

webMethods API Cloud: API Gateway User's Guide

Version 10.1

This document applies to webMethods API Cloud: API Gateway Version 10.1 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 © 2016-2017 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 <http://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 <http://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 <http://softwareag.com/licenses> and/or in the root installation directory of the licensed product(s).

Table of Contents

| | |
|--|-----------|
| About this Guide | 9 |
| Document Conventions..... | 9 |
| Online Information..... | 10 |
| 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..... | 15 |
| Using Help in API Gateway..... | 15 |
| API Gateway Administration | 17 |
| Overview of Administration Tasks..... | 18 |
| General Configuration..... | 18 |
| Configuring Extended Settings..... | 18 |
| Configuring API Fault Settings..... | 27 |
| Approval Configuration..... | 28 |
| Configuring Approvals for Creating an Application..... | 29 |
| Configuring Approvals for Registering Application..... | 31 |
| Configuring Approvals for Updating Application..... | 33 |
| Configuring Approvals for Subscribing Package..... | 35 |
| Managing Pending Requests..... | 38 |
| Deleting Requests..... | 40 |
| URL Aliases..... | 40 |
| Creating URL Alias..... | 41 |
| Example: URL Aliases..... | 42 |
| Deleting a URL Alias..... | 44 |
| Modifying a URL Alias..... | 44 |
| License Alerts..... | 44 |
| Configuring Criteria for License Alert Notification..... | 45 |
| Modifying License Alert Configurations..... | 46 |
| Deleting a License Alert Configuration..... | 46 |
| Security Configuration..... | 46 |
| Keystore and Truststore..... | 47 |
| Configuring Keystore Information..... | 47 |
| Configuring Truststore Information..... | 48 |
| Configuring Keystore and Truststore Information..... | 49 |
| Modifying Keystore Information..... | 50 |
| Deleting Keystore Information..... | 51 |
| Modifying Truststore Information..... | 51 |
| Deleting Truststore Information..... | 52 |
| SAML Issuer..... | 52 |

| | |
|---|-----|
| Custom Assertions..... | 56 |
| Creating a Custom Assertion..... | 59 |
| Viewing Custom Assertion List and Assertion Configuration..... | 60 |
| Modifying Custom Assertion..... | 61 |
| Deleting Custom Assertion..... | 61 |
| Example: Custom Assertions..... | 62 |
| OAuth 2.0..... | 66 |
| Authorization Workflows..... | 69 |
| Configuring API Gateway to Use OAuth 2.0..... | 74 |
| Adding OAuth 2.0 Partner Provider..... | 74 |
| Adding OAuth 2.0 Authorization Server..... | 80 |
| Viewing OAuth 2.0 Partner Provider List and Provider Configuration..... | 83 |
| Viewing OAuth 2.0 Authorization Server List and Server Configuration..... | 84 |
| Setting Default OAuth 2.0 Server..... | 85 |
| Modifying OAuth2.0 Partner Provider Configuration..... | 85 |
| Modifying OAuth2.0 Authorization Server Configuration..... | 86 |
| Deleting OAuth2.0 Partner Provider..... | 86 |
| Deleting OAuth2.0 Authorization Server..... | 87 |
| Kerberos Settings..... | 87 |
| Configuring API Gateway to Use Kerberos..... | 88 |
| JWT..... | 89 |
| JWT Workflow..... | 91 |
| Configuring API Gateway as JWT Issuer..... | 92 |
| Configuring API Gateway as JWT Relying Party..... | 93 |
| Identifying Applications Using JWT..... | 94 |
| OpenID Provider..... | 94 |
| Authorization Workflows..... | 98 |
| Identifying Applications Using OpenID Tokens..... | 101 |
| Adding an OpenID Provider..... | 102 |
| Viewing OpenID Provider List and Provider Configuration..... | 105 |
| Modifying the OpenID Provider Configuration..... | 106 |
| Activating an OpenID Provider..... | 106 |
| Deactivating an OpenID Provider..... | 107 |
| Deleting an OpenID Provider..... | 107 |
| Destination Configuration..... | 107 |
| Configuring Events for API Gateway Destination..... | 108 |
| Configuring API Portal Destination..... | 108 |
| Configuring Events for API Portal Destination..... | 110 |
| Configuring CentraSite Destination..... | 110 |
| Configuring Events for CentraSite Destination..... | 112 |
| Configuring Elasticsearch Destination..... | 113 |
| Configuring Events for Elasticsearch Destination..... | 114 |
| Configuring Email Templates..... | 115 |
| System Settings..... | 116 |
| Modifying API Gateway Configuration Parameters..... | 117 |

| | |
|---|------------|
| Users and Access Profiles..... | 119 |
| Managing Users and Access Profiles..... | 120 |
| APIs..... | 121 |
| Overview..... | 122 |
| Creating an API by Importing an API from a File..... | 122 |
| Creating an API by Importing an API from a URL..... | 123 |
| Creating a REST API from Scratch..... | 123 |
| Viewing API List and API Details..... | 127 |
| Filtering APIs..... | 128 |
| Publishing an API..... | 128 |
| Modifying API Details..... | 129 |
| Updating APIs..... | 129 |
| Updating an API by Importing an API from a File..... | 130 |
| Updating an API by Importing an API from a URL..... | 131 |
| SOAP to REST Transformation..... | 133 |
| Activating SOAP to Rest Transformation..... | 133 |
| Modifying the REST Definitions for SOAP Operations..... | 134 |
| Supported Content-types and Accept Headers..... | 137 |
| REST Service Endpoints..... | 138 |
| Samples for REST Request..... | 138 |
| Limitations..... | 141 |
| API Mocking..... | 141 |
| Enabling API Mocking..... | 142 |
| Modifying API Mocking Details..... | 142 |
| Custom Replacer..... | 145 |
| API Scopes..... | 145 |
| Creating an API Scope..... | 146 |
| Viewing List of API Scopes and Scope Details..... | 147 |
| Modifying API Scope Details..... | 148 |
| Deleting an API Scope..... | 149 |
| Example: Usage Scenarios of API Scopes..... | 149 |
| Exposing a REST API to Applications..... | 154 |
| Exposing a SOAP API to Applications..... | 155 |
| Versioning APIs..... | 156 |
| Creating New API Version..... | 157 |
| API Grouping..... | 157 |
| Exporting APIs..... | 158 |
| Exporting Specifications..... | 158 |
| Example: Managing an API..... | 159 |
| Policies..... | 171 |
| Overview..... | 172 |
| Policy Validation and Dependencies..... | 174 |
| System-defined Stages and Policies..... | 181 |

| | |
|--|-----|
| Transport..... | 181 |
| Identify and Access..... | 184 |
| Request Processing..... | 195 |
| Routing..... | 198 |
| Traffic Monitoring..... | 218 |
| Response Processing..... | 231 |
| Error Handling..... | 233 |
| The API for Context Variables..... | 236 |
| Managing Threat Protection Policies..... | 251 |
| Configuring Global Denial of Service Policy..... | 252 |
| Configuring Denial of Service by IP Policy..... | 253 |
| Managing Denied IP List..... | 254 |
| Configuring Rules..... | 255 |
| Registering a Mobile Device or Application..... | 261 |
| Configuring Alert Settings..... | 262 |
| Managing Global Policies..... | 262 |
| Creating a Global Policy..... | 263 |
| Modifying the Scope of a Global Policy..... | 264 |
| Refining the Scope of a Global Policy..... | 265 |
| Associating Policies to a Global Policy..... | 267 |
| Configuring Properties for a Global Policy..... | 268 |
| Viewing List of Global Policies and Policy Details..... | 270 |
| Modifying Global Policy Details..... | 271 |
| Activating a Global Policy..... | 272 |
| Deactivating a Global Policy..... | 273 |
| Deleting a Global Policy..... | 274 |
| Copying a Global Policy..... | 274 |
| Viewing Global Policy Details in an API..... | 275 |
| Exporting Global Policies..... | 276 |
| Managing API-level Policies..... | 277 |
| Assigning a Policy to an API..... | 277 |
| Viewing the List of Policies Assigned to an API..... | 278 |
| Modifying a Policy Assigned to an API..... | 278 |
| Modifying Policies Assigned to an API..... | 278 |
| Managing Scope-level Policies..... | 279 |
| Creating a Scope-level Policy..... | 280 |
| Viewing List of Scope-level Policies and Policy Details..... | 281 |
| Modifying Scope-level Policy Details..... | 282 |
| Deleting a Scope-level Policy..... | 283 |
| Managing Policy Templates..... | 284 |
| Creating a Policy Template..... | 284 |
| Associating Policies with a Policy Template..... | 285 |
| Configuring Properties for a Policy Template..... | 286 |
| Viewing List of Policy Templates and Template Details..... | 287 |
| Modifying Policy Template Details..... | 288 |

| | |
|--|------------|
| Deleting a Policy Template..... | 289 |
| Copying a Policy Template..... | 289 |
| Applying a Policy Template on the API Details Page..... | 290 |
| Modifying a Policy Template on the API Details Page..... | 291 |
| Saving Policy Definition of an API as Policy Template..... | 293 |
| Aliases..... | 295 |
| Overview..... | 296 |
| Creating a Simple Alias..... | 296 |
| Creating an Endpoint Alias..... | 297 |
| Creating an HTTP Transport Security Alias..... | 298 |
| Creating a SOAP Message Security Alias..... | 302 |
| Creating a webMethods Integration Server Service alias..... | 306 |
| Creating an XSLT Transformation alias..... | 307 |
| Applications..... | 309 |
| Overview..... | 310 |
| Creating an Application..... | 311 |
| Viewing List of Applications and Application Details..... | 313 |
| Regenerating API Access Key..... | 313 |
| Refreshing OAuth Token..... | 314 |
| Modifying Application Details..... | 314 |
| Registering an API with Consumer Applications from API Details Page..... | 315 |
| Registering APIs with Consumer Applications from Application Details Page..... | 315 |
| API Packages and Plans..... | 317 |
| Overview..... | 318 |
| Creating a Package..... | 318 |
| Creating a Plan..... | 319 |
| Viewing List of Packages and Package Details..... | 321 |
| Modifying a Package..... | 321 |
| Deleting a Package..... | 322 |
| Activating a Package..... | 323 |
| Publishing a Package..... | 323 |
| Viewing List of Plans and Plan Details..... | 324 |
| Modifying a Plan..... | 324 |
| Deleting a Plan..... | 325 |
| Import Archives..... | 327 |
| Importing Exported Files..... | 328 |
| API Gateway Analytics..... | 331 |
| Analytics Dashboards..... | 332 |
| API Gateway Dashboard..... | 333 |
| API-specific Dashboard..... | 336 |
| Purging API Analytics Data..... | 338 |

About this Guide

This guide 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 guide 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 storage locations for services on webMethods Integration Server, using the convention <i>folder.subfolder.service</i> . |
| UPPERCASE | Identifies keyboard keys. Keys you must press simultaneously are joined with a plus sign (+). |
| <i>Italic</i> | Identifies variables for which you must supply values specific to your own situation or environment. Identifies new terms the first time they occur in the text. |
| Monospace font | Identifies text you must type or messages displayed by the system. |
| { } | 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. |

| Convention | Description |
|------------|---|
| ... | Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...). |

Online Information

Software AG Documentation Website

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

Software AG Empower Product Support Website

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](#).

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.

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 | 15 |
| ■ Using Help in API Gateway | 15 |

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 and SOAP-based APIs, provides protection from malicious attacks, provides a complete run-time governance of APIs, and information about gateway-specific events and API-specific events.

API Gateway provides the following key features:

Support for SOAP APIs, and REST APIs

API Gateway supports REST-based APIs and SOAP-based 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.

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.

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. The search functionality does not support search of resource or method name for a REST API.

You can search for the following types of data:

- APIs
- Applications
- Alias
- Plans

■ Packages

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 type or keyword.

- **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 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: 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.

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.

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
```

Edit the configuration file as required. 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 located in the top-right corner under **Username > Help**. This opens the introduction to the webMethods API Gateway page in the help system. You can browse topics in the view area. Click on a topic to display detailed information. You will also find the help links in the form of a ? on several pages of the

API Gateway UI. Click the ? icon to view the detailed information of the page on which the ? resides.

2 API Gateway Administration

| | |
|--|-----|
| ■ Overview of Administration Tasks | 18 |
| ■ General Configuration | 18 |
| ■ Security Configuration | 46 |
| ■ Destination Configuration | 107 |
| ■ System Settings | 116 |

Overview of Administration Tasks

Various aspects of the way API Gateway functions can be configured:

- General configuration: extended settings, service faults, approval settings, and URL aliases.
- Security configuration: keystores and trustore, 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.
- System setting configuration: configure system-level configuration parameters.

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 the extended settings which are advanced parameters required for your server to operate properly.
- Configure global service fault settings for errors being returned by the API Gateway to the applications.
- Configure approval settings.
- Configure URL aliases.

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.

To configure the extended settings

1. Select **Username > Administration**.
2. Select **General > Extended settings**.
3. Provide the following information in the Extended settings section:

| Parameter | Description |
|---|--|
| allowExceedMaxWindowSize | <p>Provide a value <code>true</code> to exceed the maximum window size configured and <code>false</code> to avoid exceeding the maximum window size.</p> <p>The default value is <code>true</code>.</p> |
| apig_MENConfiguration_tickInterval | <p>Provide a value to specify the time interval (in seconds) between each interval processor iteration. This must be an evenly divisible fraction of the smallest policy interval, which is one minute.</p> <p>Default value is 15.</p> <p>If you have configured the result window size in elastic search, specify the configured value to fetch results accordingly.</p> |
| apig_rest_service_redirect | <p>Provide the value <code>true</code> to redirect the incoming service requests to one of the following directives that API Gateway supports in addition to the <code>/gateway</code> directive:</p> <ul style="list-style-type: none"> ■ <code>/ws</code> ■ <code>/mediator</code> <p>The default value is <code>false</code>.</p> |
| apig_schemaValidationPoolSize | <p>Provide a value that specifies the schema validation pool size.</p> <p>The default value is 10.</p> |
| apigGroupingPossibleValues | <p>Specifies the groups that the APIs would be grouped under.</p> <p>The default groups provided are <code>Finance</code>, <code>Banking</code> and <code>Insurance</code>, <code>Sales</code> and <code>Ordering</code>, <code>Search</code>, <code>Transportation</code> and <code>Warehousing</code></p> |

| Parameter | Description |
|--|--|
| | <p>You can delete or add new groups that the APIs can be grouped under.</p> |
| <p>apiKeyExpirationPeriod</p> | <p>Specifies the length of time that the API key is valid. The key expires after the set number of minutes, hours, days, months, or years. If a value is not specified, then the API key never expires. The expiration date is computed as follows:</p> <ul style="list-style-type: none"> ■ 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 current time of the application in the system + The value specified in the apiKeyExpirationPeriod parameter. |
| <p>apiKeyHeader</p> | <p>Provide the header value from which API Gateway receives the API key.</p> <p>The default value is <code>x-Gateway-APIKey</code></p> |
| <p>apiMaturityStatePossibleValues</p> | <p>Specify the API maturity state values that can be set for an API.</p> <p>The default values provided are <code>Beta</code>, <code>Deprecated</code>, <code>Experimental</code>, <code>Production</code>, <code>Test</code>.</p> |
| <p>defaultEncoding</p> | <p>Specify the default encoding format.</p> <p>The default value is <code>UTF-8</code>.</p> |
| <p>esScrollTimeout</p> | <p>Provide a value that specifies the time out interval for fetching the result that is greater than <code>maxWindowSize</code>.</p> <p>Default value is the time, in milliseconds, required to fetch the records specified in <code>maxWindowSize</code>.</p> |

| Parameter | Description |
|----------------------------------|---|
| maxWindowSize | <p>Provide a value to to set the result window size that specifies the number of records per query.</p> <p>Default value is 10000.</p> |
| pg_Active_OpenID_Provider | <p>Specifies the unique identifier (ID) of an active OpenID Provider.</p> |
| pg_JWT_isHTTPS | <p>Provide a value to specify the transport protocol over which the JSON Web Tokens (JWTs) are granted authorization.</p> <p>Available values are:</p> <ul style="list-style-type: none">■ <code>true</code>. Sets the transport protocol HTTPS. This is set by default.■ <code>false</code>. Sets the transport protocol HTTP. |
| pg_oauth2_isHTTPS | <p>Provide a value to specify the transport protocol over which the OAuth 2.0 access tokens are granted authorization.</p> <p>Available values are:</p> <ul style="list-style-type: none">■ <code>true</code>. Sets the transport protocol HTTPS. This is set by default.■ <code>false</code>. Sets the transport protocol HTTP. |
| pg_OpenID_isHTTPS | <p>Provide a value to specify the transport protocol over which the OpenID (ID) tokens are granted authorization.</p> <p>Available values are:</p> <ul style="list-style-type: none">■ <code>true</code>. Sets the transport protocol HTTPS. This is set by default.■ <code>false</code>. Sets the transport protocol HTTP. |

| Parameter | Description |
|--|---|
| pg_xslt_enableDOM | <p>Provide the value <code>true</code> to enable DOM parsing.</p> <p>The value <code>false</code> disables the DOM parsing and enables other parsers</p> |
| pg.3pSnmpSender.sendDelay | <p>This is an internal parameter. Do not modify.</p> |
| pg.cs.snmpTarget.base64Encoded | <p>This is an internal parameter. Do not modify.</p> |
| pg.cs.snmpTarget.connTimeout | <p>Specify the number of milliseconds before an inactive connection to the SNMP target server is closed. If set to 0, the socket remains open indefinitely.</p> |
| pg.cs.snmpTarget.maxRequestSize | <p>Specify the maximum size (in bytes) for SNMP traps.</p> <p>The default value is 10485760.</p> |
| pg.cs.snmpTarget.retries | <p>Specify the number of times to resend SNMP traps upon failure.</p> <p>The default value is 1.</p> <p>This parameter works with <code>pg.cs.snmpTarget.sendTimeOut</code> to determine the delay in re-sending SNMP traps to malfunctioning SNMP servers (that is, it <code>retries*sendTimeOut</code>). This means that if the <code>retries</code> parameter is set to 3, and the <code>sendTimeOut</code> 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.</p> |

| Parameter | Description |
|--|---|
| <code>pg.cs.snmpTarget.sendTimeOut</code> | <p>Specify the time (in milliseconds) to wait before the SNMP trap times out because the server destination is not responding.</p> <p>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.</p> <p>This parameter works with <code>pg.cs.snmpTarget.retries</code> to determine the delay in resending SNMP traps to malfunctioning SNMP servers (that is, it retries <code>*sendTimeOut</code>). This means that if the <code>retries</code> parameter is set to 3, and the <code>sendTimeOut</code> 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.</p> |
| <code>pg.lb.failoverOnDowntimeErrorOnly</code> | <p>Specify API Gateway's behavior for load-balanced endpoints. The default is <code>false</code>, which means that in a load-balanced routing scenario, if a service fault is encountered in the response coming from endpoint 1, then API Gateway immediately tries the next configured endpoint. There is no distinction on the type of fault present in the response from endpoint 1. However, if this parameter is set to <code>true</code>, then API Gateway failovers</p> |

| Parameter | Description |
|---|---|
| | only if the service fault is a downtime error. |
| pg.snmp.communityTarget.base64Encoded | <p>Specify whether to use a third-party SNMPv1 community-based connection.</p> <p>If set to true, the community name must be the Base64 value.</p> <p>The default value is <code>false</code>.</p> |
| pg.snmp.communityTarget.maxRequestSize | <p>Specifies the maximum size (in bytes) for SNMP traps.</p> <p>The default value is <code>65535</code>.</p> |
| pg.snmp.communityTarget.retries | <p>Specify the number of times to resend SNMP traps upon failure.</p> <p>The default value is <code>1</code>.</p> <p>This parameter works with <code>pg.snmp.communityTarget.sendTimeOut</code> to determine the delay in re-sending SNMP traps to malfunctioning SNMP servers (that is, it retries <code>*sendTimeOut</code>). This means that if the <code>retries</code> parameter is set to <code>3</code>, and the <code>sendTimeOut</code> parameter is set to <code>500</code> milliseconds, there is a <code>1.5</code> 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.</p> |
| pg.snmp.communityTarget.sendTimeOut | <p>Specify 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</p> |

| Parameter | Description |
|---|--|
| | <p>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.</p> <p>The default value is 500.</p> <p>This parameter works with <code>pg.snmp.communityTarget.retries</code> to determine the delay in re-sending SNMP traps to non-responsive SNMP servers (that is, it retries <code>*sendTimeOut</code>). This means that if the <code>retries</code> parameter is set to 3, and the <code>sendTimeOut</code> 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.</p> |
| <code>pg.snmp.customTarget.connTimeout</code> | <p>Specify 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.</p> |
| <code>pg.snmp.userTarget.maxRequestSize</code> | <p>Specify the maximum size (in bytes) for SNMP traps.</p> <p>The default value is 65535.</p> |
| <code>pg.snmp.userTarget.retries</code> | <p>Specify the number of times to resend SNMP traps upon failure.</p> <p>The default value is 1.</p> <p>This parameter works with <code>pg.snmp.userTarget.sendTimeOut</code> to determine the delay in re-sending SNMP traps to malfunctioning SNMP servers (that is, it retries</p> |

| Parameter | Description |
|--|---|
| | <p>*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.</p> |
| <p>pg.snmp.userTarget.sendTimeOut</p> | <p>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.</p> <p>The default value is 500.</p> <p>This parameter works with <code>pg.snmp.userTarget.retries</code> 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</p> |

| Parameter | Description |
|---|--|
| | events when the queue reaches its maximum level. |
| <code>pg.uddiClient.publish.maxThreads</code> | Specify the maximum allowed number of threads to publish the performance metrics data to CentraSite. The default value is 2. |
| <code>pg.uddiClient.uddiClientTimeout</code> | Specify the number of milliseconds that can elapse before not publishing performance metrics to an unavailable API Gateway server. The default value is 5000. |

- 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

- Select **Username > Administration**.
- Select **General > API fault**.
- 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.

- 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.
- 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. For more information on registering APIs with applications, see ["Registering an API with Consumer Applications from API Details Page" on page 315](#) and ["Registering APIs with Consumer Applications from Application Details Page" on page 315](#).

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. Select **Username > Administration**.
2. Select **General > Create application**.
3. Set the **Enable** toggle button to the on position to enable approval configuration to take effect.
4. Select the access profile 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 approvers access profile 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 Administrators access profile, 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

| Field | Description |
|-------|--|
| | 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. |
| 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. |

| 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.

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

This implies that, any one user associated with the approvers access profile 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 Administrators access profile, 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. Select **Username > Administration**.
2. Select **General > Update application**.
3. Set the **Enable** toggle button to the on position to enable approval configuration to take effect.
4. Select the access profile 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 approvers access profile 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 Administrators access profile, 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. |

| Field | Description |
|----------------|--|
| Content | By default, the template appears. You can customize the email content. |
| | <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> |

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. |
| | <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> |

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. Select **Username > Administration**.
2. Select **General > Subscribe package**.
3. Set the **Enable** toggle button to the on position to enable approval configuration to take effect.
4. Select the access profile 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 approvers access profile 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 Administrators access profile, 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. |

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.

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. |

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

| Field | Description |
|-------|---|
| | Approval of Subscribe Package in the email sent, where Subscribe Package is the event.type. |

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

- Select **Username > Pending Requests**.
The Pending requests page appears.
- 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. For 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. |
|------------------------|--|

| Field | Description |
|-------|--|
| | <ul style="list-style-type: none"> ■ 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. ■ Register application <ul style="list-style-type: none"> ■ Application name: The name of the application. ■ API: The API to which the application is associated. ■ 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. ■ 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 clicking the checkboxes 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. Select **Username > 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.

A confirmation dialog appears.

4. Click **Yes** to confirm deletion.

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.

All the configured and available URL aliases are listed in a table with the corresponding details of the default URL path, port it is mapped to, and so on. You must have the manage aliases functional privilege assigned to update the alias information.

Note: Any modifications to the URL aliases in the WmAPIGateway package 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.

To create a URL alias

1. Select **Username > Administration**.

Alternatively you can click on **Administration** in the Welcome screen.

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 <code>http://</code> or <code>https://</code></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> |

| Field | Description |
|-------------------------|--|
| | The URL path cannot include a space or the following characters: # % ? ' " < \ |
| Default URL path | <p>The URL path to the API on API Gateway.</p> <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> |

- Click **Save**.

Example: URL Aliases

This section explains how an API consumer can invoke an API for which a URL alias is created. In this example, we use 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 the 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 URL aliases. To create a URL alias:

- Select **Username > Administration**.
- Select **General > URL aliases**.
- Click **Add URL alias** and provide the following values:

| Field | Value |
|--------------|-------|
| Alias | calc |

| Field | Value |
|------------------|---------------------------------------|
| Default URL path | /gateway/RESTCalcService/1.0/postCalc |

- 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`.

Since the default URL path is provided the API consumer can use this alias for any ports of gateway endpoint which are shown in 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:

- Select **Username > Administration**.
- Select **General > URL aliases**.
- Click **Add URL alias** and provide the following values:

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

- 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.

Deleting a URL Alias

To delete a URL alias

1. Select **Username > 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 that appears.

Modifying a URL Alias

To modify a URL alias

1. Select **Username > 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**.

License Alerts

API Gateway supports core as well 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. Alternatively, you can also view the license file in Integration Server (in the Integration Server Administrator, go to **Settings > Licensing** link). For more information on licensing, see *webMethods Integration Server Administrator's Guide*.

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 > License alert configurations** and the **Analytics > API usage details** pages appear in API Gateway.

Configuring Criteria for License Alert Notification

To configure license alert criteria

1. Select **Username > Administration**.
2. Select **General > License alert configurations**.

This displays a list of available alerts and the corresponding details.

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

| Field | Description |
|-----------------------|---|
| Notify at | Select the usage percentage at which the notification is to be sent. 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. |
| Notify through | The user can be notified through an email, through an alert that appears in the API Gateway user interface, or through both email and the alert that appears in the UI. |
| Email | A valid email address to which the license alerts are sent. 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. |

| Field | Description |
|--------|--|
| Action | Edit (by clicking the edit icon) or delete (by clicking the delete icon) the notification criteria that appears. |

- Click **Save**.

Modifying License Alert Configurations

To modify a license alert criteria

- Select **Username > Administration**.
- Select **General > License alert configurations**.
- In the License alert configurations list, locate the alert that you want to modify and click .
- Incorporate the required changes.
- Click **OK**.

Deleting a License Alert Configuration

To delete a license alert criteria

- Select **Username > Administration**.
- Select **General > License alert configurations**.
- In the License alert configurations list, locate the row that contains the alert you want to delete, and click .

The alert is deleted from the License 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.
- 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 the OAuth 2.0 application server settings.
- Configure Kerberos settings.

- Configure JSON web token.
- Configure OpenId provider.

Note: You have to restart the server for some of the security configurations enforced to take effect.

Keystore and Truststore

Keystores and truststores are required for incoming message-level security. They provide SSL authentication, encryption and decryption, and digital signing and verification services for all message content that API Gateway receives. You must first set up keystores and truststores and configure them in API Gateway.

For an API Gateway service to be SSL enforced, it must have a valid, authorized X.509 certificate installed in a keystore file, and the private key and signing certificate for the certificate issuer (CA) installed in a truststore file.

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. Select **Username > 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 | A text identifier for the keystore file. |

| Field | Description |
|--------------------|--|
| | <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.

Configuring Truststore Information

To configure truststore information

1. Select **Username > 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 | Optional. A text description for the truststore alias. |

- 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.

Configuring Keystore and Truststore Information

API Gateway includes a list of SSL keystores and truststores.

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 information

1. Select **Username > Administration**.
2. Select **General > Security**.

A list of existing keystores and truststores loaded during 10.1 first startup and those created in API Gateway and corresponding details are displayed.

3. Provide the following information in the Configure keystore and truststore settings section:

| Field | Description |
|----------------------------|--|
| Keystore alias | <p>Select a keystore that API Gateway uses for incoming message-level security.</p> <p>Lists all available keystores. If you have not configured an Integration Server keystore the list is empty.</p> |
| Key alias (signing) | <p>Select the alias for the private key to sign the outgoing response from API Gateway to the original client.</p> <p>This alias value validates the inbound requests to API Gateway and signs the outgoing response from API Gateway to the original client. It is auto-populated based on the keystore selected. This field lists all the aliases available in the chosen keystore. If there are no configured keystores, this field is empty.</p> |
| Truststore alias | <p>Select a truststore that establishes the HTTPS connection to the client application.</p> <p>It contains public certificates that are trusted by API Gateway.</p> |

4. Click **Save**.

Modifying Keystore Information

To modify keystore information

1. Select **Username > 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. Select **Username > 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.

Modifying Truststore Information

To modify truststore information

1. Select **Username > 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. Select **Username > 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.

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.

1. Select **Username > 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. 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). |

| Field | Description |
|---|---|
| | The delegator uses a signature element to authenticate the SAML request. |
| Issuer policy | <p>Specifies the name of an issuer policy to be used to communicate with SAML issuer.</p> <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 | <p>Specifies that WSS Username mode is used to obtain the SAML assertion to access the API.</p> <p>The WSS username token supplied in the header of the SOAP request that the consumer application submits to the API.</p> |
| Kerberos | <p>Specifies that Kerberos mode is used to obtain the SAML token and assertion to access the API.</p> <p>Transports the Kerberos token over the Transport Layer Security (TLS) protocol to provide additional security features.</p> |
| 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 | <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"> ■ 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 the LDAP or central user directory 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 the LDAP or central user directory used for authentication to the KDC. |

| Field | Description |
|---|--|
| | <ul style="list-style-type: none"> ■ 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 the LDAP or central user directory 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 | |

| Field | Description |
|--|---|
| Keystore alias | Specify the keystore to be used by API Gateway while sending the request to the STS. A keystore is a repository of private keys and corresponding public certificates. |
| 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. 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  . |

- 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 | 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. |
| X509 Token | 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. |
| Kerberos Token | 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. |
| SAML Token | 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. |

Note: API Gateway supports both the SAML 1.1 and 2.0 standards.

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. Select **Username > 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 | Name of the custom assertion. 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. |

| Field | Description |
|--------------------------|---|
| | 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. |
| 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 | If you have not uploaded the policy assertion file, provide the XML representation of assertion. |

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. Select **Username > Administration**.
2. Select **Security > Custom assertions**.

The Custom assertions section displays a list of all the currently defined policy assertions in API Gateway.

You can delete a custom assertion by clicking the **Delete** icon in the Action column.

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. Select **Username > Administration**.
2. Select **Security > Custom assertions**.

The Custom assertions section displays a list of all the currently defined policy assertions in API Gateway.

3. Click the name of the 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. Select **Username > Administration**.
2. Select **Security > Custom assertions**.

The Custom assertions section displays a list of all the defined custom assertions in API Gateway.

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>
        </wsp:Policy>
      </wsp>All>
    </wsp:ExactlyOne>
  </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.

You could create custom assertions to include one or more of the following security requirements:

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>
  </wsp:Policy>

```

```

<sp:Layout>
  <wsp:Policy>
    <sp:Strict/>
  </wsp:Policy>
</sp:Layout>
<sp:IncludeTimestamp/>
<sp:ProtectTokens/>
<sp:OnlySignEntireHeadersAndBody/>
<sp:SignBeforeEncrypting/>
</wsp:Policy>
</sp:SymmetricBinding>

```

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>
            <sp:Strict/>
          </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>

```

OAuth 2.0

The Open Authorization (OAuth) 2.0 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 2.0 authorization framework includes the following terms:

OAuth 2.0 Roles

- **Resource Owner (or the End User):**

This is the holder of the protected resources that the client application needs to access. 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 (OAuth 2.0):**

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.

OAuth 2.0 Scopes

OAuth 2.0 scopes are used to limit the amount of access that is granted to a client by the resource owner. Scope defines a single resource or a collection of resources to which a client application may request access to. When an API is created, and defined with one or more scopes, the corresponding OAuth 2.0 scopes are created in API Gateway. When requesting access tokens to access the protected resources in API Gateway, the client should pass the OAuth 2.0 scope in the same request, if required by the OAuth 2.0 provider.

OAuth 2.0 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.

- **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.

Note: The Resource Owner Password Credentials grant type is currently not supported by API Gateway.

OAuth 2.0 Clients

- **Confidential:**

This is a client that supplies a client ID and client secret to the authorization server to obtain access tokens (and optionally refresh tokens). These clients hold a user account on the authorization server. If a client does not have the proper credentials (client ID and secret) for the user account, the authorization server does not grant the client an access token.

- **Public:**

This is a client that uses only a client ID for identification, with no other credentials, to obtain access token. These clients are typically implemented in a browser using a scripting language such as JavaScript.

OAuth 2.0 Endpoints

- **Authorize Endpoint:**

This is the endpoint that allows the client to authenticate with the authorization server. This endpoint is applicable for Authorization Code grant type only.

- **Token Endpoint:**

This is the endpoint that allows the client to get an access token (and optionally refresh token) from the authorization server. Refresh token is applicable for Authorization Code grant type only.

OAuth 2.0 Tokens

- **Access Token:**

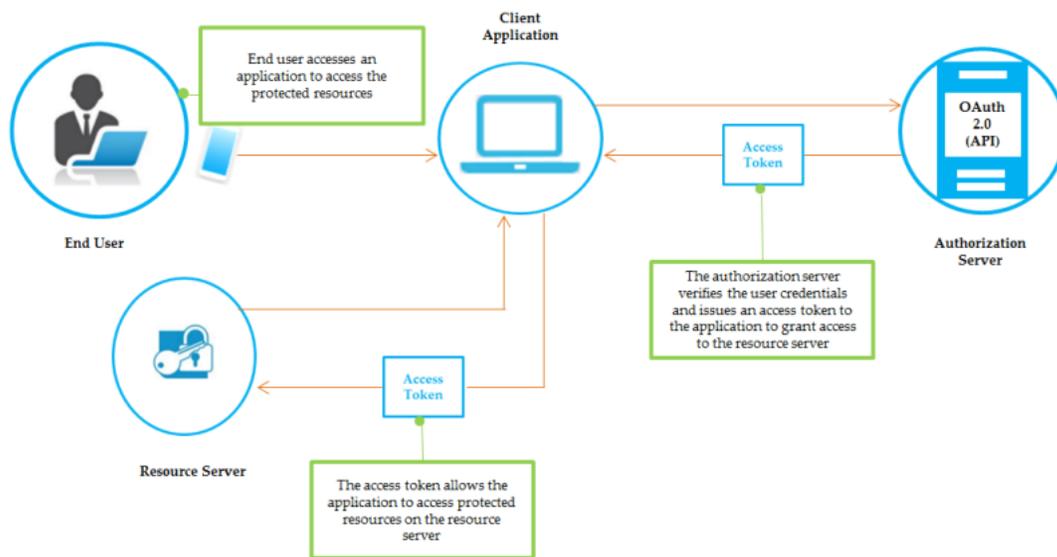
This is the token that allows the client to access protected resources on the resource server.

- **Refresh Token:**

This is the token that allows the client to obtain a renewed access token when the existing access token has expired. This token is applicable for Authorization Code grant type only.

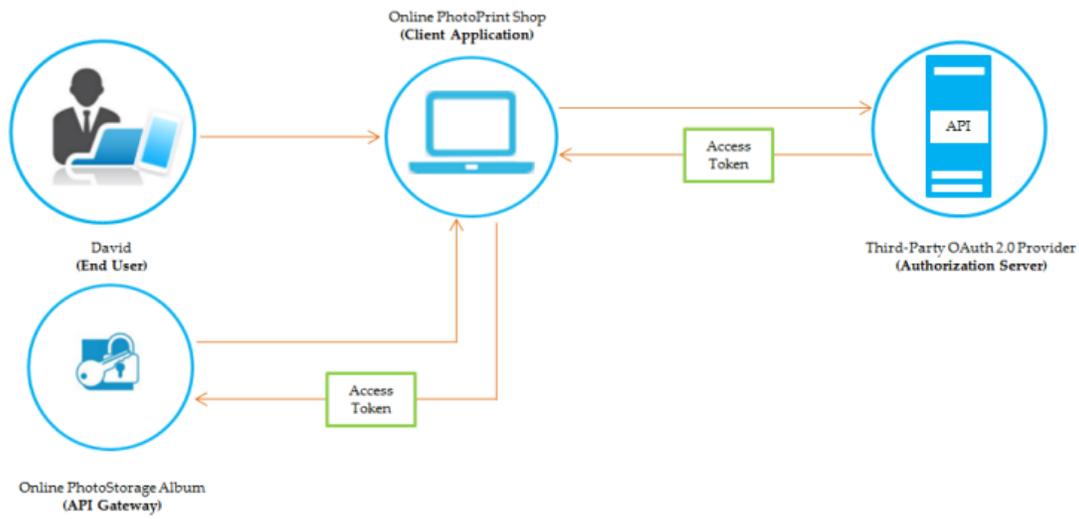
OAuth 2.0 Flow

The protected resources are stored on a **Resource Server** and the owner of the resources, **Resource Owner**, (usually the **End User**), grants the **Client Application** permission to access them. When a resource owner grants permission, the OAuth 2.0 **Authorization Server** issues an **Access Token**, depending on grant type, to the **Client Application**. When the **Client Application** passes the access token to the **Resource Server**, the **Resource Server** communicates with the **Authorization Server** to validate the token and, if valid, provides access to the resources. If the access token is expired, the **Client Application** uses the **Refresh Token** to request a new access token. The **Authorization Server** returns a new access token.



Example: OAuth 2.0 Flow

In the example, David is the **End User** (Resource Owner) who wants to access and print his photos stored on the Online PhotoStorage Album (Resource Server) using the Online PhotoPrint Shop (Client Application). PhotoPrint supplies David with an application that runs on his personal device (for example, phone). David uses that application to initiate the process. PhotoPrint sends a request to the PhotoStorage through the OAuth 2.0 Authorization Server. The OAuth 2.0 Authorization Server requests authorization from David and issues a token to PhotoPrint. PhotoPrint can then access David's photos on PhotoStorage.



An in-depth description of OAuth is beyond the scope of this guide but is available elsewhere.

Authorization Workflows

The flow of authorization requests and responses between the end user, client application, authorization server, and resource server depends on the grant type defined by the OAuth framework.

API Gateway supports the following authorization flows for OAuth 2.0 authentication:

- Authorization code grant
- Implicit grant
- Client credentials

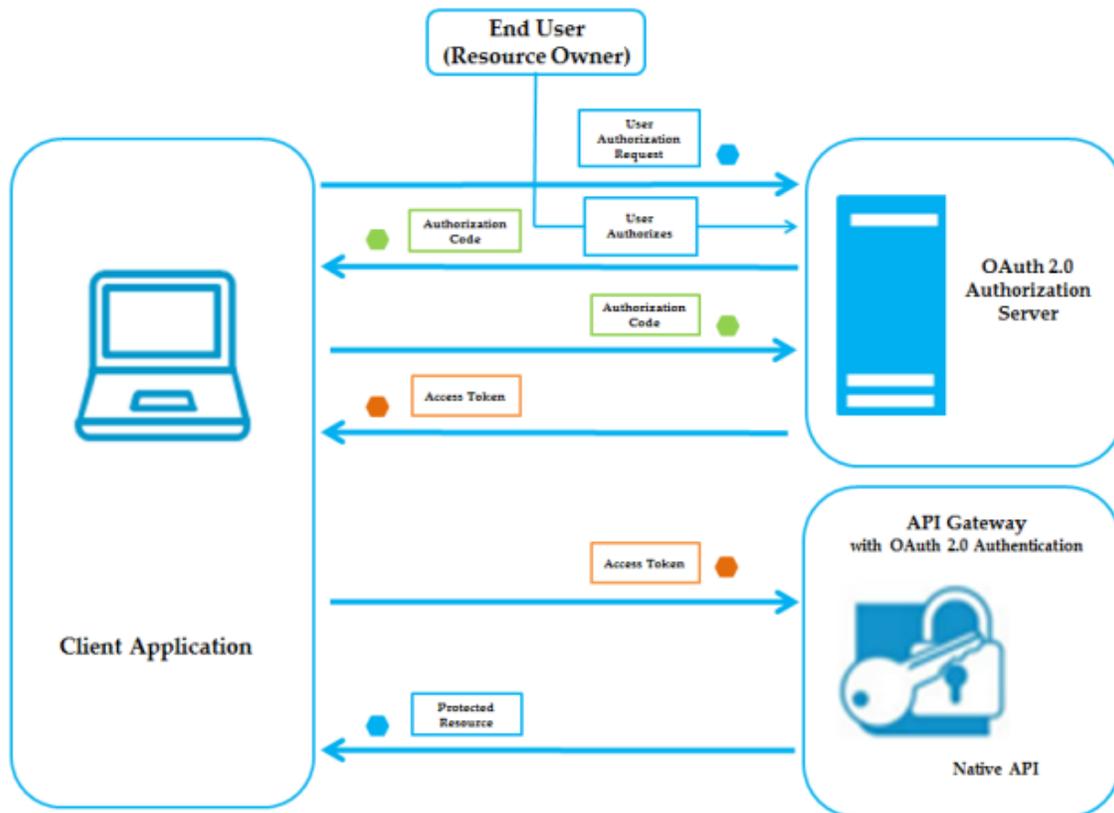
Authorization Code Grant Flow

The authorization code grant type is used to authenticate and provide access to clients that have credentials on the authorization server. This grant type requires the client to authenticate to the authorization server before obtaining an access token.

You use this type of grant to authenticate and provide access to confidential clients.

When using the authorization code grant type, the authorization server can optionally issue a *refresh token* to the client application along with the access token. A refresh token enables clients to get a new access token without requesting additional approval from the resource owner. When the access token expires, the client application can pass the refresh token to the authorization server to request a new access token.

Authorization Code Grant Flow



The Authorization Code Grant OAuth flow has the following steps:

1. Client application initiates the flow to authorize endpoint to request access to end user's data.
2. Authorization server validates the request and informs the user that the client is requesting access within a specified scope. The user then approves or denies the request.
3. When the user approves the request, authorization server generates an authorization code for the client. Authorization server transmits this authorization code through the redirection URI provided by client.

If the user denies the request, authorization server returns an error response.

4. The client's redirection URI passes the authorization code to the token endpoint to exchange the authorization code for an access token.

When making the request, the client authenticates with authorization server. The client includes the redirection URI that was used to obtain the authorization code for verification.

5. Authorization server authenticates the client, validates the authorization code, and ensures that the redirection URI received matches the URI used to redirect the client.

If the access token request is valid and authorized, authorization server responds with an access token and, optionally, a refresh token.

If the request is invalid, authorization server returns an error response.

6. Client uses this access token to send HTTP requests to API Gateway. The access token is validated by authorization server.
7. API Gateway checks with authorization server to make sure the requested API is within the scope for which the access token was issued, and whether the client is authorized to access the resources in the scope.
8. If the access token is expired, authorization server returns a specific error response.

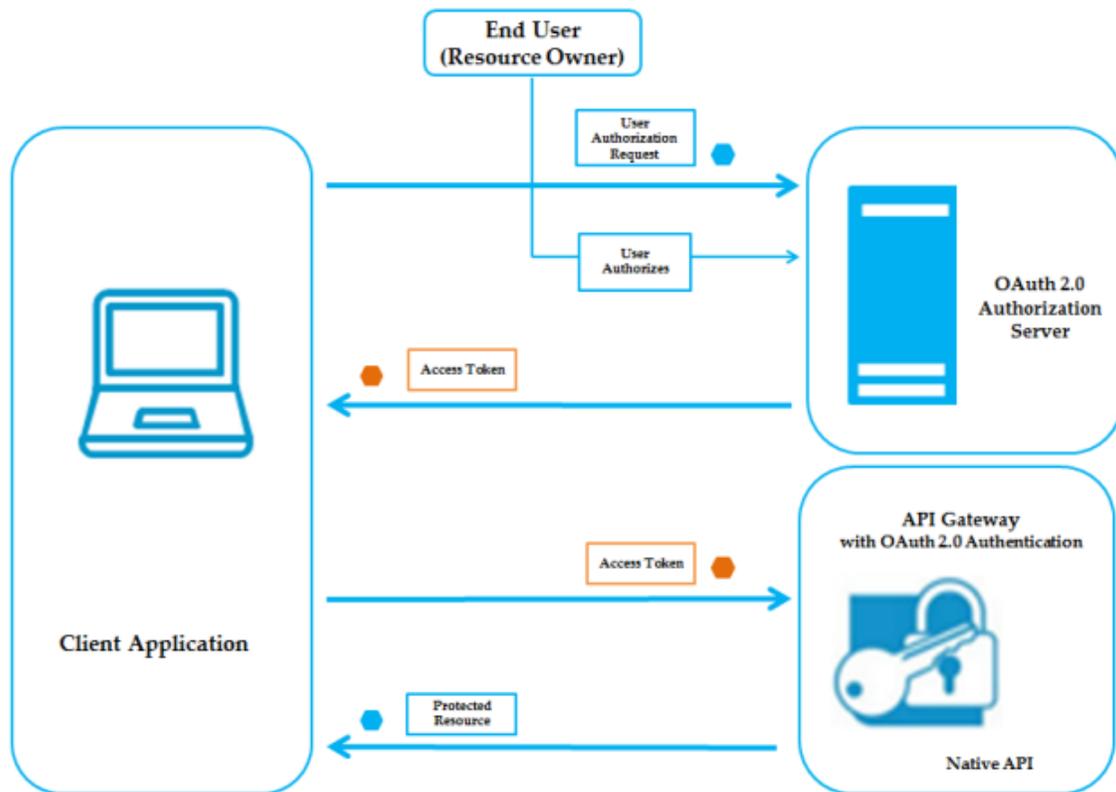
Implicit Grant Flow

The implicit grant type is used to authenticate browser-based applications and mobile applications. This grant type is less secure than the authorization code grant because it does not require the client to authenticate with the authorization server. The authorization server grants *any* client with a valid client ID an access token. In addition, the implicit grant type passes the access token through the resource owner's browser, exposing it to theft by malicious applications on the resource owner's device.

You use this type of grant to authenticate and provide access to public clients.

Note: The implicit grant type does not support refresh tokens.

Implicit Grant Flow



The Implicit Grant OAuth flow has the following steps:

1. Client application initiates the flow to authorize endpoint to request access end user's data.
2. Authorization server validates the request and informs the user that the client is requesting access within a specified scope. The user then approves or denies the request.
3. When the user approves the request, authorization server generates an access token for the client. Authorization server transmits the access token through the redirection URI provided by client.

If the user denies the request, authorization server returns an error response.

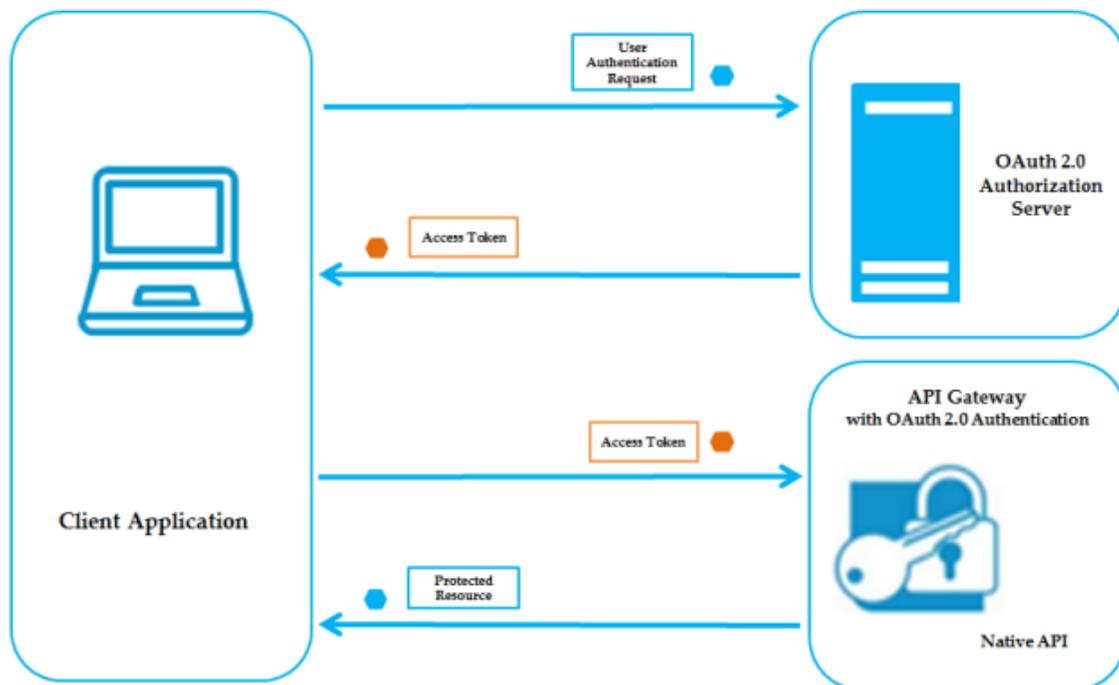
4. Client uses this access token to send HTTP requests to API Gateway. The access token is validated by authorization server.
5. API Gateway checks with authorization server to make sure the requested API is within the scope for which the access token was issued, and whether the client is authorized to access the resources in the scope.
6. If the access token is expired, authorization server returns a specific error response.

Client Credentials Flow

The client credentials grant type is used to authenticate and provide access to clients using their own credentials.

You use this type of grant for fully trusted clients since it allows them to authenticate and get access token for any user. You can use the client credentials grant for confidential clients only.

Client Credentials Grant Flow



The Client Credentials Grant OAuth flow has the following steps:

1. The client application initiates the flow to token endpoint to request access to end user's data.
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. The access token is validated by authorization server.
4. API Gateway checks with authorization server to make sure the requested API is within the scope for which the access token was issued and whether the client is authorized to access the resources in the scope.
5. If the access token is expired, authorization server returns a specific error response.

Configuring API Gateway to Use OAuth 2.0

API Gateway, by default, acts as an OAuth 2.0 resource server as well as authorization server.

- **API Gateway as a Resource Server** receives requests from client applications that include an access token. The resource server asks the authorization server to validate the access token. If the token is valid to access the APIs, the resource server executes the request. The resource server and the authorization server might be in the same API Gateway instance or might be in different API Gateway instances.
- **API Gateway as an OAuth 2.0 Authorization Server** receives authorization requests from client applications.

When API Gateway acts as an authorization server, it handles the interactions between the client application, resource server, and resource owner for approval of the request. The authorization server issues opaque access tokens as *bearer tokens*. A bearer token is an access token that allows any party in possession of the access token (Bearer) to use the token. The authorization server retains the information about the bearer tokens it issues, including the user information. When the client presents a bearer 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.

If the user is authorized to access the APIs (resources), the resource server executes the request. If the user does not have permissions to access the resources, the resource server rejects the request.

You can configure API Gateway to use a third-party OAuth 2.0 provider.

API Gateway provides out-of-the-box integration with the following third-party OAuth 2.0 authorization servers:

- **PingFederate**
- **OKTA**

In addition, you can define any third-party OAuth 2.0 provider (other than the predefined OAuth 2.0 providers shipped with API Gateway).

Adding OAuth 2.0 Partner Provider

Pre-requisites:

You must have the API Gateway's manage security configurations functional privilege assigned to add a partner provider.

The OAuth 2.0 configuration in API Gateway is split into two sections - **Partner providers** and **Authorization servers**.

The **Partner providers** section includes provider-specific information for API Gateway to communicate with the OAuth 2.0 provider.

To add an OAuth 2.0 partner provider

1. Select **Username > Administration**.
2. Select **Security > OAuth 2.0**.
3. In the Partner providers section, click **Add provider**.
4. Provide the following information:

| Field | Description |
|-------------|---|
| Name | <p>Name of a third-party OAuth 2.0 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"> ■ PingFederate ■ OKTA |

Scope management ESB service

Name of an Enterprise Service Bus (ESB) service that creates and manages the scopes in the OAuth 2.0 authorization server.

You might use the **Scope management ESB service** field when adding a third-party OAuth 2.0 provider that is not shipped with API Gateway. The OAuth 2.0 scope management does not have any predefined standards, and the implementation for managing the OAuth 2.0 scopes could be varying with different providers. API Gateway uses the ESB service to create and manage the OAuth 2.0 scopes in the configured OAuth 2.0 authorization server.

API Gateway would send a specific set of parameters to the ESB service for the scope management. For details on the parameters for an ESB service, see below.

Client metadata field mapping. Specifies the mapping of OAuth 2.0 dynamic client registration specification to that of the client implementation of the OAuth 2.0 provider.

The **Client metadata field mapping** fields are required when you are adding a third-party provider that is not shipped with API Gateway.

| Field | Description |
|---------------------------|--|
| Specification name | <p>The client metadata attributes in accordance with the OAuth 2.0 Dynamic Client Registration specification as defined in RFC 7591.</p> <hr/> <p>redirect_uris. Redirection URL that the OAuth 2.0 authorization server uses to redirect the authorization code once the authorization request is approved by end user.</p> <p>Note: If you do not specify this attribute, API Gateway automatically generates the URL.</p> <hr/> <p>token_endpoint_auth_method. The OAuth 2.0 client authentication method at the token endpoint.</p> <hr/> <p>grant_types. The grant type of OAuth 2.0 authorization flow to obtain authorization codes, ID tokens, and refresh tokens.</p> <hr/> <p>response_types. The type of response that the client application uses at the OAuth 2.0 authorization endpoint.</p> <hr/> <p>client_name. Name of the client to use to represent the client application to the end user during authorization.</p> <hr/> <p>client_uri. URL of the client application.</p> <hr/> <p>logo_uri. URL of an image to use to represent the client application to the end user during authorization.</p> <p>Note: The logo_uri is currently not supported in API Gateway.</p> <hr/> <p>scope. List of user-authorized scopes that the client uses for requesting access tokens.</p> <p>Note: If you do not specify this attribute, the authorization server registers the client with a default set of scopes.</p> <hr/> |

| Field | Description |
|----------------------------|---|
| | <p>contacts. The means (for example, Email address) by which end users can contact the client for support requests.</p> |
| | <p>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> <p>Note: The tos_uri is currently not supported in API Gateway.</p> |
| | <p>jwt_uri. URL of the JSON Web Key (JWK) Set document containing the client's public keys.</p> <p>Note: The jwt_uri is currently not supported in API Gateway.</p> |
| | <p>client_id. Identifier that is unique to the client application.</p> |
| | <p>client_secret. The password or phrase for the client application to use to authorize communication with the end user.</p> |
| Implementation name | <p>The client metadata attributes that are used by the OAuth 2.0 authorization server, but are not in accordance with the OAuth 2.0 Dynamic Client Registration specification.</p> <p>Example:</p> <ul style="list-style-type: none"> ■ For the redirect_uris field, provide the value <i>redirectUris</i>. ■ For the grant_types field, provide the value <i>grantTypes</i>. ■ For the client_name field, provide the value <i>name</i>. ■ For the logo_uri field, provide the value <i>logoUrl</i>. ■ For the client_id field, provide the value <i>clientId</i>. ■ For the client_secret field, provide the value <i>secret</i>. |

| Field | Description |
|--------------|---|
| | <p>Extended request parameters. Specifies the additional client metadata attributes that are specific to the OAuth 2.0 authorization server, and are not specified in the OAuth 2.0 Dynamic Client Registration specification.</p> <p>In PingFederate (For example):</p> <pre>forceSecretChange = true</pre> |
| Key | The client metadata attribute that is specific to the OAuth 2.0 authorization server. |
| Value | A value for the client metadata attribute. When sending requests to the authorization server, this value is appended to all requests. |

- Click **Save**.

The OAuth 2.0 partner provider is added. You can add as many partner providers as required.

Parameters for ESB Service

API Gateway would send the following configuration parameters to an ESB service for scope management:

| Parameter | Description |
|-------------------------------|---|
| <code>action</code> | <p>Specifies the action you want to perform on an OAuth 2.0 scope.</p> <p>Possible values are: <code>create</code> and <code>delete</code>.</p> |
| <code>scopeName</code> | Specifies the name of an OAuth 2.0 scope that is to be created in OAuth 2.0 authorization server. |
| <code>scopeDescription</code> | Specifies the description of an OAuth 2.0 scope that is to be created in OAuth 2.0 authorization server. |
| <code>providerName</code> | Specifies the name of an OAuth 2.0 provider. |
| <code>endpoints</code> | Specifies the Endpoint URLs to create, update, and delete OAuth 2.0 scopes. |
| <code>endpoints</code> | Specifies the Endpoint URL details for OAuth 2.0 scope management. |

| Parameter | Description | | | | | | | | |
|--------------------------------|--|-----------|-------------|-----------------------|---|----------------------|--|--------------------|---|
| | <table border="1"> <thead> <tr> <th>Parameter</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>endpoint</code></td> <td>Specifies an Endpoint URL to create, update, and delete an OAuth 2.0 scope.</td> </tr> <tr> <td><code>headers</code></td> <td>Specifies one or more HTTP authorization headers that API Gateway would use to send OAuth 2.0 scope details to OAuth 2.0 authorization server.</td> </tr> <tr> <td><code>https</code></td> <td> <p>Specifies the transport protocol that API Gateway would use to send OAuth 2.0 scope details to OAuth 2.0 authorization server.</p> <p>Possible values are: <code>true</code> and <code>false</code>.</p> <p>A value of <code>true</code> allows API Gateway send OAuth 2.0 scope details to OAuth 2.0 authorization server using the HTTP protocol.</p> <p>A value of <code>false</code> allows API Gateway send OAuth 2.0 scope details to OAuth 2.0 authorization server using the HTTPS protocol.</p> </td> </tr> </tbody> </table> | Parameter | Description | <code>endpoint</code> | Specifies an Endpoint URL to create, update, and delete an OAuth 2.0 scope. | <code>headers</code> | Specifies one or more HTTP authorization headers that API Gateway would use to send OAuth 2.0 scope details to OAuth 2.0 authorization server. | <code>https</code> | <p>Specifies the transport protocol that API Gateway would use to send OAuth 2.0 scope details to OAuth 2.0 authorization server.</p> <p>Possible values are: <code>true</code> and <code>false</code>.</p> <p>A value of <code>true</code> allows API Gateway send OAuth 2.0 scope details to OAuth 2.0 authorization server using the HTTP protocol.</p> <p>A value of <code>false</code> allows API Gateway send OAuth 2.0 scope details to OAuth 2.0 authorization server using the HTTPS protocol.</p> |
| Parameter | Description | | | | | | | | |
| <code>endpoint</code> | Specifies an Endpoint URL to create, update, and delete an OAuth 2.0 scope. | | | | | | | | |
| <code>headers</code> | Specifies one or more HTTP authorization headers that API Gateway would use to send OAuth 2.0 scope details to OAuth 2.0 authorization server. | | | | | | | | |
| <code>https</code> | <p>Specifies the transport protocol that API Gateway would use to send OAuth 2.0 scope details to OAuth 2.0 authorization server.</p> <p>Possible values are: <code>true</code> and <code>false</code>.</p> <p>A value of <code>true</code> allows API Gateway send OAuth 2.0 scope details to OAuth 2.0 authorization server using the HTTP protocol.</p> <p>A value of <code>false</code> allows API Gateway send OAuth 2.0 scope details to OAuth 2.0 authorization server using the HTTPS protocol.</p> | | | | | | | | |
| <code>keyStoreAlias</code> | Specifies the alias of keystore containing the private key to secure communication between API Gateway and OAuth 2.0 authorization server. | | | | | | | | |
| <code>keyAlias</code> | Specifies the alias for private key to validate the HTTP requests from client applications. | | | | | | | | |
| <code>authorizationInfo</code> | Specifies the alias of OAuth 2.0 authorization server. | | | | | | | | |
| | <table border="1"> <thead> <tr> <th>Parameter</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>type</code></td> <td>Specifies the type of authentication scheme that API Gateway would use to communicate</td> </tr> </tbody> </table> | Parameter | Description | <code>type</code> | Specifies the type of authentication scheme that API Gateway would use to communicate | | | | |
| Parameter | Description | | | | | | | | |
| <code>type</code> | Specifies the type of authentication scheme that API Gateway would use to communicate | | | | | | | | |

| Parameter | Description |
|------------------------|---|
| | with the OAuth 2.0 authorization server for OAuth 2.0 scope management. Possible values are: <code>basic</code> and <code>token</code> . |
| <code>username</code> | Specifies the username to access the protected resources. |
| <code>password</code> | Specifies a valid password associated with the username. |
| <code>tokenType</code> | Specifies the type of token that would be contained in the HTTP request. |
| <code>token</code> | Specifies the token that would be contained in the HTTP request. |

Adding OAuth 2.0 Authorization Server

Pre-requisites:

You must have the API Gateway's manage security configurations functional privilege assigned to add an authorization server.

The OAuth 2.0 configuration in API Gateway is split into two sections - **Partner providers** and **Authorization servers**.

The **Authorization servers** section provides information on the OAuth 2.0 endpoint URLs, such as metadata URLs, dynamic client registration endpoint URLs, and scope management URLs, SSL configuration, and user authentication (HTTP basic or Bearer token).

To add an OAuth 2.0 authorization server

1. Select **Username > Administration**.
2. Select **Security > OAuth 2.0**.
3. In the Authorization servers section, click **Add authorization server**.
4. Provide the following information:

| Field | Description |
|-----------------------------------|------------------------------------|
| Authorization server alias | Alias of the authorization server. |

| Field | Description |
|-------|-------------|
|-------|-------------|

| | |
|----------------------|---|
| Provider name | Name of a third-party OAuth 2.0 provider. |
|----------------------|---|

5. In the Endpoints section, click the chevron to expand and collapse the OAuth 2.0 endpoint configuration. Click **Add endpoints**. Provide the following information:

| Field | Description |
|-------|-------------|
|-------|-------------|

| | |
|---------------------|---|
| Use keystore | Select Use keystore to allow client authentication only over a two-way SSL connection. |
|---------------------|---|

| | |
|-------------|---|
| Type | <p>Select one or more Endpoint URL types to create, update, and delete client applications and scopes.</p> <ul style="list-style-type: none"> ■ Client read: Fetches the details of a client application specified by the <code>clientId</code>. ■ Client registration: Registers a client application in the authorization server. ■ Client update: Updates the configuration of a client application specified by the <code>clientId</code>. ■ Client delete: Deletes a client application specified by the <code>clientId</code>. ■ Scope read: Fetches the details of a scope specified by the <code>scopeId</code>. ■ Scope create: Creates a scope for the authorization server. ■ Scope update: Updates the configuration of a scope specified by the <code>scopeId</code>. ■ Scope delete: Deletes a scope specified by the <code>scopeId</code>. |
|-------------|---|

For example, to update an OAuth client application or an OAuth scope, the endpoint URL should be specified as:

```
PUT /oauth2/v1/clients/:clientId
PUT /oauth2/v1/scopes/:scopeId
```

In the above endpoint URLs, the `clientId` and `scopeId` should be specified in a set of curly braces.

```
PUT /oauth2/v1/clients/{client_id}
PUT /oauth2/v1/scopes/{scope_id}
```

| | |
|------------|---|
| URL | Specifies the corresponding REST endpoint URLs for the client configuration and scope configuration of REST APIs. |
|------------|---|

| Field | Description |
|----------------|--|
| Headers | Specifies the authorization header that API Gateway should send to the OAuth 2.0 authorization server. |
| Key | The HTTP header key that should be included in the authorization header of API requests. |
| Value | The HTTP header value that should be included in the authorization header of API requests. |

6. In the Authentication section, click the chevron to expand and collapse the OAuth 2.0 authentication scheme configuration. Provide the following information:

| Field | Description |
|-------------|--|
| Type | Specifies the type of authentication scheme that API Gateway would use to communicate with the OAuth 2.0 authorization server for client and scope management. |
| | 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. |
| | 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. |

7. In the SSL Configuration section, click the chevron to expand and collapse the OAuth 2.0 SSL configuration. Provide the following information:

| Field | Description |
|-----------------------|---|
| Keystore alias | <p>Alias of the keystore containing the private key that is used for a secured communication between API Gateway and OAuth 2.0 authorization server.</p> <p>The Keystore alias box lists all the keystore aliases available in API Gateway. If there are no configured keystore aliases, this box lists the default Integration Server keystore, <i>DEFAULT_IS_KEYSTORE</i>.</p> |
| Key alias | <p>Alias for the private key to use to validate the HTTP requests from the client.</p> <p>The Key alias box is auto-populated and lists all the aliases available in the selected keystore. If there are no configured keystores, this list box is empty.</p> |

- In the Metadata section, click the chevron to expand and collapse the OAuth 2.0 token configuration. Provide the following information:

| 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. |

- Click **Save**.

The OAuth 2.0 authorization server is added. You can add as many authorization servers as required, but only one is the default at any given time.

Viewing OAuth 2.0 Partner Provider List and Provider Configuration

You can view the list of integrated OAuth 2.0 third-party providers, delete a provider, and reset the provider to its default configuration in the Partner providers section. In addition, you can view and modify the OAuth 2.0 provider configuration in the individual OAuth 2.0 provider details page.

To view a list of OAuth 2.0 providers and provider configuration

1. Select **Username > Administration**.
2. Select **Security > OAuth 2.0**.

The Partner providers section displays a list of all the currently defined third-party OAuth 2.0 providers in API Gateway.

You can also perform the following operations in the Partner providers section by clicking the appropriate icons in the Action column.

- **Reset to system defaults:** Deletes the current configuration and restore the system configuration for an OAuth 2.0 provider.
 - **Delete:** Deletes an OAuth 2.0 provider from API Gateway.
3. Click any OAuth 2.0 provider to view the configuration details.

The OAuth 2.0 provider details page displays the client metadata attributes defined in accordance with the OAuth 2.0 specification, and the corresponding attributes defined in the displayed provider.

Viewing OAuth 2.0 Authorization Server List and Server Configuration

You can view the list of OAuth 2.0 authorization servers, delete or test an authorization server, and set a specific authorization server as default in the Authorization servers section. In addition, you can view and modify the OAuth 2.0 authorization server details in the individual OAuth 2.0 authorization server details page.

To view a list of OAuth 2.0 authorization servers and server configuration

1. Select **Username > Administration**.
2. Select **Security > OAuth 2.0**.

The Authorization servers section displays a list of all the currently defined third-party OAuth 2.0 authorization servers in API Gateway.

You can also perform the following operations in the Authorization servers section by clicking the appropriate icons in the Action column.

- **Set as default:** Sets an authorization server to use, by default, when there is more than one server defined in API Gateway.

You can change the default setting of an authorization server at any time.

- **Test:** Checks if the configuration of a specific authorization server is valid.

When API Gateway executes functional testing of the authorization server, it internally creates OAuth 2.0 clients and scopes in the server, performs operations such as, update and delete, on the created OAuth 2.0 clients and scopes, and reports if the authorization server is available for OAuth 2.0 token validation.

If the authorization server is not available for OAuth 2.0 token validation, API Gateway issues a specific error message.

- **Delete:** Deletes an authorization server from API Gateway.

3. Click any OAuth 2.0 authorization server to view the configuration details.

The OAuth 2.0 authorization server details page the client information, scope information, and token information of the displayed OAuth 2.0 authorization server.

Setting Default OAuth 2.0 Server

Pre-requisites:

You must have the API Gateway's manage security configurations functional privilege assigned to set a default OAuth 2.0 server.

A default OAuth 2.0 server is the server that you want to use when there is more than one server defined in API Gateway.

Note: If you revoke the default setting of an OAuth 2.0 server, API Gateway defaults to the local authorization server configured in Integration Server.

To set a default OAuth 2.0 authorization server

1. Select **Username > Administration**.
2. Select **Security > OAuth 2.0**.

The Authorization servers section displays a list of all the currently defined third-party OAuth 2.0 authorization servers in API Gateway.

3. Click **Set as default** in the action column of the authorization server to set as the default OAuth 2.0 server.

You can change the default OAuth 2.0 server setting at any time.

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

The OAuth2.0 server is set as the default authorization server in API Gateway.

Modifying OAuth2.0 Partner Provider Configuration

Pre-requisites:

You must have the API Gateway's manage security configurations functional privilege assigned to modify a partner provider.

You might want to modify an OAuth 2.0 partner provider to change the currently defined configuration settings.

To modify an OAuth2.0 partner provider configuration

1. Select **Username > Administration**.

2. Select **Security > OAuth 2.0**.

The Partner providers section displays a list of all the currently defined third-party OAuth 2.0 partner 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 OAuth2.0 partner provider is updated.

Modifying OAuth2.0 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 an OAuth2.0 authorization server configuration

1. Select **Username > Administration**.
2. Select **Security > OAuth 2.0**.

The Authorization servers section displays a list of all the currently defined third-party OAuth 2.0 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 OAuth2.0 authorization server is updated.

Deleting OAuth2.0 Partner Provider

Pre-requisites:

You must have the API Gateway's manage security configurations functional privilege assigned to delete a partner provider.

You delete an OAuth2.0 partner provider to remove it from API Gateway permanently.

Important: You must not delete an OAuth 2.0 provider if it is being used by an authorization server.

To delete an OAuth2.0 partner provider

1. Select **Username > Administration**.

2. Select **Security > OAuth 2.0**.

The Partner providers section displays a list of all the currently defined third-party OAuth 2.0 partner providers in API Gateway.

3. Click  in the action column of the partner provider to be deleted.
4. Click **Yes** in the confirmation dialog.

The OAuth2.0 partner provider is deleted from API Gateway.

Deleting OAuth2.0 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 OAuth2.0 server to remove it from API Gateway permanently.

Note: If there are no OAuth2.0 servers configured, API Gateway defaults to the local authorization server configured in Integration Server.

To delete an OAuth2.0 authorization server

1. Select **Username > Administration**.
2. Select **Security > OAuth 2.0**.

The Authorization servers section displays a list of all the currently defined third-party OAuth 2.0 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 OAuth2.0 authorization server is deleted from API Gateway.

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.

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. Select **Username > 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 |

| Field | Description |
|-------|--|
| | the user or service being authenticated in the JAAS login context. |

- Click **Ok**.

JWT

JSON Web Token (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.

Note: JWT authentication is supported for both REST and SOAP APIs.

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 uses the RSA-based JWS to provide stronger integrity protection to JWTs.

API Gateway can generate a JWT token itself or validate the JWT token generated by a trusted third party server.

Structure of JSON Web Token

A JSON Web Token (JWT) consists of three components. These three components are Base64Url encoded and separated by dots. The JWT can be easily passed in the HTTP Authorization header.

Sample JSON Web Token:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJsb2dnZWRJbkFzIjoieYWRtaW4iLCJpYXQiOiE0MjI3Nzk2Mzh9.gzSraSYS8EXBxLN_oWnFSRgCzcmJmMjLiuyy5CSpyHI
```

JWT Header

Contains the type of token, JWT, and the hashing algorithm used, RSA:

Token Value Encoded:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
```

Token Value Decoded:

```
{
  "typ": "JWT",
  "alg": "RS256",
}
```

JWT Payload

Contains the claims relating to the authentication and identification of the client or the user:

Token Value Encoded:

```
eyJsb2dnZWRJbkFzIjoiYWRtaW4iLCJpYXQiOjE0MjI3Nzk2Mzh9
```

Token Value Decoded:

```
{
  "iss": "John Chris",
  "sub": "Administrator",
  "aud": "SAG",
  "nbf": 1503039579,
  "exp": 1503041379,
  "iat": 1503039579
}
```

The following claims in the JWT token specify the unique identifying information for the JWT client:

| Claim | Description |
|-------|--|
| iss | Issuer of the JWT. In API Gateway, this claim represents the Token issuer field in the JWT configuration page. |
| sub | Subject of the JWT. This is the username of the end user. |
| aud | Audience for the JWT. This must match the client_id of the application. In API Gateway, this claim represents the Audience field in the JWT configuration page. |
| nbf | Time before which the JWT must not be accepted for processing. In API Gateway, this claim indicates the time at which the user has requested for JWT. |
| exp | Time on or after which the JWT is set to expire. In API Gateway, this claim indicates the current time plus that expiry duration represented by the Expiry duration field in the JWT configuration page. |
| iat | Timestamp when the JWT was issued. |

JWT Digital Signature

Contains the Base64url encoded JWS that is constructed using the JWT Header and Payload.

Token Value Encoded:

```
gzSraSYS8EXBxLN_oWnFSRgCzcmJmMjLiuyu5CSpyHI
```

Token Value Decoded:

```
RSA256 (
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload) ,
secret)
```

JWT Workflow

The JSON Web Token (JWT) flow has the following steps:

1. Client application sends an authentication request with required user credentials like username and password, to API Gateway or to any third-party JWT issuer, to obtain a JWT token.

The JSON Web Token endpoint in API Gateway is:

```
<hostname>:<port>/rest/pub/apigateway/jwt/getJsonWebToken
```

2. If API Gateway is the JWT issuer, then it validates the user credentials. If the user 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.

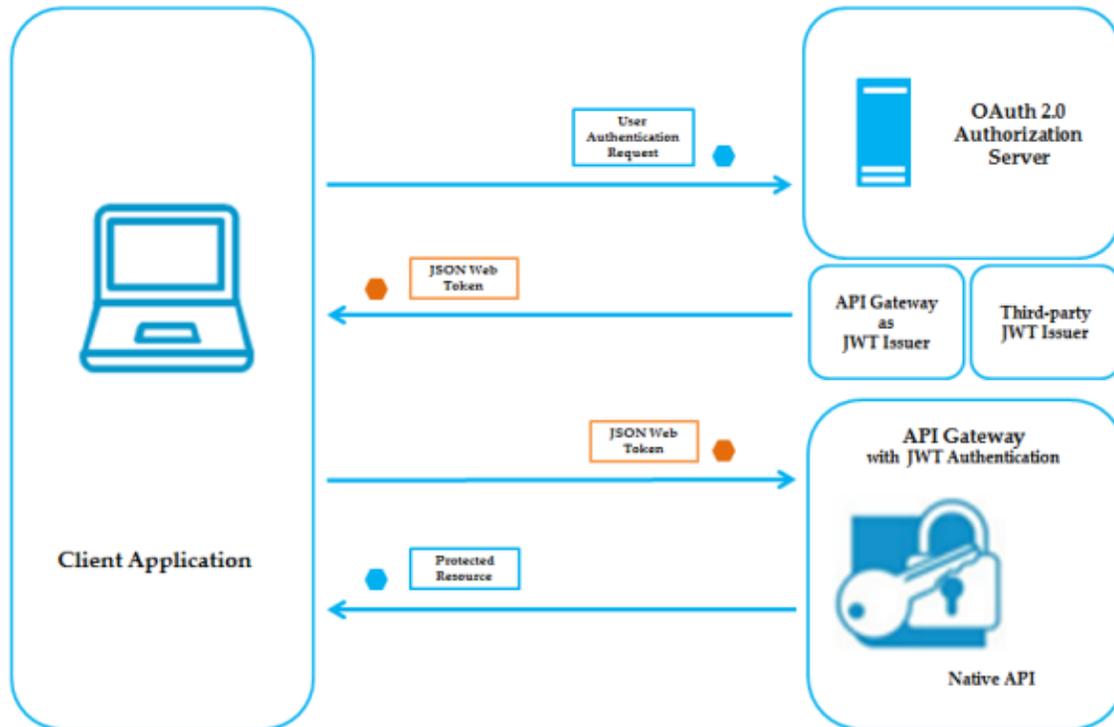
3. Client sends the generated JSON token in the HTTP Authorization request header to access the protected API in API Gateway.
4. API Gateway validates the JSON token using the same private key. If validation of the JSON token succeeds, API Gateway extracts the unique identifying information of the client that are represented as claims, identifies the client, and provides access to the secured API.

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 that was configured for that issuer in Integration Server.

The following diagram illustrates how API Gateway participates in the JWT workflow.

JSON Web Token Flow



Configuring API Gateway as JWT Issuer

Pre-requisites:

You must have the API Gateway's manage security configurations functional privilege assigned to configure API Gateway as a JWT Issuer.

You can configure API Gateway to generate JSON Web Tokens (JWT) as access tokens for authenticating and securing API requests. API Gateway provides support for using JWT authentication in both the inbound and outbound HTTP requests at the transport level.

To configure API Gateway as JWT issuer

1. Select **Username > Administration**.
2. Select **Security > JWT**.
3. Click **Edit**.
4. Provide or modify 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 the contents of JWT. 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 | Optional. 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. |
| Keystore alias | Alias of the keystore containing the private key that is used to sign the contents of JWT. The Keystore alias box lists all the keystore aliases available in API Gateway. If there are no configured keystore aliases, this box lists the default Integration Server keystore, <code>DEFAULT_IS_KEYSTORE</code> . |
| Key alias | Alias of the private key used to sign the contents of JWT. The Key alias box is auto-populated and lists all the aliases available in the selected keystore. If there are no configured keystores, this list box is empty. |

- Click **Save**.

Configuring API Gateway as JWT Relying Party

API Gateway can also validate the JWT token generated by a trusted third party server.

You must have the API Gateway's manage security configurations functional privilege assigned to configure API Gateway as a relying party.

To enable API Gateway to validate a trusted third-party's JWT, complete the following settings in Integration Server (in the Integration Server Administrator, go to **Security > JWT**).

- The trusted third-party issuer.

- Public certificate of the third-party issuer.

For detailed procedures, see *webMethods Integration Server Administrator's Guide*.

Identifying Applications Using JWT

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, and identifies the application that is using the token to access a protected resource. An application that is defined with the claims conveyed by the JWT is identified and allowed access to the protected resource.

For details on how to define JWT claims in an application, see ["Creating an Application" on page 311](#).

OpenID Provider

OpenID Connect is an open standard and decentralized authentication protocol that extends on the OAuth 2.0 authorization framework.

OpenID Connect allows the clients to:

- Obtain an OpenID Connect (ID) token to verify the identity, authenticate, and authorize an end user with an authorization server or identity provider.
- Use an UserInfo endpoint to retrieve basic profile information, preferences, and other user-specific information.

The OpenID Connect uses JWT as the format of the ID token for exchanging identity and user profile data. The ID token contains a set of claims which contain the information relating to an end user or an application. 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 an ID token are encoded as a JSON object and used as the payload of a JSON Web Signature (JWS) structure. JWTs can be signed using a shared secret (with HMAC algorithm), or a public or private key pair using RSA. API Gateway uses the RSA-based JWS to provide stronger integrity protection to JWTs.

Important: API Gateway supports OpenID Connect Discovery 1.0.

Structure of OpenID Token

An ID (`id_token`) token in the JSON Web Token (JWT) format consists of three components. These three components are Base64Url encoded and separated by dots. The ID token can be easily passed in the URL or HTTP Authorization header.

Sample ID Token:

```
eyJhbGciOiJSUzEzIiwiaWF0IjoiMTUxMjM0NTY3OTkifQ.eyJhenAiOiIzODk4MDEwNTQyMTAtNHBhOWJGEzZmYjFhNjUxZTMwNjcwOTkiLCJpc0kiOiJ1IiwiaXNjaWQiOiJ1IiwiaWF0IjoiMTUxMjM0NTY3OTkifQ.
```



```

"email_verified": true,
"at_hash": "YcO_BaJz1NrosAhUOMhydQ",
"nonce": "4bd473c3-aad4-4399-b88f-5981c8e3994e",
"auth_time": 1503298187,
"amr": [
  "rba"
],
"iss": "https://accounts.google.com",
"iat": 1503298188,
"exp": 1503301788,
"name": "John Chris",
"picture": "jc.jpg",
"given_name": "John",
"family_name": "Chris",
"locale": "en"
}

```

The following claims in the ID token specifies the unique identifying information for the OpenID client:

| Claim | Description |
|----------------|--|
| azp | The client_id of the application to which the ID token was issued. This claim value is a case-sensitive. |
| aud | Audience for the ID token. |
| sub | Username of the end user. |
| email | Email address of the user. |
| email_verified | Denotes if the email address of the user has been verified. |
| at_hash | Denotes if the email address of the user has been verified. |
| nonce | This associates a client session with an ID Token. |
| auth_time | The time at which the user authentication occurred. |
| amr | The authentication method that was used for user authentication. |
| iss | Issuer of the ID token. |
| iat | The timestamp when the ID token was issued. |
| exp | The time on or after which the ID token is set to expire. |

| Claim | Description |
|-------------|---|
| name | Full name of the user. |
| picture | The profile picture of the user. |
| given_name | First name of the user. |
| family_name | Surname or last name of the user. |
| locale | The display language of the user interface for user authentication. |

Digital Signature

Contains the Base64url encoded JWS that is constructed using the Header and Payload.

Token Value Encoded:

```
pG16c17-BGjDsXL7daoolMyxh1fE2cGIBWGIxonsKBosb1Qx7uGqEmH
PGQdB0cWTO6L0nQQ9mVUW-0BvQ6QYFD7QP8MilhGLNsF8bdgBS_-RL6bJqDGdXmAD
B81pe5SjSa8bG91g82y-clFGQpyl7JTRztsMewlxya2VuT5m9aJMSavPFyG6JfU9t
dnqkHxQlLTVoelbecJWfpFuudY58e5wz42Pjmuz5IP_TJRdoK7TbeQ7PrP2Kl18i1
8fsu4I0R2-IxR5u3YCbGGihMhSrJG2pTULcCrRTUeT5BjcS2GEzd6yjKQvwLn6dbS
ndlicWVzT3dEiR1SuN2xkBhyyjQ
```

Token Value Decoded:

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret)
```

Sample Userinfo Endpoint

The UserInfo endpoint provided, for example, by Google is located at:

```
https://www.googleapis.com/oauth2/v3/userinfo
```

An example HTTP client request to the UserInfo endpoint:

```
GET https://www.googleapis.com/oauth2/v3/userinfo
Authorization: Bearer <access token>
```

A successful response will return a HTTP 200 OK response and the users claims in JSON format:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
  "sub": "johnc",
  "email": "vinola.betsy@gmail.com",
  "email_verified": true,
  "name": "Vinola J",
  "picture": "jc.jpg",
  "given_name": "Vinola",
  "family_name": "J",
  "nickname": "John"
```

```
...[additional claims]...  
}
```

Before the client application can trust the values returned from the UserInfo endpoint, the client must verify that the `sub` claim returned from the UserInfo endpoint request matches the subject from the `id_token`.

Authorization Workflows

The OpenID Connect support in API Gateway provides two different ways for a client to obtain access to a protected resource.

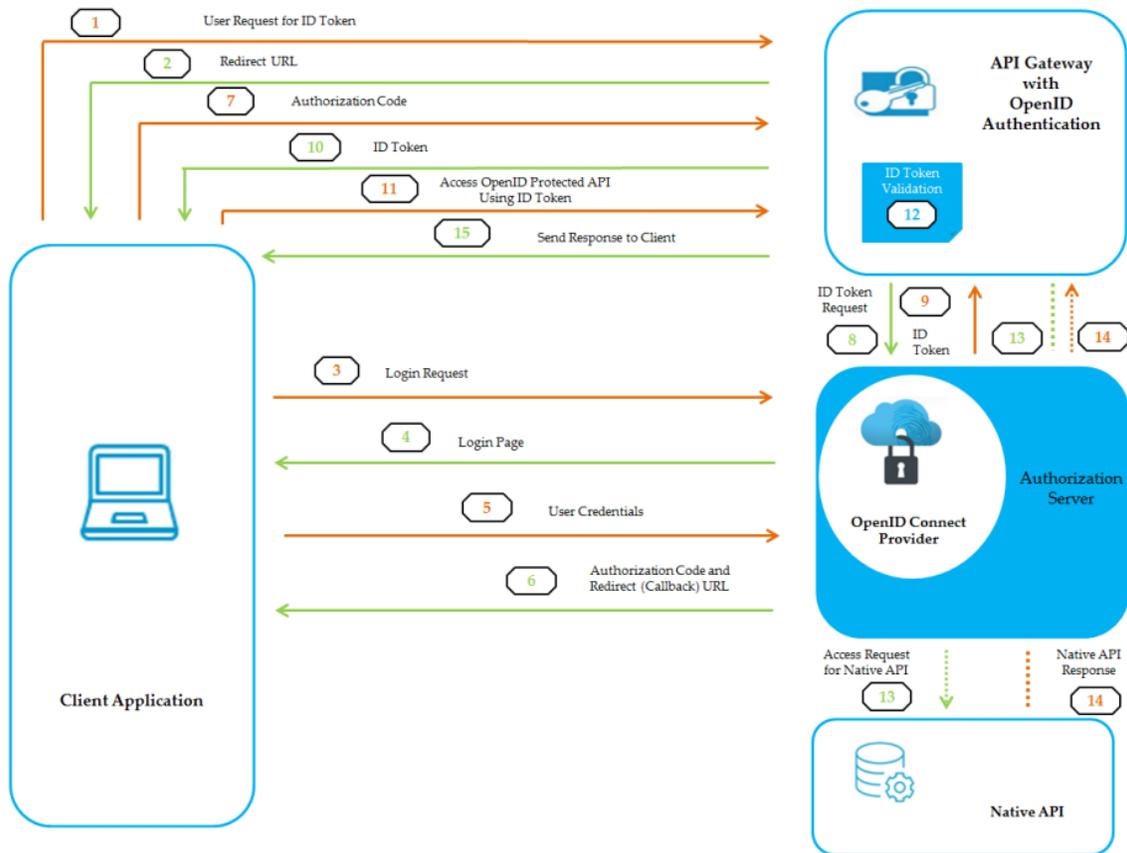
- Authorization Code
- Implicit

Authorization Code Flow

This flow returns an authorization code to API Gateway and the client application, which can then directly exchange it for an identity (`id_token`) token and, optionally, a secondary ID token (which embeds the primary ID token and application ID), and refresh token.

This flow obtains the authorization code from the authorization endpoint and all tokens are returned from the token endpoint.

Authorization Code Grant Flow



The authorization code flow involves the following steps:

1. Client application sends the OpenID token request to API Gateway to obtain an OpenID (ID) token.

The OpenID token endpoint in API Gateway is:

```
<hostname>:<port>/rest/pub/apigateway/openid/getOpenIDToken
```

Note: You can optionally specify an application ID to get an OpenID token specific to the application.

For example, `test:5555/rest/pub/apigateway/openid/getOpenIDToken?app_id=990807e0-b03b-4a25-8f26-5d73fdd6f7d0`. In this case, API Gateway generates a secondary ID token that includes the application ID and the primary ID token. As a pre-requisite, you must have the JWT settings configured in API Gateway. For information about configuring JWT settings in API Gateway, see ["Configuring API Gateway as JWT Issuer" on page 92](#).

2. API Gateway sends the redirect URL of the configured OpenID Provider to client.

3. The redirect URL opens the login page of the OpenID Provider and retrieves the user information for authentication.
4. The OpenID Provider authenticates the user. It passes the authorization code to the client using a preconfigured redirect URL.

The preconfigured redirect URL is the callback URL an API Provider will configure in the OpenID Provider's console.

The callback URL for OpenID in API Gateway is:

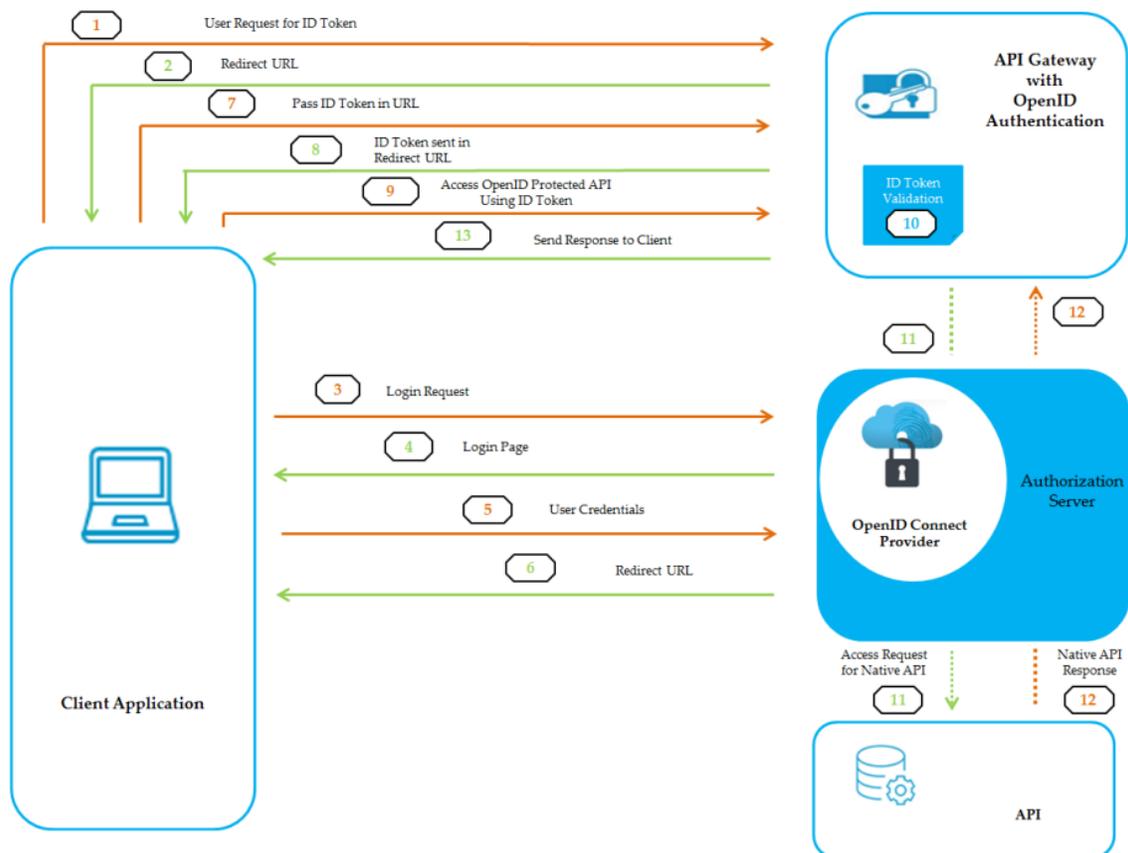
```
<hostname>:<port>/rest/pub/apigateway/openid/openIDCallback
```

5. API Gateway validates the authorization code, and, if valid, returns an ID token to the client.
6. Client uses the ID token to access the OpenID protected API in API Gateway.

Implicit Flow

This flow requests an identity (ID) token and, optionally, a secondary ID token (which embeds the primary ID token and application ID) without an explicit client authentication. This flow uses the redirect URL to verify the client identity.

Implicit Grant Flow



The implicit flow involves the following steps:

1. Client application sends the OpenID token request to API Gateway to obtain an OpenID (ID) token.

The OpenID token endpoint in API Gateway is:

```
<hostname>:<port>/rest/pub/apigateway/openid/getOpenIDToken
```

Note: You can optionally specify an application ID to get an OpenID token specific to the application.

For example, `test:5555/rest/pub/apigateway/openid/getOpenIDToken?app_id=990807e0-b03b-4a25-8f26-5d73fdd6f7d0`. In this case, API Gateway generates a secondary ID token that includes the application ID and the primary ID token. As a pre-requisite, you must have the JWT settings configured in API Gateway. For information about configuring JWT settings in API Gateway, see ["Configuring API Gateway as JWT Issuer" on page 92](#).

2. API Gateway sends the redirect URL of the configured OpenID Provider to client.
3. The redirect URL opens the login page of the OpenID Provider and retrieves the user information for authentication.
4. The OpenID Provider authenticates the user. It passes the ID token to the client to the client using a preconfigured redirect URL.

The preconfigured redirect URL is the callback URL an API Provider will configure in the OpenID Provider's console.

The callback URL for OpenID in API Gateway is:

```
<hostname>:<port>/rest/pub/apigateway/openid/openIDCallback
```

5. Client uses the ID token to access the OpenID protected API in API Gateway.

Identifying Applications Using OpenID Tokens

The JSON-based OpenID (ID) 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 ID token, and identifies the application that is using the token to access a protected resource. An application that is defined with the claims conveyed by the ID token is identified and allowed access to the protected resource.

For details on how to define OpenID claims in an application, see ["Creating an Application" on page 311](#).

Adding an OpenID Provider

Pre-requisites:

You must have the API Gateway's manage security configurations functional privilege assigned to add an OpenID provider.

Before you create an OpenID Connect (OIDC) identity provider in API Gateway, you must register API Gateway with the OpenID provider to receive a client ID and client secret. The client ID and client secret are the unique identifiers that are issued when API Gateway is registered with the OpenID provider. You must also specify a redirect URL that allows the OpenID provider to allow users to authenticate.

You can create and manage an OpenID provider using the OpenID provider configuration screen in API Gateway.

To add an OpenID provider

1. Select **Username > Administration**.
2. Select **Security > OpenID provider**.
3. Click **Add OpenID provider**.
4. Provide the following information:

| Field | Description |
|-------------------------------------|--|
| Name | Name of an OpenID provider. API Gateway supports OpenID Connect (OIDC) identity tokens provided a standards-compliant OpenID provider. For example, Google, Salesforce. |
| OpenID Connect discovery URL | The discovery endpoint of OpenID Connect. Note: When the discovery URL is specified, some of the fields are automatically populated with data provided in the URL. |
| Token issuer endpoint | The endpoint URL of an OpenID token issuer used by API Gateway. |
| Token endpoint | The endpoint URL of OpenID Provider's token (id_token) through which the client application exchanges the authorization code, client ID, and client secret, obtain ID tokens and refresh tokens. |

| Field | Description |
|--------------------------------|---|
| Authorization endpoint | The endpoint URL through which API Gateway authenticates and grants authorization to client applications. |
| JSON Web Key endpoint | The endpoint URL of JSON Web Key Signature (JWKS) through which API Gateway verifies and validates the signature of ID tokens. |
| Userinfo endpoint | Optional. The endpoint URL through which API Gateway retrieves the end user identity information to include as claims in the ID tokens. |
| Include userinfo claims | Optional. Select the check box if you want to include the end user identity information as claims in the ID tokens. <div data-bbox="581 886 1334 1411" style="background-color: #f0f0f0; padding: 10px; margin-top: 10px;"> <p>Note: Some of the OpenID Providers do not include the complete end user information in the ID tokens. In such cases, you could use this Include userinfo claims check box to include the complete user information. API Gateway generates a secondary ID token that includes the required user information as claims. The token also includes the application ID, and primary ID token, if client has specified the application ID in the GET OpenID token request to API Gateway. As a pre-requisite, you must have the JWT settings configured in API Gateway. For information about configuring JWT settings in API Gateway, see "Configuring API Gateway as JWT Issuer" on page 92.</p> </div> |
| Scope | The specific set of end user identity information to include as claims in the ID tokens. |
| Response type | The authorization grant type to obtain authorization codes, ID tokens, and refresh tokens. Supported values: <ul style="list-style-type: none"> ■ code ■ id_token ■ token id_token ■ id_token token |

| Field | Description |
|---|--|
| | <p>The value code indicates the authorization grant type is code, and an authorization code is returned in the response to obtain ID tokens and refresh tokens.</p> <p>The values id_token, token id_token, id_token token and indicate the grant type is implicit, and the ID tokens are directly sent to the client.</p> |
| Token endpoint authentication method | <p>The client authentication method that API Gateway will use to communicate with the OpenID provider to fetch the ID token in exchange of an authorization code.</p> <p>Supported values:</p> <ul style="list-style-type: none"> ■ client_secret_basic ■ client_secret_post |
| Display | <p>Optional. The display type of user interface for end user authentication and consent flow.</p> <p>Supported values:</p> <ul style="list-style-type: none"> ■ none ■ page ■ popup ■ touch ■ wap |
| Prompt | <p>Optional. The display type of user interface prompt for end user reauthentication and consent flow.</p> <p>Supported values:</p> <ul style="list-style-type: none"> ■ none ■ login ■ consent ■ select_account |
| Client id | An 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. |

| Field | Description |
|--------------------------|--|
| Redirect hostname | The host name of the redirect endpoint of API Gateway that is configured in the OpenID Provider. |
| Redirect port | The port number on which the redirect endpoint is listening for requests. |
| Max age | Optional. The duration of time (in seconds) in which the end user must have been authenticated. If this time has expired, the OpenID provider must re-authenticate the end user. |
| Locale | Optional. The display language of the user interface for end user authentication and consent flow. |

5. Click **Save**.

The OpenID provider is added. You can add as many providers as required, but only one is the default at any given time.

Viewing OpenID Provider List and Provider Configuration

You can view the list of integrated OpenID providers, delete a provider, and activate the provider in the OpenID providers section. In addition, you can view and modify the OpenID provider configuration in the individual OpenID provider details page.

To view a list of OpenID providers and provider configuration

1. Select **Username > Administration**.
2. Select **Security > OpenID provider**.

The OpenID providers section displays a list of all the currently defined OpenID providers in API Gateway.

You can also perform the following operations in the OpenID providers section by clicking the appropriate icons in the Action column.

- **Activate:** Enables client applications to authenticate users against an OpenID provider.
- **Delete:** Deletes an OpenID provider from API Gateway.

3. Click any OpenID provider to view the configuration details.

The OpenID provider details page displays the configuration details for client applications to obtain and use OpenID (ID) tokens to access protected resources in API Gateway.

Modifying the OpenID Provider Configuration

Pre-requisites:

You must have the API Gateway's manage security configurations functional privilege assigned to modify the configuration of an OpenID provider.

You might want to modify an OpenID provider to change the currently defined configuration settings.

To modify the OpenID provider configuration

1. Select **Username > Administration**.
2. Select **Security > OpenID provider**.

The OpenID providers section displays a list of all the currently defined OpenID providers in API Gateway.

3. Click the name of the OpenID provider you want to modify.
4. Modify the fields as required.
5. Click **Save**.

The OpenID provider is updated.

Activating an OpenID Provider

Pre-requisites:

You must have the API Gateway's manage security configurations functional privilege assigned to activate an OpenID provider.

You might want to activate an OpenID provider to use by default when there are more than one OpenID provider defined in API Gateway.

Important: At any point of time, only one OpenID Provider can be active in API Gateway.

To activate an OpenID provider

1. Select **Username > Administration**.
2. Select **Security > OpenID provider**.

The OpenID providers section displays a list of all the currently defined OpenID providers in API Gateway.

3. Click **Activate** in the action column of the OpenID provider to be activated.

You can switch the OpenID provider to an inactive state at any time.

Deactivating an OpenID Provider

Pre-requisites:

You must have the API Gateway's manage security configurations functional privilege assigned to deactivate an OpenID provider.

You might want to deactivate an OpenID provider to set any other OpenID provider in an active state.

To deactivate an OpenID provider

1. Select **Username > Administration**.
2. Select **Security > OpenID provider**.

The OpenID providers section displays a list of all the currently defined OpenID providers in API Gateway.

3. Click **Deactivate** in the action column of the OpenID provider to be deactivated.

You can switch the OpenID provider to an active state at any time.

Deleting an OpenID Provider

Pre-requisites:

You must have the API Gateway's manage security configurations functional privilege assigned to delete an OpenID provider.

You delete an OpenID provider to remove it from API Gateway permanently.

To delete an OpenID provider

1. Select **Username > Administration**.
2. Select **Security > OpenID provider**.

The OpenID providers section displays a list of all the currently defined OpenID providers in API Gateway.

3. Click  in the action column of the OpenID provider to be deleted.
4. Click **Yes** in the confirmation dialog.

The OpenID provider is deleted from API Gateway.

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
- CentraSite
- Database
- Elasticsearch
- Email

Configuring Events for API Gateway Destination

You have to configure the API Gateway destination so that the events and performance metrics 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. Select **Username > Administration**.
2. Select **Destinations**.
3. Select **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. 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. Select **Username > 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. |
| 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. |
| 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. Select **Username > 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**.

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. Select **Username > 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. |
| 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. |

| Field | Description |
|-------------------------------|---|
| 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: DES, 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. Select **Username > 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 configuring the events for CentraSite destination, you need to select **CentraSite** as a **Destination** in the Policy Properties page of the API to publish the transaction and monitoring events for the API. 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 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. Select **Username > 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 event types and performance metrics data can be published to Elasticsearch.

To configure Elasticsearch destination

1. Select **Username > 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.

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. Select **Username > 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
```

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**.

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. Select **Username > Administration**.
2. Select **System Settings > Configuration**.
3. To change the API Gateway timeout interval, type the required value in the **API Gateway timeout (minutes)** field in the System setting section.
4. 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.

5. 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.
6. Click **Save**.

3 Users and Access Profiles

| | |
|--|-----|
| ■ Managing Users and Access Profiles | 120 |
|--|-----|

Managing Users and Access Profiles

You can add and manage user and access profile information from the User Management page in Integration Cloud.

Users: User personas who can access API Gateway and perform tasks. A predefined user is an Administrator who has administrator privileges.

Access Profiles: The functional privileges that are grouped together to form an access profile, and associate LDAP or local groups to the access profile. An administrator can create an access profile, add functional privileges to the profile, associate an user to access profiles, and delete an access profile.

User must be associated with atleast one access profile to access and log on to API Gateway.

For information on managing users and access profiles, see *webMethods Integration Cloud Help*

4 APIs

| | |
|---|-----|
| ■ Overview | 122 |
| ■ Creating an API by Importing an API from a File | 122 |
| ■ Creating an API by Importing an API from a URL | 123 |
| ■ Creating a REST API from Scratch | 123 |
| ■ Viewing API List and API Details | 127 |
| ■ Filtering APIs | 128 |
| ■ Publishing an API | 128 |
| ■ Modifying API Details | 129 |
| ■ Updating APIs | 129 |
| ■ SOAP to REST Transformation | 133 |
| ■ API Mocking | 141 |
| ■ API Scopes | 145 |
| ■ Exposing a REST API to Applications | 154 |
| ■ Exposing a SOAP API to Applications | 155 |
| ■ Versioning APIs | 156 |
| ■ API Grouping | 157 |
| ■ Exporting APIs | 158 |
| ■ Exporting Specifications | 158 |
| ■ Example: Managing an API | 159 |

Overview

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, and view API analytics from this view.

You can create an API:

- By importing an API from a file
- By importing an API from a URL
- From scratch

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.

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. Click **Create**.

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**, and **WSDL**.
9. Provide a version for the API in the **Version** field.
10. Click **Create**.

Creating a REST API from Scratch

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**.
6. Provide the following information in the Basic information section:

| Field | Description |
|-----------------------|--|
| Name | Name of the API. |
| Version | Version of the API being created. |
| 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 categorised.</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> |
| 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. Select the protocol **HTTP** or **HTTPS** to specify the protocol to be used.
 - b. Provide the host details in the form of *hostname:port#*
 - c. Provide the Base URL in the **Base path** field.
 - d. Click **+ Add Parameter** and provide the following information to add the required 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 |
| Data type | Specifies the data type. Available values - String, Date, Date time, Integer, Double, Boolean. |
| Required | Specifies the parameter is required if selected. |
| Array | Specifies the array is required if selected. |
| Value | Specifies the possible values. |

- e. Click **+ Add Schema** and provide the following information.

| Field | Description |
|--------------------|---|
| Name | Name of the schema. |
| Schema type | Specifies the schema type. Available types - Inline schema, External schema |

| Field | Description |
|--------------|--|
| Value | <p>Specifies the value for the schema type selected.</p> <p>For Inline schema - Type the request and response values.</p> <p>For External schema - Click Browse to upload the request and response values.</p> |

The schema specified here can be used and referred in the resource and method specification section across multiple methods and resources.

9. Click **Continue to provide Resource and methods for this API**.

Alternatively, you can click **Resources and methods** to go to the Resources and methods section.

Click **Save** to save the API at this stage and provide the resources and methods for the API at a later time.

10. Provide the following information in the Resources and methods section:
 - a. Click **Add Resources** and provide the following information.

| Field | Description |
|--------------------------|---|
| Resource name | <p>Name of the resource.</p> <p>This is the display name of the resource and resource path is used for execution.</p> |
| Resource path | <p>Specifies the path of the resource.</p> <p>The resource path should start with /</p> |
| Description | Description of the resource. |
| Supported methods | <p>Specifies the supported methods.</p> <p>Available values - GET, POST, PUT, DELETE, PATCH</p> |

- b. Click **Add**.

The resource is added. You can add as many resources you require.

- c. Provide the following information in the Supported methods section for each of the supported methods selected:

| Field | Description |
|------------------------------|---|
| Description | Brief description of the supported method. |
| Add method parameters | <p>Specifies the method parameters.</p> <p>Provide the following information:</p> <ul style="list-style-type: none"> ■ Name - Name of the parameter. ■ Description - Brief description of the parameter. ■ Type - Specifies the parameter type. ■ Data type - Specifies the data type. ■ Required - Specifies the parameter is required if selected. ■ Array - Specifies the array. ■ Values - Specifies the possible values for the parameter. |

- d. Click **Add Request** and add a Content type, provide a schema or select a schema and a sample.
- e. Click **Add Response** and provide the status code and a description.

11. Click **Save**.

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.

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** or **Activation status**.
- Activate an API by clicking  that denotes an inactive state.

Activating an API creates a virtual copy of the API without publishing it to a gateway. 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.

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**, or both 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.

Publishing an API

You must have the API Gateway's publish to API portal functional privilege assigned to perform this task.

You can publish an API to the configured destination, for example API Portal. Once the API is published it is available.

Ensure that a destination is configured before publishing an API.

To publish an API

1. Click **APIs** in the title navigation bar.
A list of all APIs appears.
2. Click the **Publish** icon for the API that has to be published.
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 communities to which the API needs to be published.

By default an API is published to the Public Community of API Portal.

5. Click **Publish**.

The API is now published to the destination, for example API Portal, that is configured and is available on API Portal for the consumers to consume it.

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.

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

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. Modify the information as required.
5. Click **Save**.

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 existing API:

- 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.

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 | <p>Click Browse to browse to the location of file to be imported and select the required file or ZIP format file.</p> <p>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.</p> |
| Root File Name | <p>If you have selected a file in ZIP format, type the relative path of the main file within the ZIP file.</p> |
| Name | <p>Name for the API. Edit or delete the name of the existing API displayed.</p> <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 | <p>Select the required type. The available types are RAML, Swagger, and WSDL.</p> <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. |

| Field | Description |
|--------------------|--|
| 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. 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 |

| Field | Description |
|------------------|--|
| | be updated using only the WSDL file type information that the URL is pointing to. |
| 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. ■ 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 | Select the required type. The available types are RAML , Swagger , and WSDL . 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 file referred to by URL does not have a version number, |

| Field | Description |
|--------------------|---|
| | <p>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> |

- Click **Update**.

The API definition is updated with the latest changes from the URL and is displayed in the API details page.

SOAP to REST Transformation

SOAP web services 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 web services 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

- Click **APIs** in the title navigation bar.
- Select the required API from the list of available APIs.
The API details page for the selected API appears.
- Click **Edit**.
- 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. |

| Field | Description |
|----------------------|---|
| | 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, or //axis:name</code> (If the SOAP request has only one element such as name). |

| Field | Description |
|-------------------------|--|
| Namespace prefix | Specifies the namespace prefix of the element that appears in the XPath . |
| Namespace URI | Specifies the namespace URI for the XPath element. 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/xml or text/xml multipart/form-data or multipart/mixed application/x-www-form-urlencoded | application/json application/xml or text/xml multipart/form-data or multipart/mixed |
| PUT | application/json application/xml or text/xml multipart/form-data or multipart/mixed application/x-www-form-urlencoded | application/json application/xml or text/xml multipart/form-data or multipart/mixed |
| 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 Service 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 service does not change the service name. The only change to the SOAP invocation is that the `<resource-path-for-the-operation>` is appended:

```
/ws/<SERVICE-NAME>/<version-number>/<resource-path-for-the-resource>
```

Note: The REST-enabled SOAP services 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>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://schemas.xmlsoap.org/soap/envelope/">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></pre> |

| | Request | Equivalent SOAP Request |
|-----------------------------------|---|--|
| | or /ws/ CalcService/ add?num1=10 | |
| Multiple query or path parameters | /ws/ CalcService/ add/{num1}/ {num2} | <pre><ws:addInts xmlns:ws="http:test"> <num1></num1> <num2></ num2></ws:addInts></pre> |
| | or /ws/ CalcService/ add? num1=10&num2=3 | |
| | or /ws/ CalcService/ add/ {num1}&num2=3 | |
| Hierarchical elements | /ws/ CalcService/ add/{num1}/ anotherNumber/ {num2} | <pre><ws:addInts xmlns:ws="http:test"> <num1></num1> <num2></ num2></ws:addInts></pre> |

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.

API Mocking

Using API Gateway, you can mock an API by simulating the native service. 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.

As an API Provider, you can create or modify the default mock response. You can specify conditions and associate an ESB service with the mocked API. When an ESB service is associated with a mocked API, the associated ESB service must adhere to the *apigateway.specifications:mockService* specification.

At runtime, when the mocked API is invoked, instead of calling the native service, 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 ESB service is configured for the API, then API Gateway invokes the ESB service and returns the ESB service response.
3. If any of the conditions for the invoked operation is not satisfied, and if an ESB 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** to enable API 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.
2. Click the required API.
The API details page appears.
3. Click **Edit**.
4. Click **API mocking**.
5. 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 run the ESB 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  .

- 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  .

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 | <p>Select the type to which the condition is to be applied. The available options are:</p> <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. |
| 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 | <p>Select the status code from the drop-down list.</p> <div style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;"> <p>Note:■ 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.</p> <ul style="list-style-type: none"> ■ 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> |
| + 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 |

| Field | Description |
|---------------------------|--|
| | <ul style="list-style-type: none"> 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  .

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.

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.

For information on the usage scenarios of scopes in an API, see ["Example: Usage Scenarios of API Scopes" on page 149](#).

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 280.

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 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 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**.

Post-requisites:

Activate the API when you are ready 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 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 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. When you are prompted to confirm the delete operation, click **Yes** in the confirmation dialog box.
The scope is removed from the List of scopes section.
8. Click **Save** to save the updated API.

Post-requisites:

Activate the API when you are ready 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 |

In the following section, we demonstrate 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 | |
| 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 Inbound Authentication - Transport 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 Application policy with HTTP Basic Authentication |
| Scope-level Policy for PAYMENT Scope | Inbound Authentication - Transport |
| Scope-level Policy for WRITE Scope | Identify and Authorize Application policy with API Key Throttling Traffic Optimization |

The API Scopes definition looks like this:

| | | | |
|--------------------------|--|------|---|
| API-level Policy | Identify and Authorize Application policy with HTTP Basic Authentication | | |
| Policy for PAYMENT Scope | Resource C: /phones/orders/{order-id}/paymentdetails | | Inbound Authentication - Transport |
| Policy for WRITE Scope | Resource C: /phones/orders/{order-id}/paymentdetails | POST | Identify and Authorize Application policy with API Key Throttling 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): Inbound Authentication - Transport
- API-level Policy: Identify and Authorize Application policy with HTTP Basic Authentication

As per the precedence of policy enforcement, the Inbound Authentication - Transport 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:

- Inbound Authentication - Transport
- 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): Inbound Authentication - Transport
- 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:

- Inbound Authentication - Transport
- 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): Inbound Authentication - Transport
- 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:

- Inbound Authentication - Transport
- 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 Policy | Identify and Authorize Application policy with HTTP Basic Authentication | | |
| Policy for PAYMENT Scope | Resource C: /phones/orders/{order-id}/paymentdetails | | Inbound Authentication - Transport |
| Policy for WRITE Scope | Resource C: /phones/orders/{order-id}/paymentdetails | POST | Identify and Authorize Application policy with API Key Throttling Traffic Optimization |

| | | | |
|-------------------------|--|------|---|
| Policy for UPDATE Scope | Resource C: /phones/orders/{order-id}/paymentdetails | POST | Identify and Authorize Application policy with IP Address Range |
|-------------------------|--|------|---|

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): Inbound Authentication - Transport
- 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 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 will 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 prompts you to deactivate it.
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. When you have selected the required resources and methods, click **Save** to save the updated API.

Post-requisites:

Activate the API when you are ready to put it into effect.

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 prompts you to deactivate it.
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. When you have selected the required operations, click **Save** to save the updated API.

Post-requisites:

Activate the API when you are ready to put it into effect.

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.

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 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.

Exporting APIs

You must have the API Gateway's export assets privilege assigned to perform this task.

To export an API

1. Click **APIs** in the title navigation bar.
2. Select the required API from the list of available APIs.
3. Click  to export a single API.

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 Export archive window appears.

4. Select **Include applications** if you want to export the applications associated with the APIs.
5. Click **Export**.

The browser prompts you to either open or save the export archive.

6. 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 **Specifications**.
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.
 - **Artifacts** link to export the WSDL specification.The browser prompts you to either open or save the export archive.
5. Select the appropriate option and click **OK**.

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. |
| <code>http://www.phonestore.com/api/phones/412456</code> | Accesses a phone referenced by the product code <code>412456</code> . |

| URL | Description |
|---|---|
| <code>http://www.phonestore.com/api/phones/412456/reviews</code> | Specifies a set of reviews posted for a phone of code 412456. |
| <code>http://www.phonestore.com/api/phones/412456/reviews/78</code> | 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 |
|-----------------------------|-------------|---|
| <code>/phones/orders</code> | GET | Asks for a representation of all of the orders. |
| <code>/phones/orders</code> | POST | Attempts to create a new order, returning the location (in the Location |

| Resource URI | HTTP Method | Description |
|--|-------------|--|
| | | HTTP Header) of the newly created resource. |
| /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 by either 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
```

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. |

| Data Type | Description |
|-----------|--|
| | 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 |
|-----------------------------|------------------------|---|
| <code>/phones/orders</code> | GET | 200 (OK, Success) |
| <code>/phones/orders</code> | POST | 201 (Created) if the Order resource is successfully created, in addition to a Location header |

| Resource URI | Supported HTTP Methods | Supported HTTP Status Codes |
|--|------------------------|--|
| | | 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 |
| /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 | 172 |
| ■ Policy Validation and Dependencies | 174 |
| ■ System-defined Stages and Policies | 181 |
| ■ Managing Threat Protection Policies | 251 |
| ■ Managing Global Policies | 262 |
| ■ Managing API-level Policies | 277 |
| ■ Managing Scope-level Policies | 279 |
| ■ Managing Policy Templates | 284 |

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. For information on the stages and policies that API Gateway provides out-of-the-box, see ["System-defined Stages and Policies" on page 181](#).

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-specific Scopes:** 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.

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 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-level) 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-level) 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-level) 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.

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:

- The order in which API Gateway evaluates the policies.
- 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- | 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? |
|------------------------------------|---------------------|------------------|---|---|
| | | | based Routing | |
| 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. |
| 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. |
| Identify and Authorize Application | 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? |
|--|---------------------|------------------|--|--|
| 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. |
| 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. |
| Load Balancer Routing | REST SOAP | None. | Straight Through Routing, Dynamic Routing, Content-based Routing, Context- | 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? |
|-------------------------------------|---------------------|------------------------------------|---------------------------|---|
| | | | based Routing | |
| 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. |
| 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 |

| Policy | Applicable API Type | Dependent Policy | Mutually Exclusive Policy | Can include multiple times in an API? |
|---------------------------------|---------------------|------------------------------------|--|---|
| | | | | the effective policy. |
| Require HTTP / HTTPS | REST SOAP | None. | None. | No. API Gateway includes only one policy 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 |

| Policy | Applicable API Type | Dependent Policy | Mutually Exclusive Policy | Can include multiple times in an API? |
|---|---------------------|------------------|---------------------------|---|
| | | | | the effective policy. |
| Validate HTTP Headers | REST SOAP | None. | None. | No. API Gateway includes only one policy in the effective policy. |
| Validate Schema (Response Processing) | REST SOAP | None. | None. | No. API Gateway includes only one policy in the effective policy. |
| Validate Schema (Request Processing) | REST SOAP | None. | None. | No. API Gateway includes only one policy in the effective policy. |
| XSLT Transformation (Response Processing) | REST SOAP | None. | None. | Yes. API Gateway includes all XSLT Transformation policies in the effective policy. |
| XSLT Transformation (Request Processing) | REST SOAP | None. | None. | Yes. API Gateway includes all XSLT Transformation policies in the effective policy. |

System-defined Stages and Policies

API Gateway provides system-defined policies that are grouped into stages depending on their usage:

- Threat Protection
- 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:

- Require HTTP/HTTPS
- Set Media Type
- Validate HTTP Headers

Require 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 Require 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. ■ SOAP 1.2. API Gateway accepts requests that are in the SOAP 1.2 format. |

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 | <p>Specifies the default content type for REST request received from a client.</p> <p>Examples for content types: <code>application/json</code>, <code>application/xml</code>.</p> |
| Default Accept Header | <p>Specifies the default accept header for REST request received from a client.</p> |

Validate HTTP Headers

This policy validates presence of HTTP headers, or header values, or both in an incoming API request. This policy is applicable for both SOAP and REST APIs.

The table lists the properties that you can specify for this policy:

| Parameter | Description |
|--------------------|--|
| Condition | <p>Specifies the logical operator to use to validate multiple HTTP headers in the incoming API requests.</p> <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 | <p>Provide the following information:</p> <ul style="list-style-type: none"> ■ HTTP Header Key. Specifies a key that must be passed through the HTTP header of the incoming API requests. ■ Header Value. Optional. 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> |

Use Case Examples: Validate HTTP Headers Policy

Example 1:

```
HTTP Header KEY: Authorization
HTTP Header value:
```

The HTTP header is defined with an empty value. API Gateway verifies the presence of an HTTP Authorization header in the incoming API request.

Example 2:

```
HTTP Header KEY: Content-Type
HTTP Header value: application/xml
```

API Gateway verifies the presence of a Content-Type Authorization header with the corresponding header value, `application/xml`, in the incoming API request.

Example 3:

```
HTTP Header KEY: Content-Type
HTTP Header value: ^application.[a-z]*
```

API Gateway verifies the presence of a Content-Type Authorization header with the corresponding header value to start with `application` in the incoming API request. For example, `application/json`, `application/xml`, and so on.

Example 4:

```
HTTP Header KEY: Authorization
HTTP Header value:
```

API Gateway verifies the presence of a Content-Type Authorization header with the corresponding header value to end with `xml` in the incoming API request. For example, `application/xml`, `text/xml`, and so on.

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 - Transport
- Inbound Authentication - Message
- Identify and Authorize Application

The Inbound authentication policies at transport and message level 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 policy is used to identify the application and authorize it against all application registered in API Gateway.

Inbound Authentication - Transport

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 transport headers that will be added to the request and sent to the native API. API Gateway supports the HTTP basic and Kerberos authentication schemes 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:

| Property | Description |
|----------------------------------|--|
| HTTP Basic Authentication | Specifies using the HTTP authentication mechanism to validate incoming requests from clients. API Gateway authorizes the credentials (username and password) against a list of all users available in API Gateway. |
| OpenID Authentication | <p>Specifies that an OpenID (ID) Token is required when a client application accesses an API enforced with the OpenID authentication policy.</p> <p>API Gateway extracts the ID token from the transport authorization header and validates the token with the claims configured in the application that is requesting access for the API. If the ID token sent by the client is valid, API Gateway forwards the request to the native API and the response to the client.</p> |
| JWT Authentication | <p>Specifies that a JSON Web Token (JWT) is required when a client application accesses an API enforced with the JWT authentication policy.</p> <p>API Gateway extracts the JWT from the transport authorization header and validates the token using a public certificate that was specified in the JWT configuration. If the JWT sent by the client is valid, API Gateway forwards the request to the native API and the response to the client.</p> |

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. |
| <p>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.</p> | |
| Encrypted Parts | <p>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.</p> <p>Provide the following information:</p> <ul style="list-style-type: none"> ■ Entire SOAP Body. Specifies encryption of the entire SOAP body. ■ All SOAP Attachments. Specifies encryption of all the SOAP attachments. <p>In the SOAP Header section, provide the following information:</p> <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 encrypted. <p>You can add more SOAP headers by clicking  .</p> |
| 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. |

| Parameter | Description |
|-------------------------------|---|
| | <p>You can add more namespace prefixes and URIs by clicking  .</p> |
| | <p>Require Signature. 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> |
| <p>Signed Elements</p> | <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> |
| <p>Signed Parts</p> | <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> <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. |

| Parameter | Description |
|-----------------------|---|
| | 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.</p> <p>This property is used in the following scenarios:</p> <ul style="list-style-type: none"> ■ 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. <p>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.</p> |

| Parameter | Description |
|------------------|--|
| Token Assertions | <p>Select the type of token assertion required to authenticate the client.</p> <p>Select any of the following:</p> <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 style="border: 1px solid #ccc; 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 the LDAP or central user directory 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: |

| Parameter | Description |
|-----------|---|
| | <ul style="list-style-type: none"> <li data-bbox="634 323 1299 386">■ SAML Version. Specifies the supported SAML version. Available values are SAML 1.0, SAML 2.0 <li data-bbox="634 411 1346 1083">■ SAML Subject Configuration. Select one of the following: <ul style="list-style-type: none"> <li data-bbox="683 499 1346 604">■ Bearer of Token. Select the bearer method when the client wants a security token to be issued without a proof of possession. <li data-bbox="683 621 1346 852">■ Holder of Key - Symmetric. Select the Holder of Key (Symmetric) method when either the client or the server 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. <li data-bbox="683 877 1346 1083">■ 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. <li data-bbox="634 1104 1346 1209">■ WS-Trust Version. Specifies the WS-Trust version to be used. Available values are WS-Trust 1.0, WS-Trust 1.3 <li data-bbox="634 1230 1218 1293">■ Encrypt Signature. Select Yes to encrypt the signature. <li data-bbox="634 1314 1325 1346">■ Issuer Address. Specifies the SAML issuer address. <li data-bbox="634 1367 1305 1472">■ Metadata Reference Address. Specifies the address from where the metadata reference document is obtained. <li data-bbox="634 1493 1317 1556">■ Algorithm Suite. Specifies the applicable algorithm suite. <li data-bbox="634 1577 1300 1640">■ Key. Specifies the Key type of the security token template. <li data-bbox="634 1661 1346 1818">■ Value. Specifies a value for the request token. You can add more values for the key-value pair by clicking  . <li data-bbox="586 1839 1284 1904">■ Custom Token Assertion. Type a search string, select a custom token assertion name to authenticate the |

| Parameter | Description |
|--------------------------|--|
| | <p>client, and click  to add. You can add more custom token assertions in a similar way.</p> <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 | 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. |

Identify and Authorize Application

This policy authorizes and allows access to the applications that are trying to access the APIs, for example, through IP address or hostname, and validate the clients credentials.

The table lists the properties that you can specify for this policy:

| Parameter | Description |
|---|--|
| Condition | <p>Specifies the condition operator for the authorization types selected.</p> <p>Select any of the following condition operators:</p> <ul style="list-style-type: none"> ■ AND. Applies all the identification types selected. ■ OR. Applies one of the identification types selected. |
| Allow anonymous | Specifies whether to allow all users to access the API without restriction. |
| Identification Type. Specifies the identification type. You can select any of the following. | |
| 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. |
| Hostname Address | Specifies using host name address to identify the client, extract the client's hostname from the HTTP request |

| Parameter | Description |
|---|--|
| | <p>header and verify the client's identity in 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. Tries to verify the client's hostname against a list of registered applications for the specified API. ■ Global applications. Tries to verify the client's hostname against a list of all global applications available in API Gateway. <p>Note: If JMS is selected as the entry protocol policy, extract the client's hostname from the X-Forwarded-For JMS message property.</p> |
| <p>HTTP Basic Authentication</p> | <p>Specifies using Authorization Header in the request to identify and authorize the client application against the list of applications with the identifier <code>username</code> 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. Tries to verify the client's credentials against the list of registered applications for the specified API. ■ Global applications. Tries to verify the client's credentials against a list of all global applications available in API Gateway. ■ Do not identify. Checks for the existence of the criterion but does not validate if the specified value is a valid application and forwards the request to the native API. For example, HTTP Basic Authentication is checked by the HTTP transport level property <code>Authorization: Basic Base64encodesusernamepassword</code> |
| <p>IP Address Range</p> | <p>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>Select one of the Application Lookup condition:</p> |

| Parameter | Description |
|-----------------------|---|
| | <ul style="list-style-type: none"> ■ Registered applications. Tries to verify the client's credentials against a list of registered applications for the specified API. ■ Global applications. Tries to verify the client's credentials against a list of all global applications available in API Gateway. <p>Note: If JMS is selected as the entry protocol policy, extract the client's IP address from the X-Forwarded-For JMS message property.</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. Tries to verify the JWT against a list of registered applications for the specified API. ■ Global applications. Tries to verify the JWT against a list of all global applications available in API Gateway. |
| 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>Select one of the Application Lookup condition:</p> <ul style="list-style-type: none"> ■ Registered applications. Tries to verify the Kerberos token against a list of registered applications for the specified API. ■ Global applications. Tries to verify the Kerberos token against a list of all global applications available in API Gateway. |
| OAuth2 Token | <p>Specifies using the OAuth2 token to identify the client, extract the client's credentials from the HTTP request header, and verify the client's identity against the specified list of applications in API Gateway.</p> |
| OpenID Connect | <p>Specifies using the OpenID (ID) token to identify the client, extract the client's credentials from the ID token,</p> |

| Parameter | Description |
|-----------------------------------|--|
| | <p>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. Tries to verify the ID token against a list of registered applications for the specified API.■ Global applications. Tries to verify the ID token against a list of all global applications available in API Gateway. |
| 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.</p> <p>Select one of the Application Lookup condition:</p> <ul style="list-style-type: none">■ Registered applications. Tries to verify the client certificate against a list of registered applications for the specified API.■ Global applications. Tries to verify the client certificate against a list of all global applications available in API Gateway. |
| 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>Select one of the Application Lookup condition:</p> <ul style="list-style-type: none">■ Registered applications. Tries to verify the client's WSS username token against a list of registered applications for the specified API.■ Global applications. Tries to verify the client's WSS username token against a list of all global applications available in API Gateway. |

| Parameter | Description |
|--------------------------------------|---|
| WS Security X.509 Certificate | <p>This is applicable only for SOAP APIs.</p> <p>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>Select one of the Application Lookup condition:</p> <ul style="list-style-type: none"> ■ Registered applications. Tries to verify the client's X.509 certificate against a list of registered applications for the specified API. ■ Global applications. Tries to verify the client's X.509 certificate against a list of all global applications available in API Gateway. |
| XPath expression | <p>Specifies using the XPath expression to identify the client, extract the custom authentication credentials supplied in the request represented using an XPath expression, and verify the client's identity against the specified list of applications in API Gateway.</p> <ul style="list-style-type: none"> ■ Select one of the Application Lookup condition: <ul style="list-style-type: none"> ■ Registered applications. Tries to verify the client's OAuth access token against a list of registered applications for the specified API. ■ Global applications. Tries to verify the client's identify credentials against a list of all global applications available in API Gateway. ■ Query Expression. Specifies an argument to evaluate the XPath expression contained in the request. ■ Namespace URI. The namespace URI of the XPath expression to be validated. ■ Namespace Prefix. The namespace prefix of the XPath expression to be validated. |

Request Processing

These policies are used to specify how the request message from an application has to be transformed or pre-processed before it is submitted to the native API. This is required to 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 Integration Server
- XSLT Transformation
- Validate Schema

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.

The table lists the properties that you can specify for this policy:

| Parameter | Description |
|------------------------------------|--|
| webMethods IS Service | <p>Specifies the webMethods IS service to be invoked to pre-process the request messages.</p> <p>Note: The webMethods IS service must be running on the same Integration Server as API Gateway.</p> <p>You can add multiple entries by clicking  .</p> |
| webMethods IS Service alias | <p>Specifies the webMethods IS service alias to be invoked to pre-process the request messages.</p> <p>You can add multiple entries by clicking  .</p> |

XSLT Transformation

This policy specifies the XSLT transformation file to transform request messages from clients into a format required by the native API in the SOAP request before it is submitted to the native API.

The table lists the properties that you can specify for this policy:

| Parameter | Description |
|----------------------------------|--|
| XSLT Document | <p>Click + Add xslt document to add an xslt document.</p> <p>Provide the following information:</p> <ul style="list-style-type: none"> ■ XSLT file. Specifies the XSLT file used to transform the request messages as required. <p>Click Browse to browse and select a file.</p> <p>In the XSLT Features section, provide the following information:</p> <ul style="list-style-type: none"> ■ Feature Name. Specifies the name of the XSLT feature. ■ Feature value. Specifies the value of the XSLT feature. <p>You can add more XSLT features and xslt documents by clicking  .</p> |
| XSLT Transformation alias | <p>Click + Add xslt transformation alias to add an xslt transformation alias in the XSLT Transformation alias field.</p> <p>You can add more XSLT transformation aliases by clicking  .</p> |

Validate Schema

This policy validates the incoming requests to an API against a schema referenced in the WSDL. 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 an XML schema specified in this policy to conform to the structure or format expected by the API. An XML schema precisely defines the elements and attributes that constitute an instance XML document. The XML schema also specifies the data types of these elements to ensure that only appropriate data is allowed through to the API. For example, an XML schema might stipulate that all requests to a particular API must contain a <name> element, which contains at the most a ten-character string. If the request contains a message with an improperly formed <name> element, there is a policy violation and the message is rejected.

For a REST API, the schema can be added inline or uploaded in the Technical Information section on the API Details page.

Note: For a REST API, the schema validation works only for XML schema provided the schema is mapped to the resource. The schema validation does not work for a JSON schema.

For a SOAP API, the WSDL and the referenced schema must be provided in a zip format.

The runtime invocations that fail the schema validation are considered as policy violations. You can view such policy violation events in the dashboard.

The table lists the schema properties that you can specify for this policy:

| Parameter | Description |
|----------------------|---|
| Feature name | Specifies the name for the schema configuration. For example: TOLERATE_DUPLICATES, NAMESPACE_GROWTH |
| Feature value | Specifies whether the feature value is True or False. |

You can add multiple entries for the feature name and value by clicking  .

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

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 | Specifies the URI of the native API endpoint to route the request to in case all routing rules evaluate to False. |
| Routing Method | This is applicable to REST APIs. Specifies the available routing methods: GET , POST , PUT , DELETE , and CUSTOM (default). |
| HTTP Connection Timeout (seconds) | Specifies the time interval (in seconds) after which a connection attempt times out. If a value 0 is specified (or if the value is not specified), API Gateway uses the default value 30 seconds. |
| Read Timeout (seconds) | Specifies the time interval (in seconds) after which a socket read attempt times out. If a value 0 is specified (or if the value is not specified), API Gateway uses the default value 30 seconds. |
| SSL Configuration | Specifies values to enable SSL client authentication that API Gateway uses to authenticate incoming requests for the native API. |
| Keystore Alias | Specifies the keystore alias of the instance of Integration Server on which API Gateway is running. This value (along with the value of Client Certificate Alias) is used for performing SSL client authentication. |
| Key Alias | Specifies the alias for the private key, which must be stored in the keystore specified by the keystore alias. |
| 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: |

| Parameter | Description |
|---------------------------------|--|
| | <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 | 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. |
| Rule. | Defines the routing decisions based on one of the following routing options. |
| XPath Expression | <p>Specifies that the XPath expression contained in the request is used for routing.</p> <p>Provide the following information:</p> <p>XPath Expression. Specifies an argument to evaluate the XPath expression contained in the request.</p> <p>Value. Specifies the XPath value.</p> |
| Namespace | <p>Specifies that the namespace identifiers contained in the request are used for routing.</p> <p>Provide the following information:</p> <ul style="list-style-type: none"> ■ Namespace Prefix. Specifies the namespace declaration indicated by the Prefix. ■ Namespace URI. Namespace declaration indicated by URI. <p>You can add multiple entries for the Namespace prefixes and URIs by clicking  .</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. |

| Parameter | Description |
|--|--|
| Routing method | Specifies the available routing methods: GET , POST , PUT , DELETE , and CUSTOM (default). |
| HTTP Connection Timeout (seconds) | <p>Specifies the time interval (in seconds) after which a connection attempt times out.</p> <p>If a value 0 is specified (or if the value is not specified), API Gateway uses the default value 30 seconds.</p> |
| Read Timeout (seconds) | <p>Specifies the time interval (in seconds) after which a socket read attempt times out.</p> <p>If a value 0 is specified (or if the value is not specified), API Gateway uses the default value 30 seconds.</p> |
| SSL Configuration | <p>Specifies values to enable SSL client authentication that API Gateway uses to authenticate incoming requests for the native API.</p> <p>Provide the following information:</p> <ul style="list-style-type: none">■ Keystore Alias. Specifies the keystore alias of the instance of Integration Server on which API Gateway is running. This value (along with the value of Client Certificate Alias) is used for performing SSL client authentication.■ Key Alias. Specifies the alias for the private key, which must be stored in the keystore specified by the keystore alias. |
| 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. |

| Parameter | Description |
|---------------------------------|--|
| 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. |

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 | Specifies the URL of the native API endpoint to route the request to in case all routing rules evaluate to False. |
| Routing method | This is applicable to REST APIs. Specifies the available routing methods: GET , POST , PUT , DELETE , and CUSTOM (default). |
| HTTP Connection Timeout (seconds) | Specifies the time interval (in seconds) after which a connection attempt times out. If a value 0 is specified (or if the value is not specified), API Gateway uses the default value 30 seconds. |
| Read Timeout (seconds) | Specifies the time interval (in seconds) after which a socket read attempt times out. If a value 0 is specified (or if the value is not specified), API Gateway uses the default value 30 seconds. |
| SSL Configuration | Specifies values to enable SSL client authentication that API Gateway uses to authenticate incoming requests for the native API. |

| Parameter | Description |
|---|--|
| Keystore Alias | Specifies the keystore alias of the instance of Integration Server on which API Gateway is running. This value (along with the value of Client Certificate Alias) is used for performing SSL client authentication. |
| Key Alias | Specifies the alias for the private key, which must be stored in the keystore specified by the keystore alias. |
| 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> |
| 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. |

| Parameter | Description |
|------------------|---|
| 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. ■ To IP. Type an IP address range. ■ IPV6. Specifies that IP version to be IPV4. <ul style="list-style-type: none"> ■ 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. ■ Custom Context Variable <ul style="list-style-type: none"> ■ Variable Name. Type a name for the custom context variable. Use the <i>pub.apigateway.ctxvar:setContextVariable</i> 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 mx (for example, mx:CUSTOM_VARIABLE). All parameter names are case-sensitive. For more information on <i>pub.apigateway.ctxvar:setContextVariable</i> service, see "The API for Context Variables" on page 236. ■ Data Type. Select the data type: String or Integer. |

| Parameter | Description |
|--|--|
| | <ul style="list-style-type: none"> ■ Operator. Select one of the following operators based on the Data Type selected: <ul style="list-style-type: none"> ■ String: Equal To or Not Equal To ■ Integer: Equal To, Not Equal To, Greater Than, or Less Than ■ Variable Value. Type a value for the selected custom context. ■ Time <ul style="list-style-type: none"> ■ Operator. Select an operator: After or Before. ■ Hours. Type a time value in hours. ■ Minutes. Type a time value in minutes. |
| | <p>Route to. Specifies the endpoint URI of native services in a pool to which the requests are routed.</p> |
| Endpoint URI | Specifies the URI of the native API endpoint to route the request to. |
| Routing method | Specifies the available routing methods: GET , POST , PUT , DELETE , and CUSTOM (default). |
| HTTP Connection Timeout (seconds) | <p>Specifies the time interval (in seconds) after which a connection attempt times out.</p> <p>If a value 0 is specified (or if the value is not specified), API Gateway uses the default value 30 seconds.</p> |
| Read Timeout (seconds) | <p>Specifies the time interval (in seconds) after which a socket read attempt times out.</p> <p>If a value 0 is specified (or if the value is not specified), API Gateway uses the default value 30 seconds.</p> |
| SSL Configuration | <p>Specifies values to enable SSL client authentication that API Gateway uses to authenticate incoming requests for the native API.</p> <p>Provide the following information:</p> <ul style="list-style-type: none"> ■ Keystore Alias. Specifies the keystore alias of the instance of Integration Server on which API Gateway is running. This value (along with the value of Client |

| Parameter | Description |
|---------------------------------|--|
| | <p>Certificate Alias) is used for performing SSL client authentication</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. |
| Soap Optimization Method | <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. |
| 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> |

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 |
|--------------------------|---|
| Default 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. |

| Parameter | Description |
|--|--|
| Routing Method | <p>This is applicable to REST APIs.</p> <p>Specifies the available routing methods: GET, POST, PUT, DELETE, and CUSTOM (default).</p> |
| HTTP Connection Timeout (seconds) | <p>Specifies the time interval (in seconds) after which a connection attempt times out.</p> <p>If a value 0 is specified (or if the value is not specified), API Gateway uses the default value 30 seconds.</p> |
| Read Timeout (seconds) | <p>Specifies the time interval (in seconds) after which a socket read attempt times out.</p> <p>If a value 0 is specified (or if the value is not specified), API Gateway uses the default value 30 seconds.</p> |
| SSL Configuration. Specifies values to enable SSL client authentication that API Gateway uses to authenticate incoming requests for the native API. | |
| Keystore Alias | <p>Specifies the keystore alias of the instance of Integration Server on which API Gateway is running. This value (along with the value of Client Certificate Alias) is used for performing SSL client authentication.</p> |
| Key Alias | <p>Specifies the alias for the private key, which must be stored in the keystore specified by the keystore alias.</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. |

| Parameter | Description |
|---|---|
| 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. |
| Rule. Defines the routing decisions based on one of the following routing options. | |
| Route Using | <p>Defines the dynamic URL based on the HTTP header value sent by the client or the context variable value.</p> <p>Select one of the following:</p> <ul style="list-style-type: none"> ■ 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. ■ 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. |
| 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 ■ Routing method. Specifies the available routing methods: GET, POST, PUT, DELETE, and CUSTOM (default). ■ HTTP Connection Timeout (seconds). Specifies the time interval (in seconds) after which a connection attempt times out. ■ Read Timeout (seconds). Specifies the time interval (in seconds) after which a socket read attempt times out. ■ SSL Configuration. Specifies values to enable SSL client authentication that API Gateway uses to authenticate |

| Parameter | Description |
|--|--|
| | <p>incoming requests for the native API. Provide the following information:</p> <ul style="list-style-type: none"> ■ Keystore Alias. Specifies the keystore alias of the instance of Integration Server on which API Gateway is running. This value (along with the value of Client Certificate Alias) is used for performing SSL client authentication ■ Key Alias. Specifies the alias for the private key, which must be stored in the keystore specified by the keystore alias. |
| <p>SOAP Optimization Method</p> | <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. |
| <p>Pass WS-Security Headers</p> | <p>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.</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. |
| Routing Method | This is applicable to REST APIs. Specifies the available routing methods: GET , POST , PUT , DELETE , and CUSTOM (default). |
| HTTP Connection Timeout (seconds) | Specifies the time interval (in seconds) after which a connection attempt times out. If a value 0 is specified (or if the value is not specified), API Gateway uses the default value 30 seconds. |
| Read Timeout (seconds) | Specifies the time interval (in seconds) after which a socket read attempt times out. If a value 0 is specified (or if the value is not specified), API Gateway uses the default value 30 seconds. |
| SSL Configuration | Specifies values to enable SSL client authentication that API Gateway uses to authenticate incoming requests for the native API. |
| Keystore Alias | Specifies the keystore alias of the instance of Integration Server on which API Gateway is running. This value (along with the value of Client Certificate Alias) is used for performing SSL client authentication. |
| Key Alias | Specifies the alias for the private key, which must be stored in the keystore specified by the above keystore alias. |
| 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: |

| Parameter | Description |
|---------------------------------|--|
| | <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. |
| Suspend duration | <p>A numeric timeout value (in seconds). The default value is 30.</p> <p>When this timeout value expires, the system routes the execution of the API to the next consecutive endpoint specified in the Route to field.</p> |
| 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. |

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 | Specifies the URI of the native API endpoint to route the request to in case all routing rules evaluate to False. |
| Routing Method | <p>This is applicable to REST APIs.</p> <p>Specifies the available routing methods: GET, POST, PUT, DELETE, and CUSTOM (default).</p> |
| HTTP Connection Timeout (seconds) | Specifies the time interval (in seconds) after which a connection attempt times out. |

| Parameter | Value |
|---|--|
| | If a value 0 is specified (or if the value is not specified), API Gateway uses the default value 30 seconds. |
| Read Timeout (seconds) | <p>Specifies the time interval (in seconds) after which a socket read attempt times out.</p> <p>If a value 0 is specified (or if the value is not specified), API Gateway uses the default value 30 seconds.</p> |
| SSL configuration - Specifies values to enable SSL client authentication that API Gateway uses to authenticate incoming requests for the native API. | |
| Keystore Alias | Specifies the keystore alias of the instance of Integration Server on which API Gateway is running. This value (along with the value of Client Certificate Alias) is used for performing SSL client authentication. |
| Key Alias | Specifies the alias for the private key, which must be stored in the keystore specified by the keystore alias. |
| 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> |

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 | <p>Select one of the following schemes for outbound authentication at the transport level:</p> <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. |

| Parameter | Description |
|----------------------------|---|
| | <ul style="list-style-type: none"> ■ Anonymous. Authenticates the client without any credentials. ■ Alias. Uses the configured alias name for authentication. |
| Authentication mode | <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 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> <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 Name. 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 <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 Name. 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. |

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 authenticate the client. |

| Parameter | Description |
|----------------------------|---|
| | <ul style="list-style-type: none"> ■ 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. |
| Authentication mode | <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. |

| Parameter | Description |
|--|---|
| | <ul style="list-style-type: none"> ■ 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. |
| 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. |

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 request or response payloads 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 Payload | Logs all request payloads. |
| Store Response Payload | Logs all response payloads. |
| Compress Payload Data | Compresses the logged payload data. |
| 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 ■ CentraSite |

Note: This option is applicable only for the APIs published from CentraSite to API Gateway.

| Parameter | Description |
|-----------|--|
| | <ul style="list-style-type: none"> <li data-bbox="639 323 846 352">■ Elasticsearch <li data-bbox="639 380 1300 443">■ Email (you can add multiple email addresses by clicking ). <div data-bbox="688 464 1338 663" 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"> <li data-bbox="639 684 1317 814">■ 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="688 835 1338 1640" style="background-color: #f0f0f0; padding: 10px; margin: 10px 0;"> <p>Note:</p> <ul style="list-style-type: none"> <li data-bbox="808 852 1317 1402">■ 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. <li data-bbox="808 1423 1317 1625">■ 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>. </div> |

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 for a specific application.

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 API was available to the specified clients in the current interval. ■ Average Response Time. Indicates the average time taken by the service to complete all invocations in the current interval. ■ Fault Count. Indicates the number of faults returned in the current interval. ■ Maximum Response Time. Indicates the maximum time to respond to a request in the current interval. ■ Minimum Response Time. Indicates the minimum time 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 | Specifies the alert value for which the monitoring is applied. |
| 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 |

| Parameter | Value |
|-----------------------|---|
| | <p data-bbox="683 323 1317 394">Note: This option is applicable only for the APIs published from CentraSite to API Gateway.</p> <ul style="list-style-type: none"> <li data-bbox="623 426 829 453">■ Elasticsearch <li data-bbox="623 478 1284 550">■ Email (you can add multiple email addresses by clicking ). <p data-bbox="683 581 1325 751">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> <ul style="list-style-type: none"> <li data-bbox="623 783 1325 919">■ 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 data-bbox="683 951 1338 1503">Note: ■ 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.</p> <ul style="list-style-type: none"> <li data-bbox="748 1524 1308 1728">■ 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]. |
| Alert Interval | Specifies the time period in which to monitor performance before sending an alert if a condition is violated. |

| Parameter | Value |
|------------------------|---|
| Unit | Specifies the unit for the time interval in minutes, hours, days, or weeks for the alert interval. |
| 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. |
| 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.

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 API was available to the specified clients in the current interval ■ Average Response Time. Indicates the average time taken by the service to complete all invocations in the current interval. |

| Parameter | Value |
|--------------------|--|
| | <ul style="list-style-type: none"> ■ Fault Count. Indicates the number of faults returned in the current interval. ■ Maximum Response Time. Indicates the maximum time to respond to a request in the current interval. ■ Minimum Response Time. Indicates the minimum time 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"> ■ Elasticsearch ■ Email (you can add multiple email addresses by clicking ). <div style="background-color: #f0f0f0; padding: 5px; margin: 5px 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> |

| Parameter | Value |
|------------------------|--|
| | <ul style="list-style-type: none"> <li data-bbox="621 323 1328 457">■ 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="672 478 1336 1283" style="background-color: #f0f0f0; padding: 10px;"> <p data-bbox="683 491 1336 1045">Note:■ 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.</p> <ul style="list-style-type: none"> <li data-bbox="743 1066 1312 1268">■ 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> |
| Alert Interval | Specifies the time period (in minutes) in which to monitor performance before sending an alert if a condition is violated. |
| Alert Frequency | <p data-bbox="621 1465 1304 1562">Specifies how frequently to issue alerts for the counter-based metrics (Total Request Count, Success Count, Fault Count).</p> <p data-bbox="621 1583 946 1617">Select one of the options:</p> <ul style="list-style-type: none"> <li data-bbox="621 1638 1312 1709">■ Only Once. Triggers an alert only the first time one of the specified conditions is violated. <li data-bbox="621 1730 1312 1801">■ 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. |

| Parameter | Value |
|------------------------------|---|
| Consumer Applications | <p>Specifies the application to which this Service Level Agreement applies.</p> <p>You can type in a search term to match and application and click  to add it.</p> <p>You can add multiple applications or delete an added application by clicking .</p> |

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 action to avoid overloading the back-end services and their infrastructure, to limit specific clients in terms of resource usage, and so on.

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 | <p>Specifies the operator that connects the rule to the value specified.</p> <p>Select one of the operators: Greater Than, Less Than, Equals To.</p> |
| Value | Specifies the value of the request count beyond which the policy is violated. |
| 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 |

Note: This option is applicable only for the APIs published from CentraSite to API Gateway.

| Parameter | Description |
|-----------------------|---|
| | <ul style="list-style-type: none"> <li data-bbox="613 323 821 352">■ Elasticsearch <li data-bbox="613 375 1276 443">■ Email (you can add multiple email addresses by clicking ). <div data-bbox="667 464 1338 667" style="border: 1px solid #ccc; padding: 5px; margin: 5px 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"> <li data-bbox="613 682 1317 814">■ 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="667 835 1338 1646" style="border: 1px solid #ccc; padding: 5px; margin: 5px 0;"> <p>Note: ■ 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.</p> <ul style="list-style-type: none"> <li data-bbox="740 1423 1300 1625">■ 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>. </div> |
| 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. |

| Parameter | Description |
|------------------------------|---|
| Alert Frequency | <p>Specifies the frequency at which the alerts are issued.</p> <p>Specify one of the options:</p> <ul style="list-style-type: none"> ■ Only Once. Triggers an alert only the first time the specified condition is violated. ■ Every Time. Triggers an alert every time the specified condition is violated. |
| 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 in 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.

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. |

| Parameter | Description |
|------------------|--|
| HTTP Header | <p>Uses the HTTP header in the API request. 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 this field is empty, all the values that satisfy the criterion are cached.</p> |
| Query Parameters | <p>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. You can use this criterion for REST-based API requests.</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>Uses the XPath expression in the API request. You can use this criterion for SOAP-based API requests whose payload is a SOAP envelope.</p> <ul style="list-style-type: none"> ■ Name Space. Specifies the namespace of the XPath expression. <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> |

| Parameter | Description |
|--|---|
| | <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 this field is empty, all the values that satisfy the criteria 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>The default time format is minutes if the input is a number.</p> |
| Maximum Response Payload Size (in KB, -1=unlimited) | <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 before it is submitted to the application. This is required to 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 Integration Server
- XSLT Transformation
- Validate Schema

Invoke webMethods Integration Server

This policy pre-processes the native API's response messages into the format required by the application, before API Gateway returns the responses to the application.

The table lists the properties that you can specify for this policy:

| Parameter | Description |
|------------------------------------|---|
| webMethods IS Service | <p>Specifies the webMethods IS service to be invoked to pre-process the response messages.</p> <p>Note: The webMethods IS service must be running on the same Integration Server as API Gateway.</p> <p>You can add multiple entries by clicking  .</p> |
| webMethods IS Service alias | <p>Specifies the webMethods IS service alias to be invoked to pre-process the response messages.</p> <p>You can add multiple entries by clicking  .</p> |

XSLT Transformation

This policy specifies the XSLT transformation file to transform response messages from native APIs into a format required by the client.

The table lists the properties that you can specify for this policy:

| Parameter | Description |
|----------------------------------|---|
| XSLT Document | <p>Click + Add xslt document to add an xslt document.</p> <p>Provide the following information:</p> <ul style="list-style-type: none"> ■ XSLT file. Specifies the XSLT file used to transform the response messages as required. <p>Click Browse to browse and select a file.</p> <p>In the XSLT Features section, provide the following information:</p> <ul style="list-style-type: none"> ■ Feature name. Specifies the name of the XSLT feature. ■ Feature value. Specifies the value of the XSLT feature. <p>You can add more XSLT features and xslt documents by clicking  .</p> |
| XSLT Transformation alias | <p>Click + Add xslt transformation alias to add an xslt transformation alias in the XSLT Transformation alias field.</p> <p>You can add more XSLT transformation aliases by clicking  .</p> |

Validate Schema

This policy validates the responses from an application against a schema referenced in the WSDL. The response sent to the API by an application should conform to the structure or format expected by the API. The incoming responses are validated against an XML schema specified in this policy to conform to the structure or format expected by the API. An XML schema precisely defines the elements and attributes that constitute an instance XML document. The XML schema also specifies the data types of these elements to ensure that only appropriate data is allowed through to the API. For example, an XML schema might stipulate that all responses from a particular API must contain a <name> element, which contains at the most a ten-character string. If the response contains a message with an improperly formed <name> element, there is a policy violation and the message is rejected.

For a REST API, the schema can be added inline or uploaded in the Technical information section on the API details page.

Note: For REST API, the schema validation works only for XML schema provided the schema is mapped to the resource. The schema validation does not work for JSON schema. For a JSON response to be validated against a schema, the schema has to be defined with a fake main node element as shown below:

```
<xs:schema attributeFormDefault="unqualified"
           elementFormDefault="qualified"
           xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="fake-main-node">
    <!-- Schema Definition goes here -->
  </xs:element>
</xs:schema>
```

For a SOAP API, the WSDL and the referenced schema have to be provided in a zip format.

The run-time invocations that fail the schema validation are considered as policy violations. Such policy violation events that are generated can be viewed in the dashboard.

The table lists the schema properties that you can specify for this policy:

| Parameter | Description |
|----------------------|--|
| Feature name | Specifies the name for the schema configuration. For example - TOLERATE_DUPLICATES, NAMESPACE_GROWTH |
| Feature value | Specifies whether the feature value is TRUE or FALSE. |

You can add multiple entries for the feature names and values by clicking  .

Error Handling

The policy in this stage enables you to specify the error conditions and lets you determine how these error conditions are to be processed. The policy included in this stage is:

- Conditional Error Processing

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. |
| XPath Expression | Provide the details of the XPath expression in the API request. Provide the following information: <ul style="list-style-type: none"> ■ XPath Expression. Specifies the XPath expression. ■ Namespace Prefix. Specifies the prefix for the namespace. ■ Namespace URI. Specifies the namespace URI. You can add multiple entries for the namespace prefix and namespace URI by clicking  . |
| Pre-Processing. | Specifies how the native service error response is to be processed before sending the response to API Gateway. |
| Invoke webMethods Integration Server Service | Specify the webMethods IS service. Provide the webMethods IS Service to be invoked to pre-process the request messages. You can add multiple entries for webMethods IS service by clicking  . |

| Parameter | Description |
|--------------------------------|---|
| 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 a 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> |
| Custom Error Variables. | Specifies the error variables to be used in the custom error message. |
| 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 | Provide a name for the payload type. |
| XPath Expression | Provide the details of the XPath expression in the API request. |
| Namespace | <p>Specifies the namespace of the XPath expression.</p> <p>Provide the following information:</p> <ul style="list-style-type: none"> ■ Namespace Prefix. Specifies the prefix for the namespace. ■ Namespace URI. Specifies the namespace URI. <p>You can add multiple entries for the namespace prefix and namespace URI by clicking  .</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. |

| Parameter | Description |
|--|---|
| Send Native Provider Fault Message | <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 message sent by the native service is to be processed before sending the same to the application.</p> | |
| Invoke webMethods Integration Server Service | <p>Specify the webMethods IS Service.</p> <p>Provide the webMethods IS Service that should be invoked to post-process the request messages.</p> |
| 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> |

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 (predefined or custom). | PROTOCOL_HEADERS mx: CUSTOM_VAR |
| serValue | out | Object ref | Java.io.serializable value. (Usually a string). | |

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 Type | Data Type | Description | Examples |
|----------------|---------------|------------|---|----------|
| MessageContext | in | Object ref | This object is inserted into the pipeline by API Gateway. | N/A |

| Parameter | Pipeline Type | Data Type | Description | Examples |
|-----------|---------------|------------|---|-----------------------------------|
| varName | in | String | Context variable name (predefined or custom). | PROTOCOL_HEADERS mx:CUSTOM_VAR |
| serValue | in | Object ref | Java.io.Serializable 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 Type | Data Type | Description |
|-----------|---------------|------------|---|
| ctxVar | in | Object ref | The document type defining the context variable object. Use the <code>ctxVar</code> 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 <code>schema_type</code> (string or int), and <code>isReadOnly</code> (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 <code>QName</code> 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 Type | 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. | mx: CUSTOM_VAR |

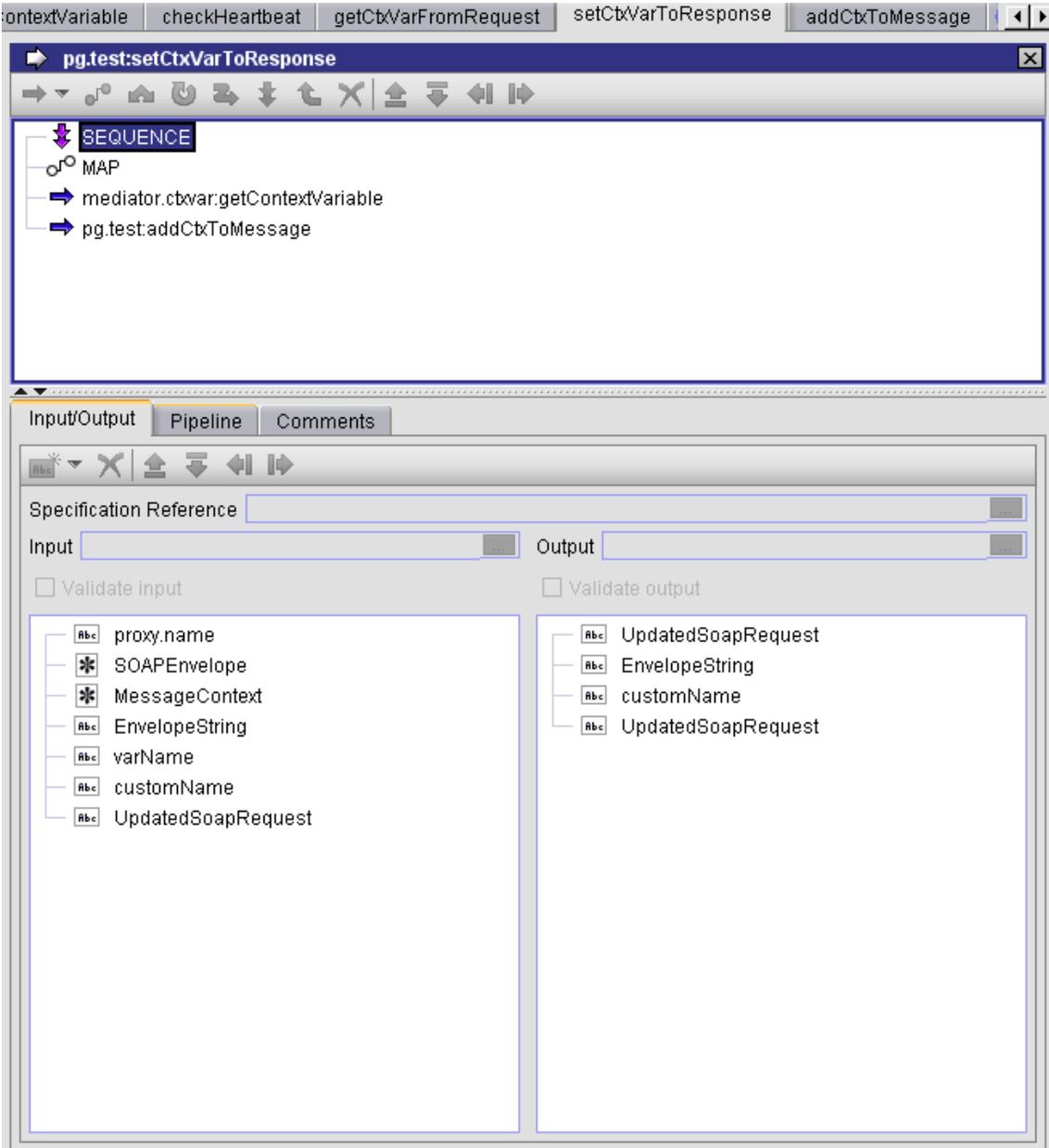
Sample Flow Service: Getting a Context Variable Value

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



We define the `customName` variable value to be `mx:COMP_TEST` so we 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

The screenshot shows the configuration of a policy named `pg.test:setCtxVarToResponse`. The policy is a **SEQUENCE** containing a **MAP** transformer. The **MAP** transformer has two entries:

- `mediator.ctxvar:getContextVariable`
- `pg.test:addCtxToMessage`

Below the policy configuration, the **Input/Output** tab is selected, showing the pipeline variables. The **Pipeline In** variables are:

- `proxy.name`
- `SOAPEnvelope`
- `MessageContext`
- `EnvelopeString`
- `varName`
- `customName`
- `UpdatedSoapRequest`

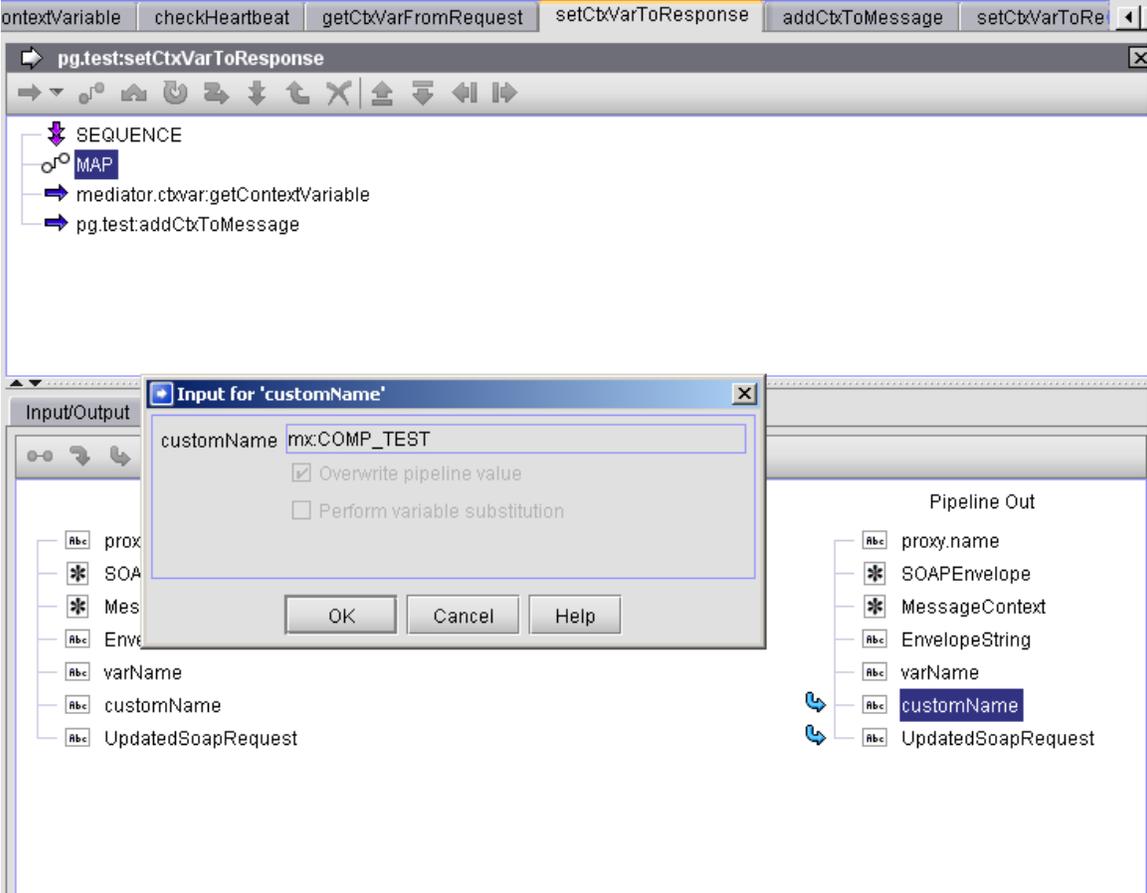
The **Pipeline Out** variables are:

- `proxy.name`
- `SOAPEnvelope`
- `MessageContext`
- `EnvelopeString`
- `varName`
- `customName`
- `UpdatedSoapRequest`

The `UpdatedSoapRequest` variable in the **Pipeline Out** section is highlighted with a blue selection box, indicating it is the current focus.

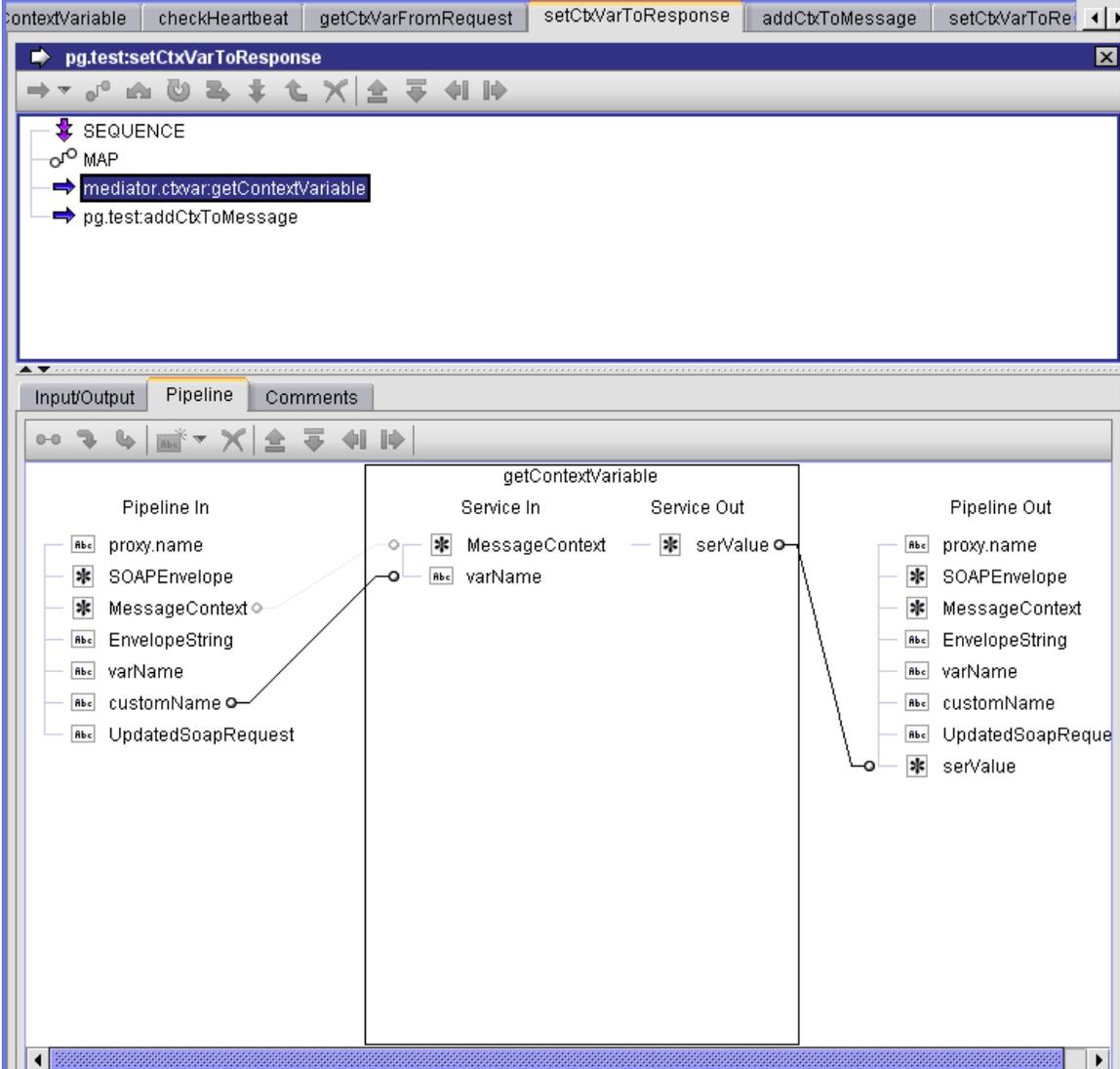
Clicking on the `customName` pipeline variable displays the name.

Step 3. Displaying the value of customName



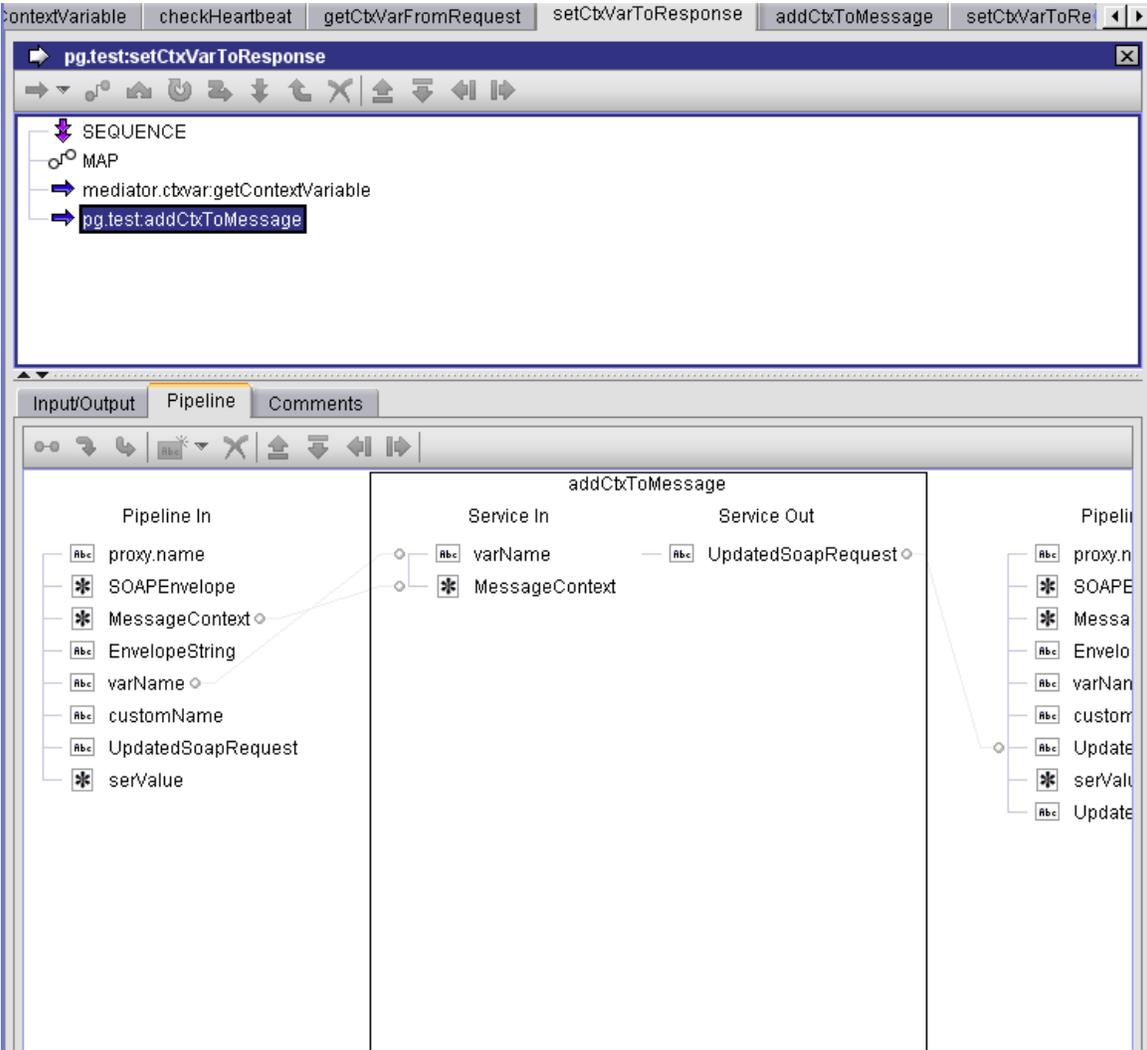
The call to `pub.mediator.ctxvar:getContextVariable` retrieves the value of the custom context variable from the context variable map.

Step 4. Calling 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



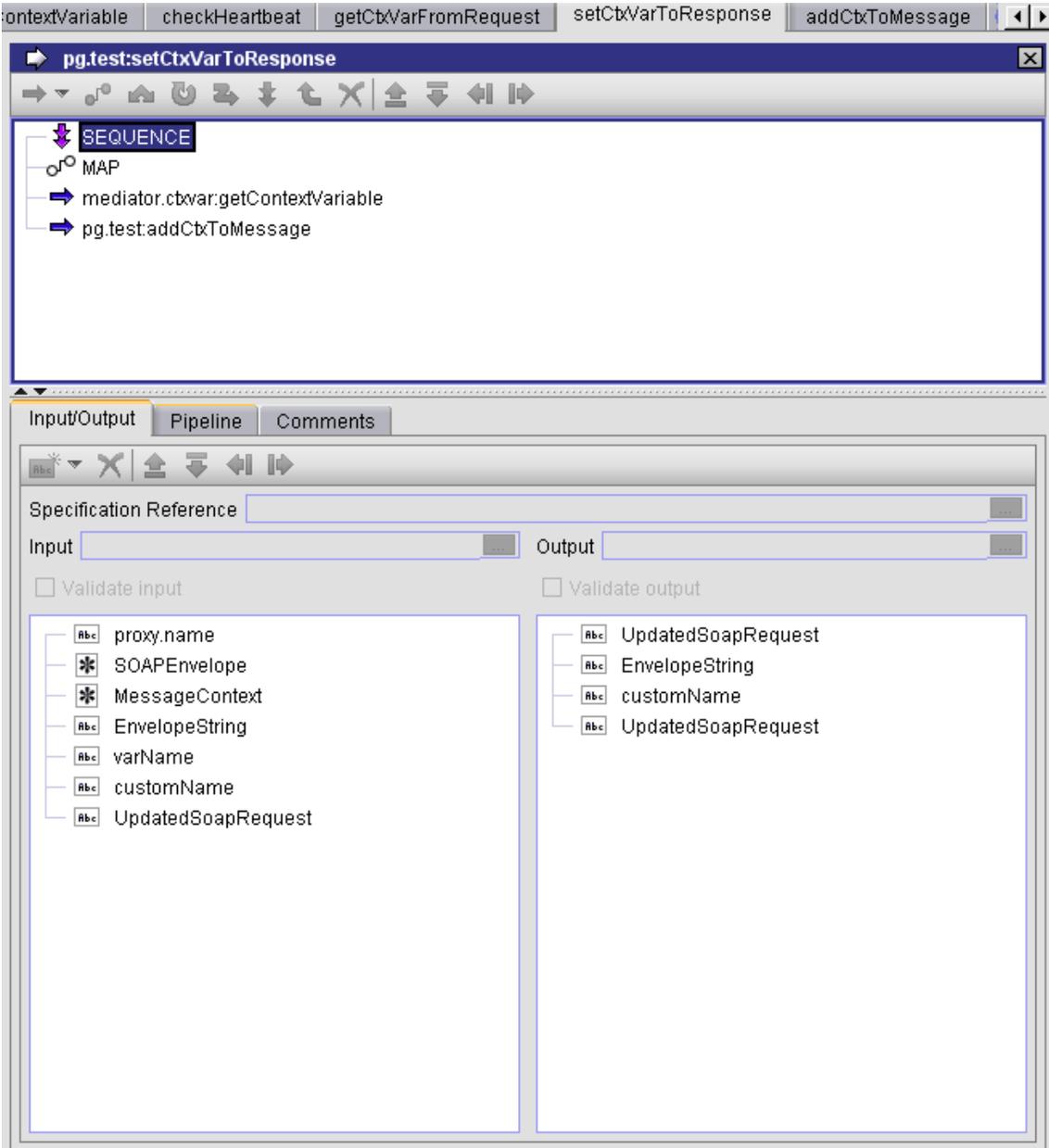
Sample Flow Service: Setting a Context Variable Value

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`.

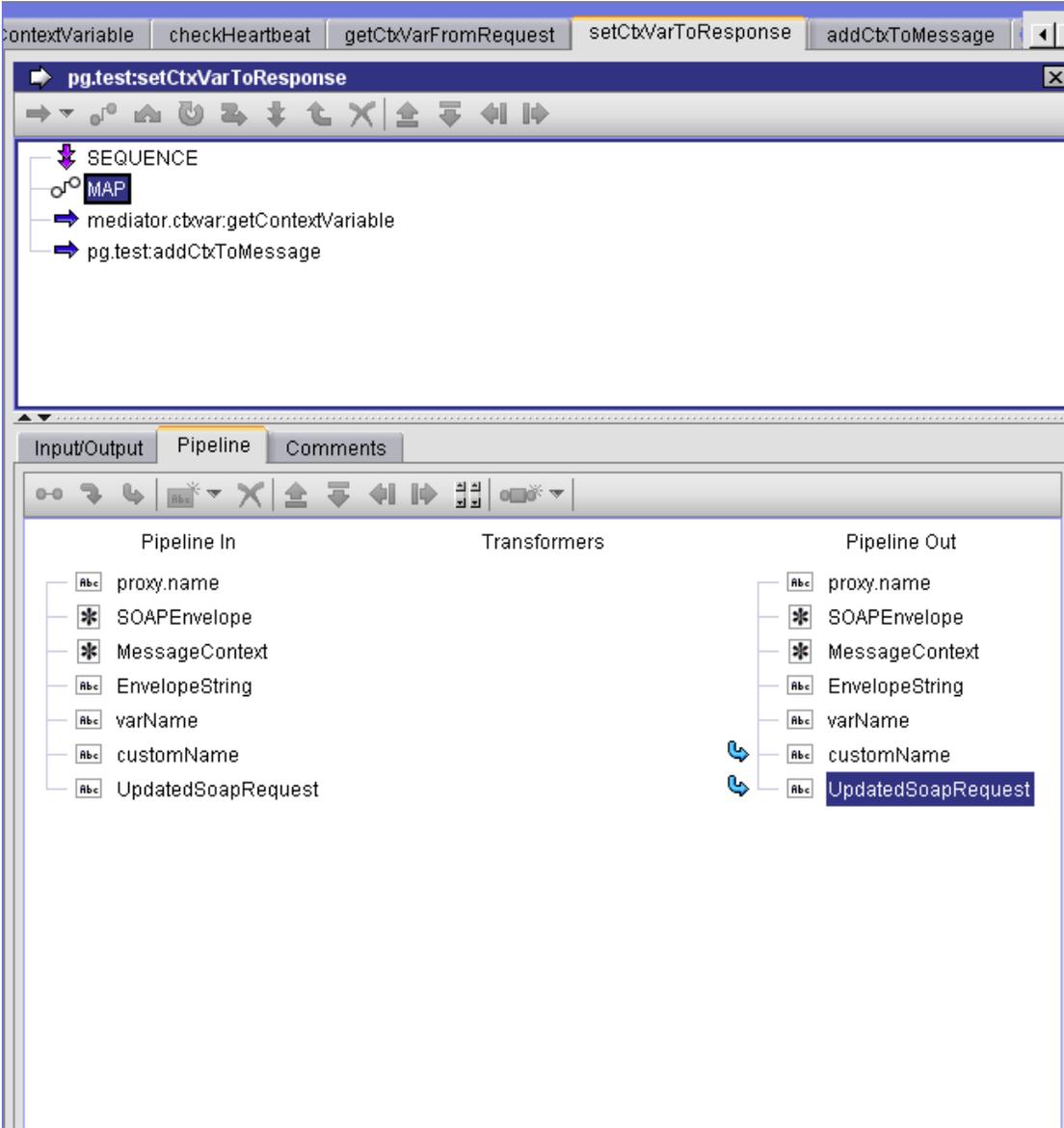
This flow service will retrieve the context variable for `customName` and create an element for its context variable value in the response message return to the consumer.

Step 1. Declaring customName



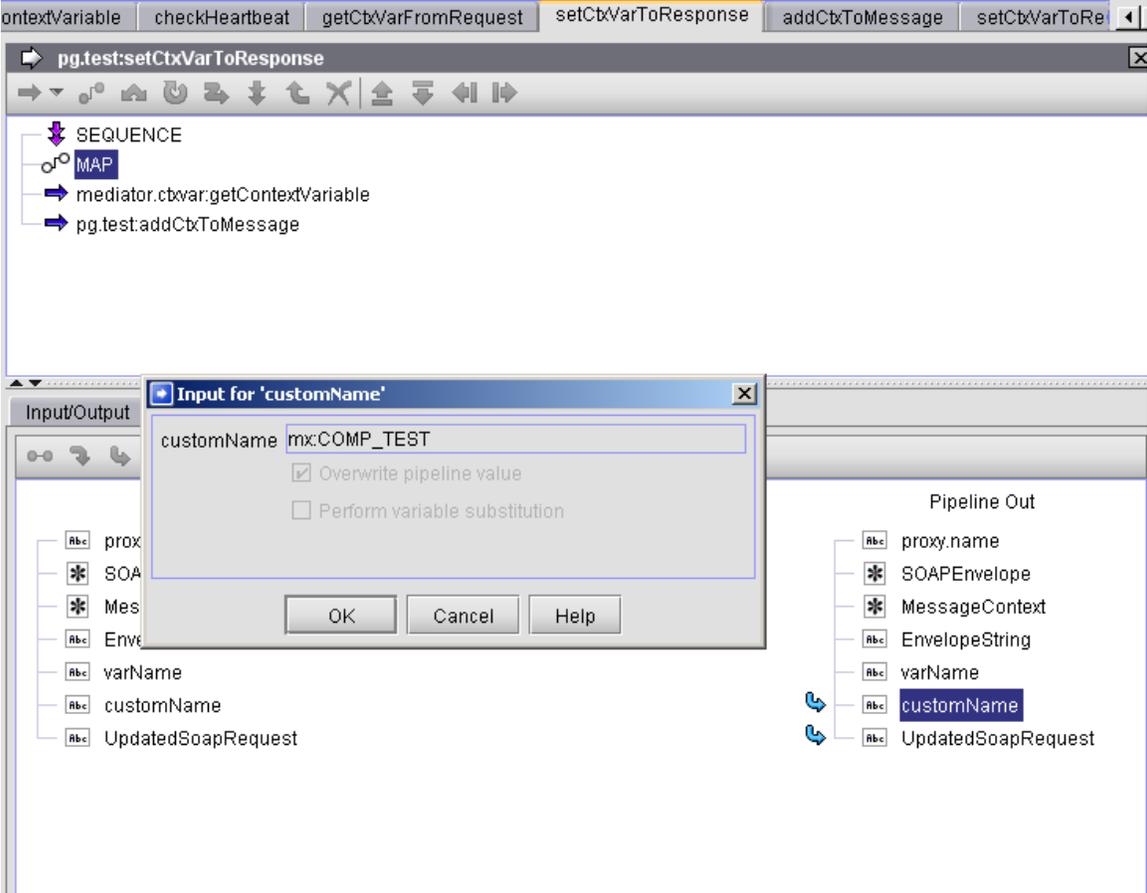
We define the `customName` variable value to be `mx:COMP_TEST` so we can use this variable to lookup the custom variable name that was seeded in the previous example.

Step 2. Setting customName to mx:COMP_TEST



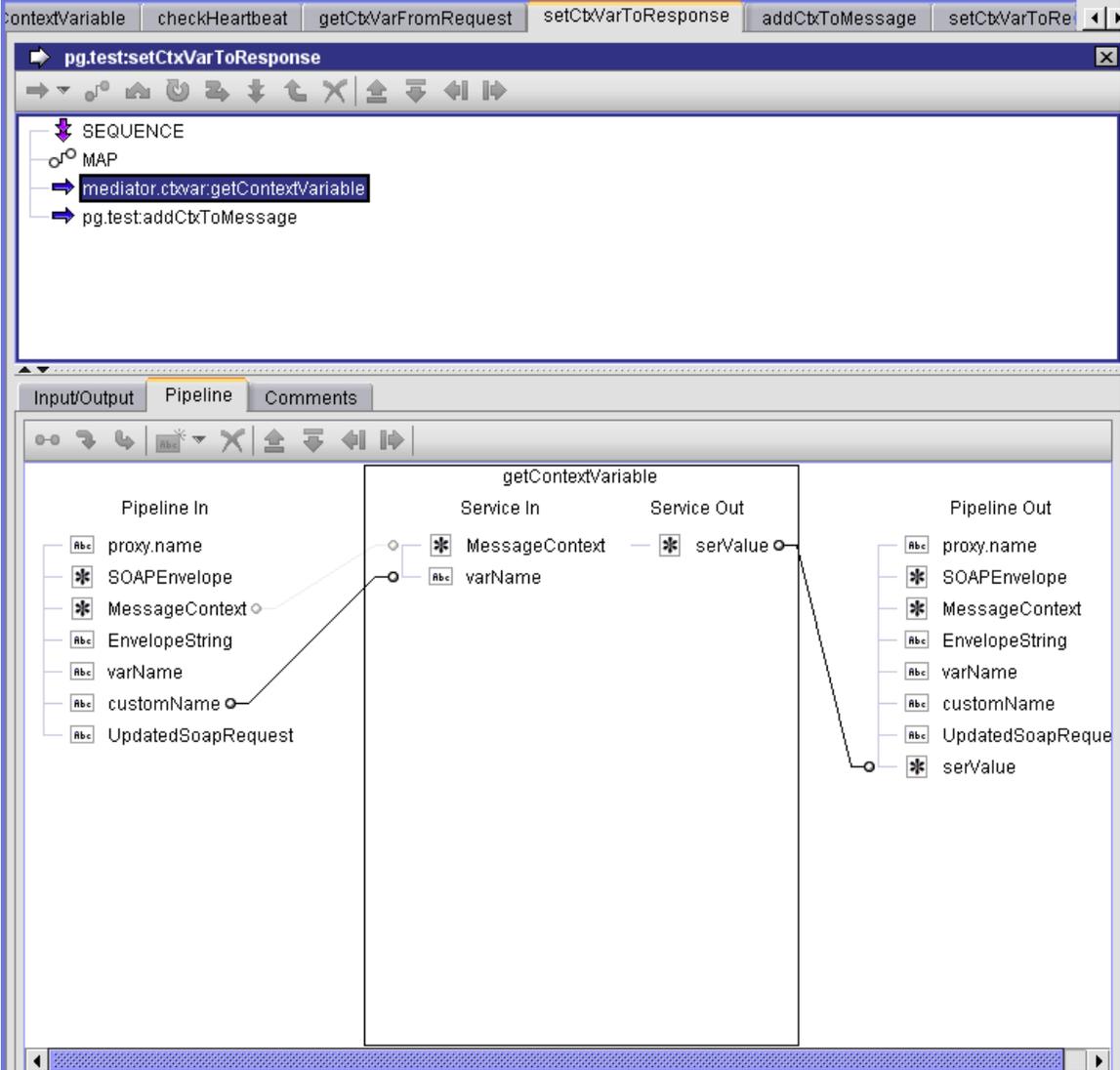
Clicking on the customName pipeline variable will display the name.

Step 3. Displaying the value of customName



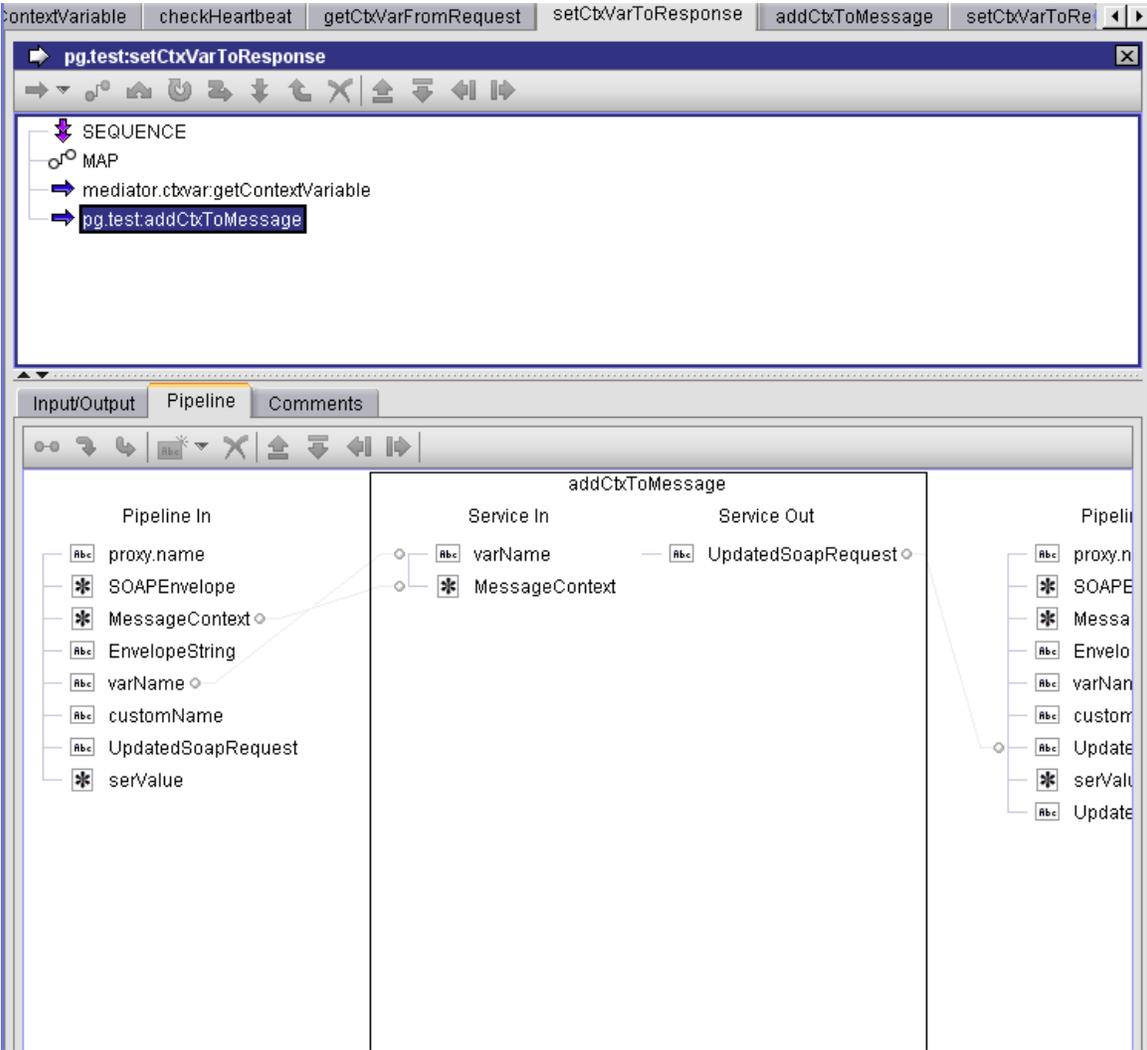
The call to `pub.mediator.ctxvar:getContextVariable` retrieves the value of the custom context variable from the context variable map.

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 Threat Protection Policies

Threat protection policies prevent malicious attacks on 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, maximum node depth and text node length, in the XML document.

The threat protection policies apply to an API globally for all requests coming into API Gateway. These policies are executed only for requests coming to the external port of API Gateway. You can configure the global threat protection rules to filter requests that API Gateway receives. You can also configure API Gateway to send an alert when a request violates a rule. When a rule is configured to send an alert and a violation occurs, API Gateway logs the details and generates an alert. The alert message contains

detailed information that includes the IP address from which the request was sent, user information, and the name of the rule filter that matched.

API Gateway applies rules in the order in which they are displayed on the Global policies screen. Because a violation of a denial rule causes API Gateway to stop processing a request, it is important to prioritize the rules based on the order in which you want them to be evaluated. The server processes denial rules before alert rules.

You must have the API Gateway's manage threat protection functional privilege assigned to perform this task.

- Global Denial of Service
- Denial of Service by IP
- Rules

In addition, the API Gateway administrator can configure the necessary mobile devices and applications, configure alert options, and manage the IPs that are denied access.

Note: If you have deployed API Gateway in a paired gateway deployment scenario with multiple instances of API Gateway connected using a load balancer for threat protection, when you make a change in enforced rules on one of the API Gateway instances you have to restart the other instances to synchronize the rule enforcement across all the API Gateway instances.

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 consider the total number of requests from all IP addresses, or to consider the number of requests from individual IP addresses. For example, you might want to limit the total number of requests received to 10 requests in 10 seconds, and limit the number of requests coming from any single IP address to 2 requests in 10 seconds. When API Gateway detects that a limit has been exceeded, it sends an alert. Depending on your configuration, API Gateway can temporarily block requests from all clients, or deny requests from particular IP addresses.

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 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. 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

This policy is configured to ensure that requests from trusted IPs are not denied. You can configure a list of IP addresses so that requests from these IP addresses are always allowed. You can configure a time interval for the maximum number of requests that can be processed from an IP address or a range of IP addresses and when this limit of maximum number is exceeded the IP is moved to the Denied IP List.

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 IPs that were denied access due to breach of one of the policies for denial of service policies. You can delete the IP from the list and make it available.

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 IPs or IP address range that were denied 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, requests from specific mobile devices and applications that could be harmful, requests that could cause an SQL injection attack, or use custom filters to avoid malicious attacks.

The logs generated by the threat protection rules are logged under Security log in Integration Server. This logging action is disabled by default, you have to enable it to log the threat protection logs. For details on how to enable the this logging action in Integration server, see *webMethods Integration Server Administrator's Guide*.

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 default alert settings to be used.
 - **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. Type `pub.apigateway.threatProtection:violationListener` that is used as the signature of the flow service and provide the user who has permissions to execute the service, 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. |

| Field | Description |
|---------------------------------------|--|
| | 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. |
| Namespace URI length | Specifies a character limit for any namespace URIs present in the XML document. |

| Field | Description |
|---|---|
| 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. |

| Field | Description |
|-------|-------------|
|-------|-------------|

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.

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.
 - Type `pub.apigateway.threatProtection:violationListener` that is used as the signature of the flow service.
 - Provide `Administrator` as the user type.
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**.

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.

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.

This displays a list of global policies available in API Gateway.

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 264](#) and [" Refining the Scope of a Global Policy" on page 265](#).

- 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 267](#) and [" Configuring Properties for a Global Policy" on page 268](#).
- c. Activate the global policy when you are ready to put it into effect. For details, see [" Activating a Global Policy" on page 272](#).

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.
This displays a list of global policies available in API Gateway.
3. Click the name of the required policy.
This opens the Global Policy details page.
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**, or both) to which the policy will be applied.
 - b. *Optional.* In the Filter using... 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 265](#).

- Click **Save** to save the modified policy.

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.

This displays a list of global policies available in API Gateway.

3. Click the name of the required policy.
This opens the Global Policy details page.
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.
This displays a list of global policies available in API Gateway.
3. Click the name of the required policy.
This opens the Global Policy details page.
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 **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 global policy, see "[Configuring Properties for a Global Policy](#)" on page 268.
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.
10. 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.

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.

This displays a list of global policies available in API Gateway.

3. Click the name of the required policy.
This opens the Global Policy details page.

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.

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 out of 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.
10. 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.

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

1. Click **Policies** in the title navigation bar.
2. Click the **Global Policies** tab.

This displays a list of global policies available in API Gateway.

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. Click the name of the required policy whose details you want to examine.

This opens the Global Policy details page. 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 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.

This displays a list of global policies available in API Gateway.

3. Click the name of the required policy.

This opens the Global Policy details page. 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 273.
- 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.
This displays a list of global policies available in API Gateway.
3. Click the name of the required policy.
This opens the Global Policy details page.
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 272.

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.

This displays a list of global policies available in API Gateway.

3. Click the name of the required policy.

This opens the Global Policy details page.

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.

This displays a list of global policies available in API Gateway.

3. Click the **Delete**  icon adjacent to 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. When you are prompted to confirm the delete operation, click **Yes** in the confirmation dialog box.

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.

This displays a list of global policies available in API Gateway.

3. Click the **Copy** icon adjacent to 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.

Viewing Global Policy Details in an API

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. |
|  | 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. |
|  | Policy is applied for the API's scope. This policy is applicable across a set of resources / methods / operations of that particular API. |

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 global policy details in an 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 the **Policies** tab.
This displays the Infographic view of the policies configured for the particular API.

Exporting Global Policies

You must have the API Gateway's export assets functional privilege assigned.

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 an API in the list of APIs displayed.
This displays the API Details page by default.
3. Click the **Policies** tab.
4. Select the policy stage and click 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 the List of Policies Assigned to an API

To view the list of policies assigned to an API

1. Click **APIs** in the title navigation bar.
2. Select an API in the list of APIs displayed.
This displays the API Details page by default.

3. Click the **Policies** tab.

4. Click  .

This displays all the policies enforced on the selected API.

Modifying a Policy Assigned to an API

Ensure that the API is in **Edit** mode before you modify a policy that is assigned to the API.

To modify a policy assigned to an API

1. Click **APIs** in the title navigation bar.
2. Select an API in the list of APIs displayed.
This displays the API Details page by default.
3. Click the **Policies** tab.
4. Select the policy stage and click the required policy.

The policy is displayed in the infographic with its properties displayed in properties section.

5. Modify the properties as required.
6. Click **Save**.

The modified policy assigned to the API is saved.

Modifying Policies Assigned to an API

Ensure that the API is in **Edit** mode before you modify the policies assigned to the API.

To modify policies assigned to an API

1. Click **APIs** in the title navigation bar.
2. Select an API in the list of APIs displayed.
This displays the API Details page by default.
3. Click the **Policies** tab.
4. Select the policy stage, and click the required policy.
You can do any of the following:
 - To assign a new policy, click on the policy and provide the required properties.
 - To remove the policy assigned, click **x** on the policy displayed in the infographic.
5. Click **Save**.
The API with the new policies assigned is saved.

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.

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
- Traffic Monitoring

For information on the usage scenarios of policies configured for the scopes of an API, see ["Example: Usage Scenarios of API Scopes" on page 149](#).

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.

Note: You cannot create a scope-level policy in an active API. If the API in which you want to create a scope-level policy is in **Active** state, you must deactivate it.

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 prompts you to deactivate it.
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. Set the properties of the displayed policy, and then click **OK**.

To exit out of full-screen mode, click the **Minimize** icon.

10. When the policy list is complete and you have configured all of the properties for the scope-level policy correctly, click **Save** to create the new scope-level policy.

11. 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.

Post-requisites:

Activate the API when you are ready 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 prompts you to deactivate it.
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. Modify the properties of the displayed policy, and then click **OK**.

To exit full-screen mode, click the **Minimize** icon.

- When you have completed the required modifications for the scope-level policy, click **Save** to save the updated scope-level policy.

- 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.

- When you are ready to put the API into effect, activate it.

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:

- You cannot remove a scope-level policy in an active API. If the API in which you want to remove a policy at the scope-level is in the **Active** state, you must deactivate it.
- 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

- Click **APIs** in the title navigation bar.

This displays a list of APIs available in API Gateway.

- Click the name of the required API.

This opens the API details page.

- Click **Edit**.

If the API is active, API Gateway prompts you to deactivate it.

- Click the **Policies** tab.

This displays a list of scopes and policies available with the API.

- In the **API Scope** box, select the scope whose policy you want to remove.

- On the Infographic section, click the **Delete** (X) icon in any individual policy to remove that particular policy from the scope.

- When you have removed the policy, click **Save** to save the updated scope-level policy.

8. 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.

Post-requisites:

Activate the API when you are ready 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.
This displays a list of policy templates available in API Gateway.
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. |

- Click **Continue to policy configuration**.
- 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 285 and "[Configuring Properties for a Policy Template](#)" on page 286.
- Click **Save** to save the new policy template.

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

- Click **Policies** in the title navigation bar.
- Click the **Policy Templates** tab.
This displays a list of policy templates available in API Gateway.
- Click the name of the required template.
This opens the Policy Template details page.
- Click **Edit**.
- 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
- 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.
- 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 286.
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.

This displays a list of policy templates available in API Gateway.

3. Click the name of the required template.
This opens the Policy Template details page.
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  **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.

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.

This displays a list of policy templates available in API Gateway. The **Policy Templates** 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. Click the name of the required policy template.

This opens the Policy Template details page. 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.

This displays a list of policy templates available in API Gateway.

3. Click the name of the required template.

This opens the Policy Template details page. 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.
This displays a list of policy templates available in API Gateway.
3. Click the **Delete**  icon adjacent to the required template.
4. When you are prompted to confirm the delete operation, click **Yes** in the confirmation dialog box.

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.
This displays a list of policy templates available in API Gateway.

- Click the **Copy** icon adjacent to the required template.
- 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. 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. |

- Click **Copy** to save the new policy template.
- 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

- Click **APIs** in the title navigation bar.
This displays a list of APIs available in API Gateway.
- Click the name of the required API.
This opens the API details page.
- Click **Edit**.
- 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 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 modify the details of a policy template on the API details page

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 prompts you to deactivate it.

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. Set the properties of the displayed policy, and then click **OK**.

To exit full screen mode, click the **Minimize** icon.

7. When you have completed the required modifications on the **Policies** tab of the API details page, click **Save** to save the updated API.

Post-requisites:

Activate the API when you are ready 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 290.

To save policy definition as policy template

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.

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** to save the new policy template.

6 Aliases

| | |
|--|-----|
| ■ Overview | 296 |
| ■ Creating a Simple Alias | 296 |
| ■ Creating an Endpoint Alias | 297 |
| ■ Creating an HTTP Transport Security Alias | 298 |
| ■ Creating a SOAP Message Security Alias | 302 |
| ■ Creating a webMethods Integration Server Service alias | 306 |
| ■ Creating an XSLT Transformation alias | 307 |

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, outbound authentication tokens, and outbound HTTP headers 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. You can create four 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. Select **Username > Aliases**.
2. Select **Create alias**.
3. In the Basic information section, provide the following information:

| Field | Description |
|-------------|------------------------------|
| Name | Name of the alias. |
| Type | Select Simple alias . |

| Field | Description |
|-------------|---------------------------|
| Description | Description of the alias. |

- Click **Technical information** and specify a value in the **Default value** field.

Note: You can specify multiple email addresses, if you are creating an email alias, for example, abc@gmail.com, test@gmail.com, and so on.

- Click **Save**.

Note: If you want to configure this alias in the routing policies, you can follow the syntax `${aliasname}`. For example, if you want to route it to an endpoint `http/mydevenv.com:7000/api`, you can create a simple alias with the name `mystage` and its value being `http/mydevenv.com:7000`. The endpoint URL can be specified in the properties as `${mystage}/api`.

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

- Select **Username > Aliases**.
- Select **Create alias**.
- 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. |

- 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 default value is 30 seconds.</p> |
| Read timeout | Specify the time interval (in seconds) after which a socket read attempt times out. |
| Keystore alias | Specify the keystore alias of the truststore that contains all the trusted public certificates. |
| Key alias | Specifies the key alias that is used by the SSL connection. |

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

To create an HTTP transport secure alias

1. Select **Username > Aliases**.
2. Select **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. ■ NTLM. Uses NTLM authentication. ■ OAuth2. Uses OAuth2 authentication. |

For the Authentication type **Basic**, authenticate using the following:

| | |
|---------------------------|--|
| Custom credentials | Specifies the values provided in the policy required to access the native API. |
|---------------------------|--|

| Field | Description |
|--|--|
| | <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 the LDAP or central user directory 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 the LDAP or central user directory 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 the LDAP or central user directory 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. |
| <p>For Authentication type NTLM, authenticate using any of the following:</p> | |

| Field | Description |
|--|--|
| Custom credentials | <p>Specifies the credentials that are required for the NTLM handshake.</p> <p>Provide the following information:</p> <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. 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 required 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. Select **Username > Aliases**.
2. Select **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 Specifies the values provided in the policy to be used to obtain the WSS username token to access the native API.

Provide the following information:

| Field | Description |
|-------|--|
| | <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> <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 the LDAP or central user directory 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. |
|--------------------------------------|--|

| Field | Description |
|---|--|
| | <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 the LDAP or central user directory 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> <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 the LDAP or central user directory 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 | |

| Field | Description |
|----------------------------------|--|
| 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. |

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 ESB 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. Select **Username > Aliases**.
2. Select **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 . |

| Field | Description |
|-------------|---------------------------|
| Description | Description of the alias. |

- Click **Technical information** and specify the webMethods Integration Server service name in the **Service name** field.

Note: The webMethods Integration Server service must be available in the Integration Server, to which the aliases are deployed.

- 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

- Select **Username > Aliases**.
- Select **Create alias**.
- 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. |

- Click **Technical information** and browse and select an XSLT style sheet in the **Select transformation file** field.
- Click **Save**.

7 Applications

| | |
|---|-----|
| ■ Overview | 310 |
| ■ Creating an Application | 311 |
| ■ Viewing List of Applications and Application Details | 313 |
| ■ Regenerating API Access Key | 313 |
| ■ Refreshing OAuth Token | 314 |
| ■ Modifying Application Details | 314 |
| ■ Registering an API with Consumer Applications from API Details Page | 315 |
| ■ Registering APIs with Consumer Applications from Application Details Page | 315 |

Overview

An application defines the precise identifiers by which messages from a particular consumer 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 consumers that are requesting an API.

The ability of API Gateway to relate a message to a specific consumer application enables it to:

- Control access to an API at run time (that is, allow only authorized consumer applications to invoke an API).
- Monitor an API for violations of a Service-Level Agreement (SLA) for a specified application.
- Indicate the consumer 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 consumer application. Example: `192.168.0.10`

This attribute is queried when the Identify and Authorize Application policy is configured to identify consumer applications using IP address.

- Claims set, which specifies one or more claims that identify requests from a particular consumer 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 consumer applications using a JWT token or an OpenID token.

- Consumer certificate, which specifies the X.509 certificates that identify requests from a particular consumer.

This attribute is queried when the Identify and authenticate consumer policy is configured to identify the consumer applications by a consumer certificate.

- Identification token, which specifies the host names, user names or other distinguishing strings that identify requests from a particular consumer application.

This attribute is queried when the Identify and authenticate consumer policy action is configured to identify consumer applications by host name, token, HTTP user name, and WSS user name.

An application has a partner ID, which is a definition for applications that can be leveraged within a B2B scenario.

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.

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.
 - **Description:** Type a description of the application.
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:
 - **IP address range:** Provide the IP address range. You can add more range options by clicking **+Add** and adding the required information.
 - **Partner identifier:** Provides the third-party partner's identity.
 - **Client certificates:** Click **Browse** and select the client certificate to be uploaded. You can add multiple certificates by clicking **+Add**.
 - **Claims:** Provide a set of claims for the JWT and OpenID clients. A claim is a unique identifying information that is defined as a name-value pair and consists of a **Claim name** and a **Claim value**. You can add more claims and claims sets by clicking **+Add** and adding the required information.
 - **Other identifiers:** Select one of the options and provide the required value:
 - **Hostname**
 - **Token**
 - **Username**
 - **WS-Security username**
 - **XPath**

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 **+Add** 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 and only then does API Gateway identifies the application.

8. Click **Save**.

The application is created and listed in the list of applications in the Manage applications page.

9. Select the application and click **Edit** to add the OAuth credentials.

10. Click **OAuth credentials**.

11. Provide the following information in the OAuth credentials section:

- **Type:** Select **Public** or **Confidential** as required.
- **Token lifetime:** Select **Unlimited** or **Limited** as required.

- **Token lifetime (in seconds):** Provide the time for which the token is active.
- **Token refresh limit:** Select **Unlimited** or **Limited** as required.
- **Token refresh limit (in seconds):** Provide the time for which the token refresh is applicable.
- **Redirect URIs:** Specifies the URIs that the authorization server uses to redirect the resource owner's browser during the grant process. You can add multiple URIs by clicking **+Add**.

12. Click **Save**.

Once the application is created successfully, you will be redirected to the created application's details page.

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 basic information, identifiers, access tokens, API key, OAuth credentials, and APIs registered for that application.

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, OAuth credentials, and APIs registered for that application.

3. Click .

The API access key is regenerated and the new API access key appears in the **API access key** field.

Refreshing OAuth Token

You must have the API Gateway's manage applications functional privilege assigned to perform this task.

You can refresh an OAuth token in the Application details page from where you can view application details.

To refresh an OAuth token

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, OAuth credentials, and APIs registered for that application.
3. Click .
New values for **Client ID**, **Client secret**, and the **client name** for OAuth2 Credentials are generated.

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. Click **OAuth credentials**.

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 displays the list of applications starting with the characters that you provide.
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 starting with the characters that you have typed in.

6. Select the required API and click **+**.

You can add more APIs in a similar way.

7. Click **Save**.

8 API Packages and Plans

| | |
|--|-----|
| ■ Overview | 318 |
| ■ Creating a Package | 318 |
| ■ Creating a Plan | 319 |
| ■ Viewing List of Packages and Package Details | 321 |
| ■ Modifying a Package | 321 |
| ■ Deleting a Package | 322 |
| ■ Activating a Package | 323 |
| ■ Publishing a Package | 323 |
| ■ Viewing List of Plans and Plan Details | 324 |
| ■ Modifying a Plan | 324 |
| ■ Deleting a Plan | 325 |

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.

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.

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 **API 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. |
| Description | A brief description for the API package. |
| Icon | An icon that is displayed for the API package. |

| Field | Description |
|-------|---|
| | 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.

- Click **Continue to add plans**.

Alternatively, click **Plans** in the left navigation pane.

- 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.

- Click **Continue to add APIs**.

Alternatively, click **APIs** in the left navigation pane.

- 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.

- Select the required APIs to be associated with the API 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.

- 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

- Click **API Packages** in the title navigation bar.
- Click **Create** in the Manage packages and plans section.
- Select **Plan**.
- Click **Create**.
- Provide the following information in the Basic information section:

| Field | Description |
|--------------------|---|
| Name | Name of the plan. |
| Version | Version assigned for the plan. |
| 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.

- Click **Continue to Pricing**.
Alternatively, click **Pricing** in the left navigation pane.
- 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 **Throttling traffic optimization** in the left navigation pane.
- Provide the following information in the Create Rule section:

| Field | Description |
|------------------------------|---|
| Maximum request quota | Specifies the maximum number of requests handled. Value provided should be an integer. |

| Field | Description |
|--------------------------|--|
| Interval | Specifies the value for the interval for which the maximum allowed quota is handled. |
| Interval unit | Specifies the unit of measurement of the interval. Example: minutes, seconds, and so on Value provided should be an integer. |
| Violation message | Specifies the text that is displayed when the policy is violated. |

- Click **Ok**.

This creates the rule and displays it in the list of rules. Click **Add rule** to add more rules. You can edit or delete the rules by clicking the **Edit** and the **Delete** icons respectively.

- 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

- Click **API 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.

- 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 **API 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 **API 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.
A confirmation dialog appears.
3. Click **Yes** to confirm deletion.

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 **API 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 **API 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.

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 **API 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 **API 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 **Throttling traffic optimization** in case 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 **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 **API 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.
A confirmation dialog appears.
4. Click **Yes** to confirm deletion.

9 Import Archives

- Importing Exported Files 328

Importing Exported Files

You can import already exported archives of APIs, global policies, and other related assets and re-create them in API Gateway. For more information about exporting APIs, see ["Exporting APIs" on page 158](#). For information about exporting global policies, see ["Exporting Global Policies" on page 276](#).

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.

Note: You must have the API Gateway's import assets functional privilege assigned to import an archive.

To import the exported files

1. Select **Username > Import**.
2. Provide the following information:

| Parameter | Description |
|---------------------|---|
| Select archive file | Click Browse to select a file or ZIP format file. |
| Advanced options | Select this option to overwrite either or all the following with the imported content: <ul style="list-style-type: none"> ■ API ■ Policy ■ Applications ■ Alias |

3. Click **Import**.

The **Import report** displays the following information:

| Parameter | Description |
|-----------|--|
| Type | The artifact type. The available values are: <ul style="list-style-type: none"> ■ API ■ Policy |

| Parameter | Description |
|---------------------|--|
| | <ul style="list-style-type: none"> ■ Policy Actions ■ Applications ■ Alias |
| 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 | <p>The artifact type. The available values are:</p> <ul style="list-style-type: none"> ■ API ■ Policy ■ Policy Actions ■ Application ■ Alias |
| Status | <p>The status of the imported artifact. The available values are:</p> <ul style="list-style-type: none"> ■ Success ■ Replaced |

| Parameter | Description |
|--------------------------|---|
| | <ul style="list-style-type: none"><li data-bbox="769 327 922 359">■ Warning<li data-bbox="769 380 902 411">■ Failure |
| <hr/> Explanation | The reason if the import fails or if a warning occurs. <hr/> |

10 API Gateway Analytics

| | |
|------------------------------------|-----|
| ■ Analytics Dashboards | 332 |
| ■ Purging API Analytics Data | 338 |

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, applications and package related event information. This can be accessed from the menu option in the right top corner of the API Gateway screen.
- 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 event 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 is shipped with 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, 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.
- 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.

- **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 events:** 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 selecting **Username > Analytics** in the top right corner. The dashboard displays the API Gateway-wide analytics based on the metrics monitored.

You can select the time interval and click **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 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 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 type as flow service in **Policies > Global threat protection > Alert settings** section. The threat protection analytics information is visible only if you select the Alert type as flow service in **Policies > Global 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 | <p>Displays the payload size of the request and responses during data transfer in the specified time.</p> <p>This data is picked up from the transactional event that is triggered when a log invocation policy is applied to the API.</p> |
| | Package performance | Displays a pie chart depicting package performance during the specified time. The different colours 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. |

| Category | Metric | Description |
|---------------------|---|--|
| | Overall error trends | Displays a graph depicting the performance of all the APIs in the system based on the error event 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. |

| Category | Metric | Description |
|--------------------------|--|--|
| | Trending subscription for package | Displays the trending subscriptions for the package based on the number of invocations. The different colours in the donut pie chart depict the trending behavior of the different applications in the package. |
| | 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. |
| | 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. |

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. |
| 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 | Displays the method level invocations per operation for the API during the specified time. |

| Metric | Description |
|--------|--|
| | 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. |

Purging API Analytics Data

The process of systematically deleting unwanted data from the database is called purging. You can purge events from the API Gateway store by setting the required date or duration in the API Gateway. You must have the API Gateway's manage purge functional privilege assigned to purge the API analytics data.

To purge API analytics data

1. Select **Username > Analytics**.
2. Click **Purge data**.
3. Select one of the following event types for which the data has to be purged:
 - **ALL**
 - **Policy violation events**
 - **Transaction events**
 - **Monitor events**
 - **Error events**
 - **Lifecycle events**
 - **Performance events**
 - **Threat protection events**
4. Select one of the following options:
 - Select **Until** and select a date by which you want the data to be purged. The rest of the data is retained.
 - Select **Duration** and type the duration value for which you want the data to be retained. The rest of the data is purged.
5. Click **Submit**.

The required data is purged.