for a transaction event.</p> <p>Example: {"customfield":"customvalue"}</p>
ErrorOrigin	<p>The origin of error.</p> <p>Example: Nativeservice</p>
Event Type	<p>The type of event that occurred.</p> <p>Example: Transaction Event</p>
External Calls	<p>List the external calls from API Gateway. These external calls can be to a native service or service registry.</p> <p>Example:</p> <pre>[{ "externalCallType":"SERVICE_REGISTRY_CALL", "externalURL":"http://service.registry.com", "callDuration":49, "callStartTime":1562244570486, "callEndTime":1562244570535, "responseCode": "200" }, { "externalCallType":"NATIVE_SERVICE_CALL", "externalURL": "https://petstore.swagger.io/v2/store/inventory", "callDuration":1285, "callStartTime":1562244569252, "callEndTime":1562244570537, "responseCode":"200" }]</pre>
Native HTTP Method	<p>The HTTP method used to invoke the native service.</p> <p>Example: GET</p>
Native Request Headers	<p>Request header in the incoming request from the API Gateway to native service.</p> <p>Example:</p> <pre>{ "Authorization":"*****", "Accept": "*/*", "Authorization": "*****", "Accept":"*/*", "Cache-Control": "no-cache", "User-Agent": "PostmanRuntime/7.13.0", "Postman-Token": "381424fa-e3b3-4058-8df9-4abf9d72c899", "postmanHeader": "hello",</pre>

Column	Description
	<pre>"accept-encoding": "gzip, deflate", "Content-Type": "application/x-www-form-urlencoded" }</pre>
Native Req Payload	<p>The native service request data.</p> <p>Example:</p> <pre>{ "param1" : "value1", "param2" : 10 }</pre>
Native Response Headers	<p>Response header in the outgoing response from the native service to API Gateway.</p> <p>Example:</p> <pre>{ "Server":"Jetty(9.2.9.v20150224)", "Access-Control-Allow-Origin":"*", "Access-Control-Allow-Methods": "GET, POST, DELETE, PUT", "Connection":"close", "Date": "Fri, 07 Jun 2019 12:44:13 GMT", "Access-Control-Allow-Headers": "Content-Type, api_key, Authorization", "Content-Type": "application/json" }</pre>
Native Res Payload	<p>The native service response data.</p> <p>Example:</p> <pre>{ "id":2, "category": { " id":2, " name":"string" }, " name":"pysen", "photoUrls":["string"], "tags": [{ " id":0, " name":"string" }], "status":"available" }</pre>
Native URL	<p>URL of the native service.</p> <p>Example: http://petstore.swagger.io/v2/pet/2</p>

Column	Description
PartnerId	<p>The unique identifier for the partner that generated the audit record.</p> <p>Example: unknown</p>
Provider Round Trip Time	<p>Time in milliseconds required for API Gateway to invoke a native provider and receive a response. This time includes the overhead incurred by API Gateway. Overhead includes the time it takes for a provider to process a request and return a response, plus any network latency to or from the provider. Subtracting total time from provider time must give a rough indicator of the API Gateway overhead.</p> <p>Example: 1700</p>
QueryParameters	<p>This is applicable only for REST APIs. Query parameters present in the incoming REST request.</p> <p>Example: {"status":"available"}</p>
RequestHeaders	<p>Request header in the incoming request from the client.</p> <p>Example:</p> <pre>{ "Cache-Control":"max-age=0", "Accept":"text/plain,application/json;q=0.9,image/webp,image/apng,*/*;q=0.8", "Upgrade-Insecure-Requests":"1", "Connection":"keep-alive", "User-Agent":"Mozilla/5.0(Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181Safari/537.36", "Host":"mcdaso02:5555", "Accept-Encoding":"gzip,deflate", "Accept-Language":"en-US,en;q=0.9,ta;q=0.8", "Content-Type":"application/x-www-form-urlencoded" }</pre>
RequestPayload	<p>The API request payload data.</p> <p>Example: <RequestPayload></p>
ResponseHeaders	<p>Response header in the outgoing response.</p> <p>Example:</p> <pre>{ "Server":"Jetty(9.2.9.v20150224)", "Access-Control-Allow-Origin":"*", "Access-Control-Allow-Methods":"GET,POST,DELETE,PUT", "Connection":"close", }</pre>

Column	Description
	<pre>"Date": "Fri, 30 Mar 2018 08:25:45 GMT", "Access-Control-Allow-Headers": "Content-Type, api_key, Authorization", "Content-Type": "application/xml" }</pre>
ResponsePayload	<p>The API response payload data.</p> <p>Example: <ResponsePayload></p>
Source Gateway Node	<p>Source API Gateway's IP address.</p> <p>Example: 10.0.75.1</p>
Total Round Trip time	<p>Time in milliseconds required to invoke the API provider. This time includes the overhead incurred by API Gateway. Overhead includes security overhead for encryption, decryption, and load-balance retries.</p> <p>Example: 1707</p>
Gateway	<p>Name of the API Gateway instance reporting the event.</p> <p>Example: API_Gateway_Instance</p>
Request Status	<p>Status of the API request.</p> <p>Possible values are: SUCCESS, FAILURE</p>

Error Events

Column	Description
ApiUserVersion	<p>The system-assigned version identifier for the API.</p> <p>Example: 1.0</p>
Consumer	<p>Name of the consumer associated with the API invocation.</p> <p>A consumer name is populated as unknown when API Gateway is unable to identify the consumer using a policy that is configured for the API.</p> <p>Example: SampleApplication</p>
ConsumerId	<p>The unique identifier for the consumer associated with the API invocation.</p> <p>Example: be8b27d6-8f79-4c6e-b06c-a628d2ba30c3</p>
Consumer IP Address	<p>IP address of the consumer associated with the API invocation.</p>

Column	Description
	Example: 10.60.20.169
CorrelationID	The unique identifier that is automatically generated for every request coming to API Gateway and can be used to query the log. Example: MED38e9cfa4-2348-408b-9462-124b2181c1a6:656
Created Time	Date and time when the event was generated in API Gateway. Example: 2017-08-09 08:24:04 AM
Error Source	The source where the error occurred. Example: e1cc3c7b-495d-11e7-a5a6-88cf17308ba4
Error Description	Message that describes the error that occurred. Example: Resource / not found
Event Type	The type of event that occurred. Example: Error Event
Gateway	Name of the API Gateway instance reporting the event. Example: API_Gateway_Instance

Monitoring Events

Column	Description
PolicyName	Name of the policy that is enforced on the API. Example: Log Invocation
Alert Description	Text of the alert message sent to a configured destination when the performance conditions are violated. The alert message is specified in the policy definition of an API. Example: MSLA_ALERT MESSAGE
Alert Source	Name of the API Gateway policy that generated the alert message. Example: Monitorpolicy, EnforcePolicy-HardLimit
Alert Type	The type of alert generated for the event. Possible values are: Monitor, Sla
apiName	Name of the API in which the event occurred.

Column	Description
	Example: pet1
apiVersion	The system-assigned version identifier for the API. Example: 1.0
applicationId	The unique identifier for the application associated with the API invocation. Example: c0f84954-9732-11e5-b9f4-f159eafe47b2
applicationIp	IP address of the application associated with the API invocation. Example: 10.20.248.33
applicationName	Name of the application associated with the API invocation. An application name is populated as unknown when API Gateway is unable to identify the application using a security policy that is configured for the API. Example: SampleApplication
creationDate	Date and time when the event was generated in API Gateway. Example: 1501671101509
Monitored Attribute	The monitored attribute which has breached the configured SLA. Example: AVGRESPOSTIME GT 1.0, SUCCESSCOUNT EQ 3, REQUESTCOUNT GT 10
native_endpoint	The endpoint URL of the native API that is invoked. Example: http://petstore.swagger.io/v2/pet/55
operationName	Name of the API operation that is invoked. Example: Using a Calculator API, you can perform various operations such as addition, subtraction, multiplication, and division. When an addition operation is invoked in API Gateway, then the operation field name is populated as addInts .
sessionId	A string the API Gateway server generates to uniquely identify each session. This is either the IS session token or the automatically generated GUID if the token is missing from the message context. Example: c0f84954-9732-11e5-b9f4-f159eafe47b2
targetName	Name of the API Gateway instance reporting the event. Example: API_Gateway_Instance

Policy Violation Events

Column	Description
ApiUserVersion	The system-assigned version identifier for the API. Example: 1.0
Alert Source	Name of the API Gateway policy that generated the alert message. Example: Unknown-Policy
Alert Type	The type of alert generated for the event. Example: PolicyViolation
Alert Description	Text of the alert message sent to a configured destination when the performance conditions are violated. The alert message is specified in the policy definition of an API. Example: A violation of policy was detected : Unable to identify the application for the request
Consumer	Name of the consumer associated with the API invocation. A consumer name is populated as unknown when API Gateway is unable to identify the consumer using a policy that is configured for the API. Example: unknown
ConsumerId	The unique identifier for the consumer associated with the API invocation. Example: unknown
Consumer IP Address	IP address of the consumer associated with the API invocation. Example: 10.60.37.118
Created Time	Date and time when the event was generated in API Gateway. Example: 2017-08-09 08:25:52 AM
Event Type	The type of event that occurred. Example: Policy Violation Event
Gateway	Name of the API Gateway instance reporting the event. Example: API_Gateway_Instance

Lifecycle Events

Column	Description
TimeStamp	Date and time when the event was generated in API Gateway. Example: 2017-08-26 04:13:35 PM
Target	Name of the API Gateway instance reporting the event. Example: API_Gateway_Instance
LifeCycleStatus	Status of the API Gateway instance. Possible values are: STARTED or STOPPED
LifeCycleAlertDescription	The alert notification message for the lifecycle event. Example: Alert_Message

Performance Metrics

Column	Description
AVG_RESP_TIME	The average amount of time it took the API to complete each invocation in the current interval. Response time is measured from the moment API Gateway receives the request until the moment it returns the response to the caller. Example: 1376
FAULT_COUNT	The number of failed API invocations in the current interval. Example: 1
INCLUDE_FAULTS	Includes failed API invocations. Possible values are: true, false
INTERVAL_START	The starting date and time from which you want to examine metrics. Example: 02 Aug 2017 10:51:31 GMT
INTERVAL_STOP	The ending date and time until which you want to examine metrics. Example: 02 Aug 2017 10:52:31 GMT
MAX_RESP_TIME	The maximum amount of time (in milliseconds) it took for the API to complete an invocation in the current interval. Example: 1401

Column	Description
MIN_RESP_TIME	The minimum amount of time (in milliseconds) it took for the API to complete an invocation in the current interval. Example: 1352
OPERATION_NAME	Name of the API operation that is invoked. Example: /pet/{petId}
SERVICE_KEY	The Universally Unique Identifier (UUID) for the service in which the event occurred. This column is currently not used by APIs created in API Gateway. It is used to support the APIs that are migrated from CentraSite or Mediator to API Gateway.
SUCCESS_COUNT	The number of successful API invocations in the current interval. Example: 1
TARGET_NAME	Name of the API Gateway instance reporting the event. Example: API_Gateway_Instance
totalCount	The total number of API invocations (successful and unsuccessful) in the current interval. Example: 2

Elasticsearch

The runtime events and metrics payload generated by API Gateway at run-time is published to the configured Elasticsearch destination. The columns that make up the events and metrics data model for Elasticsearch are listed below:

Transactional Events

Column	Description
apild	The unique identifier for the API. Example: af70b2de-c9c5-4f40-94be-7d8622743e42
apiName	Name of the API in which the event occurred. Example: SampleAPI
apiVersion	The system-assigned version identifier for the API.

Column	Description
	Example: 1.0
applicationId	The unique identifier for the application associated with the API invocation. Example: c0f84954-9732-11e5-b9f4-f159eafe47b2
applicationIp	IP address of the application associated with the API invocation. Example: 10.60.37.42
applicationName	Name of the application associated with the API invocation. An application name is populated as unknown when API Gateway is unable to identify the application using a security policy that is configured for the API. Example: SampleApplication
cachedResponse	Indicates whether the response is sent to the client from the cached data present in API Gateway through the Service result caching policy or the response is received from Native service and sent to client. Possible values are: Cached, Not-Cached
correlationID	The unique identifier that is automatically generated for every request coming to API Gateway and can be used to query the log. Example: MED38e9cfa4-2348-408b-9462-124b2181c1a6:656
creationDate	Date and time when the event was generated in API Gateway. Example: 1501671101509
customFields	The custom fields an API Provider can provide to log a new field and value for a transaction event. Example: {"customfield":"customvalue"}
errorOrigin	The origin of error. Example: Nativeservice
eventType	The type of event that occurred. Example: Transactional
externalCalls	List the external calls from API Gateway. These external calls can be to a native service or service registry. Example: <pre>[{</pre>

Column	Description
	<pre>"externalCallType":"SERVICE_REGISTRY_CALL", "externalURL":"http://service.registry.com", "callDuration":49, "callStartTime":1562244570486, "callEndTime":1562244570535, "responseCode": "200" }, { "externalCallType":"NATIVE_SERVICE_CALL", "externalURL": "https://petstore.swagger.io/v2/store/inventory", "callDuration":1285, "callStartTime":1562244569252, "callEndTime":1562244570537, "responseCode":"200" }]</pre>
httpMethod	<p>The HTTP method used to invoke the API.</p> <p>Example: GET</p>
isCallbackRequest	<p>Indicates whether the event is generated for a callback request.</p> <p>Possible values are:</p> <ul style="list-style-type: none"> ■ true. This denotes that the event is generated for a callback request. ■ false. This denotes that the event is generated for a normal response.
messageType	<p>This is applicable only for WebSocket APIs. This indicates the type of a WebSocket message.</p> <p>Possible values are: binary, text</p>
nativeHttpMethod	<p>The HTTP method used to invoke the native service.</p> <p>Example: GET</p>
nativeRequestHeaders	<p>Request header in the incoming request from the API Gateway to native service.</p> <p>Example:</p> <pre>{ "Authorization":"*****", "Accept": "*/*", "Authorization": "*****", "Accept":"*/*", "Cache-Control": "no-cache", "User-Agent": "PostmanRuntime/7.13.0", "Postman-Token": "381424fa-e3b3-4058-8df9-4abf9d72c899", "postmanHeader": "hello",</pre>

Column	Description
	<pre>"accept-encoding": "gzip, deflate", "Content-Type": "application/x-www-form-urlencoded" }</pre>
nativeRequestPayload	<p>The native service request data.</p> <p>Example:</p> <pre>{ "param1" : "value1", "param2" : 10 }</pre>
nativeResponseHeaders	<p>Response header in the outgoing response from the native service to API Gateway.</p> <p>Example:</p> <pre>{ "Server":"Jetty(9.2.9.v20150224)", "Access-Control-Allow-Origin":"*", "Access-Control-Allow-Methods": "GET, POST, DELETE, PUT", "Connection":"close", "Date": "Fri, 07 Jun 2019 12:44:13 GMT", "Access-Control-Allow-Headers": "Content-Type, api_key, Authorization", "Content-Type": "application/json" }</pre>
nativeResponsePayload	<p>The native service response data.</p> <p>Example:</p> <pre>{ "id":2, "category": { " id":2, " name":"string" }, " name":"pysen", "photoUrls":["string"], "tags": [{ " id":0, " name":"string" }], "status":"available" }</pre>
nativeURL	<p>URL of the native service.</p> <p>Example: http://petstore.swagger.io/v2/pet/2</p>
operationName	<p>Name of the API operation that is invoked.</p>

Column	Description
	Example: /pet/{petId}
origin	This is applicable only for WebSocket APIs. The origin of the request. Possible values are: client, server
packageId	The unique identifier for the API package. Example: c0f84954-9732-11e5-b9f4-f159eafe47b2
packageName	Name of the API package. Example: Travel Package
planId	The unique identifier for the API plan. Example: d0f84954-9732-11e5-b9f4-f159eafe47b2
planName	Name of the API plan. Example: Gold Plan
providerTime	Time in milliseconds required for API Gateway to invoke a native provider and receive a response. This time includes the overhead incurred by API Gateway. Overhead includes the time it takes for a provider to process a request and return a response, plus any network latency to or from the provider. Subtracting total time from provider time must give a rough indicator of the API Gateway overhead. Example: 1367
queryParameters	This is applicable only for REST APIs. Query parameters present in the incoming REST request. Example: {"status":"available"}
reqPayload	The API request payload data. Example: <RequestPayload>
requestHeaders	Request header in the incoming request from the client. Example: <pre>{ "Cache-Control":"max-age=0", "Accept":"text/plain,application/json; q=0.9,image/webp,image/apng,*/*; q=0.8", "Upgrade-Insecure-Requests":"1", "Connection":"keep-alive", "User-Agent":"Mozilla/5.0(Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181Safari/537.36",</pre>

Column	Description
	<pre>"Host": "mcdaso02:5555", "Accept-Encoding": "gzip, deflate", "Accept-Language": "en-US, en; q=0.9, ta; q=0.8", "Content-Type": "application/x-www-form-urlencoded" }</pre>
resPayload	<p>The API response payload data.</p> <p>Example: <ResponsePayload></p>
responseCode	<p>The HTTP response status code that indicates success or failure of the requested operation.</p> <p>Example: 404</p>
responseHeaders	<p>Response header in the outgoing response.</p> <p>Example:</p> <pre>{ "Server": "Jetty(9.2.9.v20150224)", "Access-Control-Allow-Origin": "*", "Access-Control-Allow-Methods": "GET, POST, DELETE, PUT", "Connection": "close", "Date": "Fri, 30 Mar 2018 08:25:45 GMT", "Access-Control-Allow-Headers": "Content-Type, api_key, Authorization", "Content-Type": "application/xml" }</pre>
serverID	<p>The API Gateway server on which the transaction event occurred.</p> <p>This column is currently not used. It appears as NULL or as an empty string.</p> <p>Example: SampleHost:80</p>
sourceGatewayDetails	<p>Details of events generated only from Microgateway. The details include Microgateway Id, Source Gateway Host, Source Gateway Port, Source Gateway Version, Microgateway pool.</p>
sourceGatewayNode	<p>Source API Gateway's IP address.</p> <p>Example: 10.0.75.1</p>
sourceGateway	<p>Source of event generation.</p> <p>Possible values: APIGateway or Microgateway.</p>
status	<p>Status of the API request.</p> <p>Possible values are: SUCCESS, FAILURE</p>

Column	Description
totalDataSize	The total combined size of request and response payloads in bytes. Example: 51
totalTime	Time in milliseconds required to invoke the API provider. This time includes the overhead incurred by API Gateway. Overhead includes security overhead for encryption, decryption, and load-balance retries. Example: 1401

Error Events

Column	Description
apild	The unique identifier for the API. Example: af70b2de-c9c5-4f40-94be-7d8622743e42
apiName	Name of the API in which the event occurred. Example: SampleAPI
apiVersion	The system-assigned version identifier for the API. Example: 1.0
applicationId	The unique identifier for the application associated with the API invocation. Example: c0f84954-9732-11e5-b9f4-f159eafe47b2
applicationIp	IP address of the application associated with the API invocation. Example: 10.60.37.42
applicationName	Name of the application associated with the API invocation. An application name is populated as unknown when API Gateway is unable to identify the application using a security policy that is configured for the API. Example: SampleApplication
correlationID	The unique identifier that is automatically generated for every request coming to API Gateway and can be used to query the log. Example: MED38e9cfa4-2348-408b-9462-124b2181c1a6:656
creationDate	Date and time when the event was generated in API Gateway. Example: 1501671101509

Column	Description
errorDesc	Message that describes the error that occurred. Example: Native service provider error. Code : 404
eventSource	The source where the event occurred. Example: API_Gateway_Instance
eventType	The type of event that occurred. Example: Error
httpMethod	The HTTP method used to invoke the API. Example: GET
operationName	Name of the API operation that is invoked. Example: /pet/{petId}
responseCode	The HTTP response status code that indicates success or failure of the requested operation. Example: 404
sourceGatewayDetails	Details of events generated only from Microgateway. The details include Microgateway Id, Source Gateway Host, Source Gateway Port, Source Gateway Version, Microgateway pool.
sourceGatewayNode	Source API Gateway's IP address. Example: 10.0.75.1
sourceGateway	Source of event generation. Possible values: APIGateway or Microgateway.

Monitoring Events

Column	Description
alertDesc	Text of the alert message sent to a configured destination when the performance conditions are violated. The alert message is specified in the policy definition of an API. Example: EnforcePolicy-HardLimit
alertSource	Name of the API Gateway policy that generated the alert message. Example: Monitorpolicy

Column	Description
alertType	The type of alert generated for the event. Example: Monitor
apild	The unique identifier for the API. Example: af70b2de-c9c5-4f40-94be-7d8622743e42
apiName	Name of the API in which the event occurred. Example: SampleAPI
apiVersion	The system-assigned version identifier for the API. Example: 1.0
applicationId	The unique identifier for the application associated with the API invocation. Example: c0f84954-9732-11e5-b9f4-f159eafe47b2
applicationIp	IP address of the application associated with the API invocation. Example: 10.60.37.42
applicationName	Name of the application associated with the API invocation. An application name is populated as unknown when API Gateway is unable to identify the application using a security policy that is configured for the API. Example: SampleApplication
creationDate	Date and time when the event was generated in API Gateway. Example: 1501671101509
eventSource	The source where the event occurred. Example: API_Gateway_Instance
eventType	The type of event that occurred. Example: Monitor
httpMethod	The HTTP method used to request the API access. Example: GET
monitorAttr	The monitored attribute which has breached the configured SLA. Example: AVGRESPONSETIME GT 1.0, SUCCESSCOUNT EQ 3, REQUESTCOUNT GT 10

Column	Description
operationName	Name of the API operation that is invoked. Example: /pet/{petId}
responseCode	The HTTP response status code that indicates success or failure of the requested operation. Example: 200
sourceGateway	Source of event generation. Possible values: APIGateway or Microgateway.
sourceGatewayDetails	Details of events generated only from Microgateway. The details include Microgateway Id, Source Gateway Host, Source Gateway Port, Source Gateway Version, Microgateway pool.

Policy Violation Events

Column	Description
alertDesc	Text of the alert message sent to a configured destination when the performance conditions are violated. The alert message is specified in the policy definition of an API. Example: A violation was detected for policy (Unknown-Policyuser): application could not be identified. Anonymous access is not allowed for this service!
alertSource	Name of the API Gateway policy that generated the alert message. Example: Unknown-Policy
alertType	The type of alert generated for the event. Example: PolicyViolation
apild	The unique identifier for the API. Example: af70b2de-c9c5-4f40-94be-7d8622743e42
apiName	Name of the API in which the event occurred. Example: SampleAPI
apiVersion	The system-assigned version identifier for the API. Example: 1.0
applicationId	The unique identifier for the application associated with the API invocation.

Column	Description
	Example: 9434e90d-65c3-4e37-8ccb-595b8df3e645
applicationIp	IP address of the application associated with the API invocation. Example: 10.60.37.42
applicationName	Name of the application associated with the API invocation. An application name is populated as unknown when API Gateway is unable to identify the application using a security policy that is configured for the API. Example: SampleApplication
creationDate	Date and time when the event was generated in API Gateway. Example: 1501671101509
eventSource	The source where the event occurred. Example: API_Gateway_Instance
eventType	The type of event that occurred. Example: PolicyViolation
httpMethod	The HTTP method used to request the API access. Example: GET
operationName	Name of the API operation that is invoked. Example: /pet/{petId}
responseCode	The HTTP response status code that indicates success or failure of the requested operation. Example: 503
sourceGateway	Source of event generation. Possible values: APIGateway or Microgateway.
sourceGatewayDetails	Details of events generated only from Microgateway. The details include Microgateway Id, Source Gateway Host, Source Gateway Port, Source Gateway Version, Microgateway pool.

Performance Metrics

Column	Description
apild	The unique identifier for the API. Example: af70b2de-c9c5-4f40-94be-7d8622743e42
apiName	Name of the API in which the event occurred. Example: SampleAPI
apiVersion	The system-assigned version identifier for the API. Example: 1.0
availability	The percentage of time that an API was available during the current interval. A value of 100 indicates that the API was always available. If invocations fail due to policy violations, this parameter could still be as high as 100. Example: 100.0
avgResponseTime	The average amount of time it took the API to complete each invocation in the current interval. Response time is measured from the moment API Gateway receives the request until the moment it returns the response to the caller. Example: 1376
creationDate	Date and time when the event was generated in API Gateway. Example: 1501671101509
eventType	The type of event that occurred. Example: PerformanceData
faultCount	The number of failed API invocations in the current interval. Example: 1
includeFaults	Includes failed API invocations. Possible values are: true, false
intervalStart	The starting date and time from which you want to examine metrics. Example: 02 Aug 2017 10:51:31 GMT
intervalStop	The ending date and time until which you want to examine metrics. Example: 02 Aug 2017 10:52:31 GMT

Column	Description
maxResponseTime	The maximum amount of time (in milliseconds) it took for the API to complete an invocation in the current interval. Example: 1401
minResponseTime	The minimum amount of time (in milliseconds) it took for the API to complete an invocation in the current interval. Example: 1352
sourceGatewayDetails	Details of events generated only from Microgateway. The details include Microgateway Id, Source Gateway Host, Source Gateway Port, Source Gateway Version, Microgateway pool.
sourceGateway	Source of event generation. Possible values: APIGateway or Microgateway.
operationName	Name of the API operation that is invoked. Example: /pet/{petId}
successCount	The number of successful API invocations in the current interval. Example: 1
totalCount	The total number of API invocations (successful and unsuccessful) in the current interval. Example: 2

Email

The runtime events and metrics payload generated by API Gateway at run-time is published to the configured Email destination. The columns that make up the events and metrics data model for Email are listed below:

Transactional Events

Column	Description
apiName	Name of the API in which the event occurred. Example: SampleAPI
consumerId	The unique identifier for the consumer associated with the API invocation. Example: c0f84954-9732-11e5-b9f4-f159eafe47b2

Column	Description
consumerName	<p>Name of the consumer associated with the API invocation.</p> <p>A consumer name is populated as unknown when API Gateway is unable to identify the consumer using a security policy that is configured for the API.</p> <p>Example: SampleApplication</p>
correlationID	<p>The unique identifier that is automatically generated for every request coming to API Gateway and can be used to query the log.</p> <p>Example: MED38e9cfa4-2348-408b-9462-124b2181c1a6:656</p>
Description	<p>Message that describes the date and time the API was invoked and the application associated with the API invocation.</p> <p>Example: Invoked at 4/24/18 1:50 PM Consumer Name: Unknown Consumer ID: Unknown</p>
ErrorOrigin	<p>The origin of error.</p> <p>Example: Nativeservice</p>
External Calls	<p>List the external calls from API Gateway. These external calls can be to a native service or service registry.</p> <p>Example:</p> <pre>[{ "externalCallType": "SERVICE_REGISTRY_CALL", "externalURL": "http://service.registry.com", "callDuration": 49, "callStartTime": 1562244570486, "callEndTime": 1562244570535, "responseCode": "200" }, { "externalCallType": "NATIVE_SERVICE_CALL", "externalURL": "https://petstore.swagger.io/v2/store/inventory", "callDuration": 1285, "callStartTime": 1562244569252, "callEndTime": 1562244570537, "responseCode": "200" }]</pre>
Native Endpoint	<p>The endpoint URL of the native API being invoked.</p> <p>Example: http://petstore.swagger.io/v2/pet/55</p>
Native HTTP Method	<p>The HTTP method used to invoke the native service.</p> <p>Example: GET</p>

Column	Description
Native Request Headers	<p>Request header in the incoming request from the API Gateway to native service.</p> <p>Example:</p> <pre>{ "Authorization": "*****", "Accept": "*/*", "Authorization": "*****", "Accept": "*/*", "Cache-Control": "no-cache", "User-Agent": "PostmanRuntime/7.13.0", "Postman-Token": "381424fa-e3b3-4058-8df9-4abf9d72c899", "postmanHeader": "hello", "accept-encoding": "gzip, deflate", "Content-Type": "application/x-www-form-urlencoded" }</pre>
Native Request Payload	<p>The native service request data.</p> <p>Example:</p> <pre>{ "param1" : "value1", "param2" : 10 }</pre>
Native Response Headers	<p>Response header in the outgoing response from the native service to API Gateway.</p> <p>Example:</p> <pre>{ "Server": "Jetty(9.2.9.v20150224)", "Access-Control-Allow-Origin": "*", "Access-Control-Allow-Methods": "GET, POST, DELETE, PUT", "Connection": "close", "Date": "Fri, 07 Jun 2019 12:44:13 GMT", "Access-Control-Allow-Headers": "Content-Type, api_key, Authorization", "Content-Type": "application/json" }</pre>
Native Response Payload	<p>The native service response data.</p> <p>Example:</p> <pre>{ "id": 2, "category": { "id": 2, "name": "string" } }</pre>

Column	Description
	<pre> }, "name":"pysen", "photoUrls":["string"], "tags": [{ "id":0, "name":"string" }], "status":"available" } </pre>
Native URL	<p>URL of the native service.</p> <p>Example: http://petstore.swagger.io/v2/pet/2</p>
Operation/Resource Name	<p>Name of the operation or resource that is being invoked on the API.</p> <p>Example: /pet/{petId}</p>
Policy Action Name	<p>Name of the runtime policy that is enforced on the API.</p> <p>Example: Log Invocation</p>
Source Gateway Node	<p>Source API Gateway's IP address.</p> <p>Example: 10.0.75.1</p>
Status	<p>Status of the API invocation.</p> <p>Possible values are: SUCCESS, FAILURE</p>
Version	<p>The system-assigned version identifier for the API.</p> <p>Example: 1.0</p>

Monitoring Events

Column	Description
alertDesc	<p>Text of the alert message sent to a configured destination when the performance conditions are violated. The alert message is specified in the policy definition of an API.</p> <p>Example: EnforcePolicy-HardLimit</p>
alertSource	<p>The type of alert generated for the event.</p> <p>Example: Monitor</p>
alertType	<p>Text of the alert message sent to a configured destination when the performance conditions are violated. The alert message is specified in the policy definition of an API.</p>

Column	Description
	Example: Test
apiGWHostName	Name of the host which serves the request. Example: SAG-HS09MG2
apiName	Name of the API in which the event occurred. Example: SampleAPI
apiVersion	The system-assigned version identifier for the API. Example: 1.0
applicationId	The unique identifier for the application associated with the API invocation. Example: c0f84954-9732-11e5-b9f4-f159eafe47b2
applicationName	Name of the application associated with the API invocation. An application name is populated as unknown when API Gateway is unable to identify the application using a security policy that is configured for the API. Example: SampleApplication
monitorAttr	The monitored attribute which has breached the configured SLA. Example: AVGRESPONSETIME GT 1.0, SUCCESSCOUNT EQ 3, REQUESTCOUNT GT 10
Native Endpoint	The endpoint URL of the native API that is being invoked. Example: http://petstore.swagger.io/v2/pet/55

JDBC

The events and metrics payload generated by API Gateway at run-time is published to the configured JDBC destination. The columns that make up the events and metrics data model for JDBC are listed below:

API Gateway supports three types of database, Oracle, DB2 and MSSQL. Based on the database selected, API Gateway publishes the events and metrics payload to the JDBC destination.

Transactional Events

Column	Description	Oracle	DB2	MSSQL
API_ID	The unique identifier for the API. Example: c0f84954-9732-11e5-b9f4-f159eafe47b1	Varchar(256)	Varchar(256)	NVarchar(256)
API_NAME	Name of the API in which the event occurred. Example: SampleAPI	Varchar(256)	Varchar(256)	NVarchar(256)
API_VERSION	The system-assigned version identifier for the API. Example: 1.0	Varchar(256)	Varchar(256)	NVarchar(256)
APPLICATION_ID	The unique identifier for the application associated with the API invocation. Example: c0f84954-9732-11e5-b9f4-f159eafe47b2	Varchar(256)	Varchar(256)	NVarchar(256)
APPLICATION_IP	IP address of the application associated with the API invocation. Example: 10.20.248.33	Varchar(64)	Varchar(64)	NVarchar(64)
APPLICATION_NAME	Name of the application associated with the API invocation. An application name is populated as unknown when API Gateway is unable to identify the application using a security policy that is configured for the API. Example: SampleApplication	Varchar(128)	Varchar(128)	NVarchar(128)
AUDITTIMESTAMP	Date and time when the event was written to the log.	TIMESTAMP	TIMESTAMP	DATETIME
BINDING_NAME		Varchar(256)	Varchar(256)	NVarchar(256)

Column Description	Oracle	DB2	MSSQL
Name of the binding that identifies a specific access URI. This column is currently not used. It appears as NULL or as an empty string.			
CACHED_RESPONSE	Varchar(12)	Varchar(12)	NVarchar(12)
Indicates whether the response is sent to the client from the cached data present in API Gateway through the Service result caching policy or the response is received from Native service and sent to client. Possible values are: Cached, Not-Cached			
CONSUMER_ID	Varchar(256)	Varchar(256)	NVarchar(256)
The unique identifier for the consumer associated with the API invocation. Example: c0f84954-9732-11e5-b9f4-f159eafe47b2			
CONSUMER_IP	Varchar(64)	Varchar(64)	NVarchar(64)
IP address of the consumer associated with the API invocation. Example: 10.20.248.33			
CONSUMER_NAME	Varchar(128)	Varchar(128)	NVarchar(128)
Name of the consumer associated with the API invocation. A consumer name is populated as unknown when API Gateway is unable to identify the consumer using a policy that is configured for the API. Example: SampleApplication			
CONTEXTID	CHAR(36)	CHAR(36)	NCHAR(36)
The unique identifier for the current context information API Gateway uses to connect related entries from different logs. This column is currently not used. It appears as NULL or as an empty string. Example: 81546147-41a8-4998-8150-02ba67bb08c2			
CORRELATIONID	Varchar(256)	Varchar(256)	NVarchar(256)

Column Description	Oracle	DB2	MSSQL
<p>The unique identifier that is automatically generated for every request coming to API Gateway and can be used to query the log.</p> <p>Example: MED38e9cfa4-2348-408b-9462-124b2181c1a6:656</p>			
<p>CUSTOM_FIELD</p> <p>The custom fields an API Provider can provide to log a new field and value for a transaction event.</p> <p>Example: {"customfield": "customvalue"}</p>	BLOB	BLOB	IMAGE
<p>ERROR_ORIGIN</p> <p>The origin of error.</p>	Varchar(256)	Varchar(256)	NVarchar(256)
<p>EVENT_CREATE_TS</p> <p>Date and time when the event was generated in API Gateway.</p> <p>This is not the time the database performed its insert (for example, this is calculated by the Mediator policy engine and not a database function).</p>	TIMESTAMP	TIMESTAMP	DATETIME
<p>EVENT_SOURCE</p>			
<p>EVENT_TYPE</p> <p>The type of event that occurred.</p> <p>Example: Transactional</p>	Varchar(256)	Varchar(256)	NVarchar(256)
<p>EVENT_USERNAME</p> <p>Name of the user on API Gateway that invoked the API.</p>	Varchar(256)	Varchar(256)	NVarchar(256)
<p>EXTERNAL_CALLS</p> <p>List the external calls from API Gateway. These external calls can be to a native service or service registry.</p> <p>Example:</p> <pre>[{ "externalCallType": "SERVICE_REGISTRY_CALL", "externalURL": "http://service.registry.com",</pre>	BLOB	BLOB	IMAGE

Column Description	Oracle	DB2	MSSQL
<pre>"callDuration":49, "callStartTime":1562244570486, "callEndTime":1562244570535, "responseCode": "200" }, { "externalCallType": "NATIVE_SERVICE_CALL", "externalURL": "https://petstore.swagger.io/v2/store/inventory", "callDuration":1285, "callStartTime":1562244569252, "callEndTime":1562244570537, "responseCode":"200" }]</pre>			
HTTP_METHOD	Varchar(8)	Varchar(8)	NVarchar(8)
The HTTP method used to invoke the API.			
Example: GET			
INSERTTIMESTAMP	TIMESTAMP	TIMESTAMP	DATETIME
Date and time when the event was generated in API Gateway.			
MSGID	CHAR(36)	CHAR(36)	NCHAR(36)
The ID assigned to the message by the API provider.			
This column is currently not used.			
NATIVE_ENDPOINT	Varchar(4000)	Varchar(4000)	NVarchar(4000)
The endpoint URL of the native API that is invoked.			
Example: http://petstore.swagger.io/v2/pet/55			
NATIVE_HTTP_METHOD	Varchar2(20)	Varchar(20)	Varchar(20)
The HTTP method used to invoke the native service.			
Example: GET			
NATIVE_REQUEST_HEADERS	BLOB	BLOB	IMAGE
Request header in the incoming request from the API Gateway to native service.			
Example:			
<pre>{ "Authorization":"*****",</pre>			

Column Description	Oracle	DB2	MSSQL
<pre>"Accept": "*/*", "Authorization": "*****", "Accept": "*/*", "Cache-Control": "no-cache", "User-Agent": "PostmanRuntime/7.13.0", "Postman-Token": "381424fa-e3b3-4058-8df9-4abf9d7c899", "postmanHeader": "hello", "accept-encoding": "gzip, deflate", "Content-Type": "application/x-www-form-urlencoded" }</pre>			
NATIVE_REQUEST_PAYLOAD	CLOB	CLOB	Varchar(max)
The native service request data.			
Example:			
<pre>{ "param1" : "value1", "param2" : 10 }</pre>			
NATIVE_RESPONSE_HEADERS	BLOB	BLOB	IMAGE
Response header in the outgoing response from the native service to API Gateway.			
Example:			
<pre>{ "Server":"Jetty(9.2.9.v20150224)", "Access-Control-Allow-Origin":"*", "Access-Control-Allow-Methods": "GET, POST, DELETE, PUT", "Connection":"close", "Date": "Fri, 07 Jun 2019 12:44:13 GMT", "Access-Control-Allow-Headers": "Content-Type, api_key,Authorization", "Content-Type": "application/json" }</pre>			
NATIVE_RESPONSE_PAYLOAD	CLOB	CLOB	Varchar(max)
The native service response data.			
Example:			
<pre>{ "id":2, "category": { "id":2, "name":"string" }</pre>			

Column Description	Oracle	DB2	MSSQL
<pre> }, "name":"pysen", "photoUrls":["string"], "tags": [{ "id":0, "name":"string" }], "status":"available" } </pre>			
NATIVE_URL URL of the native service. Example: http://petstore.swagger.io/v2/pet/2	CLOB	CLOB	Varchar(max)
OPERATION_NAME Name of the API operation or resource that is invoked. Example: /pet/{petId}	Varchar(256)	Varchar(256)	NVarchar(256)
ORG_KEY The Globally Unique Identifier (GUID) for an organization. This column is currently not used by APIs created in API Gateway. It is used to support the APIs that are migrated from CentraSite or Mediatorr to API Gateway.	Varchar(128)	Varchar(128)	NVarchar(128)
PACKAGE_ID The unique identifier for the API package. Example: c0f84954-9732-11e5-b9f4-f159eafe47b2	Varchar(256)	Varchar(256)	NVarchar(256)
PACKAGE_NAME Name of the API package. Example: Travel Package	Varchar(256)	Varchar(256)	NVarchar(256)
PARENTCONTEXTID The unique identifier for the parent context information API Gateway uses to connect related entries from different logs.	CHAR(36)	CHAR(36)	NCHAR(36)

Column Description	Oracle	DB2	MSSQL
This column is currently not used. It appears as NULL or as an empty string.			
PARTNER_ID	Varchar(256)	Varchar(256)	NVarchar(256)
The unique identifier for the partner that generated the audit record.			
This column is currently not used. It appears as NULL or as an empty string.			
PLAN_ID	Varchar(256)	Varchar(256)	NVarchar(256)
The unique identifier for the API plan.			
Example: d0f84954-9732-11e5-b9f4-f159eafe47b2			
PLAN_NAME	Varchar(256)	Varchar(256)	NVarchar(256)
Name of the API plan.			
Example: Gold Plan			
PROVIDER_TIME	NUMERIC	INT	INTEGER
Time in milliseconds required for API Gateway to invoke a native provider and receive a response. This time includes the overhead incurred by API Gateway. Overhead includes the time it takes for a provider to process a request and return a response, plus any network latency to or from the provider. Subtracting total time from provider time must give a rough indicator of the API Gateway overhead.			
Example: 1367			
QUERY_PARAMS	BLOB	BLOB	IMAGE
This is applicable only for REST APIs. Query parameters present in the incoming REST request.			
Example: {"status":"available"}			
REQUEST	BLOB	BLOB	IMAGE
The API request payload data.			
Example: <RequestPayload>			
REQUEST_HEADERS	BLOB	BLOB	IMAGE

Column Description	Oracle	DB2	MSSQL
Request header in the incoming request from the client.			
RESPONSE	BLOB	BLOB	IMAGE
The API response payload data. Example: <ResponsePayload>			
RESPONSE_CODE	Numeric	INT	INTEGER
The HTTP response status code that indicates success or failure of the requested operation. Example: 200			
RESPONSE_HEADERS	BLOB	BLOB	IMAGE
Response header in the outgoing response.			
ROOTCONTEXTID	CHAR(36)	CHAR(36)	NCHAR(36)
The unique identifier for the root context information API Gateway uses to connect related entries from different logs. This column is currently not used. It appears as NULL or as an empty string. Example: 81546147-41a8-4998-8150-02ba67bb08c2			
SERVERID	Varchar(450)	Varchar(450)	NVarchar(450)
The API Gateway server on which the transaction event occurred. This column is currently not used. It appears as NULL or as an empty string.			
SERVICE_KEY	Varchar(128)	Varchar(128)	NVarchar(128)
The Universally Unique Identifier (UUID) for the service in which the event occurred. This column is currently not used by APIs created in API Gateway. It is used to support the APIs that are migrated from CentraSite or Mediator to API Gateway.			
SERVICE_NAME	Varchar(256)	Varchar(256)	NVarchar(256)
Name of the service in which the event occurred.			

Column Description	Oracle	DB2	MSSQL
Example: SampleAPI			
SERVICE_VERSION	Varchar(256)	Varchar(256)	NVarchar(256)
The system-assigned version identifier for the service.			
Example: 1.0			
SESSION_ID	Varchar(128)	Varchar(128)	NVarchar(128)
A string the API Gateway server generates to uniquely identify each session. This is either the IS session token or the automatically generated GUID if the token is missing from the message context.			
Example: c0f84954-9732-11e5-b9f4-f159eafe47b2			
SOURCE_GATEWAY_NODE	Varchar2(256)	Varchar(256)	Varchar(256)
Source API Gateway's IP address.			
Example: 10.0.75.1			
STATUS	Varchar(128)	Varchar(128)	NVarchar(128)
Status of the API request.			
Possible values are: SUCCESS, FAILURE			
TARGET_NAME	Varchar(32)	Varchar(32)	NVarchar(32)
Name of the API Gateway instance reporting the event.			
Example: API_Gateway_Instance			
TOTAL_DATA_SIZE	NUMERIC	INT	INTEGER
The total combined size of request and response payloads in bytes.			
Example: 100			
TOTAL_TIME	NUMERIC	INT	INTEGER
Time in milliseconds required to invoke the API provider. This time includes the overhead incurred by API Gateway. Overhead includes security overhead for encryption, decryption, and load-balance retries.			
Example: 120			

Column Description	Oracle	DB2	MSSQL
USER_AGENT	Varchar2(256)	Varchar(256)	NVarchar(256)
Name of the client used to invoke the API.			
Example: Postman			

Error Events

Column Description	Oracle	DB2	MSSQL
API_ID	Varchar(256)	Varchar(256)	NVarchar(256)
The unique identifier for the API.			
Example: c0f84954-9732-11e5-b9f4-f159eafe47b1			

API_NAME	Varchar(256)	Varchar(256)	NVarchar(256)
Name of the API in which the event occurred.			
Example: SampleAPI			

API_VERSION	Varchar(256)	Varchar(256)	NVarchar(256)
The system-assigned version identifier for the API.			
Example: 1.0			

APPLICATION_ID	Varchar(256)	Varchar(256)	NVarchar(256)
The unique identifier for the application associated with the API invocation.			
Example: c0f84954-9732-11e5-b9f4-f159eafe47b2			

APPLICATION_IP	Varchar(64)	Varchar(64)	NVarchar(64)
IP address of the application associated with the API invocation.			
Example: 10.20.248.33			

APPLICATION_NAME	Varchar(128)	Varchar(128)	NVarchar(128)
Name of the application associated with the API invocation.			
An application name is populated as unknown when API Gateway is unable to identify the application using a security policy that is configured for the API.			

Column Description	Oracle	DB2	MSSQL
Example: SampleApplication			
BINDING_NAME	Varchar(256)	Varchar(256)	Varchar(256)
Name of the binding that identifies a specific access URI.			
This column is currently not used. It appears as NULL or as an empty string.			
CONSUMER_ID	Varchar(256)	Varchar(256)	NVarchar(256)
The unique identifier for the consumer associated with the API invocation.			
Example: c0f84954-9732-11e5-b9f4-f159eafe47b2			
CONSUMER_IP	Varchar(64)	Varchar(64)	NVarchar(64)
IP address of the consumer associated with the API invocation.			
Example: 10.20.248.33			
CONSUMER_NAME	Varchar(128)	Varchar(128)	NVarchar(128)
Name of the consumer associated with the API invocation.			
A consumer name is populated as unknown when API Gateway is unable to identify the consumer using a policy that is configured for the API.			
Example: SampleApplication			
CORRELATION_ID	Varchar2(256)	Varchar(256)	Varchar(256)
The unique identifier that is automatically generated for every request coming to API Gateway and can be used to query the log.			
Example: MED38e9cfa4-2348-408b-9462-124b2181c1a6:656			
ERROR_SOURCE	Varchar(256)	Varchar(256)	NVarchar(256)
The source where the error occurred.			
Example: e1cc3c7b-495d-11e7-a5a6-88cf17308ba4			
ERROR_DESC	Varchar(4000)	Varchar(4000)	NVarchar(4000)

Column Description	Oracle	DB2	MSSQL
Message that describes the error that occurred. Example: Resource / not found			
EVENT_CREATE_TS	TIMESTAMP	TIMESTAMP	DATETIME
Date and time when the event was generated in API Gateway. This is not the time the database performed its insert (for example, this is calculated by the Mediator policy engine and not a database function).			
EVENT_SOURCE	Varchar(80)	Varchar(80)	NVarchar(80)
The source where the event occurred.			
EVENT_TYPE	Varchar(256)	Varchar(256)	NVarchar(256)
The type of event that occurred. Example: Error Event			
EVENT_USERNAME	Varchar(80)	Varchar(80)	NVarchar(80)
Name of the user on API Gateway that invoked the API.			
HTTP_METHOD	Varchar(8)	Varchar(8)	NVarchar(8)
The HTTP method used to invoke the API. Example: GET			
NATIVE_ENDPOINT	Varchar(4000)	Varchar(4000)	NVarchar(4000)
The endpoint URL of the native API that is invoked. Example: http://petstore.swagger.io/v2/pet/55			
OPERATION_NAME	Varchar(256)	Varchar(256)	NVarchar(256)
Name of the API operation or resource that is invoked. Example: /pet/{petId}			
ORG_KEY	Varchar(128)	Varchar(128)	NVarchar(128)
The Globally Unique Identifier (GUID) for an organization. This column is currently not used by APIs created in API Gateway. It is used to support the APIs that are			

Column Description	Oracle	DB2	MSSQL
migrated from CentraSite or Mediator to API Gateway.			
RESPONSE_CODE	Numeric	INT	INTEGER
The HTTP response status code that indicates success or failure of the requested operation. Example: 200			
SERVICE_KEY	Varchar(128)	Varchar(128)	NVarchar(128)
The Universally Unique Identifier (UUID) for the service in which the event occurred. This column is currently not used by APIs created in API Gateway. It is used to support the APIs that are migrated from CentraSite or Mediator to API Gateway.			
SERVICE_NAME	Varchar(256)	Varchar(256)	NVarchar(256)
Name of the service in which the event occurred. Example: SampleAPI			
SERVICE_VERSION	Varchar(256)	Varchar(256)	NVarchar(256)
The system-assigned version identifier for the service. Example: 1.0			
SESSION_ID	Varchar(128)	Varchar(128)	NVarchar(128)
A string the API Gateway server generates to uniquely identify each session. This is either the IS session token or the automatically generated GUID if the token is missing from the message context. Example: c0f84954-9732-11e5-b9f4-f159eafe47b2			
TARGET_NAME	Varchar(64)	Varchar(64)	NVarchar(64)
Name of the API Gateway instance reporting the event. Example: API_Gateway_Instance			
USER_AGENT	Varchar2(256)	Varchar(256)	NVarchar(256)
Name of the client used to invoke the API.			

Column Description	Oracle	DB2	MSSQL
Example: Postman			

Monitoring Events

Column Description	Oracle	DB2	MSSQL
USER_AGENT	Varchar2(256)	Varchar(256)	NVarchar(256)

Name of the client used to invoke the API.

Example: **Postman**

ALERT_DESC	Varchar(256)	Varchar(256)	NVarchar(256)
-------------------	--------------	--------------	---------------

Text of the alert message sent to a configured destination when the performance conditions are violated. The alert message is specified in the policy definition of an API.

Example: **EnforcePolicy-HardLimit**

ALERT_SOURCE	Varchar(256)	Varchar(256)	NVarchar(256)
---------------------	--------------	--------------	---------------

Name of the API Gateway policy that generated the alert message.

Example: **MonitorPolicy**

ALERT_TYPE	Varchar(128)	Varchar(128)	NVarchar(128)
-------------------	--------------	--------------	---------------

The type of alert generated for the event.

Possible values are: **Monitor, SLA**

API_ID	Varchar(256)	Varchar(256)	NVarchar(256)
---------------	--------------	--------------	---------------

The unique identifier for the API.

Example: **c0f84954-9732-11e5-b9f4-f159eafe47b1**

API_NAME	Varchar(256)	Varchar(256)	NVarchar(256)
-----------------	--------------	--------------	---------------

Name of the API in which the event occurred.

Example: **SampleAPI**

API_VERSION	Varchar(256)	Varchar(256)	NVarchar(256)
--------------------	--------------	--------------	---------------

The system-assigned version identifier for the API.

Example: **1.0**

Column Description	Oracle	DB2	MSSQL
APPLICATION_ID	Varchar(256)	Varchar(256)	NVarchar(256)
The unique identifier for the application associated with the API invocation.			
Example: c0f84954-9732-11e5-b9f4-f159eafe47b2			
APPLICATION_IP	Varchar(64)	Varchar(64)	NVarchar(64)
IP address of the application associated with the API invocation.			
Example: 10.20.248.33			
APPLICATION_NAME	Varchar(128)	Varchar(128)	NVarchar(128)
Name of the application associated with the API invocation.			
An application name is populated as unknown when API Gateway is unable to identify the application using a security policy that is configured for the API.			
Example: SampleApplication			
BINDING_NAME	Varchar(256)	Varchar(256)	NVarchar(256)
Name of the binding that identifies a specific access URI.			
This column is currently not used. It appears as NULL or as an empty string.			
creationDate	Varchar2(256)	Varchar(256)	NVarchar(256)
Date and time when the event was generated in API Gateway.			
Example: 1501671101509			
EVENT_TYPE	Varchar(256)	Varchar(256)	NVarchar(256)
The type of event that occurred.			
Example: Monitor Event			
EVENT_USERNAME	Varchar(80)	Varchar(80)	NVarchar(80)
Name of the user on API Gateway that invoked the API.			
MONITOR_ATTR	Varchar(256)	Varchar(256)	NVarchar(256)

Column Description	Oracle	DB2	MSSQL
The monitored attribute which has breached the configured SLA. Example: AVGRESPONSETIME GT 1.0, SUCCESSCOUNT EQ 3, REQUESTCOUNT GT 10			
NATIVE_ENDPOINT	Varchar(4000)	Varchar(4000)	NVarchar(4000)
The endpoint URL of the native API that is invoked. Example: http://petstore.swagger.io/v2/pet/55			
OPERATION_NAME	Varchar(256)	Varchar(256)	NVarchar(256)
Name of the API operation or resource that is invoked. Example: /pet/{petId}			
RESPONSE_CODE	Numeric	INT	INTEGER
The HTTP response status code that indicates success or failure of the requested operation. Example: 200			
SESSION_ID	Varchar(128)	Varchar(128)	NVarchar(128)
A string the API Gateway server generates to uniquely identify each session. This is either the IS session token or the automatically generated GUID if the token is missing from the message context. Example: c0f84954-9732-11e5-b9f4-f159eafe47b2			
TARGET_NAME	Varchar(64)	Varchar(64)	NVarchar(64)
Name of the API Gateway instance reporting the event. Example: API_Gateway_Instance			

Policy Violation Events

Column Description	Oracle	DB2	MSSQL
ALERT_DESC	Varchar(256)	Varchar(256)	NVarchar(256)
Text of the alert message sent to a configured destination when the performance conditions are			

Column Description	Oracle	DB2	MSSQL
violated. The alert message is specified in the policy definition of an API. Example: EnforcePolicy-HardLimit			
ALERT_SOURCE Name of the API Gateway policy that generated the alert message. Example: PolicyViolationPolicy	Varchar(256)	Varchar(256)	NVarchar(256)
ALERT_TYPE The type of alert generated for the event. Example: PolicyViolation	Varchar(128)	Varchar(128)	NVarchar(128)
API_ID The unique identifier for the API. Example: c0f84954-9732-11e5-b9f4-f159eafe47b1	Varchar(256)	Varchar(256)	NVarchar(256)
API_NAME Name of the API in which the event occurred. Example: SampleAPI	Varchar(256)	Varchar(256)	NVarchar(256)
API_VERSION The system-assigned version identifier for the API. Example: 1.0	Varchar(256)	Varchar(256)	NVarchar(256)
APPLICATION_ID The unique identifier for the application associated with the API invocation. Example: c0f84954-9732-11e5-b9f4-f159eafe47b2	Varchar(256)	Varchar(256)	NVarchar(256)
APPLICATION_IP IP address of the application associated with the API invocation. Example: 10.20.248.33	Varchar(64)	Varchar(64)	NVarchar(64)
APPLICATION_NAME	Varchar(128)	Varchar(128)	NVarchar(128)

Column Description	Oracle	DB2	MSSQL
<p>Name of the application associated with the API invocation.</p> <p>An application name is populated as unknown when API Gateway is unable to identify the application using a security policy that is configured for the API.</p> <p>Example: SampleApplication</p>			
<p>BINDING_NAME</p> <p>Name of the binding that identifies a specific access URI.</p> <p>This column is currently not used. It appears as NULL or as an empty string.</p>	Varchar(256)	Varchar(256)	NVarchar(256)
<p>CONSUMER_ID</p> <p>The unique identifier for the consumer associated with the API invocation.</p> <p>Example: c0f84954-9732-11e5-b9f4-f159eafe47b2</p>	Varchar(256)	Varchar(256)	NVarchar(256)
<p>CONSUMER_IP</p> <p>IP address of the consumer associated with the API invocation.</p> <p>Example: 10.20.248.33</p>	Varchar(32)	Varchar(32)	NVarchar(32)
<p>CONSUMER_NAME</p> <p>Name of the consumer associated with the API invocation.</p> <p>A consumer name is populated as unknown when API Gateway is unable to identify the consumer using a policy that is configured for the API.</p> <p>Example: SampleApplication</p>	Varchar(128)	Varchar(128)	NVarchar(128)
<p>EVENT_CREATE_TS</p> <p>Date and time when the event was generated in API Gateway.</p> <p>This is not the time the database performed its insert (for example, this is calculated by the Mediator policy engine and not a database function).</p>	TIMESTAMP	TIMESTAMP	DATETIME
<p>EVENT_SOURCE</p>	Varchar(80)	Varchar(80)	NVarchar(80)

Column Description	Oracle	DB2	MSSQL
The source where the event occurred.			
EVENT_TYPE The type of event that occurred. Example: Policy Violation	Varchar(256)	Varchar(256)	NVarchar(256)
EVENT_USERNAME Name of the user on API Gateway that invoked the API.	Varchar(80)	Varchar(80)	NVarchar(80)
HTTP_METHOD The HTTP method used to invoke the API. Example: GET	Varchar(8)	Varchar(8)	NVarchar(8)
NATIVE_ENDPOINT The endpoint URL of the native API that is invoked. Example: http://petstore.swagger.io/v2/pet/55	Varchar(4000)	Varchar(4000)	NVarchar(4000)
OPERATION_NAME Name of the API operation that is invoked. Example: /pet/{petId}	Varchar(256)	Varchar(256)	NVarchar(256)
ORG_KEY The Globally Unique Identifier (GUID) for an organization. This column is currently not used by APIs created in API Gateway. It is used to support the APIs that are migrated from CentraSite or Mediator to API Gateway.	Varchar(128)	Varchar(128)	NVarchar(128)
RESPONSE_CODE The HTTP response status code that indicates success or failure of the requested operation. Example: 200	Numeric	INT	INTEGER
SERVICE_KEY The Universally Unique Identifier (UUID) for the service in which the event occurred.	Varchar(128)	Varchar(128)	NVarchar(128)

Column Description	Oracle	DB2	MSSQL
This column is currently not used by APIs created in API Gateway. It is used to support the APIs that are migrated from CentraSite or Mediator to API Gateway.			
SERVICE_NAME Name of the service in which the event occurred. Example: SampleAPI	Varchar(256)	Varchar(256)	NVarchar(256)
SERVICE_VERSION The system-assigned version identifier for the service. Example: 1.0	Varchar(256)	Varchar(256)	NVarchar(256)
SESSION_ID A string the API Gateway server generates to uniquely identify each session. This is either the IS session token or the automatically generated GUID if the token is missing from the message context. Example: c0f84954-9732-11e5-b9f4-f159eafe47b2	Varchar(128)	Varchar(128)	NVarchar(128)
TARGET_NAME Name of the API Gateway instance reporting the event. Example: API_Gateway_Instance	Varchar(64)	Varchar(64)	NVarchar(64)
USER_AGENT Name of the client used to invoke the API. Example: Postman	Varchar2(256)	Varchar(256)	NVarchar(256)

Performance Metrics

Column Description	Oracle	DB2	MSSQL
API_ID The unique identifier for the API. Example: c0f84954-9732-11e5-b9f4-f159eafe47b1	Varchar(256)	Varchar(256)	NVarchar(256)
API_NAME	Varchar(256)	Varchar(256)	NVarchar(256)

Column Description	Oracle	DB2	MSSQL
Name of the API in which the event occurred. Example: SampleAPI			
API_VERSION The system-assigned version identifier for the API. Example: 1.0	Varchar(256)	Varchar(256)	NVarchar(256)
AVAIL The percentage of time that an API was available during the current interval. A value of 100 indicates that the API was always available. If invocations fail due to policy violations, this parameter could still be as high as 100. Example: 100	NUMBER	NUMBER (6, INTEGER 2)	
AVG_RESP The average amount of time it took the API to complete each invocation in the current interval. Response time is measured from the moment API Gateway receives the request until the moment it returns the response to the caller. Example: 1376	NUMBER	INT	INTEGER
BINDING_NAME Name of the binding that identifies a specific access URI. This column is currently not used. It appears as NULL or as an empty string.	Varchar(256)	Varchar(256)	NVarchar(256)
EVENT_CREATE_TS Date and time when the event was generated in API Gateway. This is not the time the database performed its insert (for example, this is calculated by the Mediator policy engine and not a database function).	TIMESTAMP	TIMESTAMP	DATETIME
EVENT_SOURCE The source where the event occurred.	Varchar(80)	Varchar(80)	NVarchar(80)
EVENT_TYPE	Varchar(256)	Varchar(256)	NVarchar(256)

Column Description	Oracle	DB2	MSSQL
The type of event that occurred. Example: Performance Data			
FAULT_COUNT	NUMBER	INT	INT
Total number of error invocations for a specified API withing the configured metrics interval. Example: 1			
INCLUDE_FAULTS	CHAR(1)	CHAR(1)	NCHAR(1)
Includes failed API invocations. Possible values are: true, false			
INTERVAL_START	TIMESTAMP	TIMESTAMP	DATETIME
The starting date and time from which you want to examine metrics. Example: 1526294632172	(6)		
INTERVAL_STOP	TIMESTAMP	TIMESTAMP	DATETIME
The ending date and time until which you want to examine metrics. Example: 1526294632182	(6)		
LIVELY	Char(1)	Char(1)	NChar(1)
Boolean. Availability of an the API at the end of the current interval.			
MAX_RESP	NUMBER	INT	INTEGER
The maximum amount of time (in milliseconds) it took for the API to complete an invocation in the current interval. Example: 1401			
MIN_RESP	NUMBER	INT	INTEGER
The minimum amount of time (in milliseconds) it took for the API to complete an invocation in the current interval. Example: 1352			
NATIVE_ENDPOINT	Varchar(4000)	Varchar(4000)	NVarchar(4000)

Column Description	Oracle	DB2	MSSQL
The endpoint URL of the native API that is invoked. Example: http://petstore.swagger.io/v2/pet/55			
OPERATION_NAME Name of the API operation or resource that is invoked. Example: /pet/{petId}	Varchar(256)	Varchar(256)	NVarchar(256)
ORG_KEY The Globally Unique Identifier (GUID) for the organization. This column is currently not used by APIs created in API Gateway. It is used to support the APIs that are migrated from CentraSite or Mediator to API Gateway.	Varchar(128)	Varchar(128)	NVarchar(128)
SERVICE_KEY The Universally Unique Identifier (UUID) for the service in which the event occurred. This column is currently not used by APIs created in API Gateway. It is used to support the APIs that are migrated from CentraSite or Mediator to API Gateway.	Varchar(128)	Varchar(128)	NVarchar(128)
SERVICE_NAME Name of the service in which the event occurred. Example: SampleAPI	Varchar(256)	Varchar(256)	NVarchar(256)
SERVICE_VERSION The system-assigned version identifier for the service. Example: 1.0	Varchar(256)	Varchar(256)	NVarchar(256)
SUCCESS_COUNT The number of successful API invocations in the current interval. Example: 1	NUMBER	INT	INTEGER
TARGET_NAME Name of the API Gateway instance reporting the event. Example: API_Gateway_Instance	Varchar(64)	Varchar(64)	NVarchar(64)

Column Description	Oracle	DB2	MSSQL
TOTAL_COUNT	NUMBER	INT	INTEGER

The total number of API invocations (successful and unsuccessful) in the current interval.

Example: **2**

USER_AGENT	Varchar2(256)	Varchar(256)	NVarchar(256)
-------------------	---------------	--------------	---------------

Name of the client used to invoke the API.

Example: **Postman**

Local Log

The runtime events and metrics payload generated by API Gateway at run-time is published to the configured Local Log destination. The columns that make up the events and metrics data model for Local Log are listed below:

Transactional Events

Column	Description
ApiName	Name of the API in which the event occurred. Example: SampleAPI
ApiVersion	The system-assigned version identifier for the API. Example: 1.0.0
ApplicationID	The unique identifier for the application associated with the API invocation. Example: 7908eb44-d107-4670-929d-89111fc9347c
ApplicationIP	IP address of the application associated with the API invocation. Example: 10.60.37.42
ApplicationName	Name of the application associated with the API invocation. An application name is populated as unknown when API Gateway is unable to identify the application using a security policy that is configured for the API. Example: SampleApplication

Column	Description
CorrelationID	The unique identifier that is automatically generated for every request coming to API Gateway and can be used to query the log. Example: MED38e9cfa4-2348-408b-9462-124b2181c1a6:656
CustomFields	The custom fields an API Provider can provide to log a new field and value for a transaction event. Example: {"customfield":"customvalue"}
ErrorOrigin	The origin of error. Example: Nativeservice
EventSource	The source where the event occurred. Example: API_Gateway_Instance
NativeHttpMethod	The HTTP method used to invoke the native service. Example: GET
nativeRequestPayload	The native service request data. Example: <pre>{ "param1" : "value1", "param2" : 10 }</pre>
nativeResponsePayload	The native service response data. Example: <pre>{ "id":2, "category": { "id":2, "name":"string" }, "name":"pysen", "photoUrls":["string"], "tags": [{ "id":0, "name":"string" }], "status":"available" }</pre>
Native URL	URL of the native service. Example: http://petstore.swagger.io/v2/pet/2

Column	Description
Operation/Resource name	Name of the API operation or resource that is invoked. Example: /pet
Partner ID	The unique identifier for the partner that generated the audit record. Example: unknown
SessionId	
queryParams	This is applicable only for REST APIs. Query parameters present in the incoming REST request. Example: {"status":"available"}
RequestHeaders	Request header in the incoming request from the client. Example: <pre>{ "Cache-Control": "max-age=0", "Accept": "text/plain,application/json;q=0.9,image/webp,image/apng,*/*;q=0.8", "Upgrade-Insecure-Requests": "1", "Connection": "keep-alive", "User-Agent": "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36", "Host": "mcdaso02:5555", "Accept-Encoding": "gzip, deflate", "Accept-Language": "en-US,en;q=0.9,ta;q=0.8", "Content-Type": "application/x-www-form-urlencoded" }</pre>
ResponseHeaders	Response header in the outgoing response. Example: <pre>{ "Server": "Jetty(9.2.9.v20150224)", "Access-Control-Allow-Origin": "*", "Access-Control-Allow-Methods": "GET,POST,DELETE,PUT", "Connection": "close", "Date": "Fri, 30 Mar 2018 08:25:45 GMT", "Access-Control-Allow-Headers": "Content-Type,api_key,Authorization", "Content-Type": "application/xml" }</pre>
Session Id	A string the API Gateway server generates to uniquely identify each session. This is either the IS session token or the automatically generated GUID if the token is missing from the message context.

Column	Description
	Example: 81439d366e874bc79d9f81490e30e6e0
Source Gateway Node	Source API Gateway's IP address. Example: 10.0.75.1
TargetEPR	The endpoint URL of the native API that is invoked. Example: http://petstore.swagger.io/v2/pet/55

Monitoring Events

Column	Description
alertDesc	Text of the alert message sent to a configured destination when the performance conditions are violated. The alert message is specified in the policy definition of an API. Example: EnforcePolicy-HardLimit
alertSource	Name of the API Gateway policy that generated the alert message. Example: Unknown-Policy
apiName	Name of the API in which the event occurred.Example: Example: SampleAPI
apiVersion	The system-assigned version identifier for the API. Example: 1.0
applicationId	The unique identifier for the application associated with the API invocation. Example: c0f84954-9732-11e5-b9f4-f159eafe47b2
applicationIp	IP address of the application associated with the API invocation. Example: 10.20.248.33
applicationName	Name of the application associated with the API invocation. An application name is populated as unknown when API Gateway is unable to identify the application using a security policy that is configured for the API. Example: SampleApplication
creationDate	Date and time when the event was generated in API Gateway.

Column	Description
	Example: 1501671101509
eventType	The type of event that occurred. Example: Policy Violation Event
monitorAttr	The monitored attribute which has breached the configured SLA. Example: AVGRESPOSTIME GT 1.0, SUCCESSCOUNT EQ 3, REQUESTCOUNT GT 10
Native Endpoint	The endpoint URL of the native API that is invoked. Example: http://petstore.swagger.io/v2/pet/55
Operation/Resource name	Name of the API operation or resource that is invoked. Example: /pet

12 Microgateway Management

■ Overview	742
------------------	-----

Overview

API Gateway enables you to monitor the Microgateways that are connected to it. You can view the active APIs and detailed analytics for each Microgateway that is connected to the API Gateway.

The **Microgateways management** page displays all the Microgateway groups that are connected to the API Gateway. A Microgateway group enables you to group Microgateways that have some common element, such as domain (finance or human resources) or type of APIs (external-facing or internal use). For each Microgateway group, the **Microgateways management** page displays the following information:

- The number of Microgateways that are part of the group.
- The number of APIs that are available in that group.

You can perform the following operation on this page:

- Click **View details** to view more information about a Microgateway group.

Note:

For information about installing, configuring, and using Microgateways, see the *webMethods Microgateway User's Guide*.

Microgateway Groups

A Microgateway group enables you to group Microgateways. The **Microgateway groups** page displays the Microgateways that are included in a particular group. The page displays the following information for each Microgateway:

- Host name
- HTTP and HTTPS ports that the Microgateway uses to expose the APIs that are provisioned on it
- A description of the Microgateway
- The number of APIs available on the Microgateway

To add a Microgateway to the group, you need to add the following information to the `custom-settings.yml` file:


```
microgatewayPool:  
  microgatewayPoolName: poolNameHere  
  microgatewayPoolDescription: poolDescriptionHere
```

Where *poolNameHere* is the name of the group and *poolDescriptionHere* is an optional description of the group. If a *poolNameHere* is not provided, the Microgateway is added to the **Default** group.

Note:

For more information about `custom-settings.yml`, see the *webMethods Microgateway User's Guide*.

You can perform the following operations on this page:

- Click  to remove a Microgateway from the group.
- Click the Microgateway name to view more information about it.

Microgateway Details

The Microgateway details page provides information about a particular Microgateway.

The **Microgateway Info** tab includes two sub-tabs:

- The **Basic information** tab provides information about the Microgateway
- The **APIs** tab lists the APIs provisioned on that Microgateway. Clicking an API opens the **API details** page. The active Microgateway endpoints of the API are also displayed in the **API details** page.

Note:

All the Service Registries to which a Microgateway is publishing an API must be configured in API Gateway.

You can perform the following operations on this page:

- Click an API to view the API details.
- Click **Analytics** to view detailed analytics based on the data received from the Microgateway.

Microgateway Analytics

The Microgateway **Analytics** tab displays detailed analytics based on the data received from the Microgateway. This tab displays the following information:

- **Overall events:** Displays a pie chart that lists different events being monitored and each of these event categories is depicted with different colors.
- **Runtime events:** Displays the run time event details such as time when the event was generated, API Name, the application that generated the event, event type, description of the alert generated due to the event, status, and the source of event.
- **Payload size:** Displays the payload size of the request and responses during data transfer in the specified time. This data is picked up from the transactional event that is triggered when a log invocation policy is applied to the API.

You can perform the following operations on this page:

- Apply filters: The Microgateway **Analytics** tab provides filters that you can use to view selective data or events. You can use the displayed duration filter and add a custom filters using the filter query builder.
 - To apply a duration filter, select the time interval from the drop-down list, and click **Apply filter** to filter the analytics based on the time interval chosen. To specify a custom duration,

select **Custom** from the drop-down list, enter the required **From Date** and **To Date** values, and click **Apply filter**.

- You can also add filters based on a filter query. To add a filter based on a filter query, click **Add a filter**; enter the desired field, operator, and value; and click **Save**.
- View specific events: You can also click on the specific event in the list under Legend to view the specific event in any of the widgets. You can view additional details for an event by hovering the cursor over a particular color in the graphical representations.

13 REST APIs in API Gateway

■ API Gateway Administration	746
■ Alias Management	754
■ Application Management	754
■ API Gateway Archive	756
■ API Gateway Availability	756
■ Document Management	757
■ Internal Service	757
■ Policy Management	758
■ Promotion Management	760
■ Public Services	761
■ API Gateway Search	762
■ Server Information	765
■ Service Management	765
■ Transaction Data	766
■ User Management	767
■ Backward compatibility support for REST APIs	768

API Gateway Administration

API Gateway provides the capability to API definitions to administer various functions of the API Gateway.

API Gateway provides the following REST API and the resources to manage API Gateway configuration:

- **GET/rest/apigateway/is/jmsConnections/{jmsConnId}** : Retrieves the specified JMS connection in API Gateway.
- **DELETE/rest/apigateway/is/jmsConnections/{jmsConnId}** : Deletes the JMS connection based on the JMS connection ID that is specified in the path.
- **GET/rest/apigateway/is/jmsConnections** : Retrieves a list of all the JMS connections in API Gateway.
- **POST/rest/apigateway/is/jmsConnections** : Creates a JMS connection in API Gateway. The API request body must contain the payload for the JMS connection.
- **PUT/rest/apigateway/is/jmsConnections** : Updates the JMS connections in API Gateway.
- **PUT/rest/apigateway/is/jmsConnections/{jmsConnId}/enable** : Enables the specified JMS connection in API Gateway.
- **PUT/rest/apigateway/is/jmsConnections/{jmsConnId}/disable** : Disables the specified JMS connection in API Gateway.
- **GET/rest/apigateway/is/truststore/{truststoreName}** : Retrieves an existing truststore matching the given name from API Gateway.
- **POST/rest/apigateway/is/truststore/{truststoreName}** : Updates an existing truststore in API Gateway.
- **DELETE/rest/apigateway/is/truststore/{truststoreName}** : Deletes an existing truststore in API Gateway.
- **GET/rest/apigateway/is/truststore** : Retrieves all available truststores from API Gateway.
- **POST/rest/apigateway/is/truststore** : Creates a truststore in API Gateway.
- **GET/rest/apigateway/is/kerberos** : Retrieves the configured Kerberos settings from API Gateway.
- **PUT/rest/apigateway/is/kerberos** : Persists the configured Kerberos settings in API Gateway.
- **GET/rest/apigateway/is/license** : Retrieves the license details from API Gateway.
- **PUT/rest/apigateway/is/license** : Updates the license details in API Gateway.
- **GET/rest/apigateway/is/jmsTriggers** : Retrieves a list of all JMS triggers in API Gateway.
- **PUT/rest/apigateway/is/jmsTriggers** : Updates the JMS trigger in API Gateway.

- **PUT/rest/apigateway/is/jmsTriggers/{jmsTriggerId}/enable** : Enables the specified JMS trigger in API Gateway.
- **PUT/rest/apigateway/is/jmsTriggers/{jmsTriggerId}/disable** : Disables the specified JMS trigger in API Gateway.
- **GET/rest/apigateway/is/jmsTriggers/{jmsTriggerId}** : Retrieves the specified JMS trigger in API Gateway.
- **DELETE/rest/apigateway/is/outboundproxy/{outboundproxyAlias}** : Deletes the specified outbound proxy server alias from API Gateway.
- **GET/rest/apigateway/is/outboundproxy** : Retrieves the list of all available outbound proxy server aliases in API Gateway.
- **POST/rest/apigateway/is/outboundproxy** : Creates the outbound proxy server alias in API Gateway.
- **PUT/rest/apigateway/is/outboundproxy** : Updates the outbound proxy server alias in API Gateway.
- **PUT/rest/apigateway/is/outboundproxy/{outboundproxyAlias}/enable**: Enables an already existing outbound proxy server alias in API Gateway.
- **PUT/rest/apigateway/is/outboundproxy/{outboundproxyAlias}/disable**: Disables an already existing outbound proxy server alias in API Gateway.
- **GET/rest/apigateway/logAggregation/downloadLogs** : Downloads logs from different components used by API Gateway, server configurations, and thread dumps.
- **GET/rest/apigateway/is/cluster** : Retrieves the configured cluster settings from API Gateway.
- **PUT/rest/apigateway/is/cluster** : Updates the cluster settings in API Gateway.
- **GET/rest/apigateway/is/keystore/{keyStoreName}** : Retrieves the keystore matching the name specified in API Gateway.
- **POST/rest/apigateway/is/keystore/{keyStoreName}** : Updates an already existing keystore in API Gateway.
- **DELETE/rest/apigateway/is/keystore/{keyStoreName}** : Deletes the keystore matching the name specified in API Gateway.
- **GET/rest/apigateway/is/keystore** : Retrieves all keystores available in API Gateway.
- **POST/rest/apigateway/is/keystore** : Creates a keystore in API Gateway.
- **DELETE/rest/apigateway/apitransactions/typedefinitions** : Retrieves the list of runtime event types. The available event types are transactionalEvents, monitorEvents, errorEvents, performanceMetrics, threatProtectionEvents, lifecycleEvents, and policyViolationEvents. You can use these eventType to scope the archive or purge operation.
- **GET/rest/apigateway/apitransactions**: Retrieves the API transactions data. The data to be downloaded is filtered based on the input parameters. The user should be part of

API-Gateway-Administrators group or should have Manage purge and restore runtime events privilege to perform this operation.

- **DELETE/rest/apigateway/apitransactions**: Purges the API transactions data and the data to be purged is filtered based on the input parameters. This method returns the job id as response and the job id is used to track the job status.
- **GET/rest/apigateway/is/webServiceEndpoints/{webServiceEndpointId}** : Retrieves the specified Webservice endpoint in API Gateway.
- **DELETE/rest/apigateway/is/webServiceEndpoints/{webServiceEndpointId}** : Deletes the specified Webservice endpoint in API Gateway.
- **GET/rest/apigateway/is/webServiceEndpoints** : Retrieves list of all Webservice endpoints in API Gateway.
- **POST/rest/apigateway/is/webServiceEndpoints** : Creates a Webservice endpoint in API Gateway. The API request body must contain the payload for the Webservice endpoint.
- **PUT/rest/apigateway/is/webServiceEndpoints** : Updates the Webservice endpoint in API Gateway.
- **GET/rest/apigateway/apitransactions/archives** : Retrieves the details of existing archive files and response of this method would be the list of archive file names. You can select one of the archive file names returned by this method and use the POST /apitransactions/archives/{fileName} method to restore.
- **POST/rest/apigateway/apitransactions/archives** : Archives the runtime events and metrics. You can additionally scope the archive data using input parameter filters. This method returns the job id as the response which is used to know the status of the job.
- **POST/rest/apigateway/apitransactions/archives/{fileName}** : Restores the runtime data of the archive file that is specified. This method returns the job id as a response to track the status further.
- **GET/rest/apigateway/apitransactions/jobs/{jobId}** : Retrieves the status of a specific job. This method returns the status and file name (in case of archive process) as a response.
- **GET/rest/apigateway/apitransactions/jobs** : Retrieves a list of pending jobs. Every time you initiate archive, restore or purge process you get the job id as a response. You can use the specific job id to query the status of the initiated operation.
- **GET/rest/apigateway/portalGateways/{portalGatewayId}** : Retrieves an API Portal configuration in API Gateway.
- **PUT/rest/apigateway/portalGateways/{portalGatewayId}** : Updates the API Portal configuration in API Gateway.
- **DELETE/rest/apigateway/portalGateways/{portalGatewayId}** : Deletes the API Portal configuration in API Gateway.
- **GET/rest/apigateway/portalGateways** : Retrieves API Portal configurations available in API Gateway.

- **POST/rest/apigateway/portalGateways** : Creates API Portal configuration in API Gateway.
- **GET/rest/apigateway/portalGateways/communities**: Retrieves the details about communities in API Portal. An API can be published from API Gateway to any of the communities available in API Portal.
- **GET/rest/apigateway/portalGateways/packages**: Retrieves the details of the published packages that the API is part of.
- **POST/rest/apigateway/assets**: Changes ownership of application or APIs.
- **GET/rest/apigateway/licenseNotifications** : Retrieves the latest notification issued for a transaction based license.
- **GET/rest/apigateway/licenseNotificationCriteria** : Retrieves the existing transaction based license notification criteria as a response. Transaction based license notification criteria are like a usage checkpoint and whenever usage reaches that checkpoint, a notification is generated.
- **POST/rest/apigateway/licenseNotificationCriteria** : Creates the transaction based license notification criteria to monitor the API Gateway usage. This notification criteria has the permitted invocations per month defined in the license file. If you want to get notified when usage reaches a limit before it breaches the license limit, then you have to add a notification criteria by mentioning the usage point so that a notification is generated when usage reaches the specified limit.
- **PUT/rest/apigateway/licenseNotificationCriteria** : Updates the existing transaction based license notification criteria in API Gateway.
- **GET/rest/apigateway/licenseNotificationCriteria/{notificationCriteriaId}** : Retrieves the transaction based license notification criteria based on the specified ID.
- **DELETE/rest/apigateway/licenseNotificationCriteria/{notificationCriteriaId}** : Deletes the transaction based license notification criteria based on the specified ID.
- **GET/rest/apigateway/configurations/loadBalancer**: Retrieves information about the load balancer configured.
- **PUT/rest/apigateway/configurations/loadBalancer**: Updates the load balancer configuration information.
- **GET/rest/apigateway/configurations/whiteListingIPs**: Retrieves the details of the whitelisting IPs configuration in API Gateway.
- **PUT/rest/apigateway/configurations/whiteListingIPs**: Updates the details of the whitelisting IPs configuration in API Gateway.
- **GET/rest/apigateway/configurations/apiCallbackSettings**: Retrieves the API callback processor settings from API Gateway.
- **PUT/rest/apigateway/configurations/apiCallbackSettings**: Updates or creates API callback processor settings in API Gateway. The user should have Manage general administration configurations privilege to update the API callback processor settings.

- **GET/rest/apigateway/configurations/settings**: Retrieves the list of the extended settings and watt properties from API Gateway.
- **PUT/rest/apigateway/configurations/settings**: Updates or creates a list of the extended settings and watt properties in API Gateway.
- **GET/rest/apigateway/configurations/errorProcessing**: Retrieves the configured error template and the value of the property `sendNativeProviderFault`, which enables the server to forward the native error as it is.
- **PUT/rest/apigateway/configurations/errorProcessing**: Updates the default error template with any custom templates and the value of the property `sendNativeProviderFault`.
- **GET/rest/apigateway/configurations/keystore**: Retrieves the details of the default keystore, truststore and alias settings in API Gateway.
- **PUT/rest/apigateway/configurations/keystore**: Updates the details of the default keystore, truststore and alias configurations in API Gateway.
- **GET/rest/apigateway/configurations/gatewayDestinationConfig**: Retrieves the details of the API Gateway destination. API Gateway can publish events and performance metrics data. By default, error events, lifecycle events, policy violation event, and performance data are published to API Gateway.
- **PUT/rest/apigateway/configurations/gatewayDestinationConfig**: Updates the details of the API Gateway destination in API Gateway.
- **GET/rest/apigateway/configurations/auditlogDestinationConfig**: Retrieves the details of the Audit Log destination in API Gateway. Audit log captures the API runtime invocations performed in API Gateway. The audit log data is written to a file or a database based on the configurations. Transactions events are written to the audit log only when the Audit Log is selected as a destination in Log Invocation policy.
- **PUT/rest/apigateway/configurations/auditlogDestinationConfig**: Updates the details of the Audit Log destination in API Gateway.
- **GET/rest/apigateway/configurations/centraSiteDestinationCommunicationConfig**: Retrieves the communication details of the CentraSite destination in API Gateway. API Gateway can publish events and metrics to the configured CentraSite destination.
- **PUT/rest/apigateway/configurations/centraSiteDestinationCommunicationConfig**: Updates the communication details of the CentraSite destination in API Gateway.
- **GET/rest/apigateway/configurations/centraSiteDestinationSNMPConfig**: Retrieves the SNMP details of the CentraSite destination in API Gateway. API Gateway can publish events and metrics to the configured CentraSite destination.
- **PUT/rest/apigateway/configurations/centraSiteDestinationSNMPConfig**: Updates the SNMP details of the CentraSite destination in API Gateway.
- **GET/rest/apigateway/configurations/jdbcDestinationConfig**: Retrieves details of the Database destination in API Gateway. API Gateway can publish events and metrics to the configured database.

- **PUT/rest/apigateway/configurations/jdbcDestinationConfig**: Updates the details of the database destination in API Gateway.
- **GET/rest/apigateway/configurations/desDestinationConfig**: Retrieves details of the Digital Events destination in API Gateway. Digital Event Services (DES) enables API Gateway to communicate by exchanging digital events. Digital events are typed and serialized data structures that are used to convey or record information about the execution of a runtime.
- **PUT/rest/apigateway/configurations/desDestinationConfig**: Updates the details of the Digital Events destination in API Gateway.
- **GET/rest/apigateway/configurations/elasticsearchDestinationConfig**: Retrieves details of the Elasticsearch destination in API Gateway. API Gateway can publish events and metrics to the configured Elasticsearch destination.
- **PUT/rest/apigateway/configurations/elasticsearchDestinationConfig**: Updates the details of the Elasticsearch destination in API Gateway.
- **GET/rest/apigateway/configurations/snmpDestinationConfig**: Retrieves details of the SNMP destination in API Gateway. API Gateway can publish events and metrics to the configured third party SNMP destination.
- **PUT/rest/apigateway/configurations/snmpDestinationConfig**: Updates the details of the SNMP destination in API Gateway.
- **GET/rest/apigateway/configurations/emailDestinationConfig**: Retrieves details of the Email destination in API Gateway. API Gateway can send alerts to the email ID specified either in the Log Invocation, Monitor Service Performance, Monitor Service Level Agreement or Throttling Traffic Optimization policies through the configured Email destination.
- **PUT/rest/apigateway/configurations/emailDestinationConfig**: Updates the details of the Email destination in API Gateway.
- **GET/rest/apigateway/configurations/apiPortalDestinationConfig**: Retrieves details of the API Portal destination configuration. API Gateway can publish events and performance metrics data. By default, error events, lifecycle events, policy violation event, and performance data are published to API Portal.
- **PUT/rest/apigateway/configurations/apiPortalDestinationConfig**: Updates the details of the API Portal destination in API Gateway.
- **GET/rest/apigateway/configurations/cache**: Retrieves the cache configuration in API Gateway.
- **PUT/rest/apigateway/configurations/cache**: Updates the cache configuration in API Gateway.
- **GET/rest/apigateway/configurations/customContentTypes**: Retrieves the configured custom content types in API Gateway. Custom content types can be defined for base types XML,JSON and Text. These Custom types can be then used for payload processing in policies like Content based routing, Identify and access and Conditional error processing.
- **PUT/rest/apigateway/configurations/customContentTypes**: Updates the configured custom content types in API Gateway. The response is a set of key/value pair where key indicates the

custom content type and value indicates the base type. The value can be application/xml or application/json or text/xml.

- **GET/rest/apigateway/configurations/logConfig:** Retrieves the log settings of various components used by API Gateway.
- **PUT/rest/apigateway/configurations/logConfig:** Updates the details of the log configuration in API Gateway.
- **GET/rest/apigateway/configurations/ldapConfig:** Retrieves the LDAP configuration settings configured in API Gateway.
- **PUT/rest/apigateway/configurations/ldapConfig:** Updates the LDAP configuration settings configured in API Gateway.
- **GET/rest/apigateway/configurations/passwordRestrictions:** Retrieves the password restrictions settings configured in API Gateway.
- **PUT/rest/apigateway/configurations/passwordRestrictions:** Saves the password restrictions settings configured in API Gateway.
- **GET/rest/apigateway/configurations/passwordExpiry:** Retrieves the password expiry settings configured in API Gateway.
- **PUT/rest/apigateway/configurations/passwordExpiry:** Saves the password expiry settings configured in API Gateway.
- **GET/rest/apigateway/configurations/accountLockSettings:** Retrieves the account lock settings configured in API Gateway.
- **PUT/rest/apigateway/configurations/accountLockSettings:** Saves the account lock expiry settings configured in API Gateway.
- **PUT/rest/apigateway/masterPassword/setExpiry:** Updates the expiry interval of the master password in API Gateway.
- **PUT/rest/apigateway/masterPassword/update:** Updates the master password in API Gateway. On successful update, all the old passwords available will be encrypted using this new master password.
- **PUT/rest/apigateway/masterPassword/reset:** Resets the master password to the default value in API Gateway. This should be performed when the master password is lost and after a successful reset, Software AG recommends to change the master password again to a secure value.
- **GET/rest/apigateway/masterPassword:** Retrieves the master password properties in API Gateway.
- **GET/rest/apigateway/configurations/jsonWebToken:** Retrieves the details of the API Gateway JSON Web Token (JWT) configuration. API Gateway can generate a JWT itself or validate the JWT generated by a trusted third party server. JWT is a JSON-based open standard (RFC 7519) means of representing a set of information to be securely transmitted between two parties. A set of information is the set of claims (claim set) represented by the JWT. A claim set consists

of zero or more claims represented by the name-value pairs, where the names are strings and the values are arbitrary JSON values.

- **PUT/rest/apigateway/configurations/jsonWebToken:** Updates the details of the JWT configuration in API Gateway.
- **GET/rest/apigateway/approvalConfigurations:** Retrieves a list of available approval configurations in API Gateway.
- **POST/rest/apigateway/approvalConfigurations:** Creates an approval configuration in API Gateway.
- **GET/rest/apigateway/approvalConfigurations/{id}:** Retrieves the details of a specified approval configuration in API Gateway.
- **PUT/rest/apigateway/approvalConfigurations/{id}:** Updates the details of a specified approval configuration in API Gateway.
- **DELETE/rest/apigateway/approvalConfigurations/{id}:** Deletes the specified approval configuration in API Gateway.
- **GET/rest/apigateway/licenseUsageDetails:** Retrieves the detailed usage information for the transaction based license. The retrieved information contains the maximum number of invocations that is allowed for the current month, the total number of invocations used, and the remaining number of invocations available for the month.
- **GET/rest/apigateway/urlaliases:** Retrieves all URL Aliases or a URL Alias with a particular ID in API Gateway (if the query parameter alias is provided).
- **POST/rest/apigateway/urlaliases:** Creates a new URL alias in API Gateway.
- **PUT/rest/apigateway/urlaliases:** Updates an existing URL alias in API Gateway.
- **DELETE/rest/apigateway/urlaliases:** Deletes a URL alias in API Gateway.
- **GET/rest/apigateway/is/jndi/{jndiId}:** Retrieves the specified JNDI configuration in API Gateway.
- **DELETE/rest/apigateway/is/jndi/{jndiId}:** Deletes the specified JNDI configuration in API Gateway.
- **GET/rest/apigateway/is/jndi:** Retrieves a list of all JNDI configurations in API Gateway.
- **POST/rest/apigateway/is/jndi:** Creates a JNDI configuration in API Gateway. The API request body must contain the payload for the JNDI configuration.
- **PUT/rest/apigateway/is/jndi:** Updates the JNDI configuration in API Gateway.
- **GET/rest/apigateway/is/jndi/{jndiId}/test:** Tests the given JNDI configuration in API Gateway.
- **GET/rest/apigateway/is/jndi/template:** Retrieves a list of all JNDI templates in API Gateway.

For details about the REST API, see the `APIGatewayAdministration.json` at *Install directory/IntegrationServer/instances/default/packages/WmAPIGateway/resources/apigatewayservices*.

For details on sample payloads, import Postman collection from the following link in Postman client: <https://github.com/SoftwareAG/webmethods-api-gateway/blob/10.5/apigatewayervices/postmancollections/apis/administration-service/AdministrationService.json>.

Alias Management

API Gateway provides the capability to create aliases, retrieve alias information, update alias properties as required, and delete the existing aliases using a REST API.

API Gateway provides the following REST API and the resources to manage aliases:

- **GET/rest/apigateway/alias:** Retrieves the list of all aliases in API Gateway. You can also use this to retrieve details for a particular alias by providing the aliasName.
- **POST/rest/apigateway/alias:** Creates an alias in API Gateway.
- **GET/rest/apigateway/alias/{aliasId}:** Retrieves the details of the specified alias in API Gateway.
- **PUT/rest/apigateway/alias/{aliasId}:** Updates the details of the specified alias in API Gateway.
- **DELETE/rest/apigateway/alias/{aliasId}:** Deletes the specified alias in API Gateway.

For details about the REST API, see the `APIGatewayAlias.json` at *Install directory/IntegrationServer/instances/default/packages/WmAPIGateway/resources/apigatewayervices*.

For details on sample payloads, import Postman collection from the following link in Postman client: <https://github.com/SoftwareAG/webmethods-api-gateway/blob/10.5/apigatewayervices/postmancollections/apis/alias-management/AliasManagement.json>.

Application Management

API Gateway provides the capability to create applications, retrieve application information, update application properties as required, and delete the existing applications using a REST API. You can use this REST API to register APIs to the application, modify details of the registered APIs for the application, and unregister APIs from the application.

API Gateway provides the following REST API and the resources to manage applications:

- **GET/rest/apigateway/applications:** Retrieves the list of available applications in API Gateway. You can also use this to retrieve details for a particular application by providing the applicationId.
- **POST/rest/apigateway/applications:** Creates an application in API Gateway.
- **DELETE/rest/apigateway/applications:** Deletes the specified application in API Gateway.
- **GET/rest/apigateway/applications/{applicationId}:** Retrieves the details of the specified application in API Gateway.
- **PUT/rest/apigateway/applications/{applicationId}:** Updates the details of the specified application in API Gateway.

- **PATCH/rest/apigateway/applications/{applicationId}**: Suspends the specified application in API Gateway.
- **GET/rest/apigateway/applications/{applicationId}/apis**: Retrieves the list of registered APIs for the specified application in API Gateway.
- **POST/rest/apigateway/applications/{applicationId}/apis**: Registers APIs with the specified application in API Gateway.
- **PUT/rest/apigateway/applications/{applicationId}/apis**: Updates the details of the APIs that are registered with the specified application in API Gateway.
- **DELETE/rest/apigateway/applications/{applicationId}/apis**: Unregisters APIs from the specified application in API Gateway. You can also use this to unregister a particular API by providing the apiIDs.
- **GET/rest/apigateway/strategies**: Retrieves a list of all strategies in API Gateway.
- **POST/rest/apigateway/strategies**: Creates a strategy in API Gateway. The API request body must contain the payload for the strategy.
- **DELETE/rest/apigateway/strategies**: Deletes the specified strategy in API Gateway.
- **GET/rest/apigateway/strategies/{strategyId}**: Retrieves the details of the specified strategy in API Gateway.
- **PUT/rest/apigateway/strategies/{strategyId}**: Updates the details of the specified strategy in API Gateway.
- **PUT/rest/apigateway/strategies/{strategyId}/refreshCredentials**: Refreshes the credentials of the specified strategy in API Gateway.
- **GET/rest/apigateway/applications/{applicationId}/accessTokens**: Retrieves a map of access token endpoints for all the authorization servers configured in API Gateway.
- **POST/rest/apigateway/applications/{applicationId}/accessTokens**: Regenerates the access tokens of an application in API Gateway.
- **PUT/rest/apigateway/applications/{applicationId}/accessTokens**: Updates the access tokens of an application in API Gateway.
- **DELETE/rest/apigateway/applications/{applicationId}/accessTokens**: Deletes the access tokens from a specified application in API Gateway.
- **GET/rest/apigateway/applications/_search**: Retrieves a list of available applications in API Gateway based on the search query parameters.

For details about the REST API, see the `APIGatewayApplication.json` at *Install directory/IntegrationServer/instances/default/packages/WmAPIGateway/resources/apigatewayservices*.

For details on sample payloads, import Postman collection from the following link in Postman client: <https://github.com/SoftwareAG/webmethods-api-gateway/blob/10.5/apigatewayservices/postmancollections/apis/application-management/ApplicationManagement.json>.

For sample use cases, see <https://github.com/SoftwareAG/webmethods-api-gateway/tree/10.5/apigatewayservices/postmancollections/usecases>.

API Gateway Archive

You can import already exported archives of APIs, global policies, and other related assets and re-create them in API Gateway. Each artifact in an archive is associated with a universally unique identifier (UUID) across all API Gateway installations. When importing an archive, the UUID helps in determining whether the corresponding artifact is already available in API Gateway. In such a situation, you can specify whether to overwrite an already existing artifact during the import process.

API Gateway provides the following REST API and the resources to export and import an archive:

- **GET /rest/apigateway/archive:** Retrieves the archive, which is a ZIP file that contains the selected assets and its dependent assets.
- **POST /rest/apigateway/archive:** Imports the API Gateway archive as well as exports the assets as an archive.

For details about the REST API, see the `APIGatewayArchive.json` at *Install directory/IntegrationServer/instances/default/packages/WmAPIGateway/resources/apigatewayservices*.

For details on sample payloads, import Postman collection from the following link in Postman client: <https://github.com/SoftwareAG/webmethods-api-gateway/blob/10.5/apigatewayservices/postmancollections/apis/archive-service/ArchiveService.json>.

API Gateway Availability

API Gateway provides the capability to monitor the health of API Gateway and report the overall health of API Gateway. Each health check request displays a `status` field as the first entry. The status can have the values `green`, `yellow` or `red` describing the overall status of the components to check. This means that when any of the components signals a problem, then the status is set to `red`.

API Gateway provides the following REST API and the resources to monitor the health of API Gateway:

- **GET /gateway/availability/admin:** Retrieves the availability and health status of the API Gateway administration service (UI, Dashboards, Admin REST API).
- **GET /gateway/availability/engine:** Retrieves the availability and health status of the Gateway policy enforcement engine (ElasticSearch cluster, IS and TerraCotta).
- **GET /gateway/availability/externalServices:** Retrieves the availability of external services accessed by API Gateway.
- **GET /gateway/availability/all:** Retrieves the availability of the administration service of the policy enforcement engine and of the external services accessed by API Gateway.

For details about the REST API, see the `APIGatewayAvailability.json` at *Install directory/IntegrationServer/instances/default/packages/WmAPIGateway/resources/apigatewayservices*.

Note:

- To perform the following API Gateway Availability REST calls you must have the *View Administration Configuration* privileges.
 - GET /gateway/availability/externalServices
 - GET /gateway/availability/all
- To perform the following API Gateway Availability REST calls you must be a valid API Gateway user.
 - GET /gateway/availability/admin
 - GET /gateway/availability/engine

You can use the existing health check request GET `http://localhost:5555/rest/apigateway/health`, without any authentication being set, to retrieve the health of API Gateway that monitors the availability and health status of Kubernetes and Docker containers . This returns a HTTP 200 response without additional data.

Document Management

API Gateway provides the capability to store and manage the documents associated with an API.

API Gateway provides the following REST API and the resources to manage the documents associated with APIs:

- **GET/rest/apigateway/documents/{documentId}**: Retrieves the requested document from API Gateway.
- **PUT/rest/apigateway/documents/{documentId}**: Updates the requested document in API Gateway.
- **DELETE/rest/apigateway/documents/{documentId}**: Deletes the requested document from API Gateway.
- **PATCH/rest/apigateway/documents/{documentId}**: Patches the requested document in API Gateway.
- **POST/rest/apigateway/documents**: Creates and stores the documents in API Gateway.

For details about the REST API, see the `APIGatewayDocumentManagement.json` at *Install directory/IntegrationServer/instances/default/packages/WmAPIGateway/resources/apigatewayservices*.

For details on sample payloads, import Postman collection from the following link in Postman client: <https://github.com/SoftwareAG/webmethods-api-gateway/blob/10.5/apigatewayservices/postmancollections/apis/document-mangement-service/DocumentManagementService.json>.

Internal Service

API Gateway provides internal APIs that work on identified applications that are identified based on identifiers such as API Key, OAuth token, IP address and so on.

API Gateway provides the following REST API and the resources to manage application identification:

- **POST/{apigateway}/security/getJsonWebToken:** Generates JSON Web token with custom claims supplied in the request.
- **POST/{apigateway}/security/exchangeIDToken:** Generate an access token for the given ID Token.

For details about the REST API, see the swagger file `APIGatewayInternalService.json`, located at *Install directory/IntegrationServer/instances/default/packages/WmAPIGateway/resources/apigatewayservices*.

Policy Management

API Gateway provides the capability to retrieve API Gateway policy related data such as policies, parameters, policy stages, policy templates, binding assertion, token assertion and service result cache. You can use this REST API to create, update or delete policies.

API Gateway provides the following REST API and the resources to manage policies:

- **GET/rest/apigateway/denialofservice/deniedIP:** Retrieves the list of denied IPs (IPs that violated the threat protection rules configured).
- **DELETE/rest/apigateway/denialofservice/deniedIP:** Deletes the specified IP from the denied IP list. Once the IP is removed from the list the request from that IP is processed.
- **GET/rest/apigateway/assertions:** Retrieves a list of available assertions in API Gateway.
- **POST/rest/apigateway/assertions:** Creates an assertion in API Gateway. Custom assertions allow the API providers to extend and provide additional security policies that are not available by default in API Gateway. In WS-Security, custom assertions are used for expressing individual security requirements, constraints, or both. The individual policy assertions can be combined to create security policies that ensure secure and reliable exchanges of SOAP messages between a client and a SOAP API.
- **GET/rest/apigateway/assertions/{assertionId}:** Retrieves the specified assertion element.
- **PUT/rest/apigateway/assertions/{assertionId}:** Updates the specified assertion.
- **DELETE/rest/apigateway/tokenAssertion/{assertionId}:** Deletes the specified assertion.
- **GET/rest/apigateway/policyActionTemplates/{policyActionTemplateId}:** Retrieves the template details of the specified policy action.
- **GET/rest/apigateway/policyActionTemplates:** Retrieves all the template detail for list of policy actions. You can also use this to retrieve template details for a particular policy action by providing the policy action template Id.
- **GET/rest/apigateway/policyStages:** Retrieves the list of policy stages available in API Gateway. It also displays the list of policies associated with each stage.
- **GET/rest/apigateway/configurations/mobileApp:** Retrieves the configuration details for the mobile applications for which access has been denied. You can use API Gateway to disable access for certain mobile application versions on a predefined set of mobile platforms. By registering the required devices and applications and disabling access to these versions, you

ensure that all users use the latest versions of the applications and take advantage of the latest security and functional updates.

- **PUT/rest/apigateway/configurations/mobileApp:** Updates the details of the mobile applications configuration in API Gateway.
- **GET/rest/apigateway/policyActions:** Retrieves the list of all policy actions from API Gateway. It can also be used to retrieve details for particular set of policy actions by specifying the policy id, policy details for list of policies of a particular policy type.
- **POST/rest/apigateway/policyActions:** Creates policy actions of different types in API Gateway. The result of this request is a policy action payload and is available in the response.
- **GET/rest/apigateway/policyActions/{policyActionId}:** Retrieves the policy action details for a specified policy action based on the id specified in API Gateway.
- **PUT/rest/apigateway/policyActions/{policyActionId}:** Updates the policy action details for a specified policy action based on the id specified in API Gateway.
- **DELETE/rest/apigateway/policyActions/{policyActionId}:** Deletes the policy action based on the id specified in API Gateway.
- **GET/rest/apigateway/policies:** Retrieves the list of all policies from API Gateway. It can also be used to retrieve details for particular set of policies by specifying the policy id, policy details for list of policies of a particular policy type.
- **POST/rest/apigateway/policies:** Creates policies of different types in API Gateway. You can also use this to clone policies.
- **GET/rest/apigateway/policies/{policyId}:** Retrieves the policy details for a specified policy in API Gateway. If policy id is available then the policy details is sent in response.
- **PUT/rest/apigateway/policies/{policyId}:** Updates the policy details for a specified policy in API Gateway. For Global policy user should have API Gateway administrator access to update global policy.
- **DELETE/rest/apigateway/policies/{policyId}:** Deletes the specified policy in API Gateway. This request will automatically delete the associated policy action for this policy.
- **GET/rest/apigateway/policies/{policyId}/apis:** Retrieves the list of applicable APIs for a global policy. An API become applicable API for a global policy only if it satisfies the scope specified in the global policy. By default it will return the basic API details of all the applicable APIs either if the API is active or inactive for a global policy.
- **GET/rest/apigateway/policies/{policyId}/conflicts:** Retrieves the conflicts for the specified global policy.
- **PUT/rest/apigateway/policies/{policyId}/activate:** Activates the specified global policy. This request does not require any request body. This request tries to activate the global policy and if any error occurs during activation it is reported as response or if the global policy is activated then its policy details active flag set to true is sent as response. If the global policy has any conflicts then it cannot be activated and the conflicts are manually resolved.

- **PUT/rest/apigateway/policies/{policyId}/deactivate**: Deactivates the specified global policy. This request does not require any request body. This request tries to deactivate the global policy and if any error occurs during deactivation it is reported as response or if the global policy deactivated the policy details of a global policy with active flag set to false is sent as response. An active global policy cannot have conflicts with other active global policy and hence the deactivation fails only when the conflict occurs between active global policy that is specified and one or more applicable active APIs. This can happen when the applicable active API policy action depends on one or more policy action from the specified global policy. If you deactivate this policy, it would cause the active API to have an unstable state. Hence the deactivation is reported as failed in this case.
- **PUT/rest/apigateway/policies/{policyId}/disable**: Disables the Threat protection policy created in API Gateway. This request does not require any request body. If the threat protection policy is disabled successfully then the policy details of specified policy will be sent as response.
- **PUT/rest/apigateway/policies/{policyId}/enable**: Enables the Threat protection policy created in API Gateway. This request does not require any request body. If the threat protection policy is enabled successfully then the policy details of specified policy is sent as response.
- **PUT/rest/apigateway/policies/{policyId}/movedown**: Moves down the execution order of the Threat protection policy created in API Gateway.
- **PUT/rest/apigateway/policies/{policyId}/moveup**: Moves up the execution order of the Threat protection policy created in API Gateway.
- **GET/rest/apigateway/serviceResultCache/{apiId}**: Retrieves the Service Result Cache size for the specified API accessed using the API Id.
- **DELETE/rest/apigateway/serviceResultCache/{apiId}**: Deletes the Service Result Cache for the specified API accessed using the API Id.
- **GET/rest/apigateway/serviceResultCache**: Retrieves the Service Result Cache size for the specified API accessed using apiName and apiVersion.
- **DELETE/rest/apigateway/serviceResultCache**: Deletes the Service Result Cache for the specified API accessed using apiName and apiVersion.

For details about the REST API, see the `APIGatewayPolicyManagement.json` at *Install directory/IntegrationServer/instances/default/packages/WmAPIGateway/resources/apigatewayservices*.

For details on sample payloads, import Postman collection from the following link in Postman client: <https://github.com/SoftwareAG/webmethods-api-gateway/blob/10.5/apigatewayservices/postmancollections/apis/policy-management/PolicyManagement.json>.

For sample use cases, see <https://github.com/SoftwareAG/webmethods-api-gateway/tree/10.5/apigatewayservices/postmancollections/usecases>.

Promotion Management

API Gateway provides supports staging and promotion of assets. Staging and promotion allows you to promote all the assets across different stages.

API Gateway provides the following REST API and the resources to manage staging and promotion:

- **GET/rest/apigateway/promotion:** Retrieves the promotions history with each promotion entry providing the details such as promotion name, promoted by whom, when it is promoted, and the promoted assets status.
- **POST/rest/apigateway/promotion:** Promote the API Gateway assets from the source machine to destination machine where the destination machine is configured as a stage.
- **GET/rest/apigateway/promotion/{promotionId}:** Retrieves a promotion based on the promotion Id.
- **DELETE/rest/apigateway/promotion/{promotionId}:** Deletes a promotion based on the promotion Id.
- **GET/rest/apigateway/stages:** Retrieves all the configured stages.
- **POST/rest/apigateway/stages:** Configures a stage in the source API Gateway where promotion is initiated.
- **GET/rest/apigateway/stages/{stageId}:** Retrieves a particular stage object based on a stage Id.
- **PUT/rest/apigateway/stages/{stageId}:** Updates a particular stage in the source API Gateway where the promotion is initiated.
- **DELETE/rest/apigateway/stages/{stageId}:** Deletes a particular stage.
- **GET/rest/apigateway/rollback:** Retrieves the list of possible rollbacks from the local (target) API Gateway instance.
- **GET/rest/apigateway/rollback/{rollbackId}:** Retrieves a rollback based on the rollback Id.
- **PUT/rest/apigateway/rollback/{rollbackId}:** Rolls back the assets to the previous state, That is, the state prior to promotion. Rollback should be initiated from the local API Gateway instance.
- **DELETE/rest/apigateway/rollback/{rollbackId}:** Deletes the rollback.

For details about the REST API, see the `APIGatewayPromotionManagement.json` at *Install directory/IntegrationServer/instances/default/packages/WmAPIGateway/resources/apigatewayservices*.

For details on sample payloads, import Postman collection from the following link in Postman client: <https://github.com/SoftwareAG/webmethods-api-gateway/blob/10.5/apigatewayservices/postmancollections/apis/promotion-management/PromotionManagement.json>.

Public Services

This API allows you to fetch a JWT from API Gateway and also fetch JSON Web key URI of API Gateway.

API Gateway provides the following REST API and the resources to manage public services:

- **GET/rest/pub/apigateway/jwt/getJsonWebToken:** Retrieves JWT from API Gateway. To obtain the JWT from API Gateway the client has to pass the basic authentication credentials.

- **GET/rest/pub/apigateway/jwt/certs:** Retrieves all the public keys of API Gateway, which can be used to validate the JWT generated by API Gateway.

For details about the REST API, see the swagger file `APIGatewayPublicServices.json`, located at `Install directory/IntegrationServer/instances/default/packages/WmAPIGateway/resources/apigatewayservices`.

API Gateway Search

The API Gateway search API allows you to execute a search query in API Gateway and retrieve search results that match the search query.

Remember:

When your search involves a large number of records, the process consumes a considerable memory space from the server, which in turn affects other business transactions. Hence, Software AG recommends that you perform large search operations when you expect lesser business transactions so that the regular business is not affected.

API Gateway provides the following REST API resources:

- **POST/rest/apigateway/search:** Executes a search query in API Gateway and returns the results that match your query. You can perform search across the different objects such as APIs, Applications, Aliases, Assertions, Policies, Administration Settings, Policy properties, Packages, Plans, Subscriptions, Users, User groups, Transactional events, Lifecycle events, Policy violation events, Monitor events, Error events, Threat protection events, and Performance metrics.

To perform a search operation, specify the following in your REST request:

REST Request Section	Description
Types	<p>Objects for which you want to perform the search operation. You can specify one or more of the listed objects.</p> <p>Note: When you specify <code>Users</code> and <code>User Groups</code> in the Types section to return the list of users and user groups from Integration Server respectively, you need not specify any search criteria.</p>
Scope	<p>Search Criteria. You must specify your search attribute and a keyword (value of the attribute) or one of the following as your search criteria:</p> <ul style="list-style-type: none"> ■ Time range - to retrieve results for a date range (from and to values), from a specified date to current date, till a specified date, or since the given amount of time (seconds, minutes, hours, days, weeks, months, quarters, or years). ■ Value range - to retrieve results for an integer value range (from and to values), from a given value to the maximum value, and from 0 to the given value.

REST Request Section	Description
	<p>You can specify multiple attributes in this section.</p> <p>Note: The search operation is performed based on the search criteria specified in this section for all objects specified in the Types section.</p>
Condition	<p>One of the following:</p> <ul style="list-style-type: none"> ■ and - to return results that match all search criteria. ■ or - to return results that match any of the given criteria.
Fields	<p>Fields to be returned in the response. You can specify only the required fields, instead of viewing all fields in your response. That is, if you want to view only the API Names and Versions that match your search criteria, you can specify <code>apiName</code> and <code>apiVersion</code> in this section of your REST request.</p>

- **POST/rest/apigateway/search/_count:** Retrieves the total number of records for the specified scope and types.

To retrieve the count of records, you can specify the required types and scope similar to the `/search` query. If you do not specify any search criteria in the **Scope** section, then the query returns total number of assets for the objects specified in the **Types** section.

- **POST/rest/apigateway/search/_aggregations:** Executes a search query and groups the results for the specified scope and types.

To perform an aggregations search, specify the following in your REST request:

REST Request Section	Description
Types	<p>Objects for which you want to perform the search operation. You can specify one or more of the listed objects.</p>
Scope	<p>Search Criteria. You must specify your search attribute and a keyword (value of the attribute) or one of the following as your search criteria:</p> <ul style="list-style-type: none"> ■ Time range - to retrieve results for a date range (from and to values), from a specified date to current date, till a specified date, or since the given amount of time (seconds, minutes, hours, days, weeks, months, quarters, or years). ■ Value range - to retrieve results for a integer value range (from and to values), from a given value to the maximum value, and from 0 to the given value.
Condition	<p>One of the following:</p>

REST Request Section	Description
	<ul style="list-style-type: none"> ■ and - to return results that match all search criteria. ■ or - to return results that match any of the given criteria.
Aggregations	<p>Values for the following:</p> <ul style="list-style-type: none"> ■ Name - Specify a name used to group the required results. For example, you can specify <code>Info by API</code>, if you are grouping the results by APIs. ■ Type - One of the following: <ul style="list-style-type: none"> ■ group - to group the results based on the given fields. ■ timeseries - to group results based on a given interval value. The interval can be seconds, minutes, hours, days, weeks, months, quarters, or years. ■ metrics - to find the average, minimum, maximum and sum of given fields. ■ Fields - Fields to be considered for the aggregation. If the type is group and there are multiple fields, separate the field names with commas.

For details about the REST API, see the `APIGatewaySearch.json` at *Install directory/IntegrationServer/instances/default/packages/WmAPIGateway/resources/apigatewayservices*.

For details on sample payloads, import Postman collection from the following link in Postman client: <https://github.com/SoftwareAG/webmethods-api-gateway/blob/10.5/apigatewayservices/postmancollections/apis/search-service/SearchService.json>.

Note:

The number of transactions returned for a search is based on the value specified in the **defaultSearchSize** extended setting. If your search result exceeds the value of this setting, then you can navigate through your search results by specifying the range of records that you want to view. For example, the value specified in the **defaultSearchSize** setting is `1000` and the count of your search result is `5000`, then only the first `1000` records are displayed. To view the consequent records, you can specify the number of the record from which you want to view, and the number of records that must be displayed. That is, to view the records from `1001` to `2000`, you can specify the range as follows:

```
POST http://localhost:5555/rest/apigateway/search
{
  "types": [
    "TRANSACTION_EVENTS"
  ],
  "scope": [
    { "attributeName": "responseCode",
      "keyword": "304"
    },
  ],
}
```

```

"from": "1001"
"size": "1000"
}

```

Server Information

API Gateway provides the capability to retrieve API Gateway server information.

API Gateway provides the following REST API and the resources to retrieve the server information:

- **GET/rest/apigateway/is/serverinfo:** Retrieves API Gateway server information.

For details about the REST API, see the `APIGatewayServerInfoSwagger.json` at *Install directory/IntegrationServer/instances/default/packages/WmAPIGateway/resources/apigatewayservices*.

For details on sample payloads, import Postman collection from the following link in Postman client: <https://github.com/SoftwareAG/webmethods-api-gateway/blob/10.5/apigatewayservices/postmancollections/apis/server-information/ServerInformation.json>.

Service Management

API Gateway provides the capability to retrieve and manage all APIs in API Gateway and the related information such as applications associated, scopes, versions and so on.

API Gateway provides the following REST API and the resources to manage services:

- **GET/rest/apigateway/apis/{apiId}:** Retrieves an API based on the `apiId` specified.
- **PUT/rest/apigateway/apis/{apiId}:** Updates an API by importing a file, URL or inline based on the `apiId` specified.
- **DELETE/rest/apigateway/apis/{apiId}:** Deletes an API based on the `apiId` specified.
- **PUT/rest/apigateway/apis/{apiId}/activate:** Activates an API so that the API is exposed to consumers.
- **PUT/rest/apigateway/apis/{apiId}/deactivate:** Deactivates an API so that the API is not exposed to consumers.
- **PUT/rest/apigateway/apis/{apiId}/publish:** Publishes API to the registered API Portal.
- **PUT/rest/apigateway/apis/{apiId}/unpublish:** Unpublishes an API from the registered API Portal.
- **PUT/rest/apigateway/apis/{apiId}/mock/enable:** Enables you to mock an API by simulating the native service.
- **PUT/rest/apigateway/apis/{apiId}/mock/disable:** Disables the mocking capability to mock an API.
- **GET/rest/apigateway/apis:** Retrieves all APIs or subset of APIs based on the `apiIds` specified.
- **POST/rest/apigateway/apis:** Creates an API as specified. You can create an API by importing a file, URL, or from scratch.

- **DELETE/rest/apigateway/apis**: Deletes APIs based on the apiIds specified.
- **GET/rest/apigateway/apis/{apiId}/applications**: Retrieves the list of registered applications of an API.
- **GET/rest/apigateway/apis/{apiId}/source**: Retrieves the source file along with the root file name that was used while creating an API.
- **GET/rest/apigateway/apis/{apiId}/globalPolicies**: Retrieves the list of active global policies applicable for the specified API.
- **GET/rest/apigateway/apis/{apiId}/versions**: Retrieves all versions of the specified API.
- **POST/rest/apigateway/apis/{apiId}/versions**: Creates a new version of an API and retains applications if required.
- **GET/rest/apigateway/apis/{apiId}/scopes**: Retrieves the scopes for the specified API.
- **GET/rest/apigateway/apis/{apiId}/scopes/{scopeName}**: Retrieves the scopes for the specified API based on the scope name.
- **PUT/rest/apis/{apiId}/implementation**: Updates the API in API Gateway after its implementation by any API provider tool. This is used by API provider tools to update the API after implementing from their end.
- **GET/rest/apis/{apiId}/providerspecification**: Downloads the provider specification of REST and SOAP based APIs. Provider specification is nothing but, the specification file (in swagger or wsdl format) with out the concrete API Gateway endpoint and contains all resources, methods, and operations irrespective of whether their exposure to consumer.
- **PUT/rest/apigateway/serviceRegistry/unpublish**: Unpublishes one or more APIs from one or more service registries.
- **GET/rest/apigateway/serviceRegistry/publish**: Retrieves the service registry publish information for the API.
- **PUT/rest/apigateway/serviceRegistry/publish**: Publishes one or more APIs from one or more service registries.

For details about the REST API, see the `APIGatewayServiceManagement.json` at *Install directory/IntegrationServer/instances/default/packages/WmAPIGateway/resources/apigatewayservices*.

For details on sample payloads, import Postman collection from the following link in Postman client: <https://github.com/SoftwareAG/webmethods-api-gateway/blob/10.5/apigatewayservices/postmancollections/apis/service-management/ServiceManagement.json>.

For sample use cases, see <https://github.com/SoftwareAG/webmethods-api-gateway/tree/10.5/apigatewayservices/postmancollections/usecases>.

Transaction Data

API Gateway provides the capability to query the API transactions. API Transactions are generated (as events) every time an API invocation happens. API Transactions may contain the details about

the invocation such as request and response headers, request and response payloads, consumer applications and so on. API Provider may choose to store these events to one or more destinations by using Log Invocation Policy. API Gateway provides different destination options to the API Provider (like API Gateway's own data store, relational databases, Elasticsearch, and so on) where the events can be stored. By default, API Gateway is chosen as a storage destination for these events. This REST API queries for the transactions data only from the API Gateway's default datastore. There are multiple use cases where you can use this transactions data. For instance, you can integrate this API with your billing system wherein this transactional data can be used to compute the usage history of your API for different consumers for monetization usecases. In other scenarios, the data extracted from this service can be used for custom report generation.

API Gateway provides the following REST API and the resources to retrieve the transaction events data:

- **GET/rest/apigateway/transactionEvents/_search:** Retrieves the transaction events for a given API, Application, Plan or Package for a specific period of time. Multiple request parameters of this method provide options to specify the request criteria to match the expected result and most of these input parameters support regular expression in their values. Along with the mandatory parameters, `fromDate` and `toDate`, any one of the other filter criteria should be passed in the request.
- **GET/rest/apigateway/transactionEvents/_count:** Retrieves the number of transaction events for a given API, Application, Plan or Package for a specific period of time. Multiple request parameters of this method provide options to specify the request criteria to match the expected result and most of these input parameters support regular expression in their values. Along with the mandatory parameters, `fromDate` and `toDate`, any one of the other filter criteria should be passed in the request.

For details about the REST API, see the `APIGatewayTransactionDataService.json` at *Install directory/IntegrationServer/instances/default/packages/WmAPIGateway/resources/apigatewayservices*.

For details on sample payloads, import Postman collection from the following link in Postman client: <https://github.com/SoftwareAG/webmethods-api-gateway/blob/10.5/apigatewayservices/postmancollections/apis/transaction-data-service/TransactionDataService.json>.

User Management

API Gateway provides the capability to manage Users, Groups and Access profiles in API Gateway.

API Gateway provides the following REST API and the resources to retrieve the User ACL list:

- **GET/rest/apigateway/accessProfiles:** Retrieves a list of all access profiles in API Gateway.
- **POST/rest/apigateway/accessProfiles:** Creates an access profile in API Gateway. The API request body must contain the payload for the access profile.
- **GET/rest/apigateway/accessProfiles/{accessProfileId}:** Retrieves the details of an access profile in API Gateway.
- **PUT/rest/apigateway/accessProfiles/{accessProfileId}:** Updates the details of a specified access profile in API Gateway. The API request body must contain the payload for the updated access profile.

- **DELETE/rest/apigateway/accessProfiles/{accessProfileId}**: Deletes an access profile from API Gateway.
- **GET/rest/apigateway/groups**: Retrieves list of all groups in API Gateway.
- **POST/rest/apigateway/groups**: Creates a group in API Gateway. The API request body must contain the payload for the group.
- **GET/rest/apigateway/groups/{groupId}**: Retrieves the details of a group in API Gateway.
- **PUT/rest/apigateway/groups/{groupId}**: Updates the details of a specified group in API Gateway. The API request body must contain the payload for the updated group.
- **DELETE/rest/apigateway/groups/{groupId}**: Deletes a group from API Gateway.
- **GET/rest/apigateway/users**: Retrieves list of all users in API Gateway.
- **POST/rest/apigateway/users**: Creates an user in API Gateway. The API request body must contain the payload for the user.
- **GET/rest/apigateway/users/{userId}**: Retrieves the details of an user in API Gateway.
- **PUT/rest/apigateway/users/{userId}**: Updates the details of a specified user in API Gateway. The API request body must contain the payload for the updated user.
- **DELETE/rest/apigateway/users/{userId}**: Deletes the a specified user in API Gateway.
- **POST/rest/apigateway/users/authenticate**: Authenticates a user in API Gateway.
- **GET/rest/apigateway/installedLanguages**: Retrieves list of installed language packs in API Gateway.
- **GET/rest/apigateway/is/lockedAccounts**: Retrieves the locked user accounts in API Gateway.
- **POST/rest/apigateway/is/lockedAccounts**: Unlocks the locked user accounts by API Gateway.

For details about the REST API, see the `APIGatewayUserManagementSwagger.json` at *Install directory/IntegrationServer/instances/default/packages/WmAPIGateway/resources/apigatewayservices*.

For details on sample payloads, import Postman collection from the following link in Postman client: <https://github.com/SoftwareAG/webmethods-api-gateway/blob/10.5/apigatewayservices/postmancollections/apis/user-management/UserManagement.json>.

Backward compatibility support for REST APIs

All the REST APIs in API Gateway are backward compatible. The backward compatibility handles payload transformation from the previous version to the current version of API Gateway. If you want to use version specific payload then use the corresponding endpoint. For example, if you want to use the 10.1 payload to create an asset, then you have to use

```
http://hostname:port/rest/apigateway/v101/asset
```

With the backward compatibility support, API Gateway exposes the following REST end points with the version number mentioned.

- `http://hostname:port/rest/apigateway/v101/assetsspecificURI`

Use this URI if you want to access the latest API Gateway with 10.1 version specific request and response.

The following policies have conflicting behavior compared to earlier versions:

- In 10.1 `invokeESB` `templateKey` is used to create Invoke webMethods IS policy that can be used for both request and response transformation stage. From 10.2 version, the `invokeESB` `templateKey` is changed to `requestInvokeESB` and `responseInvokeESB` for request and response transformation stage respectively. So when you send a payload with older version (10.1), it is not possible to create the correct policy in latest version. To solve this, you have to update the payload, and send the appropriate `templateKey` in 10.1 payload. For example if you are creating invoke webMethods IS policy for request transformation, then you have to specify `requestInvokeESB` `templateKey` instead of `invokeESB` `templateKey`.
- In 10.1 `xsltTransformation` is used to create XSLT Transformation policy that can be used for both request and response transformation stage. From 10.2 version, the `xsltTransformation` `templateKey` is changed to `requestTransformation` and `responseTransformation` for request and response transformation stage respectively. To solve this, you have to update the payload, and send the appropriate `templateKey` in 10.1 payload. For example if you are creating XSLT Transformation policy for request transformation, then you have to specify `requestTransformation` `templateKey` instead of `xsltTransformation` `templateKey`.

- `http://hostname:port/rest/apigateway/v102/assetsspecificURI`

Use this URI if you want to access the latest API Gateway with 10.2 version specific request and response.

- `http://hostname:port/rest/apigateway/v103/assetsspecificURI`

Use this URI if you want to access the latest API Gateway with 10.3 version specific request and response.

- `http://hostname:port/rest/apigateway/assetsspecificURI`

When there is no version mentioned, the URI, by default, accesses the latest version specific request and response.

Note:

The archive REST endpoint to export assets does not give a version specific archive. It always gives the archive with latest version regardless of the version specified in the REST endpoint.

14 Remove User Data from API Gateway

■ Removing User Data	772
----------------------------	-----

Removing User Data

Data protection laws and regulations, such as the General Data Protection Regulation (GDPR) might require specific handling of user data, even after a user profile is removed. Additionally, employees or other clients with user accounts on API Gateway may request that any user identifying information such as user name, email addresses, or client IP addresses be removed from API Gateway. To comply with data protection requirements and user requests, in addition to deleting the user account, you may need to complete activities such as deleting or masking the user data.

Note:

API Gateway can optionally capture the runtime transaction logs which contain the API request and response data (customer-defined) that flow through to the API Gateway. Though there are options to purge or clean up these data, this section does not define procedure for the same as the customer-defined data is out of the scope of this functionality.

Types of Data and their stores in API Gateway

- **Core data**

This consists of APIs, policies, applications, aliases, packages, plans, administration configurations, users, and groups. This is stored in Internal Data Store and Integration Server store.

- **Runtime transactions**

This consists of the transaction events, monitoring events, error events, policy violation events, threat protection events, lifecycle events and performance metrics. This information can be stored in Internal Data Store or external destinations.

- **Application logs**

This consists of UI, server, Elasticsearch, Kibana, filebeat, and Platform logs . This is stored in filesystem and Internal Data Store.

- **Audit logs**

This is stored in Internal Data Store.

Handling Core Data

API Gateway strongly recommends the use of internal or technical user for policy configurations like Authorize user, Invoke IS Service and Outbound Authentication that has user credentials. This avoids the life cycle of actual or real user object from impacting the API Gateway configurations.

In case, real users are used in policy configurations, then the API Provider or API Administrators should take the responsibility of changing the configurations once the user is deleted from the system.

Handling Application Logs

In API Gateway, as you increase the log level to Debug or Trace, there can be messages that include the userid. So when a user has to be deleted from the system, Administrators have to clean up this data. The Application logs are stored in file system till API Gateway version 10.2. From API Gateway 10.3, API Administrators have an option to persist the logs additionally in Internal Data Store.

The following is a sample query run to mask the user data in the application log stored either in the Internal Data Store or an external data store.

```
curl -X POST -H 'Accept: application/json' -H 'Content-Type: application/json'
http://hostname:port/gateway_default_log/doc/_update_by_query -d ' {
"script": { "id": "findAndReplace", "params": {
"find": "user123", "replace": "*****"} } }
```

- *hostname:port* refers to the host name and port of the system where the Internal Data Store or the external data store that contains the data resides.
- The `Find` field contains the data or user information, such as username, id, that is to be masked.
- The field `Replace` contains the string that is used to mask the data mentioned in the `Find` field and replaces the data with the string provided in the logs.

Logs are always persisted in filesystem. You can perform a search and replace using text editing tools. For example, you could search all `server.log` files for the id of the user to be deleted and replace it with anonymous or blank string. Following logs need to be cleaned up.

- *Install Directory*\Integration Server\instances*Instance_Name*\logs\server.log. For details to cleanup other Integration Server related logs, see *webMethods Integration Server Administrator's Guide*.
- *Install Directory*>\Integration Server\instances*Instance_Name*\logs\APIGateway.log.
- *Install Directory*\InternalDataStore\logs.

Handling Audit Logs

If enabled, Audit logs would have information about user actions. This contains user reference and has to be cleaned up after user deletion. You can achieve the cleanup in the following ways:

- As an Administrator, you can clean up the Audit logs persisted in Internal Data Store by running the following curl query to replace or mask the desired data.

The following is a sample query run to mask the user data in the Audit logs stored in the Internal Data Store.

```
curl -X POST -H 'Accept: application/json' -H 'Content-Type: application/json'
http://hostname:port/gateway_default_audit/auditlogs/_update_by_query -d ' {
"script": { "id": "findAndReplace", "params": {
"find": "user123", "replace": "*****"} } } '
```

- *hostname:port* refers to the host name of the system where the Internal Data Store resides and the corresponding port.

- The `Find` field contains the data or user information, such as username, id, that is to be masked.
- The field `Replace` contains the string that is used to mask the data mentioned in the `Find` field and replaces the data with the string provided in the logs.
- In API Gateway 10.3, as an Administrator, you can use the extended setting `saveAuditLogsWithPayload` to not store the request payloads in the audit logs. Though this does not completely eliminate the user information in audit logs it certainly minimizes the occurrences. Software AG recommends you to use this property with caution as turning on this option might lead to less audit data being captured.

15 Usage Scenarios

■ Change Ownership of Assets	776
■ Custom Policy Extension	786
■ Team Support	802
■ API First Implementation	826
■ Gateway Endpoints	834
■ SAML SSO	840
■ Secure API using OAuth2 with refresh token workflow	847

Change Ownership of Assets

Assets such as APIs and applications in API Gateway have an option where the ownership of the asset can be changed. Applications have confidential data like API key and client certificates which only the owner can view. Therefore, if the owner of an asset has to take up a different responsibility or leave the organization, no other user can view the secrets of the asset. The edit option available on the asset details page, enables the transfer of ownership of the asset to another user, so that the new owner of the asset can access or view the confidential data of the asset. API Gateway provides an option to configure an approval process for the assets' ownership change. Approval and auditing contribute to the governance of change ownership.

Before you begin

Ensure that you have:

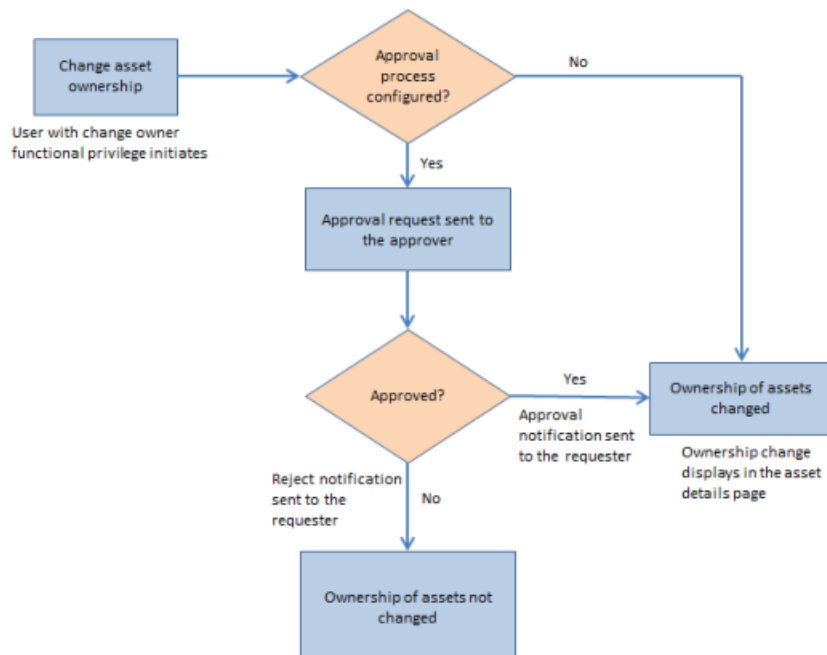
- API Gateway advanced edition version 10.5 or higher installed.
- Basic understanding of API Gateway and its related components like the API Gateway user interface.
- Change owner/team privilege.

For details on functional privileges available, see [“API Gateway Functional Privileges” on page 242](#).

- The change owner approval process configured and enabled if you want to enforce an approval process for ownership changes of assets.

For details on configuring the approval process, see [“How Do I Configure the Approval Process for Ownership Change of Assets?” on page 784](#).

The figure depicts the workflow for changing ownership of assets.



How Do I Change the Ownership of an Application?

This use case explains how to change the ownership of an application. You can configure an approval process for the change of ownership to take effect, if required.

The use case starts when an application requires a change of owner and ends when you successfully change the application's ownership.

In this example, an application *app1* is owned by *user1*. The ownership of *app1* has to be changed to *user2* through an approval process.

Before you begin

Ensure that you have the change owner privilege.

➤ To change the ownership of an application

1. Log on to API Gateway as a user with the change owner privilege.
2. Click **Applications** on the title navigation bar.
3. Click the required application **app1**.

The application details page appears. The owner of the application *app1* is *user1* as displayed in the Basic information section.

app1
View application details, identifiers, and access token information along with the APIs associated with the application. ⓘ

Application details

Basic information

Identifiers

Access tokens

APIs



Advanced

Authentication

Basic information

Name app1


Version 1.0

Owner user1  

Created 2019-07-08 05:44:56 GMT

Last updated 2019-07-16 10:36:07 GMT



4. Click .

5. Select user2 from the list and click .

Basic information

Name app1

Version 1.0

Owner user2  

Created 2019-07-08 05:44:56 GMT

Last updated 2019-07-16 10:36:07 GMT

The change approval process is initiated.

Note:

If the approval flow is not configured, the owner of the application changes to *user2* and a success message appears. Skip to step 8.

- An approval request is sent to the approver.
- The approver approves the request that resides in the Pending Requests section of the API Gateway UI.

WEBMETHODS API Gateway

APIs Policies Applications Packages Microgateways

Home Administration

Pending Requests

Manage your pending requests here. ⓘ

Approve Reject

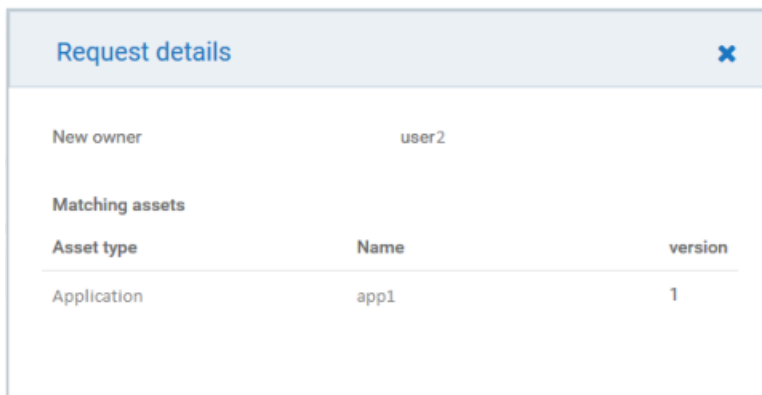
My requests Pending Requests

Requested by	Requestor comment	Event	Request details
<input type="checkbox"/> Administrator		Change ownership	Change ownership request details

Note:

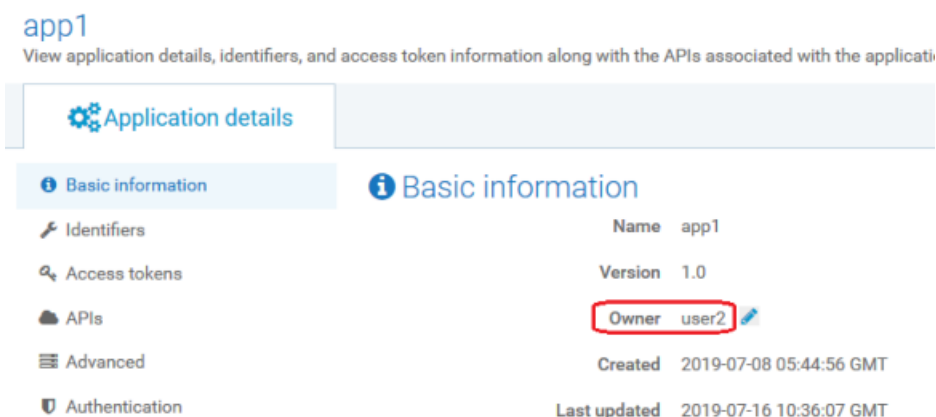
The approver can click **Reject** to reject the request for ownership change if the request is invalid. A reject notification is sent to the requester and the ownership of *app1* remains with *user1*.

Click **Change ownership request details** to view the request details. The Request details dialog box appears.



The approval notification is sent to the requester.

- The owner of the application *app1* is changed from *user1* to *user2*.



How Do I Change the Ownership of an API?

This use case explains how to change the ownership of an API. You can configure an approval process for the change of ownership to take effect, if required.

The use case starts when you have an API that requires a change of owner and ends when you successfully change the API's ownership.

In this example, an API *petstore* is owned by *user1*. The ownership of *petstore* has to be changed to *user2* through an approval process.

Before you begin

Ensure that you have the change owner privilege.

> To change the ownership of an API

1. Log on to API Gateway as a user with the change owner privilege.
2. Click **APIs** on the title navigation bar.
3. Click **petstore**.

The API details page appears. The owner of the API *petstore* is *user1* as displayed in the Basic information section.

petstore
View API details, basic and technical information, resources and methods available, and API specifications. ⓘ

API details | Scopes | Policies

Basic information | Technical information | Resources and methods | API mocking | Components | Documentation

Basic information

Name	petstore
Version	1.0
Owner	user1
Active	No
Maturity state	Beta
Created	2019-07-08 05:43:26 GMT
Last updated	2019-07-16 10:33:08 GMT

4. Click **change**.
5. Select user2 from the list and click .

Basic information

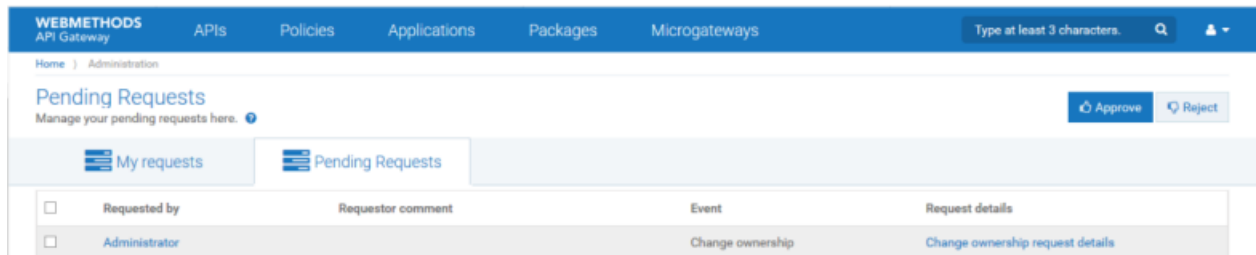
Name	petstore
Version	1.0
Owner	user2

The change approval process is initiated.

Note:

If the approval flow is not configured, the owner of the API changes to *user2* and a success message appears. Skip to step 8.

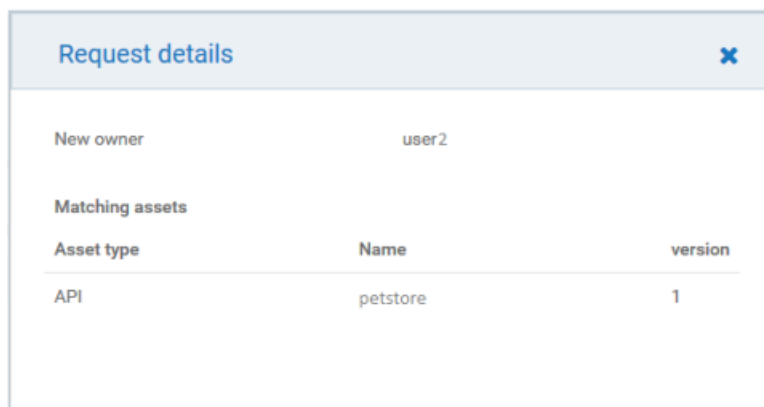
6. An approval request is sent to the approver.
7. The approver approves the request that resides in the Pending Requests section of the API Gateway UI.



Note:

The approver can click **Reject** to reject the request for ownership change if the request is invalid. A reject notification is sent to the requester and the ownership of *petstore* remains with *user1*.

Click **Change ownership request details** to view the request details. The Request details dialog box appears.



The approval notification is sent to the requester.

8. The owner of the API *petstore* is changed from *user1* to *user2*.

The screenshot shows the 'petstore' API details page. The 'Basic information' section is highlighted in the left sidebar. The main content area displays the following details:

Name	petstore
Version	1.0
Owner	user2
Active	No
Maturity state	Beta
Created	2019-07-08 05:43:26 GMT
Last updated	2019-07-16 10:40:31 GMT

How Do I Change the Ownership of Multiple Assets?

It is convenient to change the asset ownership for multiple assets with a single REST request than doing it separately for individual assets. This use case explains how to change the ownership of multiple assets by sending a REST request. You can configure an approval process, if required, for the change of ownership to take effect.

The use case starts when multiple assets require change of owner and ends when you successfully change the ownership of the assets to another user.

> To change the ownership of multiple assets

1. Use the following REST request to change the asset ownership to a new user.

```
POST http://host:port/rest/apigateway/assets/owner
Content-Type: application/json
{
  "assetType": "*", (API/APPLICATION)
  "assetIds": ["*"],
  "currentOwner": "user1",
  "newOwner": "user2"
}
```

Provide the following information in the REST request:

- **assetType.** Specifies the asset type for which you want to change the owner. Available values are `API`, `APPLICATION`, or the wildcard `*`. The wildcard `*` specifies all the assets, APIs and applications owned by the user specified in `currentOwner`.
- **assetIds.** Specifies the ID of the assets specified in `assetType`.

Note:

This is optional. `assetIds` is not required if you specify `currentOwner`.

- `currentOwner`. Specifies the user name of the owner of the assets specified in the `assetType` field.

Note:

If both `currentOwner` and `assetIds` are specified, both are validated. For example, consider *user1* and *user2* are owners of *assetID1* and *assetID2* respectively. In the request payload, if you include *assetID1* and *assetID2* in the `assetIds` field and *user1* in the **currentOwner** field, then only *assetID1* ownership changes.

- `newOwner`. Specifies the user name of the user who would be the new owner of the assets specified.

Example 1: If *user1* owns two assets, an API *petstore* and application *app1*, and you want the ownership to be transferred to *user2*, send a REST request as follows:

```
POST http://localhost:5555/rest/apigateway/assets/owner
Content-Type: application/json
{
  "assetType": "*", (API/APPLICATION)
  "currentOwner": "user1",
  "newOwner": "user2"
}
```

This request transfers the ownership of all the assets owned by *user1* to *user2*.

The change approval process is initiated.

Example 2: *user1* owns three APIs, *api1*, *api2*, and *api3* and 2 applications, *app1* and *app2*. If you want the ownership of *api1*, *api2*, and *app1* to be transferred to *user2*, send a REST request as follows:

```
POST http://localhost:5555/rest/apigateway/assets/owner
Content-Type: application/json
{
  "assetType": "*", (API/APPLICATION)
  "assetIds": ["apiID1, apiID2, appID1"],
  "currentOwner": "user1",
  "newOwner": "user2"
}
```

where *apiID1*, *apiID2*, and *appID1* are asset IDs of *api1*, *api2*, and *app1* respectively.

The change approval process is initiated.

Note:

If the approval flow is not configured, the ownership of the assets changes from *user1* to *user2*. Skip to step 4.

2. An approval request is sent to the approver.

The approval request contains information of all the assets whose ownership needs to change and the new owners' name.

3. The approver approves the request in the Pending Requests section of the API Gateway UI.

The approval notification is sent to the requester.

- The owner of the assets is changed from *user1* to *user2*.



How Do I Configure the Approval Process for Ownership Change of Assets?

If you want to enforce an approval process, configure and enable the approval process for ownership change of assets. The approver can approve or reject the request.

Before you begin


Ensure that you have Administrator privileges.

> To configure the approval process for ownership changes of assets


- On the title bar, expand the menu options icon  and select **Administration**.
- Select **General > Change owner/teams**.
- Set the **Enable** toggle button to the on position .
- Select the team of approvers from the **Approvers** list.
- Select Anyone in the **Approved by** list.

This specifies that any user associated with the approvers' access profile specified in the **Approvers** list can approve or reject the requests.

Approvers

Administrators 

Approved by

Anyone 

- In the Configure approval initiate request mail template to be sent to the approver section, provide the following information:
 - Select **Send notification** to send an email notification to the approver for the pending approval.
 - Provide the text to display in the subject line and the body of the email.

Configure approval initiate request mail template to be sent to approver

Send notification

Subject

Approval request pending

Content

Hello @approver.name,

A request by @requestor.name to @event.type needs your review and approval.

Best Regards,
API Gateway Team

*** This notification was sent automatically. Do not reply to this email.***

Note:

The at sign (@) character acts as a place holder and API Gateway automatically generates the values. For example, Hello @approver.name appears as Hello Joe in the email sent, where Joe is the approvers' login ID.

7. In the Configure request approved mail template to be sent to the requester section, provide the following information:
 - Select **Send notification** to send an email notification to the approval requester.
 - Provide the text to display in the subject line and the body of the email.

Configure request approved mail template to be sent to requester

Send notification

Subject

Approval of @event.type

Content

Congratulations @requestor.name !

Your request for @event.type has been approved.

*** This notification was sent automatically. Do not reply to this email.***

Note:

The at sign (@) character acts as a place holder and API Gateway automatically generates the values. For example, Approval of @event.type appears as Approval of Change ownership in the email sent, where Change ownership is the event.type.

8. In the Configure rejection mail template to be sent to the requester section, provide the following information:

- Select **Send notification** to send an approval rejection notification to the requester.
- Provide the text to display in the subject line and the body of the email.

[Configure rejection mail template to be sent to requester](#)

Send notification

Subject

Rejection of @event.type

Content

```

Hello @requestor.name,

Your @event.type request has been rejected.
Reasons:@rejectionReason.

Best Regards,
API Gateway Team

*** This notification was sent automatically. Do not reply to this email.***

```

Note:

The at sign (@) character acts as a place holder and API Gateway automatically generates the values. For example, Rejection of @event.type appears as Rejection of change of ownership of an asset in the email.

9. Click **Save**.

Custom Policy Extension

API Gateway provides a range of out-of-the-box policies to address common API management requirements like security, transformation, validation, error processing, and so on. In addition, API Gateway provides an option to add custom extensions. You can add these custom extensions into API Gateway policy stages to handle a requirement that might not be handled by any of the existing policies. You can use custom extensions in conjunction with the existing policies across stages. For example, if you want to invoke a third-party API or call an external endpoint during any stage of API processing, you can add custom logic in the corresponding policy stage and use it as required.

API Gateway supports the following custom extension types:

- **External endpoint**

Use this custom extension when you have an external endpoint exposed, which can be configured and invoked during any stage in API processing.

For example, if a native API expects the request in a certain format and the client application sends the request in a different format, you can add a custom extension to modify the incoming request to the required format before sending it to the native API.

■ webMethods IS service

Use this custom extension when you want to invoke the webMethods IS policy.

■ AWS Lambda

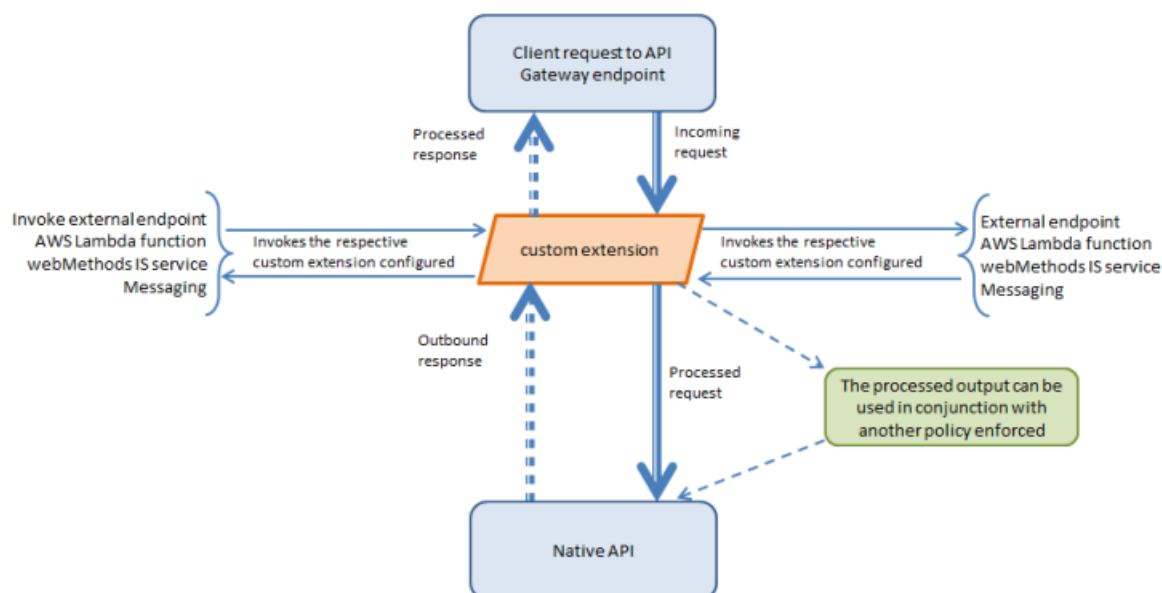
Use this custom extension to invoke an Amazon Web Services (AWS) Lambda function and use the business logic built-in the Lambda function in any stage of API processing.

■ Messaging

Use this custom extension when you want to send some data to a queue or topic during any stage in API processing and a system can read the message from the queue or topic and process it asynchronously.

Custom extensions are applicable to the REST, SOAP and OData API types. Custom extensions are supported at all levels such as, API, Scope, Global and can be added in any or all policy enforcement stages except the transport policy and the traffic monitoring policy stages.

The figure depicts a sample workflow for custom extension support in the request and response processing stages in API Gateway.



How Do I Invoke a Service through HTTP/HTTPS using Custom Extension?

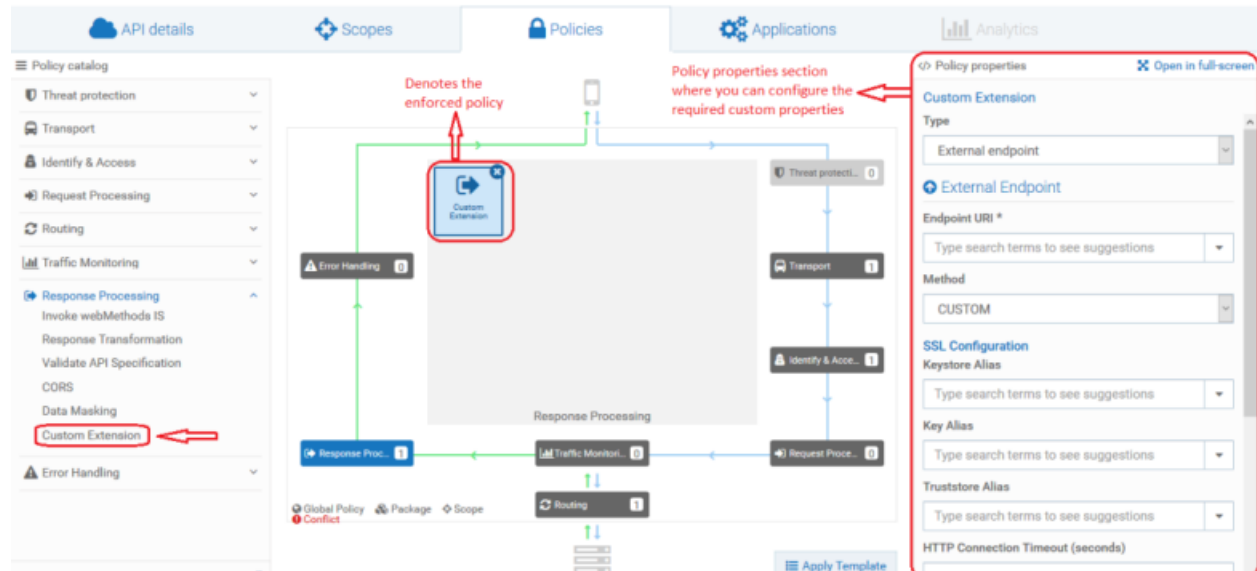
This use case explains how to invoke a service through HTTP/HTTPS using custom extension. The custom extension configured can be enforced in any of the policy stages and used during API processing.

The use case starts when you have an API that has to be enforced with a custom extension and ends when you successfully invoke the API with the custom extension enforced.

➤ **To invoke a service through HTTP/HTTPS using custom extension**

1. Ensure you have the external endpoint URL to be invoked during API processing using a custom extension.
2. Click **APIs** on the title navigation bar.
3. Click the required API.
The API details page appears.
4. Click **Edit**.
5. Select **Policies**.
6. Click **Required Policy stage > Custom Extension**.

This adds the custom extension policy where you can configure the required properties.



Click [Open in full-screen](#) to open the policy properties section in a full page.

7. Select **External endpoint** in the custom extension **Type** field.
8. Provide the following information in the External Endpoint section, as required:

Property	Description
Endpoint URI	Provide the external endpoint URI that you want to invoke.
Method	Specify the method exposed by the API. Available values are: PUT, POST, GET, DELETE, HEAD, CUSTOM .

Note:

Property	Description
	If you select CUSTOM , the HTTP method in the incoming request is sent to the native API.
SSL Configuration	<p>Specifies the required SSL configuration details of the external endpoint.</p> <p>Provide the following information:</p> <ul style="list-style-type: none"> ■ Keystore Alias. Specifies the keystore alias. For details on Keystore configuration, see “Keystore and Truststore” on page 109. ■ Key Alias. Specifies the alias for the private key, which must be stored in the keystore specified by the keystore alias. ■ Truststore Alias. Specifies the alias for the truststore. For details on Truststore configuration, see “Keystore and Truststore” on page 109. ■ HTTP Connection Timeout (seconds). Specifies the time interval (in seconds) after which a connection attempt to the external endpoint URL times out. ■ Read Timeout (seconds). Specifies the time interval (in seconds) after which a socket read attempt times out.
Path Parameters	<p>Specifies the path parameter you want to configure to your custom extension.</p> <p>Provide the following information:</p> <ul style="list-style-type: none"> ■ Path Parameter Name. Species the name of the path parameter you want to configure in your custom extension. This path parameter name should be present in the endpoint URL enclosed with {} to be replaced at runtime. For example, define external URL as <code>http://host/authors/{id}/books</code> and provide <code>id</code> as path parameter name with the value you need to populate at runtime. ■ Path Parameter Value. Specifies the value for the path parameter specified.

9. Configure the custom properties of the custom extension as required.

For details about the custom extension properties and their descriptions, see [“Custom Extension Properties” on page 797](#).

10. Click **Save**.

The API is saved with the added custom extension.

11. Invoke the API.

The applied custom extension invokes the mentioned HTTP/HTTPS endpoint and processes as configured.

How Do I Invoke an IS Service using a Custom Extension?

This use case explains how to invoke an IS service using custom extension in one of the policy stages and enforce during API processing.

For example you may want to process the request messages and transform them into a format required by the native API or perform some custom logic before API Gateway sends the requests to the native API.

The use case starts when you have an API which has to be enforced with a messaging custom extension and ends when you successfully invoke the API with the custom extension enforced.

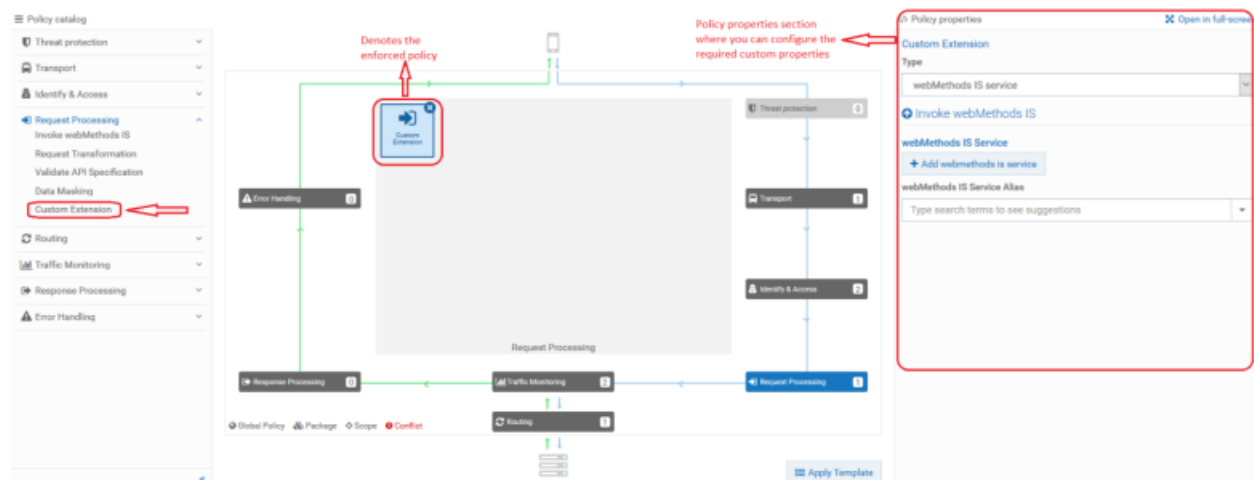
➤ To invoke an IS service using custom extension

1. Click **APIs** on the title navigation bar.
2. Click the required API.

The API details page appears.

3. Click **Edit**.
4. Select **Policies**.
5. Click **Any policy stage > Custom Extension**.

This adds the custom extension policy where you can configure the required properties.




Click [Open in full-screen](#) to open the policy properties section in a full page.

6. Select **webMethods IS service** in the custom extension **Type** field.

7. Provide the following information in the Invoke webMethods IS section, as required:

Property	Description
webMethods IS Service	<p>Specify the webMethods IS service to be invoked to process the messages.</p> <p>The webMethods IS service must be running on the same Integration Server as API Gateway.</p> <p>Note: If an exception occurs when invoking the webMethods IS service, by default API Gateway displays the status code as <i>500</i> and error message as <i>Internal Server Error</i>.</p> <p>You can set custom status code and error message by setting the following properties in the message context of the webMethods IS service:</p> <ul style="list-style-type: none"> ■ service.exception.status.code ■ service.exception.status.message <p>The sample code is given below:</p> <pre> IDataCursor idc = pipeline.getCursor(); MessageContext context = (MessageContext)IDataUtil.get(idc,"MessageContext"); if(context != null) { context.setProperty("service.exception.status.code", 404); context.setProperty("service.exception.status.message", "Object Not Found"); throw new ServiceException(); } </pre> <p>Note: If ServiceException or FlowException occurs when invoking webMethods IS Service, the message given in the exception is displayed to the client. If any other exception occurs, a generic error message is displayed to the client.</p>
Run As User	<p>Specifies the authentication mode to invoke the IS service.</p> <p>If this field is left blank the incoming credentials of the user, identified by API Gateway, are used to authenticate and invoke the IS service. You can also specify a particular user, you want API Gateway to invoke the IS service.</p>
Comply to IS Spec	<p>Select this property to mark it true, if you want the input and the output parameters to comply to the IS Spec present in pub.apigateway.invokeISService.specifications folder in wmAPIGateway package.</p>

Property	Description
webMethods IS Service Alias	<p>Specifies the webMethods IS service alias to be invoked to process the messages.</p> <p>Start typing the webMethods alias name, select the alias from the type-ahead search results displayed, and click  to add one or more aliases.</p>

- Click **Save**.

The API is saved with the added custom extension.

- Invoke the API.

The applied custom extension invokes the IS service and processes as configured.

How Do I Invoke an AWS Lambda Function using Custom Extension?

This use case explains how to invoke an AWS Lambda function using custom extension. The custom extension configured can be enforced in any of the policy stages and used during API processing.

The use case starts when you have an API that has to be enforced with a custom extension and ends when you successfully invoke the API with the custom extension enforced.

➤ To invoke an AWS Lambda function using custom extension

- Create a Lambda function and ensure it is active.

For details on how to create an AWS Lambda function, see <https://docs.aws.amazon.com/lambda/latest/dg/getting-started.html>.

- Configure AWS alias.

For details on how to configure an AWS alias, see “Configuring an AWS Alias” on page 794.

- Click **APIs** on the title navigation bar.

- Click the required API.

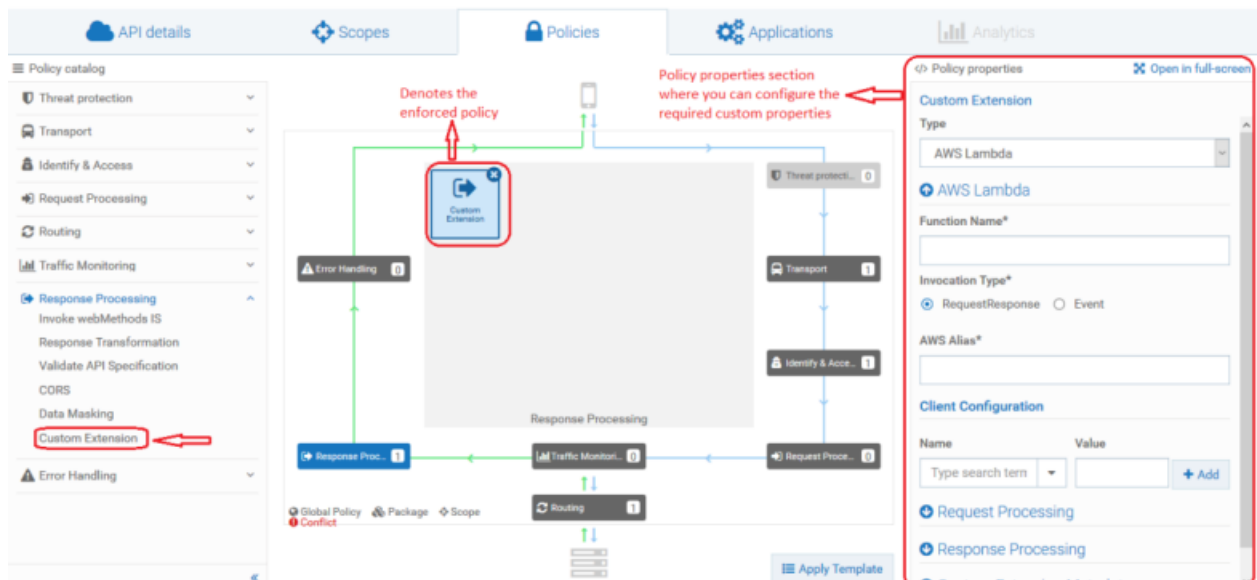
The API details page appears.

- Click **Edit**.

- Select **Policies**.

- Click **Required Policy stage > Custom Extension**.

This adds the custom extension policy where you can configure the required properties.



Click [Open in full-screen](#) to open the policy properties section in a full page.

8. Select **AWS Lambda** in the custom extension **Type** field.
9. Provide the following information in the AWS Lambda section, as required:

Property	Description
Function Name	Provide the AWS Lambda function name you want to invoke.
Invocation Type	Specify the AWS invocation type, asynchronous or synchronous. Available options are: <ul style="list-style-type: none"> ■ RequestResponse (synchronous type) ■ Event (asynchronous type)
AWS Alias	Provide the AWS alias configured for the AWS account.
Client Configuration	Provide the following client configuration details and click +Add . <ul style="list-style-type: none"> ■ Name. Start typing the client property name and select the required property from the type-ahead search results displayed. <p>API Gateway supports the following properties that you can configure: Socket timeout(ms), Connection timeout(ms), Request timeout(ms), Connection expiration timeout(ms), Maximum Connection idle time(ms), Client execution timeout(ms), Server error retry count, Enable throttle retries, Maximum client retry count, TCP send buffer size hints, TCP receive buffer size hints, Enable gzip requests, Enable Expect-Continue, Enable host prefix injection, Enable Keep-alive,</p>

Property	Description
	<p>Enable, Response metadata caching, Response metadata cache size, and Signature Algorithm.</p> <ul style="list-style-type: none"> ■ Value. Provide a value for the client property specified. <p>You can configure multiple properties.</p> <p>For details about the supported client properties, see the following AWS documents:</p> <ul style="list-style-type: none"> ■ https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/section-client-configuration.html ■ https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/amazonaws/ClientConfiguration.html

10. Configure the custom properties of the custom extension as required.

For details about the custom extension properties and their descriptions, see “[Custom Extension Properties](#)” on page 797.

11. Click **Save**.

The API is saved with the added custom extension.


12. Invoke the API.

The applied custom extension invokes the AWS lambda function and processes as configured.

Configuring an AWS Alias

Configure an AWS alias if you are using a custom extension that calls an AWS Lambda function from within a runtime policy enforcement flow of API Gateway API.

> To configure an AWS alias

1. On the title bar, expand the menu options icon  and select **Administration**.
2. Click **External accounts > AWS configuration**.
3. Click **Add new AWS account**.
4. In the Add AWS configuration section, provide the following information
 - a. **Name.** Provide the AWS alias name.
 - b. **Description.** *Optional.* Provide a description for the alias.

- c. **Region.** Provide the AWS account region where the Lambda function is running or deployed.

For details on how to create an AWS Lambda function, see <https://docs.aws.amazon.com/lambda/latest/dg/getting-started.html>

- d. **Access key ID.** Provide the access key ID of the AWS account where the Lambda function is running.
- e. **Secret access key.** Provide the secret access key of the AWS account where the Lambda function is running.

5. Click **Add**.

This creates the AWS alias and lists in the AWS configuration page.

How Do I Invoke a Service Asynchronously through JMS/AMQP using a Custom Extension?

This use case explains how to add messaging as a custom extension in one of the policy stages and invoke a service asynchronously during API processing.

You want to use the AMQP messaging setup to send some data to a queue during request processing using the configured custom extension. This data that is sent can then be read from a queue, processed, and sent in an asynchronous way.

The use case starts when you have an API which has to be enforced with a messaging custom extension and ends when you successfully invoke the API with the custom extension enforced.

» To invoke a service asynchronously through JMS/AMQP using custom extension

1. Ensure you have a JMS/AMQP environment set up with the required connection alias configured.

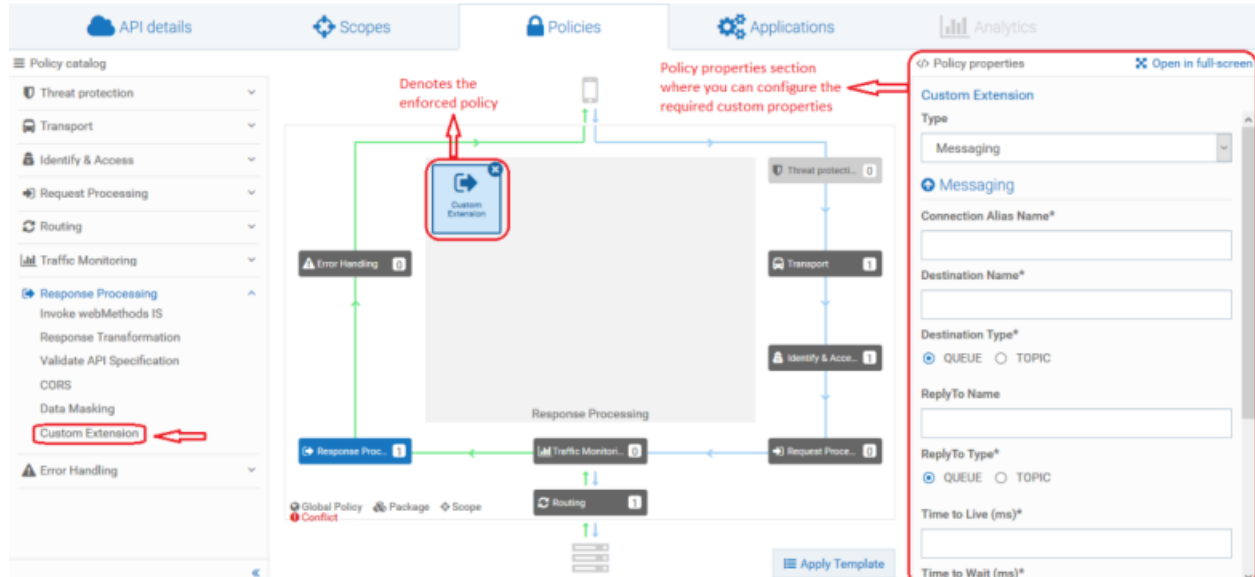
For details on setting up the JMS/AMQP setup, see [“Messaging” on page 83](#).

2. Click **APIs** on the title navigation bar.
3. Click the required API.

The API details page appears.

4. Click **Edit**.
5. Select **Policies**.
6. Click **Any policy stage > Custom Extension**.

This adds the custom extension policy where you can configure the required properties.



7. Select **Messaging** in the custom extension **Type** field.
8. Provide the following information in the Messaging section, as required:

Property	Description
Connection Alias Name	Name of the connection alias you have configured. You can configure the connection alias under Administration > Messaging section. For details on how to configure the connection alias, see “Messaging” on page 83 .
Destination Name	Specify the destination to which the request message is sent.
Destination Type	Specify the destination type to which the request message is sent.
Reply To Name	Specify the destination to which the response message is sent.
Reply To Type	Specifies the destination type to which the response message is sent. Select one of the following types: <ul style="list-style-type: none"> ■ QUEUE. Indicates that the response message is sent to a particular queue. ■ TOPIC. Indicates that the response message is sent to a particular topic.
Time to Live (ms)	Provide a numeric value that specifies the expiration time (in milliseconds) of the JMS or AMQP message. If the time-to-live is specified as zero, expiration is set to zero, which indicates that the message does not expire.

Property	Description
Time to Wait (ms)	Defines the time in milliseconds for which API Gateway listens to the Reply To Queue or Topic for the response message.
Delivery Mode	<p>The message delivery mode for the request message. This is the delivery mode that web service clients must specify in the JMS or AMQP message that serves as the request message for the web service.</p> <p>Select one of the following modes:</p> <ul style="list-style-type: none"> ■ Non-Persistent. Indicates that the request message is not persistent. The message might be lost if the JMS provider fails. ■ Persistent. Indicates that the request message should be persistent. The message is not lost if the JMS provider fails.

- Configure the custom properties of the custom extension as required.

For details on the custom extension properties and their description, see [“Custom Extension Properties” on page 797](#).

- Click **Save**.

The API is saved with the added custom extension..

- Invoke the API.


The applied custom extension calls the queue or topic that is configured.

Custom Extension Properties

The table lists the properties that you can specify for a custom extension.

Request Processing Section

The table lists the custom extension properties you can configure in the Request processing section:

Property	Description
Payload	<p>Provide the request payload to be sent to the custom extension in one of the following ways:</p> <ul style="list-style-type: none"> ■ Type the request payload in the text box. <p>For details on the data objects and variables available in the Request Processing section that you can use to configure, see “Data Objects and Variables Available in API Gateway” on page 800.</p> <ul style="list-style-type: none"> ■ Click  and select one of the following and provide the required information: <ul style="list-style-type: none"> ■ Inline Request. Type the required payload. ■ Load from Schema. Click Browse to upload a JSON or XML schema file and click Save.
Headers	<p>Provide the following information, if you want to configure the headers you need to send to the custom extension. By default, no headers are sent to the custom extension.</p> <ul style="list-style-type: none"> ■ Select Use incoming headers to use the header content in the incoming requests from the client. ■ Provide the Header Name and the Header Value in the incoming client request that has to be processed.
Query Parameters	<p>Provide the following information, if you want to configure query parameters you need to send to the custom extension.</p> <ul style="list-style-type: none"> ■ Provide the Query Parameter Name and the Query Parameter Value in the incoming client request that has to be processed. <p>For details on the data objects and variables available in the Request Processing section that you can use to configure, see “Data Objects and Variables Available in API Gateway” on page 800.</p>

Response Processing section

The table lists the custom extension properties you can configure in the Response processing section:

Property	Description
Copy the entire response	<p>Select to copy the entire response received from the external call out.</p> <p>This response is used in the subsequent step by using <code>\${request.payload}</code> or <code>\${response.payload}</code>.</p>

Property	Description
	<p>Note: Do not select this if you are using AWS Lambda custom extension with invocation type as Event as there is no response returned.</p>
<p>Abort API execution in case of failure</p>	<p>Select to abort the API execution when the external callout encounters any failures.</p> <p>If you do not select this option, API Gateway logs the failure and continues with the processing.</p>
<p>Transformation</p>	<p>Specify the following custom variables with a syntax to be accessed from the response of the custom extension and click Add.</p> <ul style="list-style-type: none"> ■ Variable. Specifies the variable type with a syntax. ■ Value. Specifies a value with a syntax. <p>Example 1</p> <p>For example if you provide a variable as <code>\${var}</code> and the corresponding value as <code>\${response[customExtension].payload.jsonPath[\$. id]}</code>, this transformation evaluates the JSON path from the custom policy response payload to get the value of the attribute <code>id</code>. The evaluated value is assigned to the variable <code>var</code> given in the Variable field. You can use the <code>\${var}</code> syntax in the subsequent custom extension.</p> <p>Example 2</p> <p>If you have a Request payload - <code>{"EmpName": "Emp1", "EmpNum": "EmpId"}</code>, and</p> <p>Response payload from Custom Extension external endpoint - <code>{"EmpId": "50001"}</code></p> <p>To transform the payload as <code>{"EmpName": "Emp1", "EmpNum": "50001"}</code>, you can use a JSON path parameter, and provide the Variable and Value as follows:</p> <p>Variable - <code>\${request.payload.jsonPath[\$. EmpNum]}</code></p> <p>Value - <code>\${response[customExtension].payload.jsonPath[\$. EmpId]}</code></p> <p>This fetches the value <code>50001</code> from the response of the external endpoint and replace the value <code>EmpId</code> in the existing payload.</p> <p>For details about the data objects and variables available in the Response Processing section that you can use to configure, see “Data Objects and Variables Available in API Gateway” on page 800.</p>
<p>Custom extension metadata</p>	<p>This is used for XML transformation.</p>

Property	Description
	<ul style="list-style-type: none"> ■ Namespace Prefix. Provide the namespace prefix of the payload expression to be validated. ■ Namespace URI. Provide the namespace URI of the payload expression to be validated.

Custom Extension Metadata section

The table lists the custom extension properties you can configure in the Custom Extension Metadata section. This is applicable only for XML transformation.

Property	Description
Namespace Prefix	Provide the namespace prefix of the payload expression to be validated.
Namespace URI	Provide the namespace URI of the payload expression to be validated.

For details about the data objects and variables that you can use to configure, see [“Data Objects and Variables Available in API Gateway” on page 800](#).

Data Objects and Variables Available in API Gateway

The following table summarizes the data objects and variables that are available in API Gateway:

Object/Variable type	Possible values
paramStage	<ul style="list-style-type: none"> ■ request ■ response
paramType	<ul style="list-style-type: none"> ■ payload OR body ■ headers ■ query ■ path ■ httpMethod ■ statusCode ■ statusMessage
queryType	<ul style="list-style-type: none"> ■ xpath ■ jsonPath ■ regex

The following data objects are available in the request processing or response processing steps:

- `${paramStage.paramType}`

You can use this syntax to access the following string variables: `path`, `statusCode`, `statusMessage`, `httpMethod`. Examples: `${request.path}`, `${response.statusCode}`

- `${paramStage.paramType.paramName}`

You can use this syntax to access map types, such as `query`, `headers`, and `path`. Example: `${request.query.var1}`, `${response.header.Content-Type}`, `${request.path.name}`.

- `${paramStage.paramType.queryType[queryValue]}`

This syntax can be used to query a `paramType`. Examples:

- `${request.payload.xpath[//ns:emp/ns:empName]}`

Where `//ns:emp/ns:empName` is the XPath to be applied on the payload if `contentType` is `application/xml`, `text/xml`, or `text/html`.

- `${response.payload.jsonPath[$.cardDetails.number]}`

Where `$.cardDetails.number` is the `jsonPath` to be applied on payload if `contentType` is `application/json` or `application/json/badgerfish`.

- `${request.payload.regex[[0-9]+]}`

Where `[0-9]+` is the regular expression to be applied on the payload if `contentType` is `text/plain`.

Note:

While `xpath` and `jsonPath` are applicable only to payload, `regex` can be used with both payload and path.

- `${paramStage[stepName].paramType.paramName}`

You can use this syntax to access header or payload from the response of the custom extension in the response processing step.

Example:

Variable: `${response.headers.id}`

Value: `${response[customExtension].payload.jsonPath[$.id]}`

This transformation adds a header to the response with name `id` and its value is derived from the json payload that is sent from the external callout as per the json path.

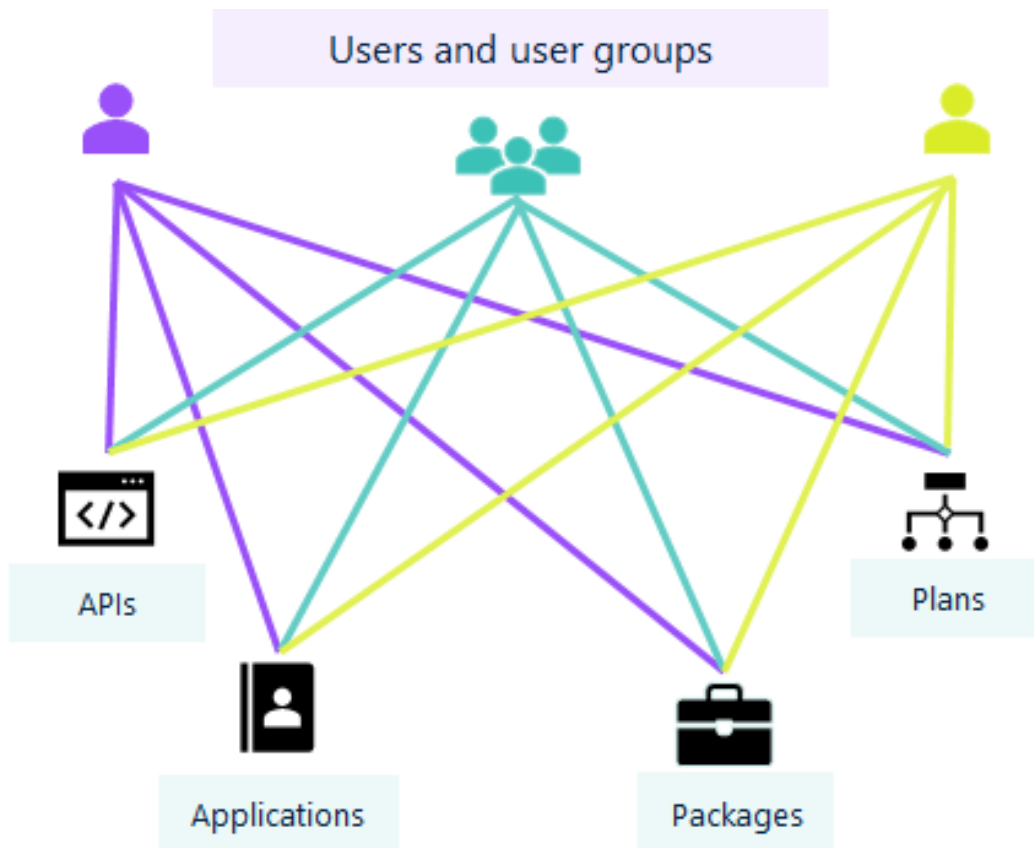
- You can define your own variables in the Transformation variables field in the response processing step.

Examples: `${key}`, `${value}`. The custom transformation variables that you define are available in subsequent steps.

Team Support

When users from multiple business units of an organization share the same API Gateway instance, by default, all users have access to all assets irrespective of their departments.

In a typical deployment scenario, all users have same level of access privileges to all API Gateway assets. However, there could be requirement for users from different business units to have different levels of access to specific assets; and they would not want interference from each other.

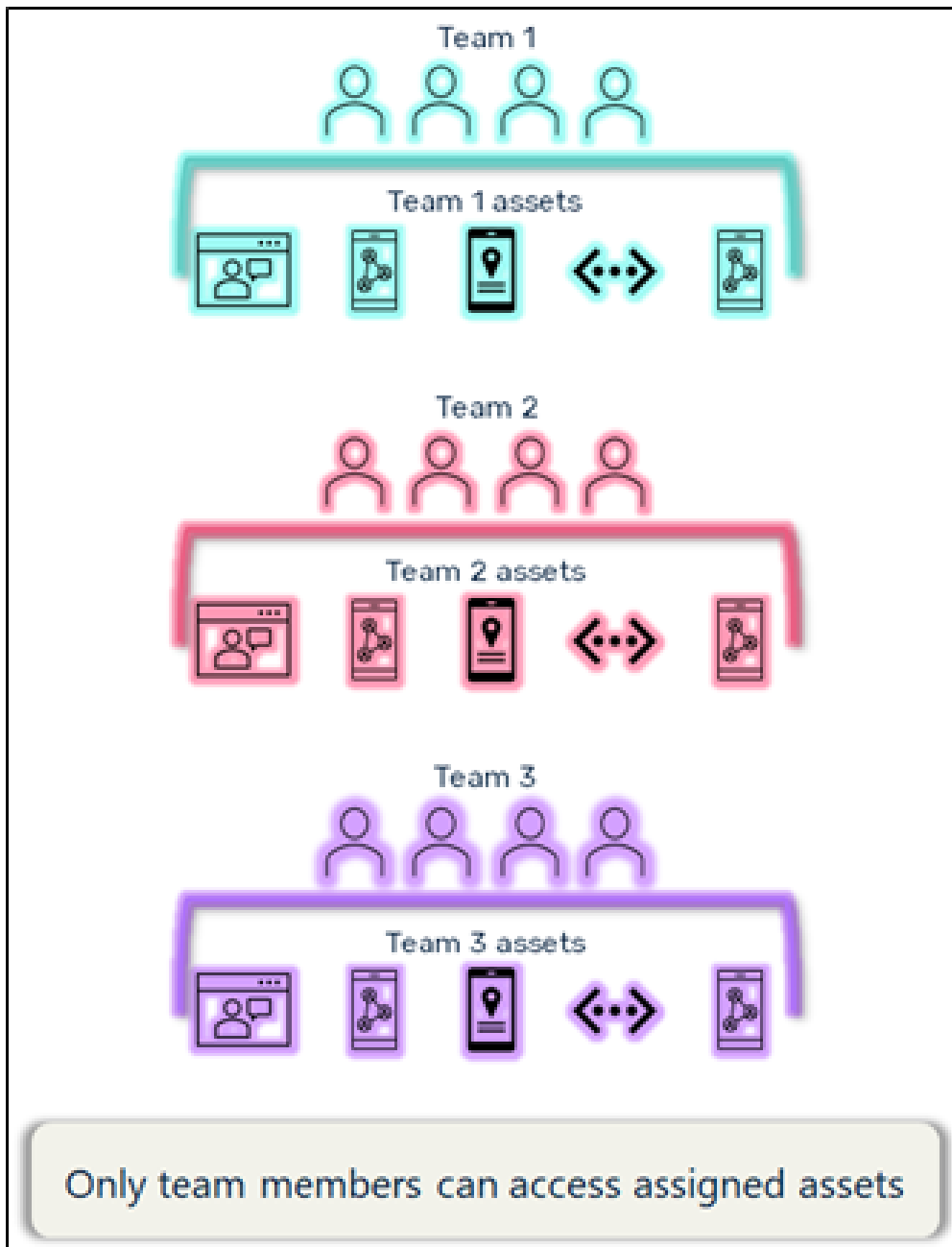


All users and user groups have access to all assets based on their access privileges

As a platform administrator, keeping in mind the various role-based access requirements, how do I

- group users of a business unit or a project with similar roles and assign certain assets to these teams?
- assign different access privilege to different set of users to specific assets?

This is where the **Team support** feature of API Gateway is useful. In a shared environment, this feature enables you to provide different level of access to different users to access specific assets.



The **Team support** is disabled by default; and you must enable the feature to use it. For information about enabling the feature, see [“Enabling Team Support” on page 806](#).

When to use Team support?

Team support can be used to group the users of a business unit or users with similar roles in your organization. Using this feature, you can assign assets for each team and specify the access level of team members based on the team members' project requirements.

This feature is helpful for organizations that have multiple business units, who work on different projects. Users can access only the assets that are assigned to them. For example, consider an organization with different teams such as Development, Configuration Management, Product Analytics, and Quality Assurance. Each of these teams needs access to different assets at different levels. That is, developers would require APIs to develop applications and they require the necessary privileges to manage APIs and applications. Similarly, analysts would want the necessary privileges to view performance dashboards of assets. In such scenarios, you can group users based on their roles as a team and assign them the necessary privileges based on their responsibility.

Prior to the 10.5 version, users were given the necessary privileges using Access Profiles. Starting version 10.5, you can limit the access of your asset to the required team members and assign access privileges using the **Team support** feature.

What is a Team in API Gateway?

A team can be defined as a group of users with a set of defined responsibilities.

The assets supported by this feature are APIs, applications, packages, and plans.

This table lists the important points on API Gateway behavior with and without the **Team support** feature:

Without Team support	With Teams support
All users can view all details for all assets.	User can view only the assets that are assigned to the teams that they are a part of.
Users can add, modify, delete, activate and deactivate assets, publish and promote assets, export and import assets based on their functional privileges.	Privileges are assigned through teams. Users can perform actions based on their team's functional privileges.

Users can manage assets based on the functional privileges assigned to the teams they belong to. For details on asset visibility and accessibility, see [“Team Support Considerations” on page 819](#).

When *not* to use Team support?

You do not use the **Team support** feature when you require:

- **Tenant isolation.** If your requirement is to allow the access of assets by tenants, then you must have multiple tenants and isolate them from each other. The **Team support** feature does not address this requirement.
- **Full access management.** Users gain access based on their team privileges. There is no role-based access for users to an asset.

Teams management using API Gateway

You can create teams from the **User Management** section of the API Gateway UI by including the required user groups and assigning them the required functional privileges. You can also assign a Team administrator for each team, who can add or modify team members.

Users with the **Manage user administration** privilege can create teams. When creating a team, you can assign:

- **Team administrator.** You can assign a user or a user group as team administrator. Team administrators can add or remove users from a team. When you assign a user group as team administrator, all users of the groups can modify team members. When team administrators, who do not have the **Manage user administration** functional privilege log on to API Gateway, they can view only the teams assigned to them in the **Teams** tab of the **Administration** page.
- **Functional privileges for the team members.** The functional privileges assigned to a team determines the accessibility of assets to the respective team members. For example, if you assign all privileges under the APIs, Policies, and Applications sections, then the team members can manage APIs and applications assigned to their teams and perform operations related to policies.
- **Team members.** You can assign users and user groups to the team. Team members can access the assets assigned to their teams and perform operations on the assets based on the functional privileges assigned to the team.

Teams - Asset Association

After you have created teams, you can assign assets to teams in one of the following ways:

- **Assign team during asset creation.** When you create an asset, API Gateway provides an option to select the teams for the asset. You can select more than one team for an asset. You can modify the teams assigned by following the Change ownership process. For information about the process, see [“How do I Modify Teams Assigned to an API?” on page 815](#) and [“How do I Change the Ownership of Multiple Teams?” on page 817](#).
- **Using Global Team Assignment rule.** This is a preferred method to assign teams when you already have assets to which you want to assign teams to. You can create global assignment rules that are applied to assets and assign teams to them. You can specify one or more conditions in a rule. When an asset satisfies the conditions specified in a rule, the asset is assigned to the teams specified in the rule. When you create and activate a rule, the rule is applied to the existing assets and teams are assigned accordingly.

If you already have assets in your API Gateway instance and when you enable the **Teams support** feature, all assets are assigned to the **Default** team. Every user is automatically assigned to this team. That means, by default, every asset (APIs, applications, packages and plans) are still visible to all users. Access rights are restricted only if the asset is explicitly assigned to one or more teams during its creation.

You must manually remove the **Default** team from the respective asset details page.

Software AG recommends that you read the Team support considerations section to see the impact of Team support on other features. For more information, see [“Team Support Considerations” on page 819](#).


Enabling Team Support

When you enable the **Team support** feature, you can manage teams from the **Teams** tab under the **User management** section of API Gateway.

If you do not enable this feature, you can still create teams, as explained in [“Creating Teams” on page 806](#), and assign functional privileges to users. However, you cannot assign required assets to a set of users and restrict the access of assets to other users.

If you have enabled this feature, created teams, and assigned assets to teams, and then disable this feature, then the team assignments that you had performed earlier become invalid. That is, the assets are available to users based on their functional privileges and not based on the assigned teams.

> To enable teams

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > Extended settings**.
3. Click **Show and hide keys**.
All configurable parameters appear.
4. Select the **enableTeamWork** from the parameter list.
The **enableTeamWork** field appears in the **Extended Settings** section.
5. Type *true* in the **enableTeamWork** field. By default, the value of the setting is set as *false*.
The feature is enabled.

Creating Teams

This use case explains how to create teams by assigning the required functional privileges and users to them.

This use case begins when you have identified the list of users who must given access to an asset or a particular set of assets and end when you have created a team including the identified users.


In this example, a team with developers called *DevTeam* is created with the *Dev* user group as the team members, *User1* as the Team administrator, and all privileges under Manage API, Policies and Applications are assigned to the team.

Before you begin

Ensure that you have:

- API Gateway Manage user administration privilege.
- The user group, *Dev* is created. For information on how to create a user group, see [“Adding a Group” on page 240](#).

> To create teams

1. Expand the menu options icon , in the title bar, and select **User management**.
2. Click **Teams**.
3. Click **Add Team**.

The **Create Team** page appears.

4. Provide the name and description of the team in respective fields.
5. In the **Team Administrators** section, provide any or both of the following:
 - Login Id of the user who want to assign as a team administrator in the **Login ID** field.
 - Name of the API Gateway user group or the LDAP group that you want to assign as team administrator in the **Group name** field.

You can search users or user groups based on the characters provided in the above fields. Select the required user from the list displayed.

For the example consider in this use case, the team is named as *DevTeam* and the *User1* is specified as the team administrator.

Create Team

Create a team by providing the basic information, functional controls and add the required groups to the team. [?](#)

Team details

- Basic Information
- Functional privileges
- Groups

Name*

Description

Team Administrator

Users

✕

Users	Action
User1	✕

Groups (all users of the groups are team administrators)

[Continue to assign functional privileges >](#)

- Click **Continue to assign functional privileges >**.

The **Functional privileges** list appears.

- Select the functional privileges to be assigned to the team members. For information on the available functional privileges, see [“API Gateway Functional Privileges” on page 824](#).

For this use case, you need to assign all privileges required to manage the APIs and applications assigned to the team. So, all functional privileges under the **APIs, Policies, and Applications** section are selected.

Team details

- Basic Information
- Functional privileges
- Groups

Functional privileges

Select all

APIs, Policies, and Applications

<input checked="" type="checkbox"/> Manage APIs	<input checked="" type="checkbox"/> Activate / Deactivate APIs	<input checked="" type="checkbox"/> Manage applications
<input checked="" type="checkbox"/> Manage aliases	<input checked="" type="checkbox"/> Manage global policies	<input checked="" type="checkbox"/> Activate / Deactivate global policies
<input checked="" type="checkbox"/> Manage policy templates	<input checked="" type="checkbox"/> Manage threat protection configurations	<input checked="" type="checkbox"/> Publish API to service registry

API Portal Management

<input type="checkbox"/> Manage packages and plans	<input type="checkbox"/> Activate / Deactivate packages	<input type="checkbox"/> Publish to API Portal
--	---	--

Administration configurations

<input type="checkbox"/> View administration configurations	<input type="checkbox"/> Manage general administration configurations	<input type="checkbox"/> Manage security configurations
<input type="checkbox"/> Execute service result cache APIs	<input type="checkbox"/> Manage destination configurations	<input type="checkbox"/> Manage system settings
<input type="checkbox"/> Manage user administration	<input type="checkbox"/> Manage promotions	<input type="checkbox"/> Manage service registries
<input type="checkbox"/> Change ownership/teams	<input type="checkbox"/> Manage scope mapping	

Export or Import assets and Purge and Archive events

<input type="checkbox"/> Import assets	<input type="checkbox"/> Export assets	<input type="checkbox"/> Manage purge and restore runtime events
--	--	--

Microgateway register and de-register

Manage microgateways

- Click **Continue to assign groups >**.

The **Find groups** section appears.

- In the **Name** field, provide the name of the user group that you want to add as team members.
For this use case, the *Dev* user group that has all developers is selected.

The screenshot shows the 'Create Team' page in the API Gateway console. The 'Find groups' section is active, showing a search for 'de'. A table lists the results, with 'Dev' selected and highlighted by a red circle.

Name	Action
Dev	

- Click **Save**.

Team with specified details is created. As per the values provided in our use case, the *DevTeam* is created and appears in the list of teams.

The screenshot shows the 'User management' page in the API Gateway console. The 'Teams' tab is selected, and a table lists the teams. 'DevTeam' is highlighted with a red circle.

Name	Description	Action
Administrators	Groups associated to this team are allowed to perform all the administration related tasks.	
API-Gateway-Providers	Groups associated to this team are allowed to access an asset based on the functional privileges assigned to this team.	
Default	All users in APIGateway will be part of this team.	
DevTeam	All developers	

After team creation, you can assign assets to the team.

How do I Assign Teams during Asset Creation?

This use case explains how to assign teams for an API Gateway asset.

The use case starts when you have an asset that you want to allow access only to a set of users in your organization and ends when you have assigned team(s) to an asset.

In this example, let us see the steps to assign the asset, *DevAPI* that is being created to a team that has all developers as team members.

Before you begin

Ensure that you have:

- API Gateway Manage user administration privilege.
- The team support feature enabled. For information on enabling the feature, refer [“Enabling Team Support” on page 806](#).
- The team, *DevTeam* is created. For information on creating team, refer [“Creating Teams” on page 806](#).

➤ To assign teams during asset creation

1. Click **APIs** in the title navigation bar.

The **Manage APIs** page appears.

2. Click **Create API**. The **Create API** page appears. The example asset, *DevAPI* is created by importing a file.
3. Click **Import from an File**.
4. Click **Browse** to select the file using which you want to create the API.
5. Provide *DevAPI* in the **Name** field.
6. From the **Team** drop-down list, select the teams that you want to assign the asset to. For the use case considered, *DevTeam* is selected in the field.

Home > APIs > Create API

Create API

Create an API by importing from a file, URL or start from scratch

Lets Get Started!

Import API from file
Create an API by importing API from a specified file.

Select file*
APIGatewayAdministration.json Browse 🗑️

Name
DevAPI

Type
Swagger

Version
1.0

Create

Description
API meant for developers

Team

Dev	✕	▼
Team		Action
DevTeam		🗑️

The assigned teams appear in the **Team** field of **Basic Information** section in the **API details** page. In our example, the *DevAPI* is assigned to *DevTeam*.

The screenshot shows the 'DevAPI' page in the API Gateway console. The navigation bar includes 'WEBMETHODS API Gateway', 'APIs', 'Policies', 'Applications', 'Packages', and 'Microgate'. The breadcrumb trail is 'Home > APIs > DevAPI'. The main heading is 'DevAPI' with a subtitle: 'View API details, basic and technical information, resources and methods available, and API specifications. ?'. Below the heading are three tabs: 'API details', 'Scopes', and 'Policies'. A left sidebar contains a list of menu items: 'Basic information' (selected), 'Technical information', 'Resources and methods', 'API mocking', 'Components', and 'Documentation'. The main content area is titled 'Basic information' and displays the following details:

Name	DevAPI
Version	1.0
Owner	Administrator
Team	Administrators DevTeam
Active	No
Maturity state	Beta
Created	2019-09-13 13:51:15 GMT
Description	For developers

By default, all assets are assigned to the Administrators team in addition to the teams that you have assigned during asset creation.

How do I Assign Teams Using Team Assignment Rule?

This use case explains how to assign teams to API Gateway assets using Team assignment rules.

Team assignment rules are used to assign teams to existing assets and the ones being created. As stated earlier, you can create rule by specifying a set of required conditions. Assets will be validated against the given conditions and they will assigned to the configured teams. If you do not provide any conditions when for a rule, the rule will be assigned to all assets in API Gateway when you activate the rule.

This section explains the steps to configure conditions and team names for creating a rule. Also, it lists the steps to activate rule to apply the rule to assets.

In this example, consider a team of users who must be enabled to view all assets. To achieve this, you can create a team called *ViewAllAssets*, and create a rule by selecting all assets and no conditions. When no conditions are specified, the rule is applied to all assets. When you activate this rule, all assets are assigned to the *ViewAllAssets* team.


The use case starts when you have a team and ends when you create a team assignment rule and activate the rule. All assets are assigned to the specified team and the team members can view all assets.

Before you begin

Ensure that you have:

- API Gateway administrator privileges.
- The team support feature enabled. For information on enabling the feature, refer [“Enabling Team Support”](#) on page 806.
- Ensure that the team, *ViewAllAssets* is created. For information on creating team, refer [“Creating Teams”](#) on page 806.

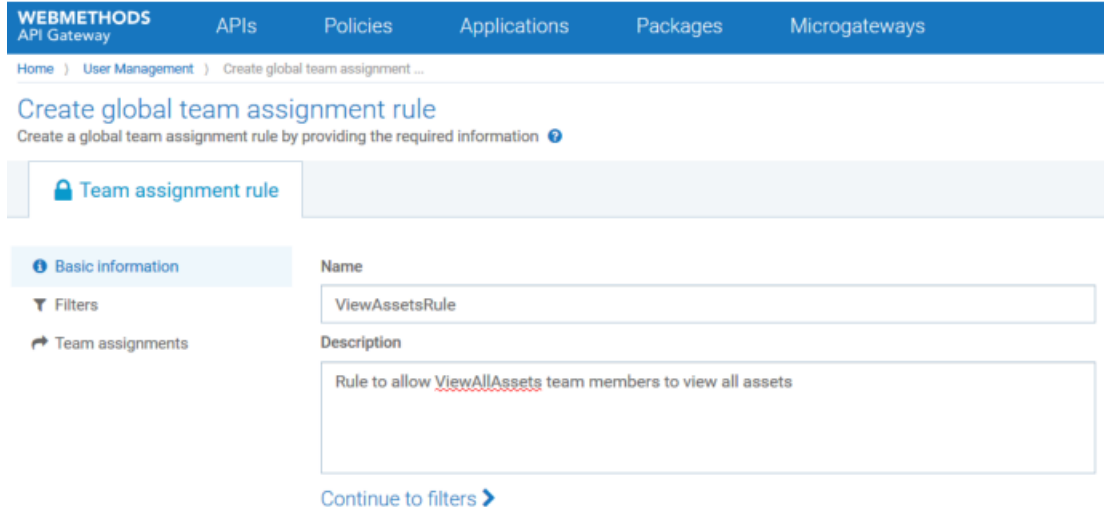
➤ To assign teams using team assignment rule

1. Expand the menu options icon , in the title bar, and select **User management**.
2. Click **Global team assignments**.
3. Click **Add global team assignment**.

The **Team assignment rule** page appears.

4. Provide the name and description of the rule in corresponding fields.

In this example, *ViewRule* is provided in the **Name** field.



The screenshot shows the 'Create global team assignment rule' page in the API Gateway console. The page has a blue header with the 'WEBMETHODS API Gateway' logo and navigation tabs for 'APIs', 'Policies', 'Applications', 'Packages', and 'Microgateways'. Below the header, there is a breadcrumb trail: 'Home > User Management > Create global team assignment ...'. The main heading is 'Create global team assignment rule' with a sub-heading 'Create a global team assignment rule by providing the required information'. A 'Team assignment rule' section is highlighted. On the left, there are tabs for 'Basic information', 'Filters', and 'Team assignments'. The 'Basic information' tab is active, showing a 'Name' field with the value 'ViewAssetsRule' and a 'Description' field with the text 'Rule to allow ViewAllAssets team members to view all assets'. A 'Continue to filters' button is located at the bottom of the form.

5. Click **Continue to filters** >.
6. Select any or all assets in the **Asset type** field to apply the rule to the selected asset types.
7. Select any one of the following from the **Logical operator** field:
 - **AND**. To apply the rule only if all conditions are satisfied by an asset.
 - **OR**. To apply the rule when any of the given condition is satisfied by an asset.

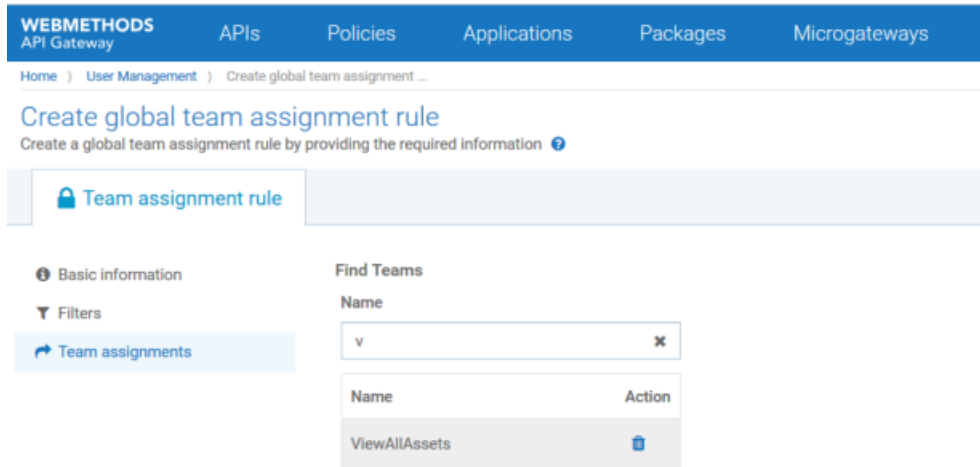
8. To specify a condition based on the asset attributes, provide the following information, and click **Add**:

Field	Description
Attribute	Name or the Description of the asset.
Operator	Operator that required to validate the attribute against the given value. The available options are: <ul style="list-style-type: none"> ■ Equals. Checks if the name or description of asset is equal to the given value. ■ Contains. Checks if asset contains the given value as a part of its name or description. ■ Start with. Checks if asset name or description starts with the given value. ■ Ends with. Checks if asset name or description ends with the given value.
Value	Value of the attribute.

For the use case considered in this section, the rule has to be applied to all assets irrespective of their attributes. So, all asset types are selected and no condition is specified.

9. Click **Continue to assign teams >**.
10. Provide the required team names in the **Name** field.

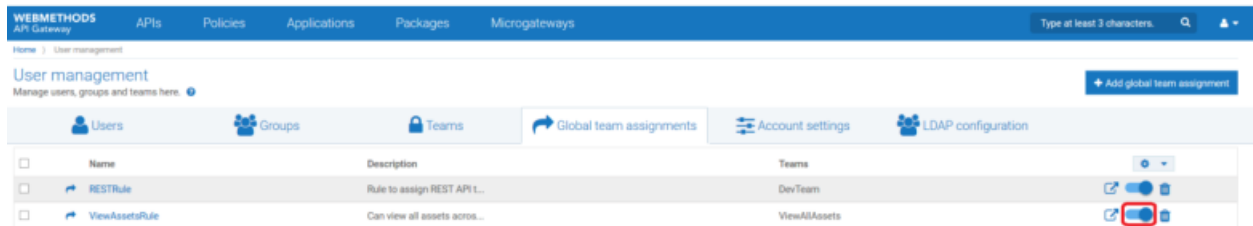
For our use case, the *ViewAll.Assets* team is selected.



11. Click **Save**.

The rule appears in the **Global team assignment** page.

12. Click the toggle button , adjacent to the rule.

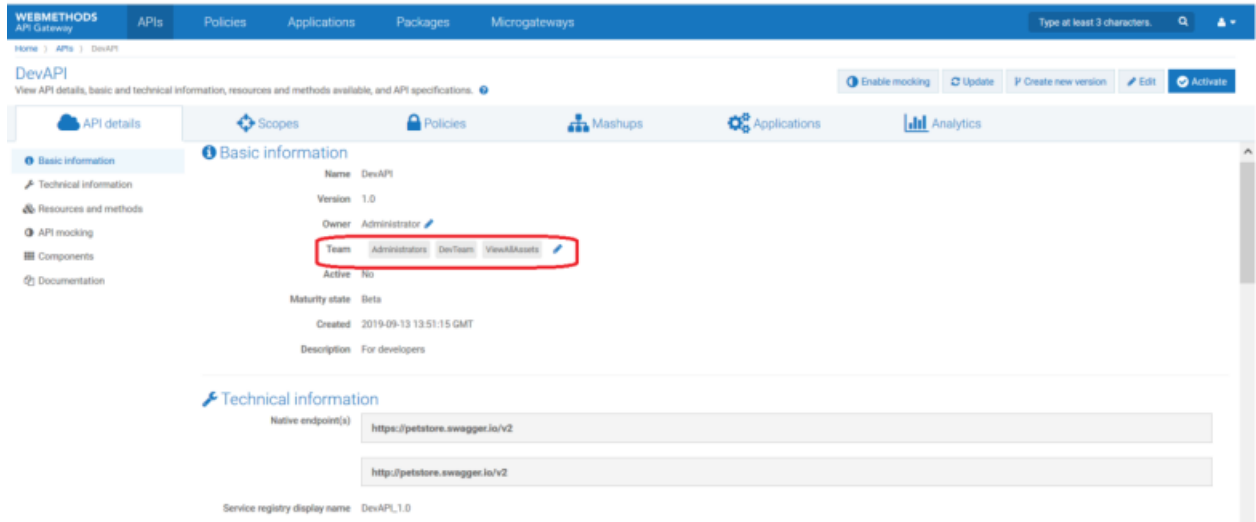


The rule is activated and applied across all existing assets. As per the rule you consider, all assets are assigned to the *ViewAllAssets* team.

13. Click **APIs** from the title bar and select *DevAPI*.

The **API details** page for the API appears.

Note that the API is assigned to the *ViewAllAssets* team as per the *ViewAssetsRule*.



How do I Modify Teams Assigned to an API?

This use case explains how to modify the list of teams associated with an API. You can configure an approval process for the modification teams assigned to an API, if required. You can only assign the API to the teams that you are part of. However, you cannot remove the Administrators team and the teams that are assigned to an API using global assignment rules.

The use case starts when you have an API that requires a modification in the list of teams assigned to it and ends when you successfully make the change.

In this example, an API *API* is assigned to the *Administrator* team (by default). The API has to be assigned to the *API-Gateway-Providers* team along with the existing team through an approval process.

Before you begin

- Ensure that you have the change owner/ team privilege.
- The change owner approval process configured and enabled if you want to enforce an approval process for ownership changes of assets. For details on configuring the approval process, see “[How Do I Configure the Approval Process for Ownership Change of Assets?](#)” on page 784.

➤ To modify the teams of an API

1. Log on to API Gateway as a user with the change ownership/ team privilege.
2. Click **APIs** on the title navigation bar.
3. Click **API**.

The API details page appears. The asset you have considered for example is not assigned to any team. So, the default team *Administrators* is displayed in the **Team** field of the Basic information section.

Home > APIs > API

API

View API details, basic and technical information, resources and methods available, and API specifications. ?

API details | Scopes | Policies

Basic information

- Basic information
- Technical information
- Resources and methods
- API mocking
- Components
- Documentation

Basic information

Name API

Version 1.0

Owner Administrator

Team Administrators

Active No

Maturity state Beta

Created 2019-08-25 03:32:57 GMT

Description New API

4. Click **change**.

5. Select the team that you want to assign and click . In our example, the API-Gateway-Providers from is selected.

API details | Scopes | Policies | Mashups

Basic information

- Basic information
- Technical information
- Resources and methods
- API mocking
- Components
- Documentation

Basic information

Name API

Version 1.0

Owner Administrator

Team

Type a keyword

- Administrators
- API-Gateway-Providers**

Active No

Maturity state Beta

Created 2019-08-25 03:32:57 GMT

Description New API

The change approval process is initiated.

Note:

If the approval flow is not configured, the *API-Gateway-Providers* team is added and a success message appears. Skip to step 8.

- An approval request is sent to the approver.
- The approver approves the request that resides in the Pending Requests section of the API Gateway UI.

Note:

The approver can click **Reject** to reject the request for ownership change if the request is invalid. A reject notification is sent to the requester and the team remains unchanged.

Click **Change ownership request details** to view the request details. The Request details dialog box appears.

The approval notification is sent to the requester.

- The *API-Gateway-Providers* team is added.

The screenshot shows the 'API Gateway' interface. The top navigation bar includes 'WEBMETHODS API Gateway', 'APIs', 'Policies', 'Applications', 'Packages', and 'Microgateway'. The breadcrumb trail is 'Home > APIs > API'. The main heading is 'API' with a sub-heading 'View API details, basic and technical information, resources and methods available, and API specifications.' Below this are three tabs: 'API details', 'Scopes', and 'Policies'. The 'API details' tab is active, showing a sidebar with options like 'Basic information', 'Technical information', 'Resources and methods', 'API mocking', 'Components', and 'Documentation'. The main content area is titled 'Basic information' and displays the following details:

- Name: API
- Version: 1.0
- Owner: Administrator
- Team: Administrators, API-Gateway-Providers (highlighted with a red box)
- Active: No
- Maturity state: Beta
- Created: 2019-08-25 03:32:57 GMT
- Description: New API

How do I Change the Ownership of Multiple Teams?

You can change the owners of multiple teams in a single step. This use case explains how to change the ownership of multiple teams by sending a REST request. You can configure an approval process, if required, for the change of ownership to take effect.

The use case starts when multiple teams require change of owner and ends when you successfully change the ownership of the teams.

> To change the ownership of multiple teams

- Use the following REST request to change the asset ownership to a new user.

```
POST http://host:port/rest/apigateway/assets/team
Content-Type: application/json
{
  "assetType": "*", (API/APPLICATION)
  "assetIds": ["*"],
  "currentTeams": "team1",
  "newTeams": ["team2"]
}
```

Provide the following information in the REST request:

- **assetType.** Specifies the asset type for which you want to assign new teams. Available values are `API`, `APPLICATION`, or the wildcard `*`. The wildcard `*` specifies all the assets, APIs and applications owned by the user specified in `currentTeams`.
- **assetIds.** Specifies the ID of the assets specified in `assetType`.

Note:

This is optional. `assetIds` is not required if you specify `currentOwner`.

- **currentTeams.** Specifies the current teams of the assets specified in the `assetType` field.

Note:

If both `currentTeams` and `assetIds` are specified, both are validated. For example, consider `user1` and `user2` are owners of `assetID1` and `assetID2` respectively. In the request payload, if you include `assetID1` and `assetID2` in the `assetIds` field and `user1` in the **currentTeams** field, then only `assetID1` ownership changes.

- **newTeams.** Specifies the teams to which you want to assign the specified assets.

Example: If all APIs assigned to the `DevTeam` must be assigned to two other teams, `Team2` and `Team3`, then send a REST request as follows:

```
POST http://localhost:5555/rest/apigateway/assets/team
Content-Type: application/json
{
  "assetType": "*", (API/APPLICATION)
  "currentTeams": "DevTeam",
  "newTeams": ["Team2", "Team3"]
}
```

This request assigns the all APIs of `DevTeam` to `Team2` and `Team3`.

The change approval process is initiated.

2. An approval request is sent to the approver.

The approval request contains information of all the assets whose ownership needs to change and the new owners' name.

3. The approver approves the request in the Pending Requests section of the API Gateway UI.

The approval notification is sent to the requester.

4. All APIs that are assigned to the `DevTeam` is assigned to `Team2` and `Team3`.

Team Support Considerations

This section lists the impact of the **Team support** feature on other API Gateway features.

As stated in previous sections, users can view only the assets that are assigned to their teams. Also, their accessibility depends on the functional privileges assigned to their team. Though this appears to be a simple behavior, there are some scenarios where it gets a little exceptional. Hence, as an administrator, read through the following scenarios carefully before you create team and assign functional privileges:

Visibility and Accessibility of Assets

You can use the **Teams support** feature to restrict the visibility and access of APIs, applications, packages, and plans. However, there are some scenarios where users from unassigned teams can view or access these assets. Also, note that the **Teams support** feature does not restrict the visibility and access of aliases, global policies, scopes, users, groups, and teams.

Scenarios that you must consider when using Team support

1. When an API assigned to a team is associated with applications that are assigned to some other teams, then the team members, to whom the API is assigned can view the applications and remove them if required.

For example, consider two teams *Team_A* and *Team_B*. The API *API_1* and *Application_1* are assigned to *Team_A* and the API *API_2* and *Application_2* are assigned to *Team_B*. If you associate *Application_1* and *Application_2* with *API_1*, then the *Team_A* members can view *Application_2* and remove the application from *API_1*, if required.

2. Users can view assets other than APIs, applications, packages, and plans based on their functional privileges. So, the **Team support** feature does not have an impact on assets like aliases, global policies, scopes, users, groups, and teams.

For example, consider two APIs, *API_1* and *API_2* assigned to two different teams; *API_1* assigned to *Team_A* and *API_2* assigned to *Team_B*. Consider a global policy applied to both APIs and an endpoint alias used in both APIs. In this case, users from both teams can view the global policy and the endpoint alias used by these APIs and there is no way to restrict this behavior.

The following table helps you to quickly recall the points discussed earlier in this topic:

Asset type	Visibility	Accessibility
APIs	Users can view the names, details, and versions of the APIs associated with their teams.	Users can access APIs based on functional privileges assigned to their teams.
	Users can view the names of applications that are associated with the APIs assigned to their teams.	Users can remove the applications that are visible (from the API).

Asset type	Visibility	Accessibility
Applications	Users can view the names, descriptions, versions of the applications associated with their teams.	Users can access applications based on functional privileges assigned to their teams.
	Users can view the names of APIs that are associated with the applications assigned to their teams.	Users can remove the APIs that are visible (from the application).
Packages and plans.	Users can view the names and details of the packages and plans assigned to their teams.	Users can access packages and plans based on functional privileges assigned to their teams.
Global search field	When users search for an asset using a keyword, the search returns only the assets that are assigned to your teams.	Users can access assets based on functional privileges assigned to their teams.
Aliases, Global Policies, Users, Groups, Teams, and Scopes	All users can view these assets, even if they have read-only access to API Gateway.	User can access the assets based on the privilege assigned to the individual users, irrespective of their teams.

Importing and Exporting Teams and Assets

Team members can import or export assets if they are assigned with the required functional privileges.

- When assets are exported, the respective team details are exported along with the assets; members of the teams are not exported.
- When assets are imported, the respective team is created if it is not present already; members of the teams cannot be imported.
- When team is exported, the users and groups that are part of team can be selected for export if required.
- When team is imported, the users and groups are imported along with the team.
- An export archive of an API will include the details of all associated applications (if this option was selected when exporting) - "visible" and "invisible"

An export archive of an Application will always include the details of all associated APIs - "visible" and "invisible"

Promotion management

Users with **Manage promotions** privilege can see all associated APIs/Applications (visible and invisible) on the Assets and dependencies page for the Applications/APIs selected on the Search page of the Promotion Management wizard

By switching between the Search page and the Assets and dependencies page using the Back and Next buttons, they can even drill down and recursively see all dependent assets (visible and invisible)

Members of a team can promote assets from one source stage to one or more target stages if they have the required functional privileges.

- When assets (all except teams) are promoted, they are promoted along with their team details; users are not promoted.
- When teams are promoted, the users and groups of teams can be promoted if required.

Scope mapping

Users with **Manage scope mapping** privilege can see all APIs (including APIs otherwise invisible to them) on the scope mapping configuration page. They can even add them to a scope mapping, but they cannot remove them again

Scope mappings are visible to all users

- API Gateway displays the names, descriptions, and versions for all associated APIs.
- The API entries are not linked with the respective API details pages

Note that there will be a scope mapping for every API if `pg_oauth2_createDefaultScopes` is set to true, so every API is listed in the list of scope mappings - visible to all users

Global policies for APIs

Global policies are visible to all users

- API Gateway displays names, descriptions, and versions for all associated APIs (by the policy's filter conditions)
- Only the entries for the visible APIs are linked with their API details pages

API mashup

Users can add APIs owned by you or assigned to your Team, but in an existing API Mashup, you can see the names, versions, selected resources and methods of all configured APIs - even those invisible to you.

API versioning

On the API details page (for a visible API version), the API Gateway displays all version numbers in the version drop-down list - for versions visible and invisible to the current user

When the user selects an invisible version, API Gateway displays an error message.

Scenarios where you cannot use Teams support

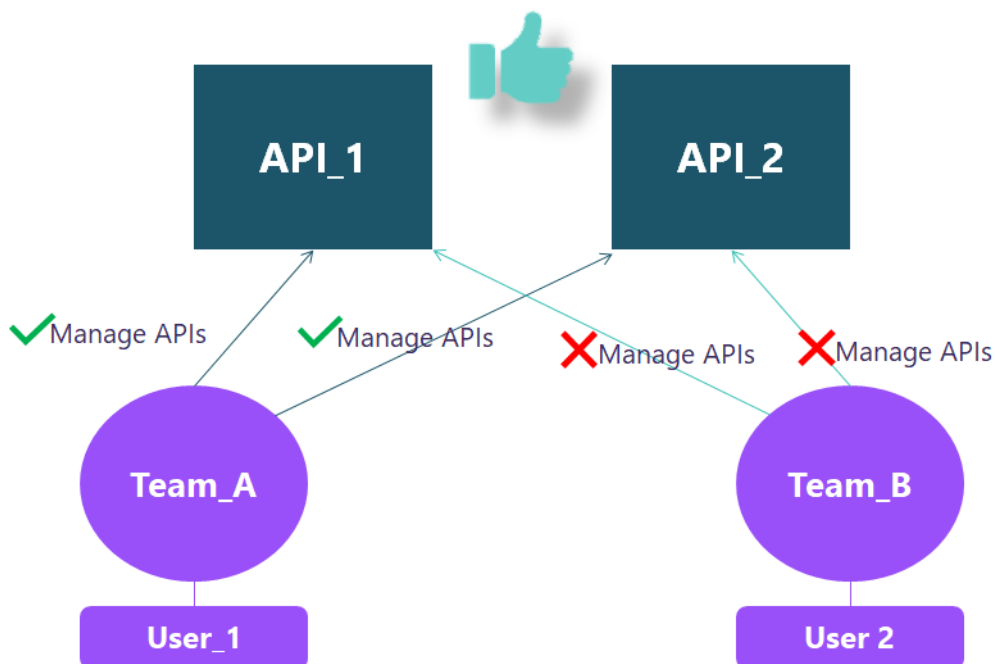
This section explains the use cases that cannot be achieved using the **Team support** feature.

Scenario 1

The functional privileges assigned to a team applies across all assets assigned to the team. You cannot assign different functional privileges to access different assets. That is, if you assign the **Manage APIs** functional privilege to a team, then the team members can perform all API-related transactions with the APIs assigned to the team. Similarly, if you assign the **Manage applications** privilege to a team, then all members of the team can manage the applications assigned to them.

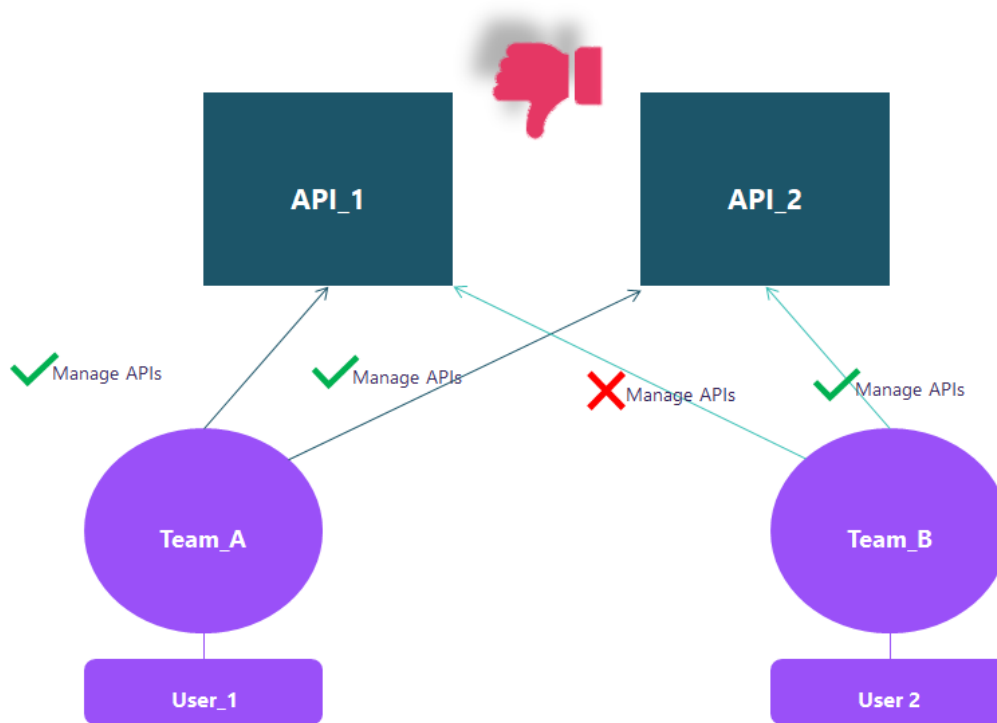
For example, if you assign the **Manage APIs** privilege to *Team_A* and assign two APIs *API_1* and *API_2*, then all users of the *Team_A* can manage both the APIs. If you do not assign the **Manage APIs** privilege to *Team_B* and assign two APIs *API_1* and *API_2*, then the *Team_B* members can only view the APIs. They cannot manage them.

This image describes the flow that you can achieve:



In the same example, you cannot allow *Team_B* members to view *API_1* and to manage *API_2*. You cannot assign different access level for different assets.

This image describes the flow that you cannot achieve:

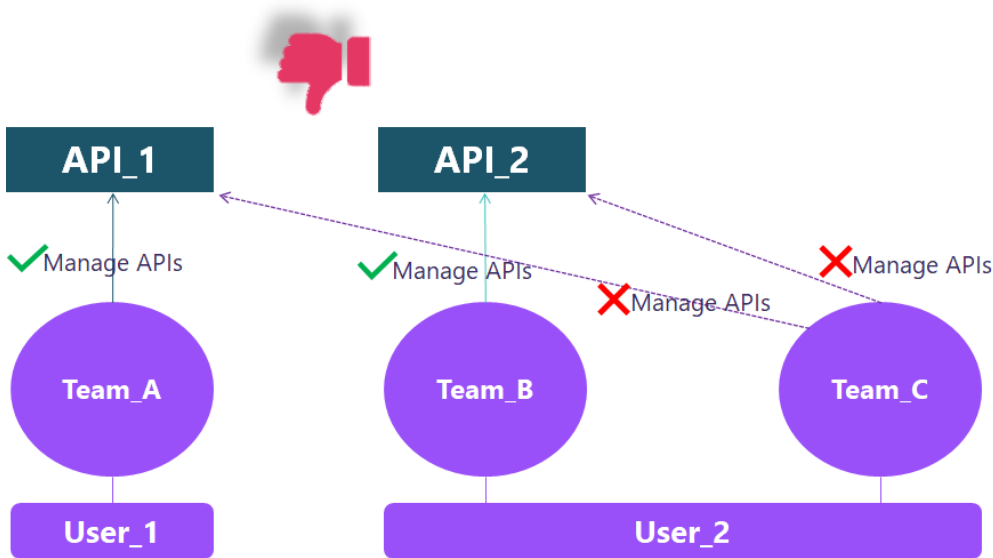


Scenario 2

The functional privileges assigned to a team applies across all users towards all assets assigned to the team. You cannot assign different privileges to different assets. That is, if a user has certain functional privilege through one of their teams, and when the user is assigned to another team that does not have the particular functional privilege, the user will still have the functional privilege assigned through the first team.

Consider *User_2* is a part of *Team_B* that has the **Manage APIs** privilege assigned with *API_2*. In this case, *User_2* can manage *API_2*. At the same time, if *User_2* is assigned to *Team_C* that does not have the **Manage APIs** privilege. If *API_2* is assigned to *Team_C*, then *User_2* can still manage *API_2*.

This image explains the flow that cannot be achieved:



API Gateway Functional Privileges

The following table lists the functional privileges and their description:

Functional Privilege	Description
Select all	To select all the listed functional controls.
Manage APIs	To create and manage APIs.
Activate/ Deactivate APIs	To activate, deactivate and manage APIs.
Manage applications	To create and manage applications and register applications with the APIs.
Manage aliases	To create and manage aliases.
Manage global policies	To apply a global policy to all APIs or the selected set of APIs.
Activate/Deactivate global policies	To activate, deactivate, and manage global policies.
Manage policy templates	To apply one or more policy templates to an API.
Manage threat protection configurations	To prevent malicious attacks on applications that typically involve large, recursive payloads, and SQL injections.
Publish API to service registry	To publish and unpublish APIs to service registry.

Functional Privilege	Description
Manage packages and plans	To create packages and plans, associate a plan with a package, and associate APIs with a package. In addition, you can view the list of packages, package details, APIs, and plans associated with the package.
Activate/ Deactivate packages	To activate, deactivate, and manage packages.
Publish to API Portal	To publish and unpublish assets to API Gateway.
View administration configurations	To view administration configurations.
Manage general administration configurations	To create and manage administration configurations.
Manage security configurations	To create and manage security configurations.
Execute service result cache APIs	To execute service result cache API.
Manage destination configurations	To publish events and performance metrics data to the configured destinations.
Manage system settings	To create and manage system settings.
Manage user administration	To create and manage users.
Manage promotions	To create stages and manage promotions.
Manage service registries	To create and manage service registries
Change ownership/ teams	To change ownership of an asset or teams.
Manage scope mapping	To manage OAuth and OpenID scopes.
Import assets	To import already exported APIs, application, policies, aliases, or other assets and configurations using the Import option in the user menu ().
Export assets	To export assets to your local system.
Manage purge and restore runtime event	To purge and restore events from the API Gateway store by setting the required date or duration in the API Gateway.
Manage microgateways	To manage the Microgateways connected to the API Gateway instance.

API First Implementation

APIs form the nerve center of software applications. So, it is very important for the providers to be clear about what they would provide and consumers to be clear about what they want to consume. Better understanding of APIs guarantee an excellent output. API First is all about the establishment of a common agreement between the providers and consumers. Thus, this design helps both the parties to be on the same page.



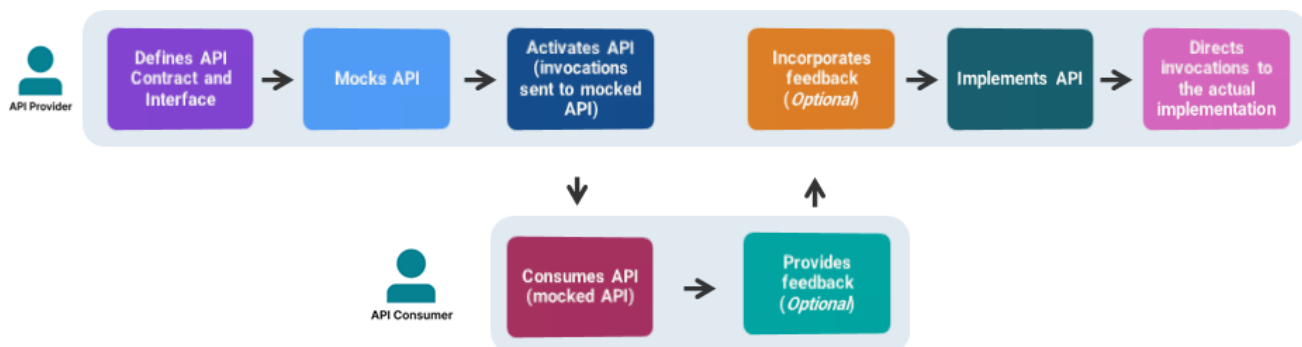
When adapting API First approach, API developers start the API development with the API contract and work on the implementation part at a later stage. This approach of prioritizing the API design over its implementation is beneficial to both, providers and consumers.

In conventional scenarios, providers expose APIs to their consumers only after the API is implemented. Consumers test the API and let the providers know their feedback about the API. Providers must then revisit the API to incorporate the feedback received from their consumers. You can optimize this process by adapting API First design.

When following API First approach, consumer does not have to wait for the provider to implement the API, they can proceed with their application development using the exposed API. The implementation status of API does not have an impact on consumers as they receive the designated responses for their requests through the mocked API. So, the API development and the application development can take place at the same time.

Once the provider implements the API, the end-point is updated to divert the invocations to the actual implementation instead of mocked response. The provider can then disable mocking.

The following diagram explains the flow of API development as per the API First design:



As per the API First design, providers expose their API to consumers when the development is underway.

API First Design using API Gateway

Starting API Gateway 10.5, the application provides seamless support for API First approach for your APIs.

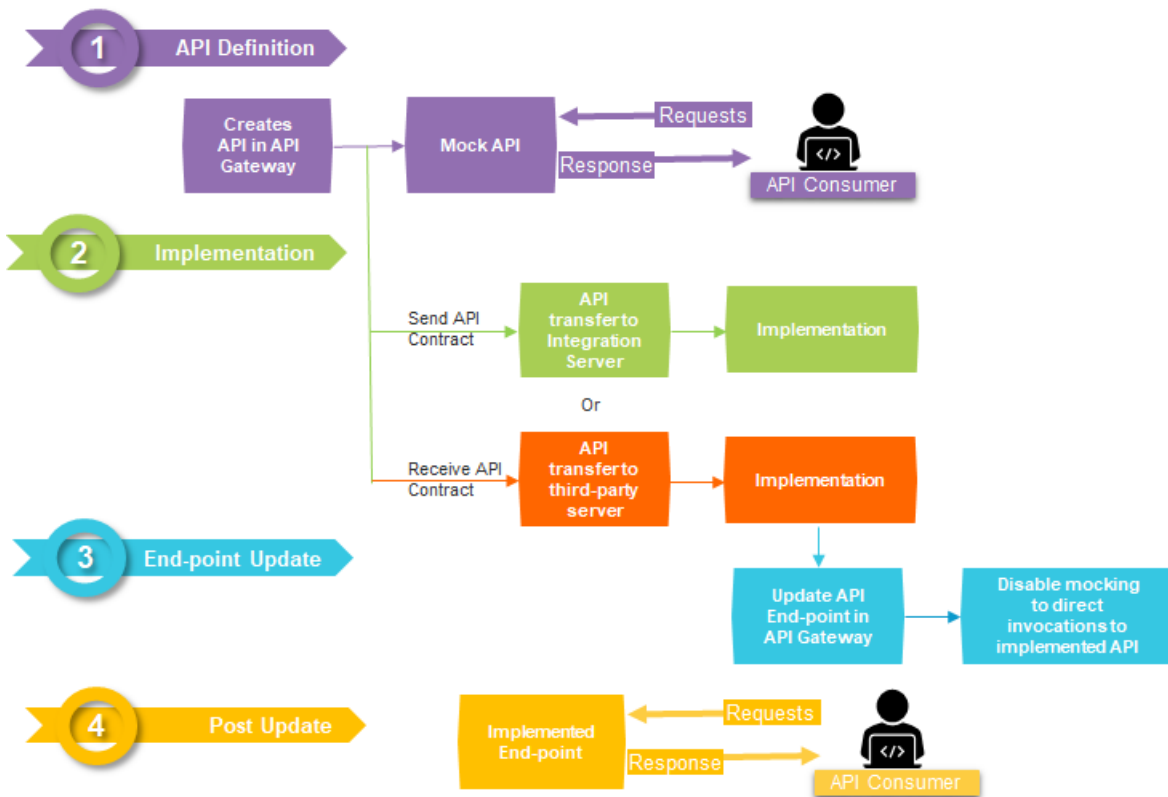
Using API Gateway, you can define API contract for the APIs and download provider specification for the APIs that you create. As a provider, you would not want to expose all resources and methods of an API to consumers. The API Contract given to the consumer has only the part of API exposed to the consumer whereas the provider specification comprises of the complete specification. This is useful for providers to implement the API.

You can enable mocking and activate the API for consumption. The mocked version of API returns respective responses for the consumer requests. This ensures the required end-user experience to the consumers.

When the API is ready to be implemented, you can implement your APIs in Integration Server or any other implementation server. If you are using Integration Server, you can add the required Integration Server instance in API Gateway. You can add multiple Integration Server instances and publish your API to the required instance. If you are using Integration Server, you can send the API contract from API Gateway. Else, the API contract has to be retrieved from API Gateway.

After implementation, you can update the actual implementation end-point to API Gateway. This step is mandatory to disable API mocking and divert the invocations to the actual end-point.

The following workflow shows the high-level workflow of API First implementation approach using API Gateway:



API First Implementation using Integration Server

This use case explains the steps involved in adapting API First from Integration Server. When an API created in API Gateway is implemented in Integration Server, then the API Contract is sent from API Gateway to Integration Server.

The use case starts when you create an API in API Gateway and ends when you communicate the API implementation endpoint to API Gateway.

In this example, the *APIFirst* API is created in API Gateway and implemented in the Integration Server instance, *IS1* that is configured in API Gateway.

Before you begin

- Ensure that you have the Manage API privilege.
- Configure the required Integration Server instances in API Gateway for implementing your APIs.

➤ To adapt API First design using Integration Server

1. Log on to API Gateway.
2. Click **APIs** in the title navigation bar.

A list of all existing APIs appears.

3. Click **Create API** to create an API with required API documentation.

The screenshot shows the 'API1' details page in the API Gateway console. The page is divided into several sections:

- Basic information:**
 - Name: API1
 - Version: 10.5
 - Owner: Administrator
 - Active: Yes
 - Maturity state: Beta
 - Created: 2019-08-28 16:53:04 GMT
 - Description: API Gateway Administration Service provides interface for you to administer various functions of the API Gateway. The user needs to have different fun functions. For example, in order to manage runtime transactions data of the API Gateway, the user needs to have Manage purge and restore runtime ev part of API-Gateway-Administrators group will have all privileges.
 - Following Administration functions are expo...[More](#)
- Technical information:**
 - Native endpoint(s): `http://localhost:5555/rest/apigateway`
 - Gateway endpoint(s): `http://SAG-92DYMH2:5555/gateway/API1/10.5`
 - Service registry display name: `API1_10.5`

4. Click **Policies** and define required policies for the API.
5. Click **Enable Mocking** to mock and generate API mock responses.

This step enables the API to send responses to the requests received from consumers.

6. From the APIs page, click **Publish** for the *APIFirst* API.

The **Publish API** dialog box appears.

7. Select **Integration Servers**.

The list of configured Integration Servers instances appears.

8. Select the *IS1* instance from the list.
9. In the **Package Name** and **Folder Name** fields, provide the package name and folder name of the IS instance in which the API must be implemented.

The API along with the API contract is published to Integration Server.

10. After implementing the API in Integration Server, invoke the REST end-point to communicate API implemented endpoint to API Gateway:

```
PUT http://<API Gateway host>:<port>/rest/apigateway/apis/{apiId}/implementation
{
  "maturityState": "string",
  "nativeBaseURLs": [
    "string"
  ]
}
```

You can provide required values for the parameters in the above command. For information on parameters, see [“List of Parameters used in API Implementation” on page 834](#).

Example:

```
PUT http://10.2.151.149:5555/rest/apigateway/apis/
94dfd243-dd54-4d7e-8ba5-396ffaf6fe4e/implementation
{
  "nativeBaseURLs": ["https://10.2.35.125:5556/ws/srvs:Calculator/
CalculatorHttpSoap11Endpoint",
"http://10.2.151.149:5555/ws/srvs:Calculator/CalculatorHttpSoap11Endpoint"],
  "maturityStatus" : "Implemented"
}
```

For details on the REST API, see the swagger file `APIGatewayServiceManagement.json`, located at `Install directory/IntegrationServer/instances/default/packages/`

WmAPIGateway/resources/apigatewayservices/APIGatewayServiceManagement.json. For more information on Service Management, see [“Service Management” on page 765](#).

As a result of the REST call, the mocking of the API is disabled and the consumers requests are directed to the actual implementation.

API First Implementation using a Third-party Server

This use case explains the steps involved in adapting API First approach using a third-party implementation server.

The use case starts when you create an API in API Gateway and ends when you communicate the API implementation endpoint to API Gateway.

Before you begin

- Ensure that you have the Manage API privilege in API Gateway.
- Configure a third-party implementation server for implementing your APIs.

➤ To adapt API First design using a third-party implementation server

1. Log on to API Gateway.
2. Click **APIs** in the title navigation bar.
A list of all existing APIs appears.
3. Click **Create API** to create an API with required API documentation.

The screenshot shows the API Gateway console interface. At the top, there are navigation tabs for 'WEBMETHODS API Gateway', 'APIs', 'Policies', 'Applications', 'Packages', and 'Microgateways'. Below the navigation, the breadcrumb path is 'Home > APIs > API1'. The main heading is 'API1' with a sub-heading 'View API details, basic and technical information, resources and methods available, and API specifications.' There are several tabs: 'API details', 'Scopes', 'Policies', 'Mashups', 'Applications', and 'Analytics'. The 'API details' tab is active, showing a left-hand menu with options like 'Basic information', 'Technical information', 'Resources and methods', 'API mocking', 'Components', and 'Documentation'. The main content area is divided into two sections: 'Basic information' and 'Technical information'. The 'Basic information' section displays a table of key-value pairs for API1, including Name, Version, Owner, Active status, Maturity state, and Created date. A description follows, explaining the API Gateway Administration Service. The 'Technical information' section shows endpoint details, including Native endpoint(s), Gateway endpoint(s), and Service registry display name.

4. Click **Policies** and define required policies for the API.
5. Click **Enable Mocking** to mock and generate API mock responses.
6. Using an external REST client such as Postman or SoapUI, run the below command to search for the API in API Gateway for implementation:

```
POST http://<API Gateway host>:<port>/rest/apigateway/search
{
  "types" : ["api"],
  "scope" : [
    {
      "attributeName" : "maturityState",
      "keyword" : "ToBeImplemented"
    }
  ]
}
```

The maturityState parameter in the above command is used search for APIs based on their maturity state. In this use case, you must search for APIs that are to be implemented. Hence, you can provide the *ToBeImplemented* value for the parameter. This command returns the list of APIs that are yet to be implemented.

7. Using the API Id of the API that you want to implement, run the following command to retrieve the API contract from API Gateway:

```
GET http://<host>:<port>/rest/apigateway/apis/{apiId}/
providerspecification?format=swagger
```


The value for the format parameter can be *swagger*, *raml*, or *openapi* for REST APIs; and *wSDL* for SOAP APIs.

Note:

You can search for an API based on its maturity status in API Gateway using the following command:

```
POST http://<API Gateway host>:<port>/rest/apigateway/search
{
  "types" : ["api"],
  "scope" : [
    {
      "attributeName" : "maturityState",
      "keyword" : "ToBeImplemented"
    }
  ]
}
```

8. Implement the API in the required implementation server.
9. After implementation, invoke the REST end-point to communicate API implemented endpoint to API Gateway:

```
PUT http://<API Gateway host>:<port>/rest/apigateway/apis/{apiId}/implementation
{
  "maturityState": "string",
  "nativeBaseURLs": [
    "string"
  ]
}
```

You can provide required values for the parameters in the above command. For information on parameters, see [“List of Parameters used in API Implementation” on page 834](#).

Example:

```
PUT http://10.2.151.149:5555/rest/apigateway/apis/
94dfd243-dd54-4d7e-8ba5-396ffaf6fe4e/implementation
{
  "nativeBaseURLs": ["https://10.2.35.125:5556/ws/srvs:Calculator/
CalculatorHttpSoap11Endpoint",
"http://10.2.151.149:5555/ws/srvs:Calculator/CalculatorHttpSoap11Endpoint"],
  "maturityStatus" : "Implemented"
}
```

For details on the REST API, see the swagger file `APIGatewayServiceManagement.json`, located at `Install directory/IntegrationServer/instances/default/packages/WmAPIGateway/resources/apigatewayservices/APIGatewayServiceManagement.json`. For more information on Service Management, see [“Service Management” on page 765](#).

As an outcome of the REST call, the mocking of the API is disabled and API Gateway starts requests for the actual implementation.

List of Parameters used in API Implementation

The following are some of the parameters used during API implementation:

Parameter	Purpose
nativeBaseURLs	<p>Endpoint URLs of the native service. This parameter is mandatory to route the requests to this implemented API. The existing endpoint values of the routing policies of the API are replaced with the URLs given against this parameter.</p> <p>You can provide multiple HTTP and HTTPS URLs for this parameter. The URLs that you provide for this parameter appears under the Native endpoint(s) section in the Technical information page of API Gateway. The first URL among the list of URLs is used in the routing policies by this update call. If you want to use any other URL in the routing policies, you can update the API policies accordingly.</p>
maturityState	<p>Indicates the maturity state of APIs. You can use this parameter to search for an API based on its maturity state and retrieve the API for implementation. Also, you can use this to update the maturity state of an API after implementation.</p> <p>Typically, the value of this parameter would be the consecutive state defined in the apiMaturityStatePossibleValues extended setting configuration.</p> <p>For example,</p> <p>If any of the following states are configured in the apiMaturityStatePossibleValues setting : Design, Implementation, Testing, Production; and current state of an API is <i>Implementation</i>, then you must specify <i>Testing</i> as the parameter value because that would be next stage as per the configuration.</p>

Gateway Endpoints

Gateway endpoint is the URL that is used to access an API through API Gateway. By default, API Gateway provides a default gateway endpoint for all active APIs. The default gateway endpoint is in the `protocol://host:port/{defaultPrefix}/{apiName}/{apiVersion}/{resourcePath}` format.

When there is a need to access the API using a different endpoint, you can define a custom gateway endpoint. In custom gateway endpoints, you can customize the portion of the URL between *port* and *resourcePath*.

Custom gateway endpoints can be specific to an API or a global template can be defined through global gateway endpoint.

How do I Define API-specific Gateway Endpoints?

This use case explains how to define custom gateway endpoints specific to an API. You can define more than one custom gateway endpoint to an API. Custom gateway endpoints can be added for all types of APIs such as REST, SOAP, OData, and WebSocket.

The use case starts when you want to define API specific gateway endpoint and ends when you have created the API specific gateway endpoint.

Here are some points that you need to consider, when you define API specific gateway endpoint:

- Custom gateway endpoints cannot be created for the APIs that have blank space or special characters in API name or API version.
- Gateway endpoint is case-sensitive.
- Gateway endpoint cannot start with pre-defined prefixes such as *rest* or *invoke* .
- URL path of one custom gateway endpoint cannot start with the URL path of the another custom gateway endpoint or default gateway endpoint. For example, if any of the API has a custom endpoint with URL path *abc/custom*, you cannot have another custom gateway endpoint with URL path *abc/customendpoint*. Similarly, if any of the API has a default gateway endpoint *gateway/myAPI/v1*, you cannot have custom endpoint with URL path *gateway/myAPI*. However, it is possible to have two valid custom gateway endpoints with URL paths *abc/custom1* and *abc/custom2*, because here one of the URL path is not the extension of another URL path.
- In order to use the gateway endpoints feature, the *watt.server.url.alias.partialMatching* property needs to be *true* . By default this property is set to *true* .
- When you create a custom gateway endpoint, API Gateway internally creates the URL aliases. Those internal URL aliases are hidden from the API Gateway users, it will be displayed only in the Integration Server. Software AG recommends that you do not modify any URL alias through Integration Server.
- A gateway endpoint can use following variables, which are resolved dynamically:
 - `${defaultPrefix}` - resolves based on API type. For REST and OData the defaultPrefix is *gateway*, SOAP the defaultPrefix is *ws*, and Websockets the defaultPrefix is *websocket*.
 - `${apiName}` - replaces with the API name value.
 - `${apiVersion}` - replaces with the API version value.

For example, when a gateway endpoint uses `${apiName}` variable, and if you change the API name, it automatically gets reflected in the gateway endpoint.

Note:

If you want to use a gateway endpoint across all versions of an API, Software AG recommends you to use the `${apiVersion}` variable so that the gateway endpoint becomes unique across different versions.

Important:

At any given point, API Gateway does not allow you to provide the same gateway endpoint for different APIs nor different versions of same API. Hence, make sure that you provide a unique gateway endpoint, so that it does not match with any of the existing APIs' default or custom gateway endpoints.

Before you begin

Ensure that you have:

- *Manage APIs* functional privilege.
- Activated the API.

➤ To define API-specific gateway endpoints

1. Click **APIs** in the title navigation bar.

A list of all registered APIs appears.

Note:

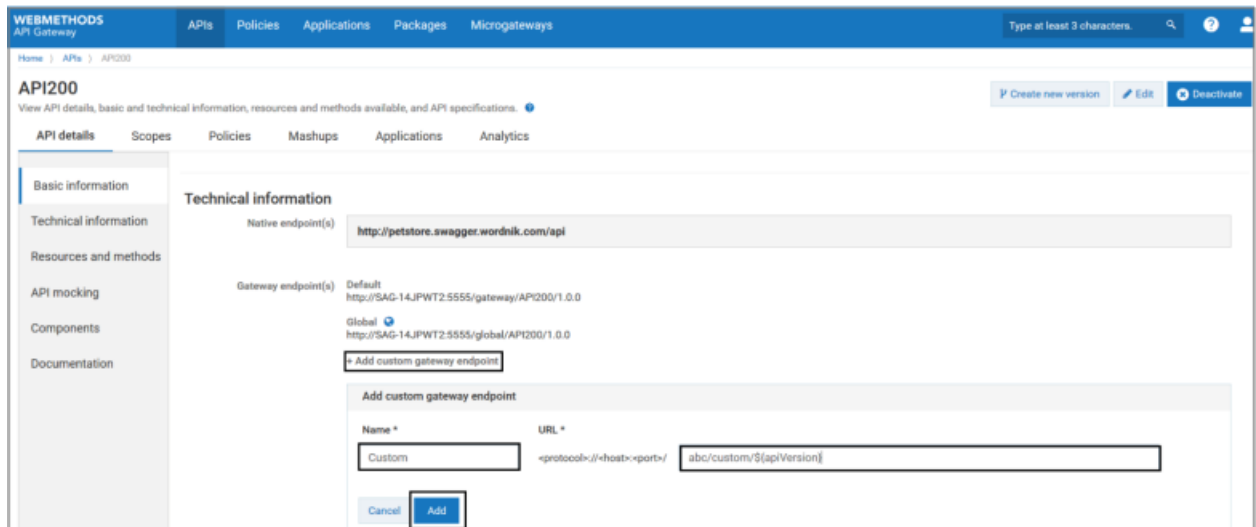
You can manage the gateway endpoints of an API, directly from the view mode of the API details screen.

2. Click the corresponding API for, which you want to customize the gateway endpoint.

The API details page appears.

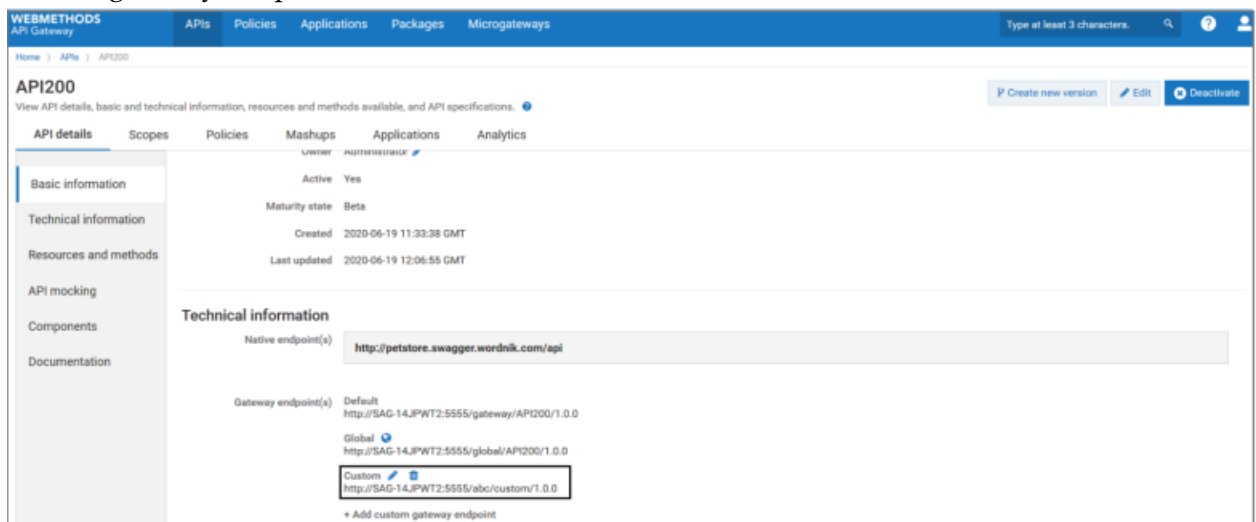
3. Click **Technical information**.
4. Click **+Add custom gateway endpoint** and provide the following information.

Field	Description
Name	Specifies the name for the custom gateway endpoint. A gateway endpoint name must be unique within an API.
URL	Specifies the custom gateway endpoint. The gateway endpoint URL cannot include a space, nor can it include the following special characters: # % ? ' " < \





5. Click **Save**.

The added custom gateway endpoint appears in the **Gateway endpoint(s)** field of the API details page. In addition to the default gateway endpoint, you can access the API using this custom gateway endpoint.



Note:

You can edit or delete the gateway endpoint from API details page either by clicking the  or  icon corresponding to the gateway endpoint that you want to edit or delete.

How do I Define Global Gateway Endpoint?

This use case explains how to define global gateway endpoint. The global gateway endpoint creates gateway endpoint template for all APIs. Each API will inherit this global endpoint in addition to the default and custom endpoints of an API.

The use case starts when you want to define global gateway endpoint and ends when you have created the global gateway endpoint.


In order to generate a unique gateway endpoint for each API version, the global gateway endpoint template must use the following variables :

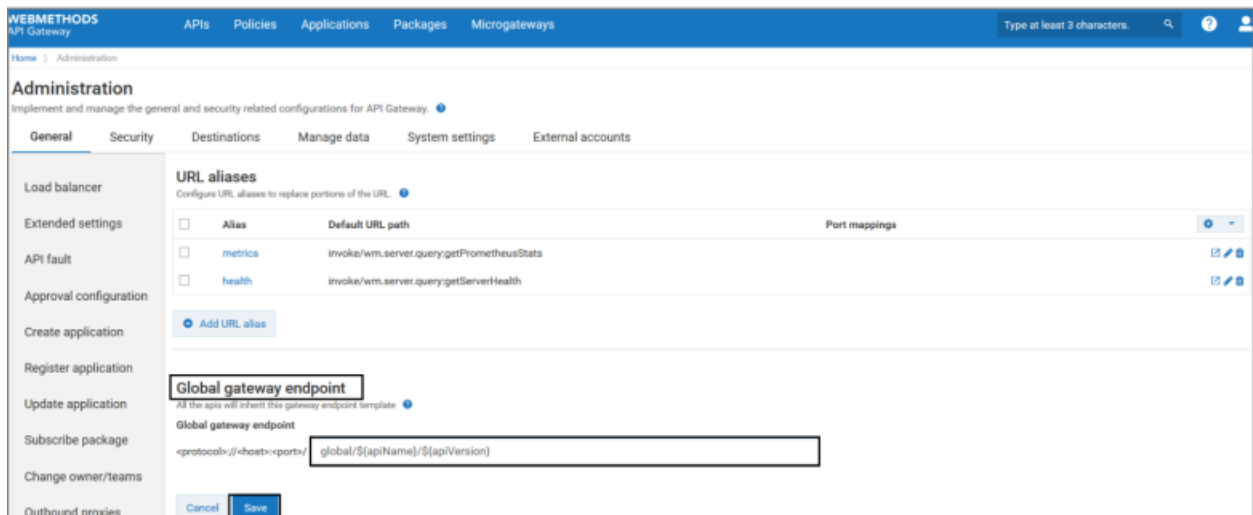
- `${apiName}`
- `${apiVersion}`

Before you begin

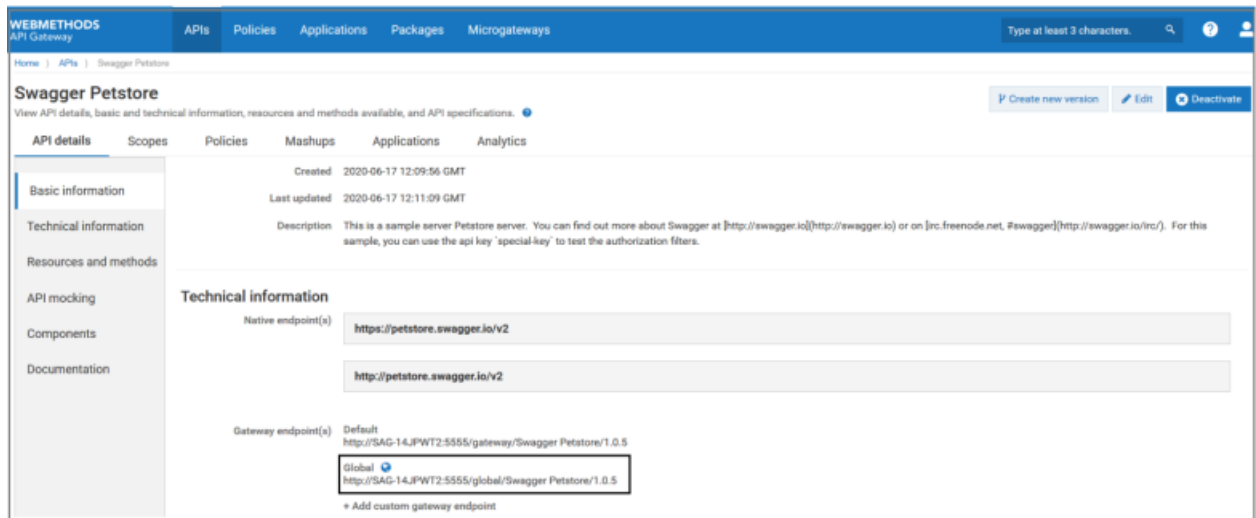
Ensure that you have *Manage APIs* functional privilege.

➤ To define global gateway endpoint

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > URL aliases**.
3. In the **Global gateway endpoint** section, provide the global gateway endpoint that you want to define across the APIs.
4. Click **Save**.



The added global gateway endpoint appears in the **Gateway endpoint(s)** field of the API details page of all APIs. In addition to the default and API-specific gateway endpoints, you can access your APIs using this global gateway endpoint.




How do I Edit Global Gateway Endpoint?

This use case explains you how to edit the global gateway endpoint. You can edit the global gateway endpoint, when you want to change or update the existing global gateway endpoint template for all the APIs.

The use case starts when you want to edit global gateway endpoint and ends when you have updated the global gateway endpoint.

> To edit global gateway endpoint

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > URL aliases**.
3. In the **Global gateway endpoint** section, update the value specified in the **Global gateway endpoint** field.
4. Click **Save**.

The updated global gateway endpoint appears in the **Gateway endpoint(s)** field of the API details page. All the APIs can be accessed using the updated global gateway endpoint.

Note:


You cannot access the APIs using the older global gateway endpoint.

How do I Delete Global Gateway Endpoint?

This use case explains you how to delete the global gateway endpoint. You can delete the global gateway endpoint, when you do not want to access any of your APIs using the existing global gateway endpoint template.

The use case starts when you want to delete global gateway endpoint and ends when you have deleted the global gateway endpoint.

➤ To delete global gateway endpoint

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **General > URL aliases**.
3. In the **Global gateway endpoint** section, delete the value specified in the **Global gateway endpoint** field.
4. Click **Save**.

The global gateway endpoint is removed from the **Gateway endpoint(s)** field of the API details page and you cannot access any of your APIs using global gateway endpoint.

Other Gateway Endpoint Usecases

Publishing APIs to API Portal

Just like publishing the default gateway endpoints, you can also publish the custom gateway endpoints to the API Portal. Published custom gateway endpoints can be accessed through the API Portal interface.

Supporting Custom Prefix in CentraSite deployed APIs

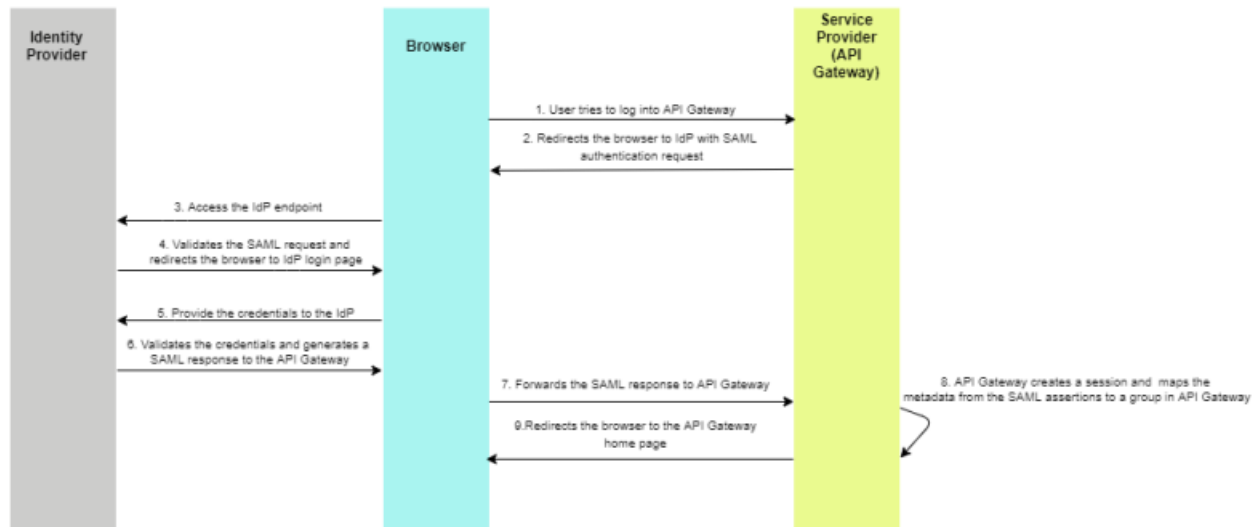
When you virtualize a service in CentraSite, you can replace the default prefix of an invocation alias with custom prefix. When you publish such services to API Gateway, the custom prefix that was specified in CentraSite will be supported in API Gateway by automatically adding the custom gateway endpoint to the respective API.

SAML SSO

Single sign-on (SSO) is a user authentication service that permits a user to use one set of login credentials to access multiple applications and service providers. In addition to the convenient factor, implementing SSO makes user logins more secure as it uses SAML protocol for communication.

Security Assertion Markup Language (SAML) is an open standard that allows identity providers to pass authorization credentials to service providers. SAML uses Extensible Markup Language (XML) for standardized communication between the identity providers and service providers. SAML provides a solution to allow your identity provider and service providers to exist separately from each other, which centralizes user management and provides access to services. In this case, API Gateway is the service provider.

- **Identity Provider (IdP)** - Performs authentication and passes the users's identity to the service provider for authorization.
- **Service Provider** - Trusts the identity provider and authorizes the given user to access the requested resource.



Limitation

When you log into API Gateway using SSO, both the IdP and API Gateway sessions are created. But when you log out from API Gateway, only the API Gateway session gets terminated, the IdP session gets terminated based on its session timeout configuration. API Gateway does not support Single Logout (SLO).

How to enable SAML SSO in API Gateway?

This use case explains the steps involved in enabling SSO for API Gateway using SAML protocol.


The use case starts when you configure the SAML settings for SSO in API Gateway and ends when you log into API Gateway using SSO.

Before you begin

Ensure that you have:

- *API Gateway Manage User* administration privilege.
- Downloaded and stored the IdP metadata file locally.

➤ To configure SAML settings for single sign-on

1. Expand the menu options icon , in the title bar, and select **Administration**.
2. Select **System Settings > SAML SSO**.

Administration
Implement and manage the general and security related configurations for API Gateway.

General Security Destinations Manage data **System settings** External accounts

Configuration
SAML SSO Export

SAML SSO
Configure the single sign-on settings

Enable SAML authentication

SAML redirect URL

Send signed SAML auth request
 Require signed assertion from IDP

Service provider identity

IDP metadata location

Gateway metadata location

Keystore properties

Keystore location

Keystore type
 JKS PKSC12

Keystore password

Default key properties

Default key alias

Default key alias password

Sign key properties

Sign key alias

Sign key alias password

Encrypt key properties

Encrypt key alias

Encrypt key alias password

Cancel **Save**

3. Set the toggle button to the on position to enable SAML.
4. Enter the SSO URL to which you want to redirect the browser in the **SAML redirect URL** field. For example, `/saml/sso/login`.
5. Select **Send signed SAML auth request**, if you want to send out the signed SAML authorization request to the Identity Provider (IdP).
6. Select **Require signed assertion from IDP** to receive a signed assertion from IdP.
7. Provide the service provider's unique URL, in the **Service provider identity** field, which is used to identify the service provider.

For example: `https://hostname:9073/apigatewayui/saml/SSO`. Make sure you specify the HTTPS port in the URL. When you configure the service provider in IdP, make sure you use the same URL.

8. Provide the location where you have saved the IdP metadata file in the **IDP metadata location** field.

For example, `C:\sso\federationmetadata.xml`.

9. Provide the location where you want to download the API Gateway metadata file in the **Gateway metadata location** field.

For example, `C:\sso\gatewaymetadata.xml`.

10. Provide the following information in the **Keystore properties** section:

- **Keystore location:** Location where the keystore file is stored.

This is used for the communication between API Gateway and IdP. By default, API Gateway is shipped with `saml_sso.jks` file. You can view this file in the `<SAGInstalDir>\profiles\IS_default\ apigateway\config\keystore\saml_sso.jks` location. In this example, the default `saml_sso.jks` file is used.

- **Keystore type:** Select the file type of the keystore file. The file type can be either **JKS** or **PKCS12**.
- **Keystore password:** The password to access the keystore file. As the default `saml_sso.jks` file is used in this example, the password to open the default `saml_sso.jks` file is `apigwstore`.

11. Provide the following information in the **Default key properties** section:

- **Default key alias:** Choose the default key alias, when you want to use the same default key alias for signing and encrypting the requests .
- **Default key alias password:** The password to access the default key alias. As the default `saml_sso.jks` file is used in this example, the default password for default key is `defaultapigw`.

12. Provide the following information in the **Sign key properties** section:

- **Sign key alias:** The alias of the default key used to digitally sign requests sent to the service provider .
- **Sign key alias password:** The password to access the sign key alias. As the default `saml_sso.jks` file is used in this example, the default password for sign key is `signapigw`.

13. Provide the following information in the **Encrypt key properties** section:

- **Encrypt key alias:** The alias of the default key used to encrypt the request that is sent to the service provider.
- **Encrypt key alias password:** The password to access the encrypt key alias. As the default `saml_sso.jks` file is used in this example, the default password for sign key is `encryptapigw`.

14. Click **Save**.

The SSO settings is configured with the entered configuration.

15. Restart the API Gateway.

16. Download the API Gateway metadata file from

`http(s)://hostname:9073/gatewayui/saml/sso/metadata` and save it in the location as specified in the step 9.

17. Restart the API Gateway.

The API Gateway Login page appears with the **Login with SSO** link.

SAML Assertion

A SAML assertion is the XML document that the IdP sends to the service provider (That is API Gateway). It informs the API Gateway that a user has logged in. It also provides the necessary information for the API Gateway to confirm the user's identity and lists the groups to which the logged user belongs to.

In the SAML assertion, the **NameID** element displays user ID, which is sent to API Gateway from the IdP.

For example, as shown in below sample, *alice* is the user ID.

```
<Subject>
  <NameID>alice</NameID>
  <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
    <SubjectConfirmationData InResponseTo="a57d9j2i936ae5de2icdedg73jce390"
      NotOnOrAfter="2021-02-19T12:51:28.106Z"
      Recipient="https://localhost:9073/apigatewayui/saml/SSO"
    />
  </SubjectConfirmation>
</Subject>
```

In the SAML assertions, under the **AttributeStatement** element, if the **AttributeName** has any of the following values, then the **AttributeValue** element displays the group name to which the user ID is associated in the IdP. This attribute value is used by API Gateway to map the user to the corresponding groups.

- `http://schemas.microsoft.com/ws/2008/06/identity/claims/role`
- `http://schemas.xmlsoap.org/claims/Group`

Example 1:

```
<AttributeStatement>
  <Attribute
    Name="http://schemas.microsoft.com/ws/2008/06/identity/claims/role">
    <AttributeValue>group1</AttributeValue>
  </Attribute>
</AttributeStatement>
```

In the example 1, based on the SAML assertion, the user is associated to the group called *group1* in the IdP. Later, API Gateway uses this value *group1* to map the user to the corresponding group.

Example 2 :

```
<AttributeStatement>
  <Attribute
Name="http://schemas.microsoft.com/ws/2008/06/identity/claims/role">
  <AttributeValue>group2</AttributeValue>
</Attribute>
</AttributeStatement>
```

In the example 2, based on the SAML assertion, the user is associated to the group called *group2* in the IdP. Later, API Gateway uses this value *group2* to map the user to the corresponding group.

Example 3:

```
<saml2:AttributeStatement xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
  <saml2:Attribute Name="http://schemas.xmlsoap.org/claims/Group"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
  <saml2:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">Everyone</saml2:AttributeValue>
  <saml2:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">group1</saml2:AttributeValue>
  <saml2:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">group2</saml2:AttributeValue>
</saml2:Attribute>
</saml2:AttributeStatement>
```

In the example 3, based on the SAML assertion, the user is associated to the groups called *Everyone*, *group1*, and *group2* in the IdP. Later, API Gateway uses these values *Everyone*, *group1*, and *group2* to map the user to the corresponding groups.

The SAML assertion is populated dynamically for each time when the user logs into API Gateway using SSO. If the user is mapped to a different group in the IdP or if the user is removed from the IdP during the subsequent login, then API Gateway maps the user to a group based on the SAML assertion of that subsequent session. This is to ensure that the mapping is always in synchronization between IdP and API Gateway.

How to map an API Gateway group based on SAML assertion from IdP?

This use case explains how to map the logged in SSO users to API Gateway groups based on the SAML assertion.

The use case starts when you log in to API Gateway using SSO and ends when you are mapped to API Gateway group based on the SAML assertion.

You must use the `saml_groups_mapping.xml` file at `<SAGInstallDir> /IntegrationServer/instances/IS_Instance_Name /packages/WmAPIGateway/config/resources/security` to map the IdP groups specified in the SAML assertion to the groups in API Gateway.

Sample `saml_groups_mapping.xml`

```
<groupsmapping>
<group source="group1" target="group3" />
</groupsmapping>
```

For example, as shown in the above sample, the *group1* displayed in the SAML assertion from IdP is mapped to the group called *group3* in API Gateway. Hence, the group in the SAML response is termed as source and the group in the API Gateway is termed as target.

Note:

Make sure you restart API Gateway instance, whenever you update the `saml_groups_mapping.xml` for the latest updates to take effect.

> Steps involved in mapping the IdP group in the SAML assertion to API Gateway group

1. API Gateway checks whether a group mapping exist in the `saml_groups_mapping.xml` for the group in the SAML assertion. If the group mapping exists, then the user is automatically mapped to target group specified in the `saml_groups_mapping.xml`.

Sample SAML assertion

```
<AttributeStatement>
  <Attribute
Name="http://schemas.microsoft.com/ws/2008/06/identity/claims/role">
    <AttributeValue>group1</AttributeValue>
  </Attribute>
</AttributeStatement>
```

Sample `saml_groups_mapping.xml`

```
<groupsmapping>
<group source="group1" target="group3" />
</groupsmapping>
```

For example, if the user is mapped to the *group1* as per to the SAML assertion and if the *group1* is mapped to *group3* in the `saml_groups_mapping.xml` file, then the user is mapped to the *group3*.

2. API Gateway checks if the group mapping doesn't exist in the `saml_groups_mapping.xml`, then check whether the group exist in the API Gateway. If the group exists in the API Gateway, then the user is mapped to that group.

Sample SAML assertion

```
<AttributeStatement>
  <Attribute
Name="http://schemas.microsoft.com/ws/2008/06/identity/claims/role">
    <AttributeValue>group4</AttributeValue>
  </Attribute>
</AttributeStatement>
```

Sample `saml_groups_mapping.xml`

```
<groupsmapping>
<group source="group1" target="group3" />
</groupsmapping>
```

For example, if the user is mapped to the *group4* as per to the SAML assertion, if there is no group mapping specified for *group4* in the `saml_groups_mapping.xml` file, and if the *group4* is available in API Gateway, then the user is mapped to the *group4*.

3. If there is no group specified in the `saml_groups_mapping.xml` file, and if there is no group exists in API Gateway, then the user is mapped to the default the *Everybody* group.

Troubleshoot tips for SSO configuration

Issue	Symptom	Solution
org.opensaml.common.SAMLException: Local entity is not the intended audience of the assertion in at least one AudienceRestriction.	The audience URL in the SAML assertion does not match with the Service provider identity in API Gateway.	Make sure the Service provider identity in API Gateway matches with the audience URL.

Note:

In case, if there is any other exception, check the `sag_osgi.log` at `<SAGInstallDir>\profiles\IS_default\logs` directory to trouble shoot.

Secure API using OAuth2 with refresh token workflow

When using the authorization code grant type to get the access token, you need to get the permission from the resource owners at least for the first time. In the subsequent attempts to get the access token, if you do not want to get the permission from the resource owners, then you can use the refresh token.

This use case explains how to secure the API using OAuth2 authentication strategy. It also explains the refresh token workflow in detail.

Configuring OAuth2 Authentication with Refresh Token

This use case explains how to secure the API using OAuth2 authentication strategy with `authorization_code` and `refresh_token` grant types.

The use case starts when you create an API and ends when you create an application strategy with OAuth2 authentication scheme.

➤ To configure OAuth2 Authentication with Refresh Token

1. Create an API.

For details about creating an API, see [“Creating a REST API” on page 273](#).

Create API

Create an API by importing from a file, URL or start from scratch

Lets Get Started!

Import API from file
Create an API by importing API from a specified file.

Import API from URL
Create an API by importing it from an URL.

URL*

Protected

Name

Type

Version

Create

Description

2. Enable the OAuth2 token identification type in the **Identify & Authorize** policy.

For details about **Identify & Authorize** policy, see [“Identify and Authorize Application” on page 398](#).

The screenshot shows the 'Swagger Petstore' API configuration page. The 'Policies' tab is active, and the 'Identify & Authorize' policy is selected in the 'Policy catalog' on the left. The main area displays a policy flow diagram with various components like 'Error Handling', 'Response Process...', 'Traffic Monitoring', 'Routing', 'Identify & Access', 'Request Process...', and 'Transport'. On the right, the 'Policy properties' panel is open, showing the 'Identify & Authorize' configuration. Under 'Application Identification condition', the 'Identification Type*' section has 'OAuth2 Token' checked, which is highlighted with a red box.

3. Create OAuth scope in the local authorization server.

Administration
Implement and manage the general and security related configurations for API Gateway.

General **Security** Destinations Manage data System settings External accounts

Keystore/Truststore
Ports
Global IP Access Settings
SAML issuer
Custom assertions
Kerberos
Master password
JWT/OAuth/OpenID
Providers
Microgateway

Internal authorization servers
Configure API Gateway as an OAuth authorization server and as a JWT issuer.

Authorization server alias	Description
local	API Gateway as an Authorization server.

local

Name
local

Description
API Gateway as an Authorization server.

- JWT configuration
Configure API Gateway as a JWT issuer.
- OAuth configuration
Configure API Gateway as an OAuth authorization server.
- OAuth tokens
Tokens that are available in the authorization server.
- OAuth scopes
Provide the scopes that are registered in the Authorization server.

Scope*	Scope description*
TestRefreshToken	Test + Add

Cancel Update

- Map the OAuth scope to the API scope.

For details about mapping OAuth scope, see “[Mapping OAuth or OpenID Scopes](#)” on page 191.

Create scope mapping
Map the authorization server scope to the API scopes to authorize the access tokens.

Cancel Save

Scope mapping

Auth server scope
API scopes

API scopes

Type a keyword

Selected API scopes

Scope	API Name	Version	Description
API Scope	Swagger Petstore	1.0.5	This is a sample server Petstore server. You can find out more about Swagger at [http://swagger.io]...

- Create an application with OAuth2 authentication strategy.

Create application
Create an application by providing the basic information, defining identifiers, and adding the required APIs.

Cancel Save

Application details

Basic information
Identifiers
APIs
Advanced
Authentication

Find APIs
s

Selected APIs

Name	Description	Version
Swagger Petstore	This is a sample server Petstore server. You can find out more about Swagger at [http://swagger.io][http://swa...	1.0.5

Continue to Advanced

- Create a new application.

For details about creating an application, see [“Creating an Application” on page 597](#).

- b. Associate the application with the API that you have created.
- c. Click the **Authentication** tab to create strategy with OAuth2 authentication.

- d. Select the **Authentication schemes** as *OAUTH2*.
- e. Specify the **Authentication server** as *local*.
- f. Enable the **Generate credentials** toggle button to generate the client dynamically in the authorization server and provide the following information:
 - a. Select the **Application Type** as *Confidential*. A confidential client is an application that can keep a client password confidential to the world. This client password is assigned to the client app by the authorization server. This password is used to identify the client to the authorization server, to avoid fraud. An example of a confidential client could be a web app, where no one but the administrator can get access to the server, and see the client password.
 - b. Select the application profile from the **Application profile** drop-down menu. For example, *web*.

- c. Specify the duration in seconds for which the access token is active in the **Token lifetime (seconds)**.
- d. Specify the number of times you can use the refresh token in the **Token refresh limit** to get a new access token.

Note:

To use refresh token unlimitedly, specify the limit as -1.

- e. Specify the URIs that the authorization server can use to redirect the resource owner's browser during the grant process. You can add multiple URIs by clicking **+Add**.
- f. Specify the grant type to be used to generate the credentials. For this specific use case, we have selected *authorization_code*, *client_credentials*, and *refresh_token*, which are dynamically populated from the authorization server.

Note:

Make sure you have selected *refresh_token* **grant_type**, if you want to get the refresh tokens.

- g. Select the scopes that are to be mapped for the authentication strategy.
- h. Click **Add** to save the strategy.
- i. Click **Save** to save the application.

Refresh Token Process Flow

This use case explains the following workflow:

1. How to get the access token with resource owner permission?
2. How to get the access token without resource owner permission using refresh token in the subsequent attempts?

How to get the access token with resource owner permission?

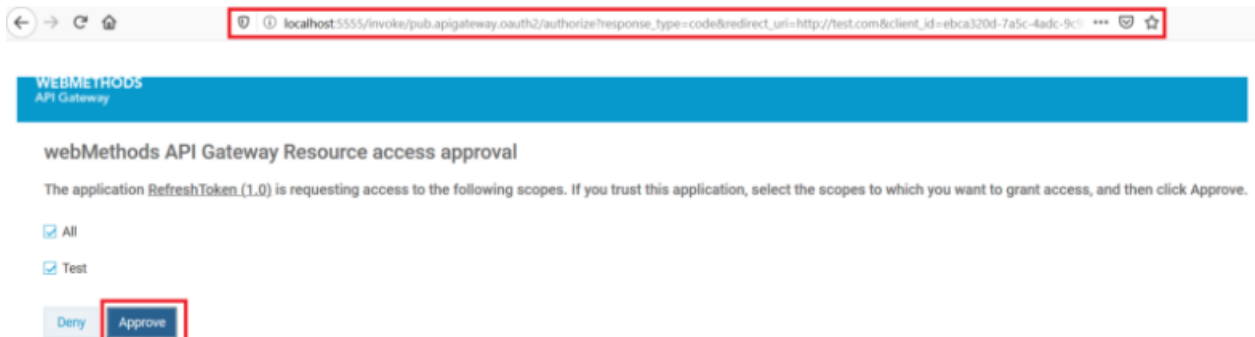
This use case starts when you get the authorization code and ends when you access then API.

➤ To get access token using authorization code grant type (With resource owner permission).

1. Get authorization code.
 - a. Click the `http(s)://hostname:port/invoke/pub.apigateway.oauth2/authorize?response_type=code&redirect_uri=<redirectURI>&client_id=<Client ID>`.

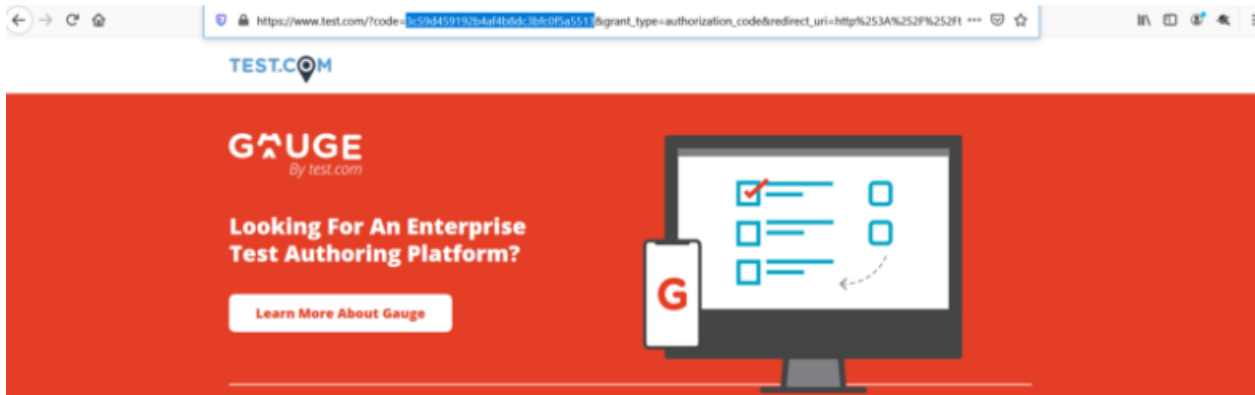
Note:

Make sure you have replaced the `<redirectURI>` and `<ClientID>` in the above mentioned URL. You can get the redirect URI and client ID from the **Authentication** tab of the **Application** screen.



- b. Click the **Approve** button.
- c. Enter the credentials of your API Gateway instance.

You will be re-directed to the redirect URI as per to the configuration. The below screenshot is just a sample, you will be redirected to a different URL based on your configuration and so the screenshot varies accordingly . If the given redirect URI is not a valid web page, you may get a *Page not found* error, which is fine, because we get the authorization code value from the browser URL.



- d. Make a note of the authorization code that is displayed in the address bar of the browser. As highlighted in the above image's URL, you can see the authorization code in the `code=` field of the URL.
2. Get Access Token.
 - a. Invoke the access token endpoint.

Request: POST `http(s):// hostname:port /invoke/pub.apigateway.oauth2/getAccessToken`

In the **Authorization** tab, select the authorization type as *Basic Auth*. Provide the client ID as username and client secret as password. You can get the client ID and client secret in the **Authentication** tab of the **Application** screen.

Sample request body

```
{
  "redirect_uri": "http://test.com",
  "scope": "email",
  "grant_type": "authorization_code",
  "code": "4b4b16c68f1c4b6fa7f26e0cb00b5daa"
}
```

Note:

You must replace the `redirect_URI`, `scope`, and `code` with appropriate values. For the code field value, make sure you use the authorization code that you have noted down in the previous step.

Sample response body

```
{
  "scope": "TestRefreshToken",
  "access_token":
  "c92b6227a19c46f1a6545bf370bb6ee6e30ff87957ef4b1aaa9577f7e86e4bd7",
  "refresh_token":
  "f78dd4fc5b8d4d799cf066427e828e26ce7e3723e4334416a7b9cd8a274e6947",
  "token_type": "Bearer",
  "expires_in": 3600
}
```

3. Access API using the REST API client.

In the **Authorization** tab, select the authorization type as *Bearer Token* and provide the access token that you get from the response payload of the previous step.

How to get the access token without resource owner permission using refresh token in the subsequent attempts?

This use case starts when you get the authorization code and ends when you access then API.

➤ To get access token using refresh token (Without resource owner permission).

When the access token expires and if you need to access the same API, you need to get another access token. If you have refresh token, you can get a new access token without getting the permission from the resource owner.

1. Invoke the refresh token endpoint.

Request: POST `http(s)://hostname:port/invoke/pub.oauth/refreshAccessToken`

In the **Authorization** tab, select the authorization type as *Basic Auth*. Provide the client ID as username and client secret as password. You can get the client ID and client secret in the **Authentication** tab of the **Application** screen.

Sample request body

```
{
  "grant_type": "refresh_token",
  "refresh_token": "f78dd4fc5b8d4d799cf066427e828e26ce7e3723e4334416a7b9cd8a274e6947"
}
```

Note:

Make sure you have replaced the refresh token that you got from the Step 2 using [“ How to get the access token with resource owner permission?” on page 851](#) use case.

Sample response body

```
{
  "grant_type": "refresh_token",
  "refresh_token":
  "f78dd4fc5b8d4d799cf066427e828e26ce7e3723e4334416a7b9cd8a274e6947",
  "scope": "TestRefreshToken ",
  "access_token":
  "c102bcaebecf451ca705bf54d26fae732ea9790a0ff64a87a010b3875b4b8da2",
  "token_type": "Bearer",
  "expires_in": 3600
}
```

2. Access API using the REST API client.

In the **Authorization** tab, select the authorization type as *Bearer Token* and provide the access token that you get from the response payload of the previous step.