

webMethods API Gateway User's Guide

Version 9.12

October 2016

This document applies to webMethods API Gateway Version 9.12 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 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..... | 5 |
| Document Conventions..... | 5 |
| Online Information..... | 6 |
| Administering API Gateway..... | 7 |
| Introduction..... | 8 |
| General Configuration..... | 9 |
| Clusters and Load Balancers..... | 9 |
| Configuring Load Balancer URLs..... | 10 |
| Configuring Extended Settings..... | 11 |
| Configuring Global Service Fault Settings..... | 12 |
| Security Configuration..... | 12 |
| Keystore and Truststore..... | 13 |
| Configuring Keystore Information..... | 13 |
| Ports..... | 14 |
| Adding an HTTP Port..... | 14 |
| Adding an HTTPS Port..... | 18 |
| Adding an API Gateway External Port..... | 22 |
| Configuring the API Gateway Internal listener..... | 26 |
| Destination Configuration..... | 29 |
| Configuring API Gateway as a Destination..... | 30 |
| Configuring Database as a Destination..... | 30 |
| Configuring API-Portal..... | 30 |
| Configuring API-Portal as a Destination..... | 32 |
| Alias Configuration..... | 32 |
| Creating a Simple Alias..... | 32 |
| Creating a Secure Alias..... | 33 |
| Creating an Endpoint Alias..... | 34 |
| Managing Users..... | 37 |
| User Roles..... | 38 |
| Managing APIs..... | 41 |
| Overview..... | 42 |
| Creating an API by Importing an API from a File..... | 42 |
| Creating an API by Importing an API from a URL..... | 43 |
| Creating an API from Scratch..... | 43 |
| Viewing API List and API Details..... | 47 |
| Publishing an API..... | 47 |
| Modifying API Details..... | 48 |
| Filtering APIs..... | 48 |
| Example: Managing an API..... | 49 |

| | |
|--|------------|
| Managing Policies..... | 61 |
| Policies..... | 62 |
| Managing Global Threat Protection Policies..... | 62 |
| Configuring the Global Denial of Service Policy..... | 63 |
| Configuring Denial of Service by IP Policy..... | 64 |
| Managing Denied IP List..... | 65 |
| Configuring Rules..... | 65 |
| Registering a Mobile Device or Application..... | 68 |
| Configuring Alert Settings..... | 68 |
| Managing API-level Policies..... | 69 |
| Transport Policies..... | 70 |
| Identity and Access Management Policies..... | 71 |
| Request Payload Processing Policies..... | 77 |
| Routing Policies..... | 78 |
| Logging, Monitoring, and Traffic Optimization Policies..... | 90 |
| Response Processing Policies..... | 95 |
| Error Handling..... | 96 |
| Managing Applications..... | 101 |
| Applications..... | 102 |
| Creating an Application..... | 103 |
| Viewing List of Applications and Application Details..... | 104 |
| Modifying Application Details..... | 105 |
| Registering an API with Consumer Applications from API Details Page..... | 105 |
| Registering APIs with Consumer Applications from Application Details Page..... | 106 |
| Managing API Packages and Plans..... | 107 |
| API Packages and Plans..... | 108 |
| Creating a Package..... | 108 |
| Creating a Plan..... | 109 |
| Viewing List of Packages and Package Details..... | 111 |
| Modifying a Package..... | 111 |
| Deleting a Package..... | 112 |
| Activating a Package..... | 113 |
| Publishing a Package..... | 113 |
| Viewing List of Plans and Plan Details..... | 114 |
| Modifying a Plan..... | 114 |
| Deleting a Plan..... | 115 |
| API Gateway Analytics..... | 117 |
| Analytics Dashboards..... | 118 |
| API Gateway Dashboard..... | 119 |
| API-specific Dashboard..... | 121 |
| Purging API Analytics Data..... | 122 |

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 consumers, 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 Administering API Gateway

| | |
|-----------------------------------|----|
| ■ Introduction | 8 |
| ■ General Configuration | 9 |
| ■ Security Configuration | 12 |
| ■ Destination Configuration | 29 |
| ■ Alias Configuration | 32 |

Introduction

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 from the API creation, policy definition and activation, creation of consumer applications and API consumption. API Gateway gives you rich dashboarding 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 both SOAP-based APIs as well as REST-based APIs, provides protection from malicious attacks, provides a complete run-time governance of APIs published to external destinations and information about Gateway specific events and API-specific events. API Gateway provides the following features:

- **Support for SOAP and REST APIs** - API Gateway supports both SOAP-based APIs as well as REST-based APIs. This support enables organizations to leverage their current investments in SOAP-based APIs while they adopt REST for new APIs.
- **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, and avoiding additional inbound firewall holes through the use of reverse invoke.
- **Mediation** - API Gateway provides complete run-time governance of APIs published to external destinations. API Gateway enforces access token and operational policies such as security policies for run-time requests between consumers and native services. API providers can enforce security, traffic management, monitoring, and SLA management policies, transform requests and responses into expected formats as necessary, perform routing and load balancing of requests, and collect events metrics on API consumption and policy evaluation.
- **Easy discovery and testing of APIs** - API Gateway provides full text search capabilities that helps developers 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 helps them decide whether to adopt a particular API. Developers can use the provided code samples and expected error and return codes to try out APIs they are interested in, directly from within API Gateway, to see how the API works.
- **Clustering Support** - Multiple instances of API Gateway can be clustered together to provide scalability of mediation. A load balancer can be placed in front of the clustered API Gateway instance to properly distribute the request messages.
- **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. This information is available by way of dashboards to API providers who have an API administrator role in API Gateway. With this information, providers can

understand how their APIs are being used, which in turn can help identify ways to improve their users' experience and increase API adoption.

- **Message transformation, pre-processing and post-processing** - 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.

API Gateway administration enables the general availability of the Gateway. Various configuration performed for API Gateway to be functional are:

- General configuration that consists of configuring the load balancer, extended settings, service fault settings, and aliases.
- Security configuration that consists of configuring keystores and ports.
- Destination configuration that consists of configuring various destinations to which the events and performance metrics data is sent.

You should have the privileges of API Gateway administrator for performing the various configurations.

General Configuration

An API Gateway Administrator can perform the following configurations in the general configuration section of API Gateway:

- Configure API Gateway to communicate with a load balancer that allows API Gateway to distribute the messages it receives across the API Gateway nodes.
- Configure the extended settings which are advanced parameters required for a proper operation of your server.
- Configure aliases that can be referred to in multiple policies so that a change in a particular alias would affect all the policy actions in which it is being referred.

All the configurations can be performed through the API Gateway user interface.

Clusters and Load Balancers

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

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

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

You can configure the load balancer URLs through the API Gateway user interface.

Load Balancer URLs

Load balancer URLs are the URLs of the load balancer where the requests are sent by the consumers. When an API is activated on a node of a cluster, the endpoint of the API is provided as the load balancer URL instead of the Gateway end point. API Gateway stores this load balancer URL endpoint. Since all nodes in the cluster use the same load balancer URL, API Gateway accepts messages from any node in the cluster as though they came from a single instance of API Gateway.

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

`http://<hostname>:<portnumber>`

or

`http://<IP-address>:<portnumber>`

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

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

Note: You can also configure the load balancer URL to use the HTTPS protocol.

Configuring Load Balancer URLs

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

To configure load balancer URLs through API Gateway user interface

1. Select **<Username> > Administration** in the top right corner.
2. Select **General > Load Balancer**.
3. Provide the following information:

| Field | Description |
|---------------------------------|--|
| Load Balancer URL (HTTP) | The primary HTTP load balancer URL and port number. For the URL, you can specify either the IP |

| Field | Description |
|----------------------------------|---|
| | address or host name of the load balancer with the port number, as follows: http://IP-address:portnumber or http://hostname:portnumber |
| Load Balancer URL (HTTPS) | The primary HTTPS load balancer URL and port number. For the URL, you can specify either the IP address or host name of the load balancer with the port number, as follows: http://IP-address:portnumber or http://hostname:portnumber |

4. Click **Save**.

Configuring Extended Settings

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

You should have the API Gateway Administrator privileges to configure the extended settings.

To configure extended settings

1. Select **<Username> > Administration** in the top right corner.
2. Select **General > Extended settings**.
3. Provide the following information. Type the API key header parameter value.

| Parameter | Description |
|--------------------------|--|
| APIKeyHeader | This parameter is used to specify the header from which API Gateway receives the API key. The default value is x-Gateway-APIKey |
| pg_xslt_enableDOM | Providing the value true enables DOM parsing. The value false disables the DOM parsing and enables other parsers |

| Parameter | Description |
|---|---|
| transformerPoolSize | Specifies the transformer pool size. |
| pg_oauth2_isHTTPS | Specifies the transport protocol over which the OAuth2 access tokens is granted authorization. Set this parameter to true for HTTPS (the default) or false for HTTP |
| apig_MENConfiguration_tickInterval | Specifies 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. Default value is 15. |

4. Click **Save**.

Configuring Global Service Fault Settings

You should have the API Gateway Administrator privileges to configure the service fault settings. You can configure global error responses for all APIs to display the default error message.

To configure service fault settings

1. Select **<Username> > Administration** in the top right corner.
2. Select **General > Service fault**.
3. Select **Send native provider fault** to send the native service provider's fault content, if available. API Gateway ignores the default error message and sends whatever content it received from the native service provider.

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

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

Security Configuration

An API Gateway administrator can perform the following in the security configuration section of API Gateway.

- Configure the keystores and truststores required for message-level security.

- Configure ports to map them to API Gateway.

Keystore and Truststore

Keystores and truststores are required for message-level security. They provide SSL authentication, encryption and decryption, and digital signing and verification services for all message content that API Gateway sends. You must first set up keystores and truststores in Integration Server (see *webMethods Integration Server Administrator's Guide* for more details) and then configure the keystore information in API Gateway.

For an API Gateway service to be SSL authenticated, 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.

SSL Keystore

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 therefore cannot be used for configuring SSL communication with API Gateway. API Gateway stores its private keys, and SSL certificates in keystore files, and the trusted roots for the certificates in truststore files. Keystores and truststores are secure files with industry-standard file formats in your production environments.

Configuring Keystore Information

To configure Keystore information

1. Select **<Username> > Administration** in the top right corner.
2. Select **General > Security**.
3. Provide the following information in the Keystore/Truststore section:

| Field | Description |
|-----------------------|---|
| Keystore name | A keystore that API Gateway uses. Lists all available keystores. If you have not configured an Integration Server keystore the list is empty. |
| Alias(Signing) | The signing alias. This alias is the value that is used to validate the inbound requests to API Gateway and to sign the outgoing response from API Gateway to the original service consumer. It is auto-populated based on the keystore selected. This field lists all |

| Field | Description |
|------------------------|---|
| | the aliases available in the chosen keystore. If there are no configured keystores, this field is empty. |
| Truststore name | The truststore used for establishing the HTTPS connection to the target service provider. It must contain the certificate of the service provider. |

4. Click **Save**.

Ports

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

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

If your API Gateway is behind an internal firewall and is not allowed to accept communications from external clients through the DMZ, you can configure the API Gateway external port to listen for requests from external clients. The API Gateway internal port passes on the requests from the external port to API Gateway thus safeguarding from any malicious attacks.

External clients send requests to API Gateway. API Gateway external port listens to this client information from each request and evaluates the request against any API Gateway rules that have been defined. It then passes requests that have not violated a rule to the API Gateway internal port. The Internal port processes the requests and sends responses to API Gateway, which then passes the responses back to the client.

You can create ports in Integration Server user interface and restrict only these ports to be used by API Gateway. For information about configuring ports in Integration Server, see *webMethods Integration Server Administrator's Guide*.

You can create additional ports on API Gateway user interface and these ports by default are mapped to the API Gateway package.

Adding an HTTP Port

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

To add an HTTP port

1. Select **<Username> > Administration** in the top right corner.

2. Select **Security > Ports**.

The ports page lists all the ports configured with API Gateway, if any.

3. Click **Add Ports**.

4. Select the type of port as **HTTP** click **Add**.

Various properties that need to be configured are displayed.

5. Provide the following information:

| Field | Description |
|------------------------------------|---|
| HTTP listener configuration | |
| Port | Specify the number you want to use for the port. Select a number that is not already in use on this host machine. |
| Alias | Specifies an alias for the port that is unique for this API Gateway. An alias must be between 1 and 255 characters in length and include one or more of the following: letters (a -z, A-Z), numbers (0-9), underscore (_), period (.), and hyphen (-). |
| Description | Provide a description of the port. |
| Bind address | Specifies the IP address to which to bind this port. Specify a bind address if your machine has multiple IP addresses and you want the port to use this specific address. If you do not specify a bind address, API Gateway picks one for you. |
| Backlog | Specifies the number of requests that can remain in the queue for an enabled port before API Gateway begins rejecting requests. The default is 200. The maximum value is 65535. |
| Keep alive timeout | Specifies when to close the connection if the server has not received a request from the client within this |

| Field | Description |
|-------------------------------|--|
| | timeout value (in milliseconds) or when to close the connection if the client has explicitly placed a close request with the server. |
| Threadpool | <p>Specifies whether to create a private thread pool for this port or use the common thread pool.</p> <ul style="list-style-type: none"> ■ Select Disable to use the common server thread pool for this port. ■ Select Enable to create a private thread pool for this port. <p>If Threadpool is enabled, provide the following information:</p> <ul style="list-style-type: none"> ■ Threadpool min - the minimum number of threads for this private threadpool. The default is 1. ■ Threadpool max - the maximum number of threads for this private thread pool. The default is 5. ■ Thread priority - the Java thread priority. The default is 5. ■ Current threads - the total number of private threadpool threads currently in use for the port. |
| Security configuration | |
| Client authentication | <p>Specifies the type of client authentication you want API Gateway to perform for requests that arrive on this HTTPS port.</p> <p>Select one of the following:</p> <ul style="list-style-type: none"> ■ Username/Password - API Gateway prompts the client for a user ID and password. ■ Digest - API Gateway uses password digest to authenticate all requests. If the client does not provide the authentication information, API Gateway returns an HTTP WWWAuthenticate header with digest scheme to the client requesting for authentication information. If the client provides the required authentication information, API Gateway verifies and validates the request. ■ Request Kerberos Ticket - API Gateway looks for a Kerberos ticket in the HTTP Authorization header using the Negotiate authentication scheme. If it does |

| Field | Description |
|-------|--|
| | <p>not find the ticket, API Gateway uses user name and password for basic authentication. If the client does not provide any authentication information, API Gateway returns an HTTP WWW-Authenticate header with negotiate scheme to the client requesting for authentication information. If the client provides the required authentication information, API Gateway verifies and validates the request.</p> <ul style="list-style-type: none"> ■ Require Kerberos Ticket - API Gateway looks for a Kerberos ticket in the HTTP Authorization header using the Negotiate authentication scheme. If it does not find the ticket, API Gateway fails the authentication. If the client does not provide any authentication information, API Gateway returns an HTTP WWW-Authenticate header with negotiate scheme to the client requesting for authentication information. If the client provides the required authentication information, API Gateway verifies and validates the request. |

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

- **JAAS context** - Specify the custom JAAS context used for Kerberos authentication.
- **Principal** - Specify the name of the principal to use for Kerberos authentication.
- **Principal password** - Specify the password for the principal that is used to authenticate the principal to the KDC.
- **Retype principal password** - Retype the principal password.
- **Service principal name format** - Specify the format in which you want to specify the principal name of the service that is registered with the principal database.
 - **host-based** - Represents the principal name using the service name and the host name, where host name is the host computer. This is the default.
 - **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"> ■ Service principal name - Specify the name of the principal used with the service that the Kerberos client wants to access. |

- Click **Save**.

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

Adding an HTTPS Port

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

To add an HTTPS port

- Select **<Username> > Administration** in the top right corner.
- Select **Security > Ports**.

The ports page lists all the ports configured with API Gateway, if any.

- Click **Add Ports**.
- Select the type of port as **HTTPS** and click **Add**.

Various properties that need to be configured are displayed.

- Provide the following information:

| Field | Description |
|-------------------------------------|--|
| HTTPS listener configuration | |
| Port | <p>Specify the number you want to use for the port.</p> <p>Select a number that is not already in use on this host machine.</p> |
| Alias | <p>Specifies an alias for the port that is unique for this API Gateway.</p> <p>An alias must be between 1 and 255 characters in length and include one or more of the following: letters (a -z, A-Z), numbers (0-9), underscore (_), period (.), and hyphen (-).</p> |

| Field | Description |
|-------------------------------|--|
| Description | Provide a description of the port. |
| Bind address | <p>Specifies the IP address to which to bind this port.</p> <p>Specify a bind address if your machine has multiple IP addresses and you want the port to use this specific address. If you do not specify a bind address, API Gateway picks one for you.</p> |
| Backlog | <p>Specifies the number of requests that can remain in the queue for an enabled port before API Gateway begins rejecting requests.</p> <p>The default is 200. The maximum value is 65535.</p> |
| Keep alive timeout | Specifies when to close the connection if the server has not received a request from the client within this timeout value (in milliseconds) or when to close the connection if the client has explicitly placed a close request with the server. |
| Threadpool | <p>Specifies whether to create a private thread pool for this port or use the common thread pool.</p> <ul style="list-style-type: none"> ■ Select Disable to use the common server thread pool for this port. ■ Select Enable to create a private thread pool for this port. <p>If Threadpool is enabled, provide the following information:</p> <ul style="list-style-type: none"> ■ Threadpool min - the minimum number of threads for this private threadpool. The default is 1. ■ Threadpool max - the maximum number of threads for this private thread pool. The default is 5. ■ Thread priority - the Java thread priority. The default is 5. ■ Current threads - the total number of private threadpool threads currently in use for the port. |
| Security configuration | |

| Field | Description |
|------------------------------|--|
| Use JSSE | <p>If this port should support TLS 1.1 or TLS 1.2, click Yes to create the port using the Java Secure Socket Extension (JSSE) socket factory. The default is Yes.</p> <p>If you set this value to No, the port supports only SSL 3.0 and TLS 1.0.</p> |
| Client authentication | <p>Specifies the type of client authentication you want API Gateway to perform for requests that arrive on this HTTPS port.</p> <p>Select one of the following:</p> <ul style="list-style-type: none"> ■ Username/Password - API Gateway prompts the client for a user ID and password. ■ Digest - API Gateway uses password digest to authenticate all requests. If the client does not provide the authentication information, API Gateway returns an HTTP WWWAuthenticate header with digest scheme to the client requesting for authentication information. If the client provides the required authentication information, API Gateway verifies and validates the request. ■ Request Client Certificate - API Gateway requests client certificates for all requests. If the client does not provide a certificate, the server prompts the client for a userid and password. The server checks whether the certificate exactly matches a client certificate on file and is signed by a trusted authority. If so, the client is logged in as the user to which the certificate is mapped in API Gateway. If not, the client request fails, unless central user management is configured. ■ Require Client Certificate - API Gateway requires client certificates for all requests. The server behaves as described for Request Client Certificates, except that the client must always provide a certificate. ■ Use Identity Provider - API Gateway uses an OpenID Provider to authenticate requests. API Gateway redirects all requests sent to this port to the OpenID Provider specified in Identity Provider. ■ Request Kerberos Ticket - API Gateway looks for a Kerberos ticket in the HTTPS Authorization header using the Negotiate authentication scheme. If it does not find the ticket, API Gateway uses user name |

| Field | Description |
|-------|---|
| | <p>and password for basic authentication. If the client does not provide any authentication information, API Gateway returns an HTTP WWW-Authenticate header with negotiate scheme to the client requesting for authentication information. If the client provides the required authentication information, API Gateway verifies and validates the request.</p> <ul style="list-style-type: none"> ■ Require Kerberos Ticket - API Gateway looks for a Kerberos ticket in the HTTPS Authorization header using the Negotiate authentication scheme. If it does not find the ticket, API Gateway fails the authentication. If the client does not provide any authentication information, API Gateway returns an HTTP WWW-Authenticate header with negotiate scheme to the client requesting for authentication information. If the client provides the required authentication information, API Gateway verifies and validates the request. <p>You have to enable Kerberos by providing the following Kerberos properties with details that will be used for handling service requests that come with a Kerberos ticket:</p> <ul style="list-style-type: none"> ■ JAAS context - Specify the custom JAAS context used for Kerberos authentication. ■ Principal - Specify the name of the principal to use for Kerberos authentication. ■ Principal password - Specify the password for the principal that is used to authenticate the principal to the KDC. ■ Retype principal password - Retype the principal password. ■ Service principal name format - Specify the format in which you want to specify the principal name of the service that is registered with the principal database. <ul style="list-style-type: none"> ■ host-based - Represents the principal name using the service name and the host name, where host name is the host computer. This is the default. ■ 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"> ■ Service principal name - Specify the name of the principal used with the service that the Kerberos client wants to access. |
| Listener specific credentials | |
| Keystore alias | <p>Specifies a user-specified, text identifier for an API Gateway keystore.</p> <p>The alias points to a repository of private keys and their associated certificates. Although each listener points to one keystore, there can be multiple keys and their certificates in the same keystore, and more than one listener can use the same keystore alias.</p> |
| Key alias (Signing) | <p>Specifies the alias for the private key, which must be stored in the keystore specified by the above keystore alias.</p> |
| Truststore alias | <p>Specifies the alias for the truststore.</p> <p>The truststore contains the trusted root certificate for the CA that signed API Gateway certificate associated with the key alias. The truststore also contains the list of CA certificates that API Gateway uses to validate the trust relationship.</p> |

6. Click **Save**.

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

Adding an API Gateway External Port

The API Gateway external and registration ports work as a pair. One port is not functional without the other.

To add an API Gateway external port

1. Select **<Username> > Administration** in the top right corner.
2. Select **Security > Ports**.

The ports page lists all the ports configured with API Gateway, if any.

3. Click **Add Ports**.
4. Select the type of port as **API Gateway external** and click **Add**.

Various properties that need to be configured are displayed.

5. Provide the following information:

| Field | Description |
|--|--|
| API Gateway external listener configuration | |
| Port | <p>Specifies the port number you want to use for the external port.</p> <p>Use a number that is not already in use. This is the port that clients will connect to through your outer firewall.</p> |
| Alias | <p>Specifies an alias for the port.</p> <p>An alias must be between 1 and 255 characters in length and include one or more of the following: letters (a - z, A-Z), numbers (0-9), underscore (_), period (.), and hyphen (-).</p> |
| Description | A description of the port. |
| Protocol | <p>Specifies the protocol to use for this port (HTTP or HTTPS).</p> <p>If you select HTTPS, additional security and credential boxes appear for which you have to provide the required values.</p> |
| Bind address | <p>Specifies the IP address to which to bind this port.</p> <p>Specify a bind address if your machine has multiple IP addresses and you want the port to use this specific address. If you do not specify a bind address, API Gateway picks one for you.</p> |
| Backlog | <p>Specifies the number of requests that can remain in the queue for an enabled port before API Gateway begins rejecting requests.</p> <p>The default is 200. The maximum value is 65535.</p> |
| Keep alive timeout | <p>Specifies when to close the connection if the server has not received a request from the client within this timeout value (in milliseconds) or when to close the connection if the client has explicitly placed a close request with the server.</p> |

| Field | Description |
|--|--|
| | The default value is 20000ms |
| Gateway registration listener configuration | |
| Port | <p>Specifies the number you want to use for the registration port.</p> <p>Use a number that is not already in use. It is best not to use a standard port such as 80 (the standard port for HTTP) or 443 (the standard port for HTTPS) because the external firewall will allow access to those ports from the outside world</p> |
| Alias | <p>Specifies an alias for the port.</p> <p>An alias must be between 1 and 255 characters in length and include one or more of the following: letters (a - z, A-Z), numbers (0-9), underscore (_), period (.), and hyphen (-).</p> |
| Description | A description of the port. |
| Protocol | <p>Specifies the protocol to use for this port (HTTP or HTTPS).</p> <p>If you select HTTPS, additional security and credential boxes appear for which you have to provide the required values.</p> |
| Bind address | <p>Specifies the IP address to which to bind this port.</p> <p>Specify a bind address if your machine has multiple IP addresses and you want the port to use this specific address. If you do not specify a bind address, API Gateway picks one for you.</p> |
| Client authentication | <p>For both the external port and registration, specify the type of client authentication required.</p> <p>Select one of the following:</p> <ul style="list-style-type: none"> ■ Username/Password - API Gateway will not request client certificates. <ul style="list-style-type: none"> ■ For external ports, the server looks for user and password information in the header of requests coming from an external client. |

| Field | Description |
|-------|--|
| | <ul style="list-style-type: none"> ■ For registration ports, the server looks for user and password information from the Internal Server. ■ Digest - For external ports, API Gateway uses password digest authentication. API Gateway looks for password digest information in the header of requests coming from an external client. ■ Request Client Certificate - API Gateway requests client certificates for all requests. <ul style="list-style-type: none"> ■ For external ports, the server requests client certificates for requests that come through this port. If the client does not present a certificate, the request proceeds using the user and password information contained in the request header. ■ For registration ports, the server requests a client certificate from the Internal Server. If the Internal Server does not present a certificate, the request proceeds using the user and password information ■ Require Client Certificate - API Gateway requires client certificates for all requests. <ul style="list-style-type: none"> ■ For external ports, the server requires client certificates for all requests that come through this port. If the client does not supply a certificate, the request fails. ■ For registration ports, API Gateway requires a client certificate from the Internal Server. If the Internal Server does not supply a client certificate, the request fails. In addition, if the certificate is not mapped to a user with Administrator privileges on API Gateway, the request fails. ■ Request Kerberos Ticket - For external ports, API Gateway requires client certificates for requests from external clients. If the external client does not supply a certificate, the request fails. ■ Require Kerberos Ticket - For external ports, API Gateway looks for a Kerberos ticket from external clients. If the external client does not present a ticket, the request proceeds using the user and password information contained in the request header. ■ Use JSSE - If this port should support TLS 1.1 or TLS 1.2, click Yes to create the port using the Java Secure |

| Field | Description |
|--------------------------------------|---|
| | <p>Socket Extension (JSSE) socket factory. The default is Yes.</p> <p>If you set this value to No, the port supports only SSL 3.0 and TLS 1.0.</p> |
| Listener specific credentials | |
| Keystore alias | <p>Specifies a user-specified, text identifier for an API Gateway keystore.</p> <p>The alias points to a repository of private keys and their associated certificates. Although each listener points to one keystore, there can be multiple keys and their certificates in the same keystore, and more than one listener can use the same keystore alias.</p> |
| Alias (Signing) | <p>Specifies the alias for the private key, which must be stored in the keystore specified by the above keystore alias.</p> |
| Truststore alias | <p>Specifies the alias for the truststore.</p> <p>The truststore contains the trusted root certificate for the CA that signed API Gateway certificate associated with the key alias. The truststore also contains the list of CA certificates that API Gateway uses to validate the trust relationship.</p> |

- Click **Save**.

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

Configuring the API Gateway Internal listener

The API Gateway Internal listener processes the requests received from the API Gateway External port and sends responses to API Gateway. This procedure describes how to connect the Internal listener to API Gateway.

To configure the API Gateway Internal listener

- Select **<Username> > Administration** in the top right corner.
- Select **Security > Ports**.

The ports page lists all the ports configured with API Gateway, if any.

- Click **Add Ports**.

4. Select the type of port as **API Gateway internal** and click **Add**.

Various properties that need to be configured are displayed.

5. Provide the following information:

| Field | Description |
|----------------------------------|--|
| Protocol | Specifies the protocol to use for this port (HTTP or HTTPS). If you select HTTPS, additional security and credential boxes appear for which you have to provide the required values. |
| Description | A description of the port. |
| Alias | Specifies an alias for the port. An alias must be between 1 and 255 characters in length and include one or more of the following: letters (a -z, A-Z), numbers (0-9), underscore (_), period (.), and hyphen (-). |
| Max connections | Specifies the number of connections maintained between API Gateway Internal port and API Gateway. The default is 5. |
| Threadpool | Specifies whether to create a private thread pool for this port or use the common thread pool. <ul style="list-style-type: none"> ■ Select Disable to use the common server thread pool for this port. ■ Select Enable to create a private thread pool for this port. If Threadpool is enabled, provide the following information: <ul style="list-style-type: none"> ■ Threadpool min - the minimum number of threads for this private threadpool. The default is 1. ■ Threadpool max - the maximum number of threads for this private thread pool. The default is 5. ■ Thread priority - the Java thread priority. The default is 5. |
| Enterprise gateway server | |
| Host | Specifies the host name or IP address of the machine on which Enterprise Gateway Server is running |

| Field | Description |
|---|--|
| Port | Specifies the port number of the registration port on Enterprise Gateway Server |
| Registration credentials (This appears only if you have selected HTTPS in the Protocol option. The registration credentials specified here must match the settings on the API Gateway registration port) | |
| User name | Specifies the name of the user on API Gateway that the Internal Server should connect as. |
| Password | Specifies the password of the user on API Gateway that the Internal Server should connect as. |
| Use JSSE | <p>If this port should support TLS 1.1 or TLS 1.2, click Yes to create the port using the Java Secure Socket Extension (JSSE) socket factory. The default is Yes.</p> <p>If you set this value to No, the port supports only SSL 3.0 and TLS 1.0.</p> |
| Keystore alias | Specifies the keystore alias created for the keystore containing the certificate that the Internal Server sends to API Gateway for client authentication. The Internal Server sends this certificate when it makes its initial registration connection to API Gateway. The Internal Server sends this certificate only if asked to by API Gateway. |
| Alias (Signing) | Specifies the key alias created for the key pair and associated certificate, in the previously specified keystore. |
| Truststore alias | Specifies the alias for the truststore file that contains the trusted root certificates associated with the CA signing authority. |
| External client security | |
| Client authentication | <p>Specifies the type of client authentication the Internal Server performs against external clients. External clients pass their authentication information to API Gateway, which in turn passes it to the Internal Server.</p> <p>Select one of the following:</p> |

| Field | Description |
|-------|--|
| | <ul style="list-style-type: none"> ■ Username/Password - API Gateway will not request client certificates. Instead it will look for user and password information in the request header. ■ Digest - The Internal Server will look for password digest information in the request header ■ Request Client Certificate - API Gateway requests client certificates for requests from external clients. If the external client does not present a certificate, the request proceeds using the user and password information contained in the request header. ■ Require Client Certificate - API Gateway requires client certificates for requests from external clients. If the external client does not supply a certificate, the request fails.. ■ Request Kerberos Ticket - API Gateway requires client certificates for requests from external clients. If the external client does not supply a certificate, the request fails ■ Require Kerberos Ticket - API Gateway looks for a Kerberos ticket from external clients. If the external client does not present a ticket, the request proceeds using the user and password information contained in the request header. |

6. Click **Save**.

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 for the APIs that it hosts, and so on.

You can configure the following destinations to which the event types and performance metrics data is published:

- API Gateway
- Database
- API-Portal

Configuring API Gateway as a Destination

You have to configure API Gateway as a destination so that the event types and performance metrics data can be published to API Gateway. By default, error events and performance data is published to API Gateway.

To configure API Gateway as a destination

1. Select **<Username> > Administration** in the top right corner.
2. Select **Destinations**.
3. Select **Gateway** to configure API Gateway as the destination.
4. Select the required Event types.
5. Select **Report performance data** to publish performance metrics data.
6. Provide a value for the publish interval.
7. Click **Save**.

The event types and performance metrics data are now published to API Gateway.

Configuring Database as a Destination

You have to configure the Database as a destination so that the event types and performance metrics data can be published to the configured database.

To configure a Database as a destination

1. Select **<Username> > Administration** in the top right corner.
2. Select **Destinations**.
3. Select **Database** to configure the database as the destination.
4. Select the required Event types.
5. Select **Report performance data** to publish performance metrics data.
6. Provide a value for the publish interval.
7. Click **Save**.

The event types and performance metrics data are now published to the database.

Configuring API-Portal

You have to configure API-Portal to establish a communication channel between API Gateway and API-Portal to exchange data.

1. Select **<Username> > Administration** in the top right corner.

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 for 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. |

Configuring API-Portal as a Destination

You have to configure API-Portal to communicate with API Gateway before you select API-Portal as a destination.

You have to configure API-Portal as a destination so that the event types and performance metrics data can be published to API-Portal.

1. Select **<Username> > Administration** in the top right corner.
2. Select **Destinations**.
3. Select **API Portal > Events** to configure API-Portal as the destination.
4. Select the required Event types.
5. Select **Report performance data** to publish performance metrics data.
6. Provide a value for the publish interval.
7. Click **Save**.

The event types and performance metrics data are now published to API-Portal.

Alias Configuration

While promoting API Gateway APIs and policies through various stages, the stage dependent environment settings like routing endpoints, routing rules, endpoint connection properties, outbound authentication tokens, and outbound HTTP headers often have to be adapted. An alias is an object that holds the above information and is stored in API Gateway. The alias name is referred in policy action's properties instead of providing a real value. The corresponding alias value is substituted in place of alias name during runtime. Thus the same alias can be referred to in multiple policies and the change in a particular alias would affect all the policy actions in which it is being referred. You can create three type of aliases - simple alias, secure alias, and endpoint alias.

Creating a Simple Alias

A simple alias stores a single value to be used for forming a routing endpoint.

To create a simple alias

1. Select **<Username> > Aliases** in the top right corner.
2. Select **Create alias**.
3. Select **Simple** as the alias type.
4. Click **Add**.

- Provide the following information:

| Field | Description |
|-------------|---|
| Name | Name of the alias. |
| Value | Specifies the value to be used for the alias. |
| Description | Description of the alias. |

- Click **Save**.

Creating a Secure Alias

Secure alias stores outbound authentication values such as user, password, domain, and OAuth2 token. The password is hashed and put into secure storage so that it is not visible in clear text.

To create a secure alias

- Select **<Username> > Aliases** in the top right corner.
- Select **Create alias**.
- Select **Secure** as the alias type.
- Click **Add**.
- Provide the following information:

| Field | Description |
|-----------------------|---|
| Name | Name of the alias. |
| Description | Description of the alias. |
| Authentication scheme | Authentication scheme applicable. Available values are Basic authentication and OAuth2. |
| Username | This is visible and required when the Authentication type selected is Basic authentication. User name used as the authentication credential. |

| Field | Description |
|--------------------|--|
| Password | This is visible and required when the Authentication type selected is Basic authentication. Password used as the authentication credential. |
| Domain | This is visible and required when the Authentication type selected is Basic authentication. Domain name. |
| Oauth token | This is visible and required when the Authentication type selected is Osuth2 Authentication. Specifies the Oauth2 token. |

- Click **Save**.

Creating an Endpoint Alias

An endpoint alias stores the endpoint value along with additional properties such as connection timeout, read timeout, and so on.

To create an endpoint alias

- Select **<Username> > Aliases** in the top right corner.
- Select **Create alias**.
- Select **Endpoint** as the alias type.
- Click **Add**.
- Provide the following information:

| Field | Description |
|-------------------------------|--|
| Name | Name of the alias. |
| Description | Description of the alias. |
| Optimization technique | This setting is applicable only for SOAP services. The available optimization techniques are: <ul style="list-style-type: none"> ■ None - This is the default value. ■ MTOM - Indicates that API Gateway expects to receive a request with a Message Transmission |

| Field | Description |
|---------------------------------|---|
| | <p>Optimization Mechanism (MTOM) attachment and forwards the attachment to the native service.</p> <ul style="list-style-type: none"> ■ SWA - Indicates that API Gateway expects to receive a SOAP with Attachment (SWA) request and forwards the attachment to the native service. |
| Endpoint URL | A default URL or components of the URL such as service name. |
| Connection timeout | <p>The time interval (in seconds) after which a connection attempt times out.</p> <p>The default value is 30 seconds.</p> |
| Read timeout | The time interval (in seconds) after which a socket read attempt times out. |
| Pass security header | Passes the security header. Selected by default. |
| Keystore alias | The keystore alias of the API Gateway instance. |
| Client certificate alias | The client's private key to be used for performing SSL client authentication. |

6. Click **Save**.

2 Managing Users

■ User Roles 38

User Roles

The primary user roles in API Gateway are API Gateway Administrator and API Provider.

Users are defined by the external LDAP user repository. Local users can be defined through the Integration Server Administration user interface. The users known to API Gateway are those that are known to the configured user repositories (Internal or LDAP). Local as well as external LDAP users and local as well as external LDAP groups can be assigned to the following groups:

- API-Gateway-Administrators
- API-Gateway-Providers

These groups are predefined and created during the installation of API Gateway. Users known to the configured user repositories can invoke APIs that are protected through user authentication policies. There are no separate user repositories for API Gateway Administrators and API Providers.

API Gateway users and groups are managed through Integration Server. For details on managing users and groups, see *webMethods Integration Server Administrator's Guide*.

The privileges based on the user role are listed in the table.

| Privileges | API-Gateway Administrator | API Provider |
|--|---------------------------|--------------|
| Manage API-Gateway configurations and integration configurations | Y | N |
| Manage APIs | Y | Y |
| Manage API Policies | Y | Y |
| Manage Threat Protection Policies | Y | N |
| Manage Applications | Y | Y |
| Manage Users | Y | N |
| Manage analytics and dashboard | Y | Y |

Authentication and Authorization

API Gateway supports the authentication against LDAP and local user repository. The UI login authentication and the user authentication for invoking an API Gateway service are performed through Integration Server authentication. The following authentication methods are supported:

- Basic authentication
- Kerberos based SSO
- HTTP header based SSO

When invoking a custom API exposed by API Gateway, user authentication is done through the inbound authentication runtime actions. The Authorization rules are applied as per the respective user roles and privileges of the user. You have to assign either the API-Gateway-Provider group or the API-Gateway-Administrator group to the Providers and Administrators of API Gateway respectively.

Note: You cannot delete predefined users, groups, and ACLs in Integration Server. These groups can be deleted in API Gateway. Deleting predefined users, groups, and ACLs may cause performance issues in API Gateway. On restarting API Gateway, the users, groups, and ACLs are repopulated.

3 Managing APIs

| | |
|---|----|
| ■ Overview | 42 |
| ■ Creating an API by Importing an API from a File | 42 |
| ■ Creating an API by Importing an API from a URL | 43 |
| ■ Creating an API from Scratch | 43 |
| ■ Viewing API List and API Details | 47 |
| ■ Publishing an API | 47 |
| ■ Modifying API Details | 48 |
| ■ Filtering APIs | 48 |
| ■ Example: Managing an API | 49 |

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 Provider or API Gateway Administrator role.

To create an API by importing an API from a file

1. Click **APIs** in the title navigation bar.
A list of all existing APIs is displayed.
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 RAML file, that is 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 **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 Provider or API Gateway Administrator role.

To create an API by importing an API from a URL

1. Click **APIs** in the title navigation bar.
A list of all existing APIs is displayed.
2. Click **Create API**.
3. Select **Importing API from URL**.
4. Type the URL from where the API is being 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 an API from Scratch

You must have the API Gateway Provider or API Gateway Administrator role.

You can create an API from scratch by providing the basic information, technical information, and defining the resources and methods as required.

To create an API from scratch

1. Click **APIs** in the title navigation bar.
A list of all existing APIs is displayed.

2. Click **Create API**.
3. Select **Create from scratch**.
4. Click **Create**.
5. Provide the following information in the Basic information section:

| Field | Description |
|--------------------|-----------------------------------|
| Name | Name of the API. |
| Version | Version of the API being created. |
| Description | Description of the API. |

6. 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.

7. Provide the following information in the Technical information section:
 - a. Type the Base URL in the **Base URL** field.
 - b. 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. |

| Field | Description |
|--------|--------------------------------|
| Values | Specifies the possible values. |

- c. 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 |
| Value | Specifies the value for the schema type selected. For Inline schema - Type the request and response values. For External schema - Click Browse to upload the request and response values. |

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

8. 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.

9. Provide the following information in the Resources and methods section:

- a. Click **Add Resources** and provide the following information.

| Field | Description |
|---------------|--|
| Resource name | Name of the resource. This is the display name of the resource and resource path is used for execution. |
| Resource path | Specifies the path of the resource. The resource path should start with / |

| Field | Description |
|--------------------------|--|
| Description | Description of the resource. |
| Supported methods | Specifies the supported methods. Available values - GET, POST, PUT, DELETE, PATCH |

- b. Click **Add**.

The resource gets 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 | Specifies the method parameters. Provide the following information: <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.

10. 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 is displayed. 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.

Publishing an API

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

Ensure the following before publishing an API:

- A destination is configured.
- The API is active.

Note: In API Gateway 9.12, while publishing an API to API-Portal, only APIs that have been imported as a file can be published to API-Portal.

To publish an API

1. Click **APIs** in the title navigation bar.
A list of all APIs is displayed.
 2. Click the **Publish** icon for the API that has to be published.
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.
- You can unpublish an API once it is published by clicking the **Unpublish** icon.

Modifying API Details

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 is displayed.
2. Click the required API.
The API details page is displayed.
3. Click **Edit**.
4. Modify the information as required.
5. Click **Save**.

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 is displayed.
2. In the Add Filter pane, select **REST** and/or **SOAP** 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.

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 412456. |
| <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 |
|--|-------------|---|
| /phones/orders | GET | Asks for a representation of all of the orders. |
| /phones/orders | POST | Attempts to create a new order, returning the location (in the Location HTTP Header) of the newly created resource. |
| /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. |

| Data Type | Description |
|-----------|--|
| | <p>The time input parameters allow hour and minute.</p> <p>This data type only accepts date values in the format <code>hh:mm:ss</code></p> |
| Date/Time | <p>Specifies a timestamp that represents a specific date and/or time.</p> <p>The date/time input parameters allow year, month, and day input as well as hour and minute. Hour and minute default to 0.</p> <p>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></p> |
| Integer | <p>Specifies an integer value for the data type.</p> <p>This is generally used as the default data type for integral values.</p> |
| Double | <p>Specifies the double data type value.</p> <p>This is a double-precision 64-bit IEEE 754 floating point and is generally used as the default data type for decimal values.</p> |
| Boolean | <p>Specifies a <code>true</code> or <code>false</code> value.</p> |

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. |

| Category | Description |
|----------|---------------|
| 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 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 |
| <code>/phones/orders/{order-id}</code> | GET | 200 (OK); 404 (Not Found) if Order Resource not found |
| <code>/phones/orders/{order-id}</code> | DELETE | 200 (OK); 404 (Not Found) if Order Resource not found |
| <code>/phones/orders/{order-id}/status</code> | GET | 200 (OK); 404 (Not Found) if Order Resource not found |
| <code>/phones/orders/{order-id}/paymentdetails</code> | GET | 200 (OK); 404 (Not Found) if Order Resource not found |
| <code>/phones/orders/{order-id}/paymentdetails</code> | 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>
```


4 Managing Policies

- Policies 62
- Managing Global Threat Protection Policies 62
- Managing API-level Policies 69

Policies

Policies on an API help API developers and administrators control the usage of the API, identify the consumers, control the acceptance of APIs, and manage their deployment into their environment.

A policy is a sequence of actions that is carried out by API Gateway when a consumer requests a particular API. An action is a single task that is included in a policy and is evaluated by API Gateway at run-time. The actions in a policy are used for preventing malicious attacks, authenticating consumers, validating digital signatures, and capturing performance measurements. Actions include one or more parameters which you configure when you insert the actions into a policy. For example, an action that identifies consumers specifies one or more identifiers to identify the consumers who are trying to access the services.

Policies are classified into the following types depending on how they are executed in API Gateway:

- **Global threat protection policies:** These 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.
- **API-level policies:** These policies apply to all APIs at the API-level within an instance of API Gateway. These policies provide run-time governance capabilities to an API.

Managing Global Threat Protection Policies

Global 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 was met.

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.

An API Gateway administrator can manage the following global threat protection policies:

- **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 having 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 the 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 **Global Policies > Global denial of service**.
3. Set the **Enable** button to the **On** position to enable the policy.
4. Type the maximum number of requests 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 that API Gateway can process concurrently from any single IP address, in the **Maximum Requests in progress** field.
7. Specify the time in minutes 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.

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 **Global Policies > 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 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 that API Gateway can process concurrently from any single IP address, in the **Maximum Requests in progress** field.
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 the inclusion of 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, enter 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 in 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 **Global Policies > Denied IPs**.
This displays a list of IPs that were denied access.
3. Click the delete icon 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.

To configure rules

1. Click **Policies** in the title navigation bar.
2. Select **Global Policies > Rules**.
This displays a list of rules that are already configured.
3. Click **Create rule**.
4. Type a name for the rule in the **Rule name** field.

Valid rule names:

- Must be unique.
- Must not be null.
- Must not contain spaces.
- Must not contain the special characters - ? ~ ` ! @ # \$ % ^ & * () - + = { } | [] \ \ : \ " ; ' < > , /

5. Type a description for the rule in the **Description** field.
6. Select an action to be followed when the policy is breached:
 - **Deny Request and Alert** to deny the access and send an alert when the policy is breached.
 - **Alert** to allow the request and send an alert when the policy is breached.
7. Type the alert message text, in the **Error message** field, to be displayed when the policy is breached.
8. Configure the required filters as follows:
 - **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.
 - **Mobile applications 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.
 - Type the mobile application version.

Note: 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 filter.
 - Set the **Enable** button to the **On** position to enable Database-specific SQL injection protection.

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 a database against which specific parameters needs to be checked.
- 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 (_).

Note: You can add multiple entries by clicking +.

- Set the **Enable** button to the **On** position to enable SQL injection protection.

■ 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.
- Type the host name or IP address of the machine on which the ICAP server is running.
- Type the port number on which the ICAP server is listening.
- Type the name of the service exposed by the ICAP server that you can use to scan your payload for viruses.

■ 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 you want API Gateway to use to run the service. The default value is Administrator.

9. 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 are ensuring that all users are using the latest versions of the applications and taking 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. Set the **Enable** button to the **On** position to enable the policy.
4. Provide the mobile application name and click **+**.

You can add more entries by clicking **+**. You can delete the added ones by clicking the delete icon.

5. Provide the mobile device type name and click **+**.

You can add more entries by clicking **+**. You can delete the added ones by clicking the delete icon.

6. 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 of the following alert types:
 - **None**: This does not send any alerts. This is the default setting.
 - Select the **On rule violation** condition to send an alert every time a request violates a rule, or the **Every** condition and specify the time interval (in minutes) to send to send alerts at specified intervals.

- **Email** :This sends email alerts.
 - Select the **On rule violation** condition to send an alert every time a request violates a rule, or the **Every** condition and specify the time interval (in minutes) to send to send alerts at specified intervals.
 - 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.
 - Select the **On rule violation** condition to send an alert every time a request violates a rule, or the **Every** condition and specify the time interval (in minutes) to send to send alerts at specified intervals.
 - Type `pub.apigateway.threatProtection:violationListener` that is used as the signature of the flow service.
 - Provide `Administrator` as the user type.
4. Click **Save**.

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. A policy is a sequence of actions that are carried out by API Gateway when a consumer requests a particular API through API Gateway. The actions in a policy are used for identifying and authenticating consumers, validating digital signatures and capturing performance measurements. An action is a single task that is included in a run-time policy and is evaluated by API Gateway at run time. Actions have one or more parameters, which you can configure in a policy when you apply it to an API. For example, an action that identifies consumers specifies one or more identifiers to identify the consumers who are trying to access the API.

The API level policies are broadly categorised as:

- Threat protection - These policies can viewed in 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
- Identity and access management
- Request payload processing
- Routing policies
- Logging, monitoring and traffic optimization
- Response processing

■ Error handling

Transport Policies

Require HTTP/HTTPS

Require HTTP/HTTPS specifies the protocol to be used 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 action 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 wish to expose the API to the consumers of API Gateway through HTTP, then you configure the incoming protocol as HTTP.

Properties:

| 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>Available values:</p> <ul style="list-style-type: none">■ HTTP: Default. API Gateway accepts requests that are sent using the HTTP protocol.■ 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>Values are:</p> <ul style="list-style-type: none">■ SOAP 1.1 - Default. API Gateway accepts requests that are in the SOAP 1.1 format.■ SOAP 1.1 - API Gateway accepts requests that are in the SOAP 1.2 format. |

Set Media Type

This policy action specifies the content type for a REST request or response. 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 service. Examples for content types: application/json, application/xml

Properties:

| Parameter | Description |
|------------------------------|--|
| Default content type | Specifies the default content type for REST request received from a client. 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. |
| Default accept header | Specifies the default accept header for REST request received from a client |

Identity and Access Management Policies

Identify & Authenticate Consumer

API Gateway provides Identify & Authenticate Consumer policy action that you can include to identify and validate clients, and then configure their parameters to suit your needs. You use these actions to identify clients who are trying to access the APIs, for example, through IP address or hostname, and validate the clients credentials.

Properties:

| Parameter | Description |
|----------------------------------|--|
| Condition | <p>Specifies the condition operator to be used for the authentication types selected.</p> <p>Available condition operators:</p> <ul style="list-style-type: none"> ■ AND - applies all the authentication actions selected. ■ OR - applies one of the authentication actions selected. |
| Allow anonymous | Specifies whether to allow all users to access the API without restriction. |
| Identification Type | |
| HTTP Basic Authentication | API Gateway tries to identify the client against either the Registered application list or the Global applications and tries to validate the client's |

| Parameter | Description |
|-------------------------|---|
| | <p>authentication credentials contained in the request's Authorization header against the specified list of API Gateway consumer applications.</p> <p>Specify 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 consumer applications who are registered as consumers for the specified API. ■ Global applications - tries to verify the client's credentials against a list of all global consumers available in API Gateway. ■ Do not identify - checks for the existence of the criteria but does not validate if the specified value is a valid consumer and forwards the request to the native API. For example, HTTP Basic Authentication is checked by the HTTP transport level property Authorization: Basic Base64encodesusernamepassword ■ Authenticate User - Selecting this specifies the users who can access the APIs. If you select the checkbox, API Gateway allows only the users specified in the Identify Consumer parameter to access the APIs. If you do not select the checkbox, API Gateway allows all users to access the API. |
| Hostname Address | <p>API Gateway tries to identify the client against either the Registered application list or the Global applications and tries to extract the client's hostname from the HTTP request header, and verify the client's identity in the specified list of API Gateway consumer applications.</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 the list of consumer applications who are registered as consumers for the specified API. ■ Global applications - tries to verify the client's hostname against a list of all global consumers available in API Gateway. |

| Parameter | Description |
|-------------------------|--|
| | <ul style="list-style-type: none"> ■ Do not identify - checks for the existence of the criteria but does not validate if the specified value is a valid consumer. |
| API Key | <p>API Gateway tries to identify the client against either the Registered application list or the Global applications and tries to validate the client's API key and verify the client's identity in the specified list of API Gateway consumer applications.</p> <p>Select one of the Application Lookup condition:</p> <ul style="list-style-type: none"> ■ Registered applications - tries to verify the client's API key against the list of consumer applications who are registered as consumers for the specified API. ■ Global applications - tries to verify the client's API key against a list of all global consumers available in API Gateway. ■ Do not identify - checks for the existence of the criteria but does not validate if the specified value is a valid consumer. |
| IP Address Range | <p>API Gateway tries to identify the client against either the Registered application list or the Global applications and tries to extract the client's IP address from the HTTP request header, and verify the client's identity against the specified list of API Gateway consumer applications.</p> <p>Select one of the Application Lookup condition:</p> <ul style="list-style-type: none"> ■ Registered applications - tries to verify the client's credentials against the list of consumer applications who are registered as consumers for the specified API. ■ Global applications - tries to verify the client's credentials against a list of all global consumers available in API Gateway. ■ Do not identify - checks for the existence of the criteria but does not validate if the specified value is a valid consumer. |
| SSL certificate | <p>API Gateway tries to identify the client against either the Registered application list or the Global applications and tries to extract the client's identity certificate, and verify the client's identity (certificate-</p> |

| Parameter | Description |
|-------------------------|---|
| | <p>based authentication) against the specified list of API Gateway consumer applications. 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 the list of consumer applications who are registered as consumers for the specified API. ■ Global applications - tries to verify the client certificate against a list of all global consumers available in API Gateway. ■ Do not identify - checks for the existence of the criteria but does not validate if the specified value is a valid consumer. |
| OAuth2 token | <p>API Gateway tries to identify the client against either the Registered application list or the Global applications and tries to extract the client's credentials from the HTTP request header, and verify the client's identity against the specified list of API Gateway consumer applications.</p> <p>Select one of the Application Lookup condition:</p> <ul style="list-style-type: none"> ■ Registered applications - tries to verify the client's OAuth access token against the list of consumer applications who are registered as consumers for the specified API. ■ Global applications - tries to verify the client identify credentials against a list of all global consumers available in API Gateway. ■ Do not identify - checks for the existence of the criteria but does not validate if the specified value is a valid consumer. |
| XPath expression | <p>API Gateway tries to identify the client against either the Registered application list or the Global applications and tries to extract the custom authentication credentials supplied in the request represented using an XPath expression to verify the consumer's identity using this information against the specified list of API Gateway consumer applications.</p> |

| Parameter | Description |
|------------------------------|--|
| | <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 the list of consumer applications who are registered as consumers for the specified API. ■ Global applications - tries to verify the client identify credentials against a list of all global consumers available in API Gateway. ■ Do not identify - checks for the existence of the criteria but does not validate if the specified value is a valid consumer. ■ 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. |
| WSS Username Token | <p>This is applicable only for SOAP APIs.</p> <p>API Gateway tries to identify the client against either the Registered application list or the Global applications and to 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 API Gateway consumer applications.</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 the list of consumer applications who are registered as consumers for the specified API. ■ Global applications - tries to verify the client's WSS username token against a list of all global consumers available in API Gateway. ■ Do not identify - checks for the existence of the criteria but does not validate if the specified value is a valid consumer. |
| WSS X.509 Certificate | <p>This is applicable only for SOAP APIs.</p> |

| Parameter | Description |
|-----------|---|
| | <p>API Gateway tries to identify the client against either the Registered application list or the Global applications and tries to to extract the client identity certificate from the WS-Security SOAP message header, and verify the client's identity against the specified list of API Gateway consumer applications.</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 the list of consumer applications who are registered as consumers for the specified API. ■ Global applications - tries to verify the client's X.509 certificate against a list of all global consumers available in API Gateway. ■ Do not identify - checks for the existence of the criteria but does not validate if the specified value is a valid consumer. |

Authorize User

The policy action authorizes consumers against a list of users and a list of groups registered in the Integration Server on which API Gateway is running. You use this action in conjunction with an authentication action (for example, Require HTTP Basic Authentication, Require WSS Username Token).

Properties:

| Parameter | Description |
|-----------------------|---|
| List of users | <p>Authorizes consumers against a list of users who are registered in the Integration Server on which API Gateway is running.</p> <p>Specify one or more users in the fields below this option.</p> |
| List of groups | <p>Authorizes consumers against a list of groups who are registered in the Integration Server on which API Gateway is running.</p> <p>Specify one or more groups in the fields below this option.</p> |

Request Payload Processing Policies

Invoke webMethods Integration Server

This policy action is required to pre-process the request messages and transform into the format required by the native API or perform 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.

Properties:

| Parameter | Description |
|------------------------------|--|
| webMethods IS service | Specifies the webMethods IS service that should be invoked to pre-process the request messages. Note: The webMethods IS service must be running on the same Integration Server as API Gateway. |

XSLT Transformation

This policy action 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.

Properties:

| Parameter | Description |
|----------------------|--|
| XSLT document | |
| XSLT file | Specifies the XSLT file that is to be used to transform the request messages as required. Click Browse to browse and select a file. |
| XSLT features | |
| Feature name | Specifies the name of the XSLT feature. |
| Feature value | Specifies the value of the XSLT feature. |

Routing Policies

Note: In cases where the internal server is protected by a firewall, the endpoint in the routing policy that is applied should be configured as `apigateway://<registrationPort-aliasname>/<relative path of the service>`. Here the registration port alias name is the alias name configured for the external registration port to communicate with the internal port. For details, see ["Adding an API Gateway External Port" on page 22](#).

Straight Through Routing

If your entry protocol is HTTP or HTTPS, you can select the Straight Through routing. When you select the Straight Through routing protocol, the API routes the requests directly to the native service endpoint you specify.

Properties:

| 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 | This is applicable to REST APIs. Specifies the available routing methods: GET, POST, PUT, DELETE, and CUSTOM (default). |
| HTTP connection timeout | Specifies the time interval (in seconds) after which a connection attempt will timeout. 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 will timeout. 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. |

| Parameter | Value |
|---------------------------------|--|
| | 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 | <p>This is applicable for SOAP-based APIs. 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 - Default. API Gateway does not use any optimization method to parse the SOAP requests to the API. |
| WS-Security header | This is applicable for SOAP-based APIs. Indicates whether API Gateway should pass the WS-Security headers of the incoming requests to the native API. |

Load Balancer Routing

If your entry protocol is HTTP or HTTPS, you can select the 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.

Properties:

| Parameter | Description |
|-----------------|---|
| Route to | Specifies the URLs of two or more native services in a pool to which the requests will be routed. |

| Parameter | Description |
|---|---|
| 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 will timeout. 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 will timeout. 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: <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. |

| Parameter | Description |
|---------------------------|--|
| | <ul style="list-style-type: none"> ■ None - Default. API Gateway does not use any optimization method to parse the SOAP requests to the API. |
| WS-Security Header | This is applicable for SOAP-based APIs. Indicates whether API Gateway should pass the WS-Security headers of the incoming requests to the native API. |
| Timeout (seconds) | Specifies the time interval (in seconds) after which the attempt will timeout. |

Dynamic Routing

This policy action enables API Gateway to support dynamic routing of virtual aliases based on policy configuration. The policies configured are enforced on the request sent to a service and these requests are forwarded to the dynamic endpoint based on specific criteria that you specify.

Properties:

| Parameter | Description |
|--|---|
| Route to | |
| 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 | <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 will timeout.</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 will timeout.</p> <p>If a value 0 is specified (or if the value is not specified), API Gateway uses the default value 30 seconds.</p> |

| Parameter | Description |
|---|--|
| 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 | <p>This is applicable for SOAP-based APIs. 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 - Default. API Gateway does not use any optimization method to parse the SOAP requests to the API. |
| WS-Security Header | This is applicable for SOAP-based APIs. Indicates whether API Gateway should pass the WS-Security headers of the incoming requests to the native API. |
| Timeout (seconds) | Specifies the time interval (in seconds) after which the attempt will timeout. |
| Rule -Defines the routing decisions based on one of the following routing options | |
| Header Name | <p>Defines the dynamic URL based on the HTTP header value sent by the client.</p> <p>This header name is configured by the API provider and is used to decide the routing decisions at the virtual service. The request message must be routed</p> |

| Parameter | Description |
|-----------------|---|
| | to the dynamic URL generated from the HTTP header value. |
| Context | <p>Defines the dynamic URL based on the context variable value.</p> <p>The API providers must provide IS service in the policy action, 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.</p> |
| Route to | <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 will timeout. ■ Read Timeout (seconds) - Specifies the time interval (in seconds) after which a socket read attempt will timeout. ■ SSL Configuration - Specifies values to enable SSL client authentication that API Gateway uses to authenticate incoming requests for the native API. Values required are Keystore alias, and key alias. ■ Timeout (seconds) - Specifies the time interval (in seconds) after which the attempt will timeout. |

Outbound Authentication

The Outbound authentication policy is used to set the client credentials to access the native API.

Properties:

| Parameter | Description |
|------------------------------|---------------------------------------|
| Authentication scheme | Specifies the mode of authentication. |

| Parameter | Description |
|-----------|---|
| | <p>Select one of the options:</p> <ul style="list-style-type: none"> ■ Basic Authentication - Used when native API enforces basic authentication. Based on the modes selected, API Gateway either uses configured basic authentication credentials to invoke a native service or it uses credentials from the authorization header of the incoming request to access the native API. Select the appropriate value - Custom Credentials or Pass Existing Credentials. ■ Alias - Used when native API enforces authentication based on the Alias configuration. API Gateway uses configured Alias name to invoke the native service. Provide the Alias name that is configured to use this mode of authentication. ■ OAuth2 Token - Used when native API enforces OAuth authorization. Based on the modes selected, API Gateway either uses configured OAuth token to invoke the native service or it uses the OAuth token of the incoming request to access native service. Select the appropriate value - Custom Credentials or Pass Existing Credentials. ■ None - API Gateway does not use any of the above authentication scheme to access the native API. |

Custom HTTP Header

This policy action routes based on the custom HTTP headers specified for the outgoing message to the native service.

Properties:

| Parameter | Description |
|------------------------|--|
| HTTP header key | Specifies the HTTP header key contained in the requests. |
| Header value | Specifies the Header value contained in the requests. |

Content-Based Routing

If your entry protocol is HTTP or HTTPS, you can select the Content-Based routing. If you have a native service 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. The requests are routed according to the content-based routing rules you create. You may specify how to authenticate requests.

Properties:

| Parameter | Description |
|---|---|
| Route to | |
| 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 will timeout. 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 will timeout. 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. |

| Parameter | Description |
|--|--|
| SOAP Optimization Method | <p>This is applicable for SOAP-based APIs. 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 - Default. API Gateway does not use any optimization method to parse the SOAP requests to the API. |
| WS-Security Header | <p>This is applicable for SOAP-based APIs. Indicates whether API Gateway should pass the WS-Security headers of the incoming requests to the native API.</p> |
| Timeout (seconds) | <p>Specifies the time interval (in seconds) after which the attempt will timeout.</p> |
| Rule -Defines the routing decisions based on one of the following routing options | |
| XPath Error Criteria | <p>Provide the following information:</p> <p>XPath Expression - An argument to evaluate the XPath expression contained in the request.</p> <p>Value - Specifies the XPath value.</p> |
| Namespace | <p>Provide the following information:</p> <p>Namespace Prefix - Specifies the namespace declaration indicated by the Prefix</p> <p>Namespace URI - namespace declaration indicated by URI</p> |
| Route to | <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 |

| Parameter | Description |
|-----------|--|
| | <ul style="list-style-type: none"> ■ 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 will timeout. ■ Read Timeout (seconds) - Specifies the time interval (in seconds) after which a socket read attempt will timeout. ■ SSL Configuration - Specifies values to enable SSL client authentication that API Gateway uses to authenticate incoming requests for the native API. Values required are Keystore alias, and key alias. ■ Soap Optimization Method - Specifies values to enable SSL authentication for SOAP APIs. MTOM, SwA or None. ■ Timeout (seconds) - Specifies the time interval (in seconds) after which the attempt will timeout. |

Context-Based Routing

If your entry protocol is HTTP or HTTPS, you can select the Context-Based routing. If you have a native service 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. 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.

Properties:

| Parameter | Description |
|-----------------------|---|
| Route to | |
| 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 | <p>This is applicable to REST APIs.</p> <p>Specifies the available routing methods: GET, POST, PUT, DELETE, and CUSTOM (default).</p> |

| Parameter | Description |
|---|--|
| HTTP Connection Timeout (seconds) | <p>Specifies the time interval (in seconds) after which a connection attempt will timeout.</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 will timeout.</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 above keystore alias.</p> |
| SOAP Optimization Method | <p>This is applicable for SOAP-based APIs. 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 - Default. API Gateway does not use any optimization method to parse the SOAP requests to the API. |
| WS-Security Header | <p>This is applicable for SOAP-based APIs. Indicates whether API Gateway should pass the WS-Security headers of the incoming requests to the native API.</p> |

| Parameter | Description |
|--|--|
| Timeout (seconds) | Specifies the time interval (in seconds) after which the attempt will timeout. |
| Rule -Defines the routing decisions based on one of the following routing options | |
| Name | Specifies the name of the rule. |
| Condition Operator | Specifies the condition operator to be used. Available operators - OR, AND |
| Condition | Specifies the context variables for processing client requests. |
| Variable | <p>The variable list displays the list of supported variables.</p> <p>Consumer - Specifies the name of the consumer application in the text box.</p> <p>Custom context variable - Select the following</p> <ul style="list-style-type: none"> ■ Select a data type - String or Integer. ■ Select an Operator - Greater Than, Less Than, Not Equal To, or Equal To ■ Type a custom variable name in the Custom Context text box. ■ Type a value in the Variable Value text box. <p>Date - Select an operator Before or After. Type a date value in the text box.</p> <p>Time - Select an operator Before or After. Type a time value in the text box.</p> <p>Plv6 - Type an IP address range in the From and To text boxes.</p> <p>Plv4 - Type an IP address range in the From and To text boxes.</p> |
| Route to | <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 |

| Parameter | Description |
|-----------|--|
| | <ul style="list-style-type: none"> ■ 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 will timeout. ■ Read Timeout (seconds) - Specifies the time interval (in seconds) after which a socket read attempt will timeout. ■ SSL Configuration - Specifies values to enable SSL client authentication that API Gateway uses to authenticate incoming requests for the native API. Values required are Keystore alias, and key alias. ■ Soap Optimization Method - Specifies values to enable SSL authentication for SOAP APIs. MTOM, SwA or None. ■ Timeout (seconds) - Specifies the time interval (in seconds) after which the attempt will timeout. |

Logging, Monitoring, and Traffic Optimization Policies

Log Invocation

This policy can be used when you want to log 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.

Properties:

| 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 | Specifies how frequently to log the payload. Select one of the following options: |

| Parameter | Description |
|--------------------|--|
| | <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 destination - JDBC, API Gateway, API-Portal.</p> |

Monitor Service Performance

This policy action is similar to the Monitor Service Level Agreement policy action and does monitor the same set of run-time performance conditions for an API, and then send alerts when the performance conditions are violated. However, this policy action monitors run-time performance for a specific client.

Properties:

| Parameter | Value |
|-----------------------------|--|
| Action Configuration | |
| 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. |

| Parameter | Value |
|------------------------|---|
| | <ul style="list-style-type: none"> ■ 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 has to be logged.</p> <p>Select the required options - JDBC, API Gateway, API-Portal</p> |
| 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>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 - Issues an alert only the first time one of the specified conditions is violated. ■ Every time - Issues 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 action 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 action enables you to monitor run-time performance for one or more specified clients. You can configure this action to define a Service Level Agreement (SLA), which is a set of conditions that defines the level of performance that a client should expect from an API. You can use this action to identify whether an API threshold rules are met or exceeded. For example, you might define an agreement with a particular client

that sends an alert to the client (consumer application) if responses are not sent within a certain maximum response time. You can configure SLAs for each API or consumer application combination.

Properties:

| Parameter | Value |
|-----------------------------|--|
| Action Configuration | |
| 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 | Specifies the destination where the alert has to be logged. |

| Parameter | Value |
|------------------------|---|
| | Select the required options - JDBC, API Gateway, API-Portal |
| 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>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: - Only once, Every time</p> <ul style="list-style-type: none"> ■ Only once - Issues an alert only the first time one of the specified conditions is violated. ■ Every time - Issues an alert every time one of the specified conditions is violated. |
| Alert Message | Specifies the text to be included in the alert. |
| Application Ids | <p>Specifies the application to which this Service Level Agreement applies.</p> <p>To specify multiple consumer applications, use + to add multiple rows.</p> |

Throttling Traffic Optimization

This policy action 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.

Properties:

| Parameter | Description |
|----------------------------|---|
| Limit Configuration | |
| Rule name | Specifies the name of throttling rule to be applied. |
| Operator | Specifies the operator that connects the rule to the value specified. |

| Parameter | Description |
|------------------------|--|
| | Select one of the operators - Greater than, Less than, or Equals to |
| Value | Specifies the value of the request count beyond which the policy is violated. |
| Destination | Specifies the destination to log the alerts. Available destinations are: JDBC, API Gateway, API-Portal |
| Alert Interval | Specifies the interval of time for the limit to be reached. |
| Unit | Specifies the unit for the time interval in minutes, hours, days, or weeks for the alert interval. |
| Alert Frequency | Specifies the frequency at which the alerts are issued. Specify one of the options: <ul style="list-style-type: none"> ■ Only Once - Issues an alert only the first time the specified condition is violated. ■ Every Time - Default. Issues an alert every time the specified condition is violated. - Only once, Every time |
| Alert Message | Specifies the text message to be included in the alert. |
| Application Ids | Specifies the consumer application that this action applies to. To specify multiple consumers, use + to add rows, or select any consumer to apply this action to any consumer application. |

Response Processing Policies

Invoke webMethods Integration Server

This policy action is required to pre-process the native API's response messages into the format required by the clients, before API Gateway returns the responses to the clients.

Properties:

| Parameter | Description |
|---|--|
| webMethods IS service | Specifies the webMethods IS service that should be invoked to pre-process the native API's response. |
| Note: The webMethods IS service must be running on the same Integration Server as API Gateway. | |

XSLT Transformation

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

Properties:

| Parameter | Description |
|----------------------|---|
| XSLT document | |
| XSLT file | Specifies the XSLT file that is to be used to transform the response messages as required. Click Browse to browse and select a file. |
| XSLT features | |
| Feature name | Specifies the name of the XSLT feature. |
| Feature value | Specifies the value of the XSLT feature. |

Error Handling

Conditional Error Handling

Error Handling is the process of passing an exception message which has been issued as a result of a run-time error to take any necessary actions. The conditional error handling policy action returns a custom error message (and the native provider's service fault content) to the client when the native provider returns a service fault. You can configure conditional error processing and use variables to create custom error messages.

Properties:

| Parameter | Description |
|---|--|
| Error conditions | - Specifies the error conditions and how each of these error conditions are to be processed |
| Status Code Error Criteria | Specifies the error status code. |
| Header Error Criteria | <p>Specifies the details of the custom HTTP header(s) included in the client requests.</p> <p>Provide the following information:</p> <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 | <p>Specifies the details of the XPath expression in the API request.</p> <p>Provide the following information:</p> <ul style="list-style-type: none"> ■ XPath Expression - Specifies the name of the HTTP header ■ Value - Specifies the value of the HTTP header. ■ Namespace Prefix - Specifies the prefix for the namespace. ■ Namespace URI - Specifies the namespace URI. |
| Pre-processing | - Specifies how the native service error response is to be processed before sending the same to API Gateway |
| Invoke webMethods Integration Server | Specifies the webMethods IS Service. |
| XSLT Transformation | <p>Specifies the XSLT file and feature you want to use to transform the service error response.</p> <p>Use the Browse icon 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. |

| Parameter | Description |
|---|--|
| Custom error variables | Specifies the error variables to be used in the custom error message. |
| Payload type | <p>Specifies the payload type.</p> <p>Available values are:</p> <ul style="list-style-type: none"> ■ Request - Selects the request payload type. ■ Response - Selects the response payload type. |
| Name | Specifies the name of the payload. |
| XPath expression | Specifies 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. |
| Failure message | Specifies the custom failure message that API Gateway should send to the client. |
| Send native provider fault message | <p>API Gateway sends the native SOAP or REST failure message to the client when this parameter is enabled.</p> <p>When you disable this parameter, the Failure Message is ignored when a fault is returned by the native API provider.</p> |
| Post-processing | Specifies how the error message sent by the native service is to be processed before sending the same to the client. |
| Invoke webMethods Integration Server | Specifies the webMethods IS Service. |
| XSLT Transformation | <p>Specifies 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> |

| Parameter | Description |
|-----------|---|
| | <ul style="list-style-type: none">■ Feature Name - Specifies the name of the XSLT feature.■ Feature Value - Specifies the value for the feature. |

5 Managing Applications

| | |
|---|-----|
| ■ Applications | 102 |
| ■ Creating an Application | 103 |
| ■ Viewing List of Applications and Application Details | 104 |
| ■ Modifying Application Details | 105 |
| ■ Registering an API with Consumer Applications from API Details Page | 105 |
| ■ Registering APIs with Consumer Applications from Application Details Page | 106 |

Applications

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, and so on, 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. This attribute is queried when the Identify Consumer action is configured to identify consumer applications by IP address. Example: 192.168.0.10
- 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 action 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, HTTP user name, and WSS user name.

As an API provider or an API Gateway Administrator 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 of administrator associates the application with the API for allowing an access to invoke. Keys are generated for applications and not for every API that the application consumes.

Note: 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 proper identifier (such as the access key or identifier) to access the API.

Creating an Application

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:** Provide the IP address range. You can add more range options by clicking **+** and adding the required information.
 - **Client Certificate:** Click **Browse** and select the client certificate to be uploaded. You can add multiple certificates by clicking **+**.
 - **Other identifiers:** Select one of the options Hostname, XPath or Username and provide the required value.
6. Click **Continue to APIs >**

Alternatively you can click **APIs** in the left navigation panel.

You can save the application by clicking **Save** at this stage and add the APIs at a later time.

7. Type a keyword to find the required API and click **+** to add the API.

Adding an API to the application enables the application to access the API. An API developer while invoking the API at runtime, has to provide the access token or identification token 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 **+**.

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, OAuth credentials, and APIs registered for that application.

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 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 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**.

6 Managing API Packages and Plans

| | |
|--|-----|
| ■ API Packages and Plans | 108 |
| ■ Creating a Package | 108 |
| ■ Creating a Plan | 109 |
| ■ Viewing List of Packages and Package Details | 111 |
| ■ Modifying a Package | 111 |
| ■ Deleting a Package | 112 |
| ■ Activating a Package | 113 |
| ■ Publishing a Package | 113 |
| ■ Viewing List of Plans and Plan Details | 114 |
| ■ Modifying a Plan | 114 |
| ■ Deleting a Plan | 115 |

API Packages and Plans

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 should have the API Gateway Administrator or API Gateway Provider privileges to manage API packages and plans.

An API Plan is the contract proposal presented to consumers who are about to subscribe APIs. Plans are offered as tiered offerings with varying availability guarantees, SLAs or cost structures associated to 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 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. Click Browse and select the required image to be displayed as the icon for the API package. The icon size should not be more than 100 KB. |

You can save the API package at this point and add the plans at a later time.

6. Click **Continue to add plans**.

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

7. Select the plans that are to be associated with the API package.

You can save the API package at this point and add APIs at a later time.

8. Click **Continue to add APIs**.

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

9. Type characters in the search box and click the search icon to search for the required APIs.

A list of APIs that contain the characters specified in the search box appears.

10. Select the required APIs to be associated with the 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.

11. Click **Save**.

Creating a Plan

You can create a Plan from the Manage packages and plans page.

To create a plan

1. Click **API Packages** in the title navigation bar.
2. Click **Create** in the Manage packages and plans section.
3. Select **Plan**.
4. Click **Create**.
5. Provide the following information in the Basic information section:

| Field | Description |
|-------------|---|
| Name | Name of the plan. |
| Version | Version assigned for the plan. |
| Description | A brief description for the plan. |
| Icon | An icon that is displayed for the plan. |

| Field | Description |
|-------|--|
| | 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 add policies**.

Alternatively click **Throttling traffic optimization** in the left navigation pane.

- Provide the following information in the Create Rule section:

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

| Field | Description |
|--------------------------|---|
| Violation message | Specifies the text that is displayed when the policy is violated. |

10. 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.

11. 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 Manage packages and plans page from where you can create, delete, and select a package to view its details.

To view the package list and package details

1. Click **API Packages** in the title navigation bar.

A list of all packages is displayed. You can perform various operations like activating a package, publishing or unpublishing a package, and deleting a package.

2. Select a package.

The package details page displays the basic information, and the associated plans and APIs for the selected package.

Modifying a Package

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 an inactive state.

To modify a package

1. Click **API Packages** in the title navigation bar.

A list of all packages is displayed.

2. Select a package.

The package details page displays the basic information, and the associated plans and APIs for the selected package.

3. Click **Edit**.

The package details are displayed.

Note: The **Edit** option is available only if the package is in an 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.
This displays a list of plans associated with the package and list of available plans.
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.
This displays a list of APIs associated with the package and a search box to search for APIs that need to be added to the package.
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 can delete a package from the Package list displayed in the Manage packages and plans page. You can not delete a package if it is in an 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 is displayed.
2. Click the **Delete** icon for the package that has to be deleted.
A confirmation dialog is displayed.
3. Click **Yes** to confirm deletion.

Activating a Package

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 for 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 in Access Token page. The consumer can test the APIs present in the package with this token in the API Try out page. The API Gateway Administrator and Provider can see a new application created in Application section of API Gateway.

To activate a package

1. Click **API Packages** in the title navigation bar.

A list of all packages is displayed 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** in the Packages details page to activate the package.

Publishing a Package

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 for the consumers to consume them. The package level token is applicable to all APIs associated with the package. The consumers do not have to request for 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 should have been already published to the destination.

To publish a package

1. Click **API Packages** in the title navigation bar.

A list of all packages is displayed.

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 for the consumers to consume it.

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 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 is displayed. You can deleting a plan by clicking the **Delete** icon for the respective plan.

3. Select a plan.

The plan details page displays the basic information, the pricing, and Quality of service associated with the selected plan.

Modifying a Plan

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 is displayed.

2. Click **Plans**.

A list of all plans is displayed.

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 are displayed with fields editable.

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.

This displays a list of rules associated with the plan.

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 can delete a plan from the Plans list displayed in the Plans section of 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 is displayed.
3. Click the **Delete** icon for the plan that has to be deleted.

A confirmation dialog is displayed.
4. Click **Yes** to confirm deletion.

7

API Gateway Analytics

| | |
|------------------------------------|-----|
| ■ Analytics Dashboards | 118 |
| ■ Purging API Analytics Data | 122 |

Analytics Dashboards

The analytics dashboards display a variety of charts to provide an overview of API Gateway performance and its API usage. 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, page views by consumers. 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 and so on. This can be accessed from the API details page.

The dashboard displays depend on the events and metrics generated in API Gateway, their types, and how customers can leverage them by rendering them as dashboards for their analytics and analyzing their business impacts. An event is a kind of notification or alert generated by API Gateway Metrics and Event Notification module. Various types of events are generated based on the behavior of the transaction that happen in the system. Events generated by API Gateway are real time events and they are persisted in the store and sent to configured destinations.

There are five types of events supported in API Gateway.

- **Transactional event**: Provides a summary of each runtime transaction occurred in the system. It is associated with the Log Invocation policy. For example, if an API has the policy attached to it, then for every invoke the system generates a transaction event.
- **Error event**: Provides the error details that occurred during an API invoke. Whenever there is an error in the system during a runtime service invocation this event is generated.
- **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, and so on. Monitoring can be done at the consumer application level too so that each consumer can be tracked individually.
- **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.
- **LifeCycle event**: Provides a summary about the life cycle of the API Gateway instance. Whenever the instance is started or stopped, a life cycle notification is generated.

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 over a particular color in the graphical representations.

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 category is depicted with different colors. |
| | Application Activity | Displays the consumer activity in API Gateway in the specified time. |
| | Runtime Events table | Displays the run time event details like 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. |
| | Payload Size | Displays the payload size of the request and responses during data transfer in the specified time. |

| Category | Metric | Description |
|-------------------|---------------------------------|--|
| | | This data is picked up from the transactional event that is triggered when a log invocation policy is applied to the API. |
| Trends | Events over Time | Displays the trending of events generated by the APIs across API Gateway over time. |
| | API trend by success | Displays the trending of APIs based on their success rate in the performance metrics. |
| | API trend by failure | Displays the trending of APIs based on their failure rate in the performance metrics. |
| | Overall error trend | Displays a graph depicting the performance of all the APIs in the system based on the error event generated. Each of these event category is depicted with different colors. |
| Application | 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 for consumed APIs | Displays the activity rate for the all the APIs that are consumed by the application in the specified time. |
| Threat protection | Threat protection filters | Displays the graphical representation of the events based on the filter violations in the specified time. |

| Category | Metric | Description |
|----------|--------------------------|---|
| | Threat protection rules | Displays the graphical representation of the events based on the rule violations in the specified time. |
| | Threat protection events | Displays the threat protection event details like 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 APIs page and by 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 in which you want to view the API-specific 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 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 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. |
| Native Service Performance | Displays information on how fast the native service responds to the request received in the specified time based on the data in the transactional event. |

| Metric | Description |
|---------------------|---|
| Response code trend | Displays the trend based on the response codes received from various events for the API in 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. |

Purging API Analytics Data

The process of systematically deleting unwanted data from the database is called purging. You can purge logs by setting the required date or duration per tenant in the API Gateway. You must have the API Gateway administrator privileges to purge the API analytics data.

To purge API analytics data

1. Select **<Username> > Analytics** in the top right corner.
2. Click **Purge**.
3. Select one of the following options in the Purge data section:
 - Select **Until** and select a date until 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.
4. Click **Purge**. A confirmation message is displayed.
5. Click **Yes** in the confirmation dialog.