API Gateway

Version 7.6.2

14 July 2020

Administrator Guide

This documentation describes the following Axway software:

Axway API Gateway 7.6.2

Contents

Preface

This guide explains how to configure and manage the components in an API Gateway domain.

Who should read this guide

The intended audience for this guide is API Gateway administrators. For details on installing API Gateway, see the *API Gateway Installation Guide*.

For details on API Gateway concepts and features, see the *API Gateway Concepts Guide*.

How to use this guide

This guide should be used with the other guides in the API Gateway documentation set. Before you begin, review this guide thoroughly. The following is a brief description of the contents:

- *API Gateway administration* on page 17: Gives an overview of the main issues involved in API Gateway administration, how to plan an API Gateway system, and how it interacts with existing infrastructure.
- *Manage an API Gateway domain* on page 40: Explains how to configure an API Gateway domain, and Admin Node Manager high availability and security.
- *Manage API Gateway operation* on page 77: Describes how to start and stop the API Gateway, Admin Node Manager, and API Gateway tools, and API Gateway high availability and disaster recovery.
- *Manage API Gateway security* on page 99: Explains how to configure API Gateway passphrases, certificates, API firewalling, and redaction of sensitive data. This part also explains how to run API Gateway in Federal Information Processing Standards (FIPS) mode.
- *Deploy API Gateway configuration* on page 139: Describes how to deploy and manage updates to API Gateway configuration.
- *Monitoring and metrics* on page 147: Explains how to perform root cause analysis using real-time monitoring of API services and message traffic, and how to configure storage of metrics for historic traffic in a relational database.
- *Troubleshoot your API Gateway installation* on page 162: Describes how to configure an API Gateway logging, events, and trace, and how to fine-tune performance.
- *Manage user access* on page 196: Explains how to manage API Gateway and Admin users, and how to configure Role-Based Access Control (RBAC) with examples from Active Directory and OpenLDAP.

- *Manage network-level settings* on page 233: Describes how to configure a DNS service with wildcards for virtual hosting.
- *Manage ActiveMQ messaging* on page 239: Explains how to manage the Apache ActiveMQ messaging broker embedded inAPI Gateway.
- *API Gateway settings reference* on page 245: Provides reference information on API Gateway configuration settings.

Related documentation

The AMPLIFY API Management solution enables you to create, publish, promote, and manage Application Programming Interfaces (APIs) in a secure and scalable environment. For more information, see the *AMPLIFY API Management Getting Started Guide*.

The following reference documents are also available on the Axway Documentation portal at https://docs.axway.com:

- *Supported Platforms*

  Lists the different operating systems, databases, browsers, and thick client platforms supported by each Axway product.

- *Interoperability Matrix*

  Provides product version and interoperability information for Axway products.

Support services

The Axway Global Support team provides worldwide 24 x 7 support for customers with active support agreements.

Email support@axway.com or visit Axway Support at https://support.axway.com.

See "Get help with API Gateway" in the *API Gateway Administrator Guide* for the information that you should be prepared to provide when you contact Axway Support.

Training services

Axway offers training across the globe, including on-site instructor-led classes and self-paced online learning. For details, go to: http://www.axway.com/support-services/training

Accessibility

Axway strives to create accessible products and documentation for users.

This documentation provides the following accessibility features:

Screen reader support

- Alternative text is provided for images whenever necessary.
- The PDF documents are tagged to provide a logical reading order.

Support for high contrast and accessible use of colors

- The documentation can be used in high-contrast mode.
- There is sufficient contrast between the text and the background color.
- The graphics have the right level of contrast and take into account the way color-blind people perceive colors.

Updates and revisions

This guide includes the following documentation changes.

Changes in version 7.6.2

- Renamed the topic on running API Gateway as non-root to "Run API Gateway on privileged ports", and updated the topic to describe alternative options for adding API Gateway library paths to the system path. For details, see *Run API Gateway on privileged ports* on page 99.
- Updated the topic on Embedded ActiveMQ settings to describe new fields that enable you to specify the maximum memory and disk usage for ActiveMQ messages and to enable reporting of memory and disk usage. For more information, see *Embedded ActiveMQ settings* on page 276.
- Updated the topic on traffic monitoring settings to describe new and updated fields that enable you to configure transaction file management. For more details, see *Traffic monitoring settings* on page 279.
- Added an appendix describing the open logging schema. For more information, see *Open logging schema* on page 286.
- Restructured the API Gateway Analytics information and added links to the new *API Gateway Analytics User Guide*.

Changes in version 7.6.1

No changes.

Changes in version 7.6.0

- Added information on storing the ModSecurity threat report in a message attribute. For details, see *Manage API firewalling* on page 132.
- Removed references to the following:
    - API Gateway server support for Windows
    - API Gateway Appliance
    - API Gateway Analytics (replaced with Embedded Analytics and API Manager metrics where appropriate)
- Renamed the existing section on "Monitoring and reporting" to "Monitoring and metrics".

- Updated the existing topic on configuring API Gateway Analytics with details on configuring API Gateway for a metrics database.

API Gateway administration 1

This part contains the following:

Introduction to API Gateway administration

Axway API Gateway is a comprehensive platform for managing, delivering, and securing APIs. This document explains how to administer and manage the API Gateway platform. This topic introduces the main issues involved in API Gateway administration.

Tip For an introduction to API Gateway features, tools, and architecture, see the *API Gateway Concepts Guide*.

API Gateway form factors

The API Gateway is available in software and hardware form factors. The available platforms are as follows:

- Software installation
- Container deployment in Docker containers

For more details, see the *API Gateway Installation Guide*.

Who owns the API Gateway platform and how is it administered?

The API Gateway platform is administered by the following groups:

- Operations—Runtime management of message traffic, logs and alerts, and high availability is performed by Operations staff.
- Architecture—Design-time policy definition, which determines the behavior of the API Gateway platform, is performed by Security Architects and Systems Architects.

Operations team

Operations staff are responsible for making sure that the API Gateway platform is running correctly. They are concerned with the following problems:

- System status and health
- Network connectivity
- Security alerts
- System security
- Backups and recovery
- Maintenance of logs

The API Gateway platform provides a web-based console named API Gateway Manager that is dedicated to the Operations team. The API Gateway Manager includes the following features:

- Dashboard for monitoring overall system health and network topology.
- Real-time monitoring and metrics for all messages processed by the API Gateway.
- Traffic monitoring to quickly isolate failed or blocked message transactions and provide detailed information about each transaction, payload, and so on.
- API Gateway logs, trace, events, and alerts.
- Messaging based on Java Message System (JMS). This includes managing queues, topics, subscribers, consumers, and messages in Apache ActiveMQ.
- Dynamic system settings, user roles, and credentials.

As well as providing operational management functionality of its own, the API Gateway also interoperates with third-party network operations tools such as HP OpenView, BMC Control, and CA UniCenter. Finally, all functionality available in the API Gateway Manager is also available as a REST API.

Architecture team

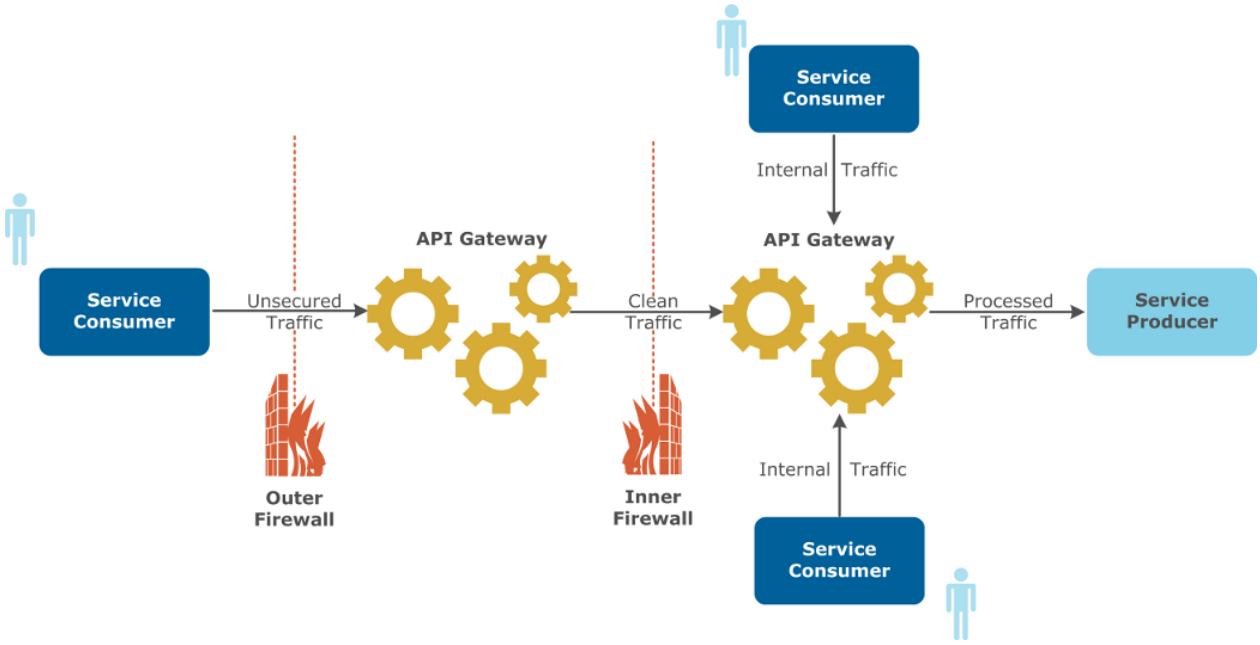
System architects and security architects have an overarching view of enterprise IT infrastructures, and so are more concerned with using the API Gateway to help integrate and secure existing enterprise systems. Architects wish to create API policies and integrate with third-party systems.

Policy Studio is a rich API Gateway policy development tool that enables architects and policy developers to create and control API Gateway policies. You can use Policy Studio to visually define policy workflows in a drag-n-drop environment. This means that configuration is performed at the systems architecture level, without needing to write code.

For more details on using Policy Studio to create API Gateway policies, see the *API Gateway Policy Developer Guide*.

Where do you deploy an API Gateway?

API Gateways can be deployed in the Demilitarized Zone (DMZ) or in the Local Area Network (LAN) depending on policies or requirements, as shown in the following diagram:



The following guidelines help you to decide where to deploy the API Gateway:

- If you are processing only traffic from external sources, consider locating theAPI Gateway in the DMZ. If the API Gateway is also processing internal traffic, consider locating it in the LAN.
- If you are processing traffic internally and externally, a combination of API Gateways in the DMZ and internally on the LAN is considered best practice. The reason for this is that different policies should be applied to traffic depending on its origin.
- Both internal and external traffic should be checked for threats and to make sure that they contain the correct parameters for REST API requests, or correspond to Web service definitions.
- External traffic carries a greater potential risk and should be scanned by the API Gateway located in the DMZ to make sure that it does not in any way affect the performance of internal applications.
- Internal traffic and pre-scanned external traffic should then be processed by the API Gateway located in the LAN. This type of checking includes:
    - Checking API service level agreements and enforcing throttle threshold levels
    - Integration with a wide range of third-party systems
    - Web service standards support

Where do you deploy API Gateway Analytics?

Although you can select the API Gateway Analytics component in the API Gateway installer, it is good practice to install API Gateway Analytics on a separate host from API Gateway installations. You should ensure that API Gateway Analytics runs on a dedicated host, or on a host that is not a running an API Gateway instance or Node Manager.

You can deploy API Gateway Analytics on any supported host platform depending on its availability in your architecture. For more details on supported platform versions, see the *API Gateway Installation Guide*.

Note API Gateway Analytics supports a range of databases for storing historic reports and metrics (for example, Oracle, DB2, MySQL, and Microsoft SQL Server). You should not install the database used for API Gateway Analytics in the DMZ. You should install this database in the LAN on a separate host from your API Gateway installations.

You can secure the connection to the API Gateway Analytics database by dedicating it to one IP address. For more details on configuring the API Gateway Analytics database, see the *API Gateway Analytics User Guide*.

Secure the last mile

Securing the last mile refers to preventing internal users from directly accessing services without going through the API Gateway. This can be achieved in multiple ways. You should carefully choose which option is best for your use case, taking into account the security level you want to achieve, and the impact on performance the solution will have. You should choose from the following approaches:

- **Controlling traffic at the network level**: Services can only be accessed if the traffic is coming from pre-approved IP addresses. This is the simplest solution to put in place, is very secure, and has no impact on performance or existing applications.
- **Establishing a mutual SSL connection between API Gateways and services**: This solution is the easiest to put in place and has little to no impact on existing applications. However, it does have a non-negligible impact on latency.
- **Passing authentication tokens from API Gateways to back-end services**: This involves passing authentication tokens for WS-Security, Security Assertion Markup Language (SAML), and so on. This solution has a low impact on latency but requires some development because the target service container must validate the presence and the contents of the token.

For more details on configuring mutual SSL, and configuring WS-Security and SAML authentication tokens, see the *API Gateway Policy Developer Guide*.

API Gateway administration lifecycle

The main stages in the overall API Gateway administration lifecycle are as follows:

1. Planning an API Gateway system. This is described in *Plan an API Gateway system* on page 21.
2. Installing API Gateway components. See the *API Gateway Installation Guide*.
3. Configuring a domain. This is described in *Configure an API Gateway domain* on page 40.
4. Operating and managing the API Gateway. This is described in the rest of this guide.
5. Upgrading an API Gateway. See the *API Gateway Upgrade Guide*.

Plan an API Gateway system

One of the most important tasks when deploying an API Gateway system is confirming that the system is fit for purpose. Enterprise software systems are hugely valuable to the overall success of the business operation. For any organization, there are many implications of system downtime with important consequences to contend with. This topic describes the most important factors to look at when architecting an API Gateway deployment.

Policy development

The functional characteristics of any given policy run by an API Gateway can have a huge effect on the overall system throughput and latency times. Depending on the purpose of a particular policy, the demand on valuable processing power will vary. The following guidelines apply in terms of processing power:

- Threat analysis and transport-based authentication tasks are relatively undemanding.
- XML processing such as XML Schema and WS-Security user name/password authentication are slightly more intensive.
- Calling out to third-party systems is expensive due to network latency.
- Cryptographic operations like encryption and signing are processor intensive.

The key point is that API Gateway policy performance depends on the underlying requirements, and customers should test their policies before deploying them into a production environment.

Policy development guidelines

Architects and policy developers should adhere to the following guidelines when developing API Gateway policies:

- Decide what type of policy you need to process your message traffic. Think in terms of functional requirements instead of technologies. Axway can help you to map the technologies to the requirements. Example functional requirements include the following:
    - *Only trusted clients should be allowed send messages into the network*
    - *An evidential audit trail should be kept*
- Think about what you already have in your architecture that could help to achieve these aims. Examples include LDAP directories, databases that already have replication strategies in place, and network monitoring tools.

- Create a policy to match these requirements and test its performance. Axway provides an integrated performance testing tool (API Tester) to help you with this process.
- Use the API Gateway Manager, API Gateway Analytics, Embedded Analytics, or third-party monitoring consoles to help identify what the bottlenecks are in your system. If part of the solution is slowing the overall system, try to find alternatives to meet your requirements.
- Test the performance capability of the back-end services.

Example policy requirements

Supplier A is creating a service that will accept Purchase Order (PO) documents from customers. The PO documents are formatted using XML. The functional requirements are:

- The service should not accept anything that will damage the PO system.
- Incoming messages must to be authenticated against a customer database to make sure they come from a valid customer account.
- The supplier already has an LDAP directory and would like to use it to store the customer accounts.
- The supplier must be able prove that the message came from the customer.

These requirements can be achieved using a policy that includes processing for Threatening Content and checking the XML Signature, which verifies the certificate against the LDAP directory. For details on how to develop API Gateway policies using Policy Studio, see the *API Gateway Policy Developer Guide*.

Traffic analysis

In the real world, messages do not arrive in a continuous stream with a fixed size like a lab-based performance test. Message traffic distribution has a major impact on system performance. Some of the questions that need to be answered are as follows:

- Is the traffic smooth or does it arrive in bursts?
- Are the messages all of the same size? If not, what is the size distribution?
- Is the traffic spread out over 24 hours or only during the work day?

Traffic analysis guidelines

You should adhere to the following guidelines when analyzing message traffic:

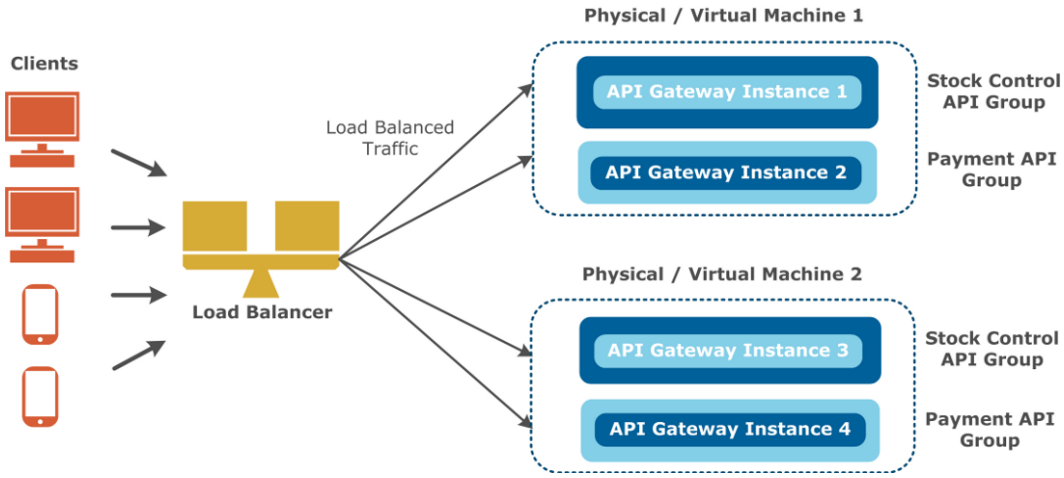
- Use the **Traffic** tab in the API Gateway Manager web console to analyze message traffic. For more details, see *Monitor services in API Gateway Manager* on page 147.
- Use the API Gateway Analytics web console to analyze historical message traffic. For more details, see the *API Gateway Analytics User Guide*.

- Use the Embedded Analytics web dashboards to analyze API, infrastructure, and client application health and API usage. For more details, see the *Embedded Analytics for AMPLIFY API Management documentation*.
- Take traffic distribution into account when calculating performance requirements.
- Take message size distribution into account when running performance tests.
- If traffic bursts cause problems for service producers, consider using the API Gateway to smooth the traffic (for example, using the Throttling filter). For more details, see the *API Gateway Policy Developer Guide*.

Load balancing and scalability

The API Gateway scales well both horizontally and vertically. Customers can scale horizontally by adding more API Gateways to a cluster and load balancing across it using a standard load balancer. API Gateways being load balanced run the same configuration to virtualize the same APIs and execute the same policies. If multiple API Gateway groups are deployed, load balancing should be across groups also.

For example, the following diagram shows load balancing across two groups of API Gateways deployed on two hosts:



The API Gateway imposes no special requirements on load balancers. Loads are balanced on a number of characteristics including the response time or system load. The execution of API Gateway policies is stateless, and the route through which a message takes on a particular system has no bearing on its processing. Some items such as caches and counters are held on a distributed cache, which is updated on a per message basis. As a result, API Gateways can operate successfully in both sticky and non-sticky modes. For more details on caching, see the *API Gateway Policy Developer Guide*.

The distributed state poses a number of questions in terms of active/active and active/passive clustering. For example, if the counter and cache state is important, you must design your overall system so that at least one API Gateway is active at all times. This means that for a resilient HA system, a minimum of at least two active API Gateways at any one time, with a third and fourth in passive mode is recommended.

The API Gateway ensures zero downtime by implementing configuration deployment in a rolling fashion. For example, while each API Gateway instance in the cluster or group takes a few seconds to update its configuration, it stops serving new requests, but all existing in-flight requests are honored. Meanwhile, the rest of the cluster or group can still receive new requests. The load balancer ensures that requests are pushed to the nodes that are still receiving requests. For more details on deploying API Gateway configuration, see the *API Gateway DevOps Deployment Guide*.

Load balancing guidelines

Axway recommends the following guidelines for load balancing:

- Use the Admin Node Manager and API Gateway groups to maintain the same policies on load-balanced API Gateways. For more details, see the *API Gateway Concepts Guide*.
- Configure alerts to identify when API Gateways and back-end services are approaching maximum capacity and need to be scaled.
- Use the API Gateway Manager console to see which parts of the system are processing the most traffic.

SSL termination

Secure Socket Layer (SSL) connections can be terminated at the load balancer or API Gateway level. These options are described as follows:

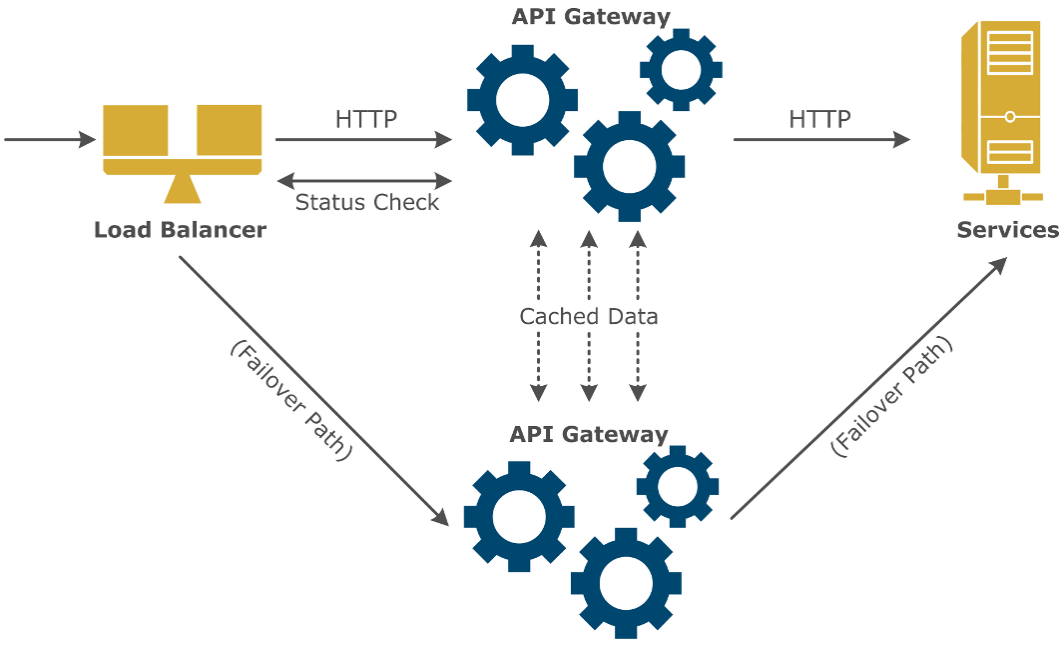
- **SSL connection terminated at load balancer**:
    - The SSL certificate and associated private key are deployed on the load balancer, and not on the API Gateway. The subject name in the SSL certificate is the fully qualified domain name (FQDN) of the server (for example, `axway.com`).
    - The traffic between the load balancer can be in the clear or over a new SSL connection. The disadvantage of a new SSL connection is that it puts additional processing load on the load balancer (SSL termination and SSL establishment).
    - If mutual (two-way) SSL is used, the load balancer can insert the client certificate into the HTTP header. For example, the F5 load balancer can insert the entire client certificate in `.pem` format as a multi-line HTTP header named `XClient-Cert` into the incoming HTTP request. It sends this header to theAPI Gateway, which uses it for validation and authentication.
- **Load balancer configured for SSL pass-through, all traffic passed to API Gateway**: With SSL pass-through, the traffic is encrypted so the load balancer cannot make any layer seven decisions (for example, if HTTP 500 is returned by API Gateway, route to HA API Gateway). To avoid this problem, you can configure the API Gateway so that it closes external ports on defined error conditions. In this way, the load balancer is alerted to switch to the HA API Gateway.

Note The API Gateway can also optionally use a cryptographic accelerator for SSL termination.For more details, see the *API Gateway Policy Developer Guide*.

High Availability and failover

API Gateways are used in high value systems, and customers typically deploy them in High Availability (HA) mode to protect their investments. The API Gateway architecture enables this process as follows:

- The Admin Node Manager is the central administration server responsible for performing all management operations across an API Gateway domain. It provides policy synchronization by ensuring that all API Gateways in an HA cluster have the same policy versions and configuration. For details on Admin Node Manager HA and API Gateway group-based architecture, see the *API Gateway Concepts Guide*.
- API Gateway instances are stateless by nature. No session data is created, and therefore there is no need to replicate session state across API Gateways. However, API Gateways can maintain cached data, which can be replicated using a peer-to-peer relationship across a cluster of API Gateways. For more details, see the *API Gateway Policy Developer Guide*.
- API Gateway instances are usually deployed behind standard load balancers which periodically query the state of the API Gateway. If a problem occurs, the load balancer redirects traffic to the hot stand-by machine.
- If an event or alert is triggered, the issue can be identified using API Gateway Manager, API Gateway Analytics,Embedded Analytics, or third-party monitoring consoles, and the active API Gateway can then be repaired.



HA stand-by systems

High Availability can be maintained using hot, cold, or warm stand-by systems. These are described as follows:

- **Cold stand-by**: System is turned off.
- **Warm (passive) stand-by**: System is operational but not containing state.
- **Hot (active) stand-by**: System is fully operational and with current system state.

HA and failover guidelines

Axway recommends the following guidelines for HA stand-by systems:

- For maximum availability, use an API Gateway in hot stand-by for each production API Gateway.
- Use API Gateways to protect against malicious attacks that undermine availability.
- Limit traffic to back-end services to protect against message flooding. This is particularly important with legacy systems that have been recently service-enabled. Legacy systems may not have been designed for the traffic patterns to which they are now subjected.
- Monitor the network infrastructure carefully to identify issues early. You can do this using API Gateway Manager, API Gateway Analytics, Embedded Analytics, or third-party monitoring consoles. Interfaces are also provided to standard monitoring tools such as syslog and Simple Network Management Protocol (SNMP).

For more details, see *Configure API Gateway high availability* on page 83.

Backup and recovery

Most customers have a requirement to keep a mirrored backup and disaster recovery site with full capacity to be able to recover from any major incidents. These systems are typically kept in a separate physical location on cold stand-by until the need arises for them to be brought into action.

Disaster recovery guidelines

The following applies for backup and recovery:

- The backup and disaster site must be a full replica of the production site (for example, with the same number of API Gateway instances).
- The Admin Node Manager helps get backup and recovery sites up and running fast by synchronizing the API Gateway policies in the backup solution.
- Remember to also include any third-party systems in backup and recovery solutions.

For more details, see *API Gateway backup and disaster recovery* on page 90.

Development staging and testing

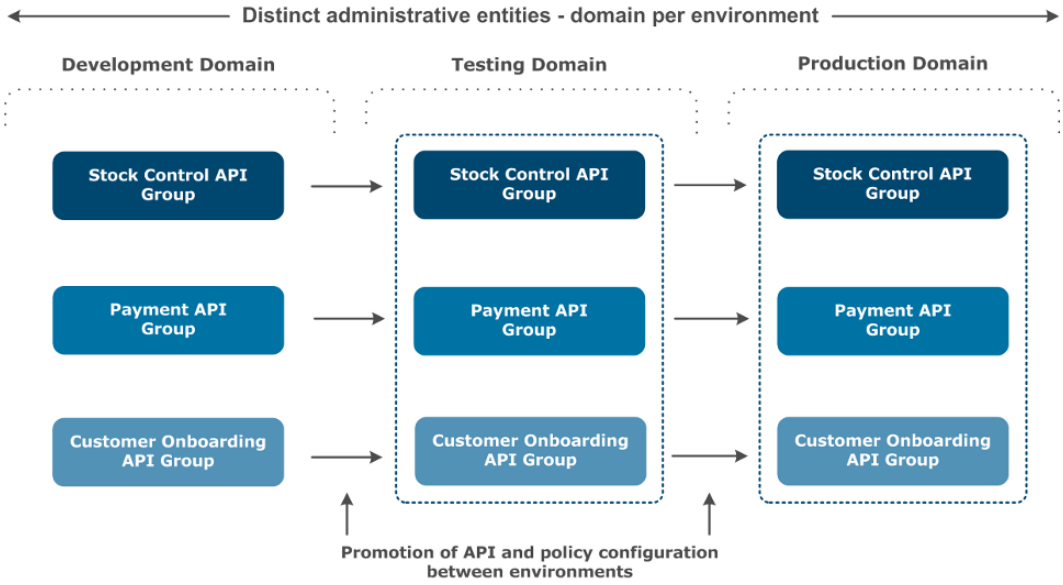
The most common reason for system downtime is change. Customers successfully alleviate this problem through effective change management as part of a mature software development lifecycle. A software development lifecycle controls change by gradually pushing it through a series of stages

until it reaches production.

Each customer will have their own approach to staging depending on the value of the service and the importance of the data. Staging can be broken into a number of different milestones. Each milestone is intended to isolate a specific type of issue that could lead to system downtime. For example:

- The development stage is where the policy and service are created.
- Functional testing makes sure the system works as intended.
- Performance testing makes sure the system meets performance requirements.
- System testing makes sure the changes to the system do not adversely affect other parts.

In some cases, each stage is managed by a different group. The number of API Gateways depends on the number of stages and requirements of each of these stages. The following diagram shows a typical environment topology that includes separate API Gateway domains for each environment:



Staging and testing guidelines

The following guidelines apply to development staging and testing: For more details, see the *API Gateway DevOps Deployment Guide*.

- Use API Gateway configuration packages (`.fed`, `.pol`, and `.env`) to control the migration of policies from development through to production.
- Keep an audit trail of all system changes.
- Have a plan in place to roll back quickly in the event of a problem occurring.
- Test all systems and policy updates before promoting them to production.
- Test High Availability and resiliency before going into production.

Hardening–secure the API Gateway

The API Gateway platform is SSL-enabled by default, so you do not need to SSL-enable each API Gateway component. However, in a production environment, you must take additional precautions to ensure that the API Gateway environment is secure from external and internal threats.

Hardening guidelines

The following guidelines apply to securing the API Gateway:

- You must change all default passwords (for example, change the password for Policy Studio and API Gateway Manager). For more details, see *Manage admin users* on page 198.
- The default X.509 certificates used to secure API Gateway components are self-signed (for example, the certificate used by the API Gateway Manager on port `8090`). You can replace these self-signed certificates with certificates issued by a Certificate Authority (CA). For more details, see *Configure Admin Node Manager high availability* on page 50.
- By default, API Gateway management ports bind on all HTTP interfaces (IP address set to `*`). You can change these ports or restrict them to bind on specific IP addresses instead. For more details, see the topic on "Configuring HTTP Services" in the *API Gateway Policy Developer Guide*.
- By default, API Gateway configuration is unencrypted. You can specify a passphrase to encrypt API Gateway instance configuration. For more details, see *Configure an API Gateway encryption passphrase* on page 102.
- You can configure user access at the following levels:
    - Policy Studio, API Gateway Manager, and Configuration Studio users—see *Manage API Gateway users* on page 196
    - API Gateway users—see *Manage API Gateway users* on page 196
    - Role-based access—see *Configure Role-Based Access Control (RBAC)* on page 202.

Capacity planning example

The following is an example formula for deciding how big your API Gateway deployment will need to be. This example is purely intended for illustration purposes only:

```
numberOfGateways = (ceiling (requiredThroughput x factorOfSafety /
testedPerformance) x HA) x (1 + numDR) + staging + eng))
```

This formula is described as follows:

- `factorOfSafety` has a value of 2 for normal.
- Typically High Availability (`HA`) has a value of 2

- *ceiling* implies that the number is rounded up. It is very important that this is not under provisioned by accident.
- Typically customers have a single backup and disaster recovery site (*numDR*=1).
- There is a large variation in performance between API Gateways depending on the policy deployed so it important to performance test policies prior to final capacity planning.
- *staging* is the amount of stage testing licenses. Customers with very demanding applications should have a mirror of their production system. This is the only way to be certain.

Example required throughput

For example, a software system is being designed that will connect an automobile manufacturer's purchasing system with a supplier. The system is designed to meet the most stringent load and availability requirements. An API Gateway system is being used to verify the integrity of the purchase orders.

The system as a whole is required to process 10,000,000 transactions per day. Each of these transactions will result in 8 individual request/responses. 90% of the load will happen between 9 am in the morning and 5 pm in the evening. The customer wishes to have a factor of safety of 2 for load spikes.

```
Overall throughput = (8 x 10,000,000 x 0.9)/8/60/60= 2,500 transactions per second
(tps)
```

Example development process

The development process for the example system should be as follows:

1. A Signature Verification policy is developed by one of the system engineers using a software license on his PC. When he is satisfied with the policy, he pushes it forward to the test environment.
2. Test engineers deploy the policy to the interoperability testing environment. They make sure that all of the systems work correctly together, and test the system as a whole. The API Gateway must show it can process the signature and integrate with the service back-end and PKI systems. When the system as a whole is working correctly, the testing moves to system testing.
3. System testing will prove that the system as a whole is reliable and has the capacity to meet the performance requirements. It will also make sure that the system interoperates with monitoring software and behaves properly during failover and recovery. During system test the API Gateway is found to be capable of processing 1,000 messages per second for this policy.
4. The customer will have the following infrastructure:
    - 1 backup and disaster site which is a full replica of the production site
    - A full HA system on hot standby
    - 2 API Gateways for testing on staging
    - A factor of safety on the load of 2

5. The total number of API Gateways across all environments (development, staging, production) should be as follows:

```
((ceiling(2500/1000)*2)*2)*(1+1)+2+1 = 27
```

6. The resulting architecture provisioned should be 27 API Gateways made up of the following:
    - 12 production licenses (6 live, 6 standby)
    - 12 backup and disaster recovery licenses
    - 2 staging licenses
    - 1 engineering license

How API Gateway interacts with existing infrastructure

As part of API Gateway policy execution, the API Gateway needs to interact with various components of the existing infrastructure. For example, these include external connections to systems such as databases, LDAP servers, third-party identity management products, and so on. This topic describes how the API Gateway interacts with these components in your existing network infrastructure.

Databases

API Gateway interoperates with a range of commonly used databases for a wide variety of purposes. For example, this includes managing access control credentials, authorization attributes, identity and role information, monitoring metrics, and reporting. API Gateway uses Java Database Connectivity (JDBC) to manage database connections. The following databases are supported:

- MySQL
- Oracle
- DB2
- Microsoft SQL Server
- MariaDB

API Gateway also supports Apache Cassandra for internal data storage.

For details on supported database versions, see the *API Gateway Installation Guide*.

Anti-virus

API Gateway supports Anti-Virus (AV) scanning using virus scanning engines from third-party vendors such as McAfee, Sophos, and ClamAV. Depending on the engine, this scanning can happen in-process or remotely using a client SDK or Internet Content Adaptation Protocol (ICAP). The API

Gateway takes message attachments, passes them to the AV scanner, and then acts on the decision. The following conditions apply:

- AV signature distribution and updates are performed using mechanisms of the anti-virus vendor.
- Licensing for the AV engine is performed through the AV vendor distribution channels.

For more details, see the *API Gateway Policy Developer Guide*.

Operations and management

API Gateway has a number of different options for operations and management:

- Detailed transaction logs can be sent to syslog (UDP), relational databases, or flat files. These contain detailed records of processed messages, their contents, how long processing took, and decisions taken during message processing. This type of logging can also include information alerts about policy execution failures and breached Service Level Agreements (SLAs), and information about critical events such as connection or disk failures. Logging levels can be controlled for an API Gateway or policy as a whole or on a filter-by-filter basis.
- Auditing information can be viewed in real time in the API Gateway Manager console, and can be pushed to your metrics database (for example, for monitoring in API Gateway Analytics, API Manager, or a third-party tool), or pushed to Embedded Analytics dashboards. For more details on logging, see *Troubleshoot your API Gateway installation* on page 162.
- The API Gateway Manager console is used to provide a current snapshot of how the API Gateway is behaving. For example, it displays how many messages are being processed, and what services are under the most load. API Gateway Manager displays what is happening now on API Gateway instances, and can be viewed by pointing a browser at an Admin Node Manager. For more details, see *Monitor services in API Gateway Manager* on page 147.
- Flexible alerts can be sent out to email, SNMP, OPSEC, syslog, Twitter, and Windows Event Log based on a condition being met in a policy. An example might be to email a service owner for every 1000 failures or to generate an alert if a service is processing more than 10000 messages per second. They can also be used to generate alerts on client usage. For more details, see the *API Gateway Policy Developer Guide*.
- Service Level Agreement filters can be used to perform a statistical measure of the services quality of service. They are used to make sure that the amount of network connection errors, response times and server errors are below a certain threshold. For more details, see the *API Gateway Policy Developer Guide*.

Network firewalls

When deployed in a Demilitarized Zone (DMZ) or perimeter network, the API Gateway sits behind network firewalls. Network Address Translation (NAT) firewall functionality is used on the network firewall to provide the API Gateway with a publicly routable address in the DMZ. This allows the API Gateway to route traffic internally to a local IP address range. API Gateways may be dual-homed to pass messages between the DMZ and the internal trusted network.

The API Gateway can then perform security processing on the incoming message traffic. For example, this includes the following:

- Looking for known attack patterns.
- Checking the validity and structure of the message for anomalies.
- Looking for traffic patterns that suggest a Denial-of-Service (DoS) attack.

Advantages over traditional application firewalls

API Gateways have a number of advantages over traditional application firewalls for message processing: If an attack or unusual traffic pattern is detected, the API Gateway can notify the firewall to block the traffic at source using OPSEC or some other notification mechanism.

- Can understand the structure of the traffic, and can detect subtle attack mechanisms such as entity expansion attacks and external reference attacks.
- Can consume information in the messages such as security and platform-specific tokens.
- Can use well understood standards such as JSON or XML Schema, JSON Path or XPath, WSDL, and OAuth to properly content filter the traffic.

Firewall modes

You can configure the API Gateway to act as a network endpoint or a network proxy, or both in tandem.

- Block unidentified traffic. This is the default setting. If there is no policy configured for this traffic, block it and, depending on configuration, raise an alert. In this way, rogue message traffic is detected and blocked.
- Pass unidentified traffic.

Application servers

Application servers are the infrastructure alongside which API Gateways are most commonly deployed. For example, API Gateways are often deployed in production systems with IBM WebSphere, Oracle WebLogic Server, Microsoft Biztalk Server, Fiorano SOA Platform, TIBCO ActiveMatrix BusinessWorks, and many others.

API Gateways interact with application servers in a number of modes, for example: For increased integration with such application server systems, the API Gateway supports numerous transport protocols including HTTP, HTTPS, JMS, email (SMTP/POP), FTP, and so on.

- Intercepting application server traffic by acting as a proxy.
- Offloading processing handed over by the application server (for example, to offload message transformation, encryption, or signature validation).

Enterprise Service Buses

API Gateways and Enterprise Service Buses (ESBs) have similar functionality and complement each other very well. Example ESB systems include Oracle Enterprise Service Bus, IBM WebSphere ESB, Progress Sonic ESB, Software AG WebMethods, and JBoss ESB. API Gateways are primarily used to perform the following tasks:

- Protect ESBs and downstream systems from traffic surge, potential DoS attacks, and threats.
- Offload expensive operations such as message validation and cryptography operations from ESBs.

Similarities between API Gateways and ESBs

API Gateways and ESBs typically both perform the following tasks:

- Protocol mediation
- Message routing and transformation
- Service composition
- Message processing

Differences between API Gateways and ESBs

The main differences between API Gateways and ESBs are as follows:

- API Gateways can be used for simple composite services (chained), but do not support Business Process Execution Language (BPEL), and are not suitable for long duration composite services.
- ESBs are usually delivered with backend adapters for systems such as CICS, IMS, Siebel, or SAP.
- API Gateways are stateless and cannot maintain transaction state.
- API Gateways are targeted at performance and application acceleration.
- API Gateways have been designed to provide superior security capabilities, without impacting on performance.

Directories and user stores

API Gateway supports a wide variety of user stores including LDAP, Active Directory, and access control products such as CA SiteMinder and Oracle Access Manager. User stores contain some of the most valuable information in an organization. For example, this includes private identity information such as phone numbers, addresses, email addresses, medical plan IDs, user names and passwords, certificates, organization structures, and so on. API Gateways must be able to interact with user stores without compromising them.

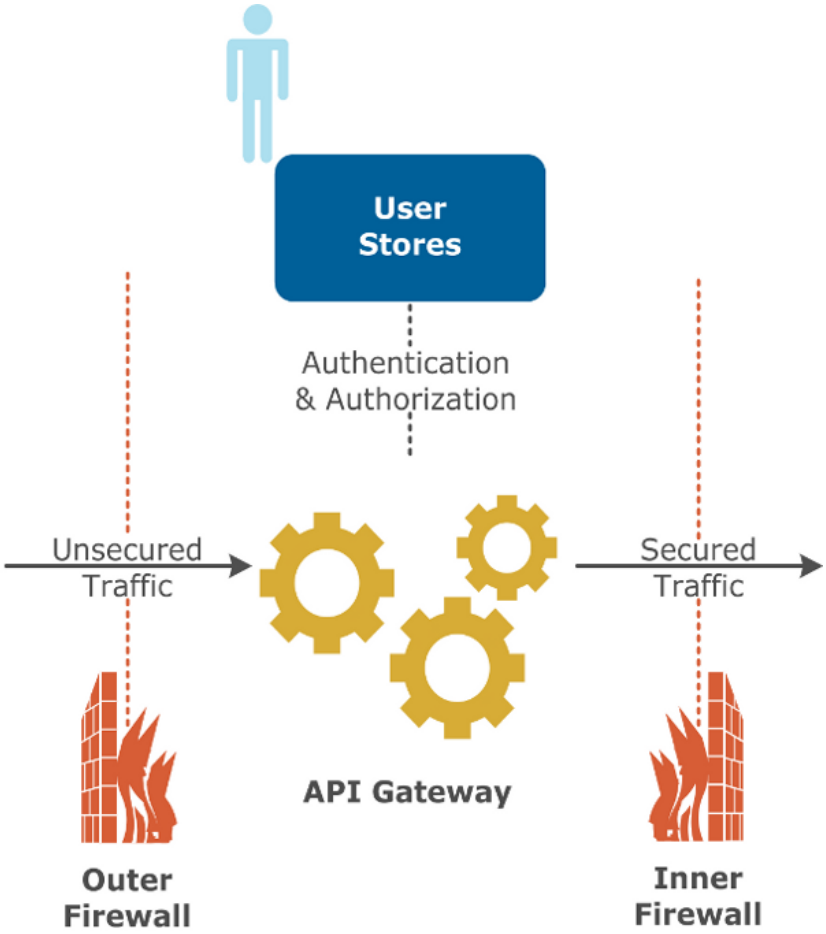
The API Gateway can use LDAP directories to retrieve user information such as the following:

- Authentication using username/password.
- Retrieve certificates and checking signatures.
- Authorization of clients based on attribute values.
- Retrieval of attributes for placing into SAML assertions.
- Checking certificate validity using Certificate Revocation Lists (CRL) retrieved from user stores.

For more details on integration with LDAP servers, see the *API Gateway Policy Developer Guide*.

Simple inline user store deployment

The following diagram shows a simple inline user store deployment:



Advantages

This architecture has the following advantages:

- This is simple and easy to set up.
- There is a single entry point through the DMZ for all message traffic.
- This is the least expensive option.

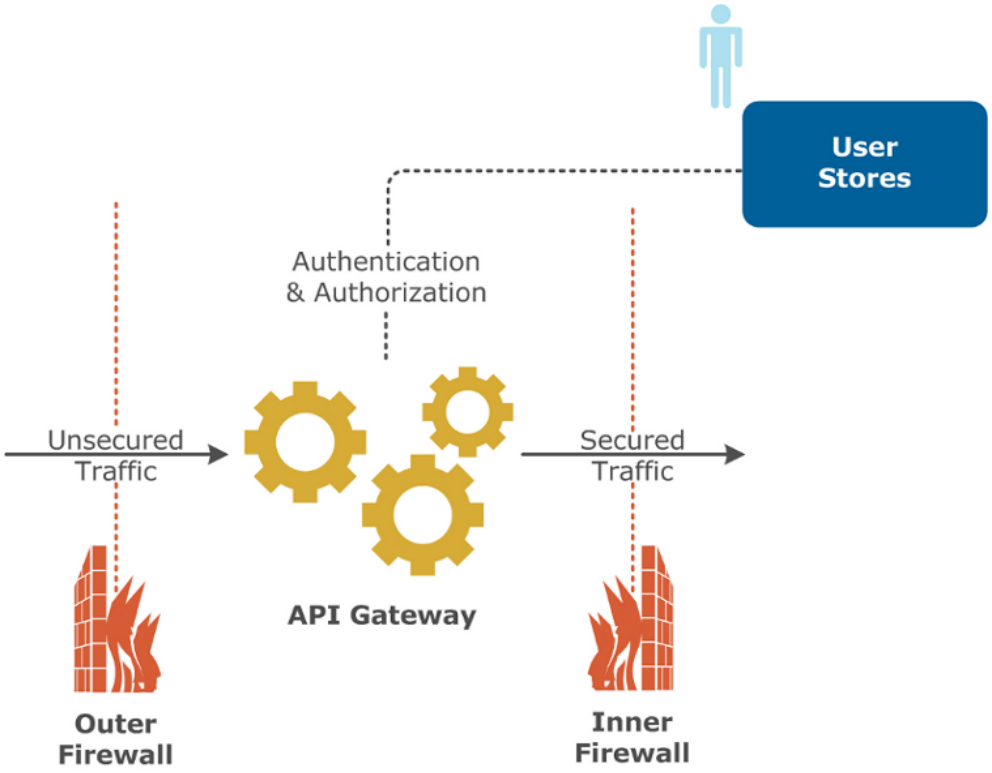
Disadvantages

This architecture has the following disadvantages:

- Exposing important user information in the DMZ is a potential security risk.
- The LDAP server is only being used for external traffic.

API Gateway in DMZ–LDAP in LAN

This is a very common setup where the API Gateway is located in the DMZ, and where it communicates with a user store located in the LAN:



Advantages

This architecture has the following advantages:

- The user store is protected from external access.
- This option is only moderately expensive.

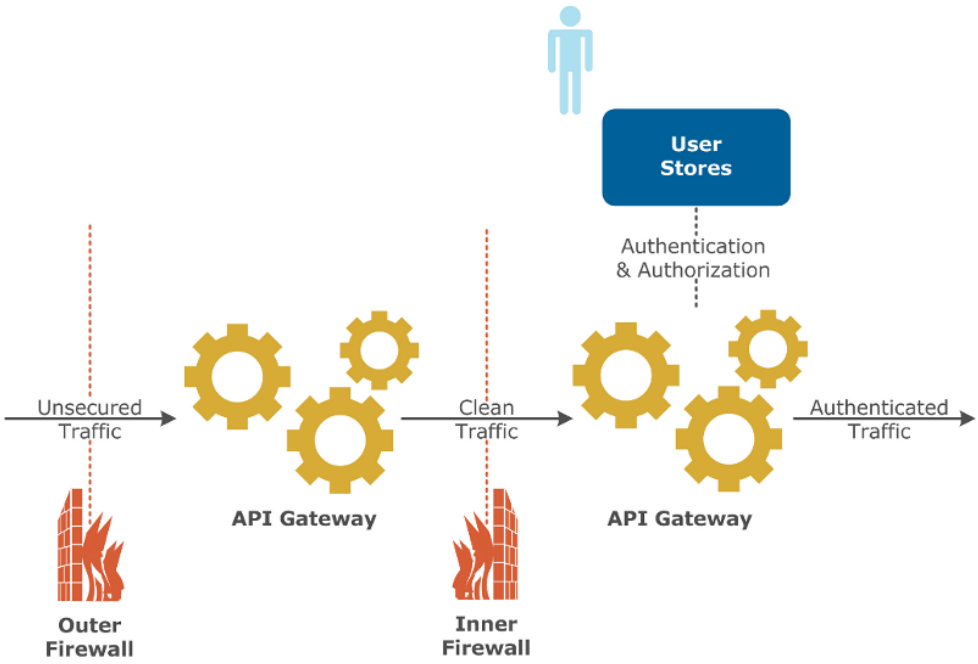
Disadvantages

This architecture has the following disadvantages:

- Administrators must maintain two entry points into the LAN from the DMZ for a single application.
- The user store is addressable from the DMZ, which is contrary to the security policies enforced by many organizations.

Split deployment between DMZ and LAN

In this scenario, two API Gateways are used to split the security checks across the DMZ and the LAN. For example, threats and JSON and XML schema validity are performed in the DMZ, while authentication and/or authorization is performed in the LAN:



Advantages

This architecture has the following advantage:

- This is the most secure deployment available.
- By separating threats from access control, it is easy to run separate security policies for internal and external traffic.

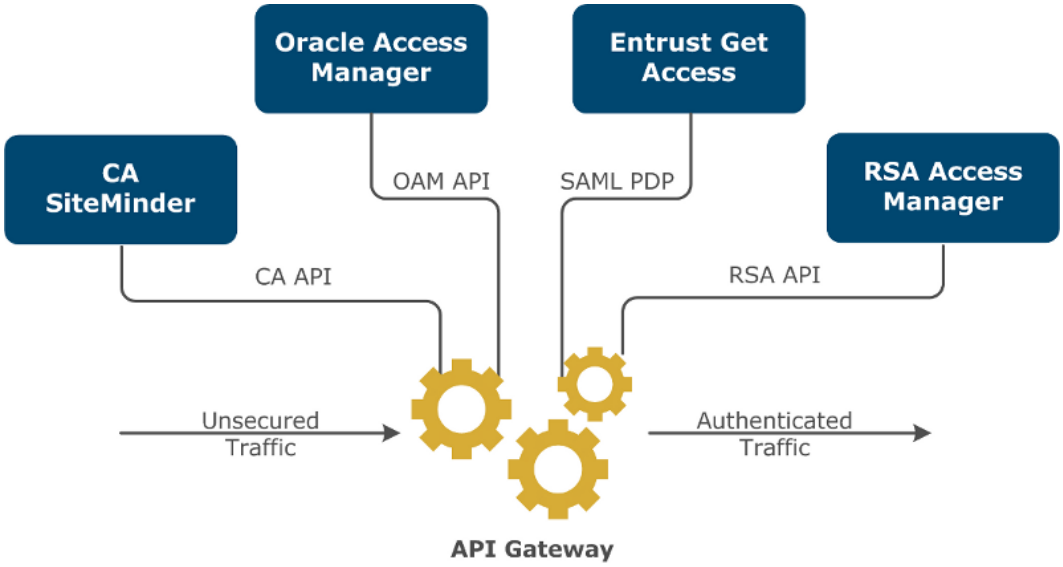
Disadvantages

This architecture has the following disadvantages:

- This is a more expensive option involving multiple API Gateways.

Access control

API Gateway interoperates with third-party Access Control and Identity Management products at a number of different levels. For example:



The options shown in the diagram are described as follows:

- API Gateway can directly connect into the Identity Management system as an agent. This solution is currently available for third-party products such as Oracle Access Manager, Oracle Entitlements Server, RSA Access Manager, CA SiteMinder, and IBM Tivoli Access Manager. The Identity Management policy is defined in the Identity Management product to which the API Gateway delegates the authentication and authorization.
- API Gateway can connect to the Identity Manager using the XML Access Control Markup Language (XACML) and Security Assertion Markup Language (SAML) protocols. API Gateway can request an authorization decision from a SAML Policy Decision Point (PDP) for an authenticated client using the SAML Protocol (SAMLP), which is defined in XACML. In such cases, the API Gateway presents evidence to the PDP in the form of user credentials, such as the Distinguished Name of a client X.509 certificate, or a username/password combination.
- The PDP decides whether a user is authorized to access the requested resource. It then creates an authorization assertion, signs it, and returns it to the API Gateway in a SAMLP response. The API Gateway can then perform a number of checks on the response, such as validating the PDP signature and certificate, and examining the assertion. It can also insert the SAML authorization assertion into the message for consumption by a downstream service. This enables propagation of the access control decision to occur.

Public Key Infrastructure

API Gateway supports Secure Sockets Layer (SSL) and Transport Layer Security (TLS) for transport-based authentication, encryption, and integrity checks. The API Gateway can interact with Public Key Infrastructure (PKI) systems in the following ways: Certificates and keys can be stored in the API Gateway keystore, or in a network or an optional Hardware Security Module (HSM). For more details, see *Manage X.509 certificates and keys* on page 106.

- Connecting to Online Certificate Status Protocol (OCSP) and XML Key Management Specification (XKMS) to query certificate status online.
- Using a Certificate Revocation List (CRL) retrieved from a directory, file, or LDAP to check certificate status.
- Checking a certificate chain.

Registries and repositories

API Gateway includes the following support for service registries and repositories:

- Architects and policy developers can use Policy Studio to pull Web service definitions (WSDL) from UDDI directories or HTTP-based repositories. These WSDL files are then used to generate the required security policies. The API Gateway can update UDDI registries with updated WSDL files or can serve them directly to the client. For more details, see the *API Gateway Policy Developer Guide*.
- When the API Manager product is installed, architects and policy developers can use Policy Studio to register REST APIs with the API Gateway. API administrators can then use API Manager to make these APIs available to client application developers in a Client Application Registry. For more details, see the *API Manager User Guide*.

Software Configuration Management

The API Gateway does not include its own built-in Software Configuration Management or Source Code Management (SCM) features. It is recommended that you use your existing SCM tools to manage your API Gateway configuration packages (`.fed`, `.pol`, and `.env` files) and API Gateway policy or general configuration exports (`.xml` files). For example, commonly used source code management tools include Concurrent Version System (CVS), Subversion (SVN), and Git.

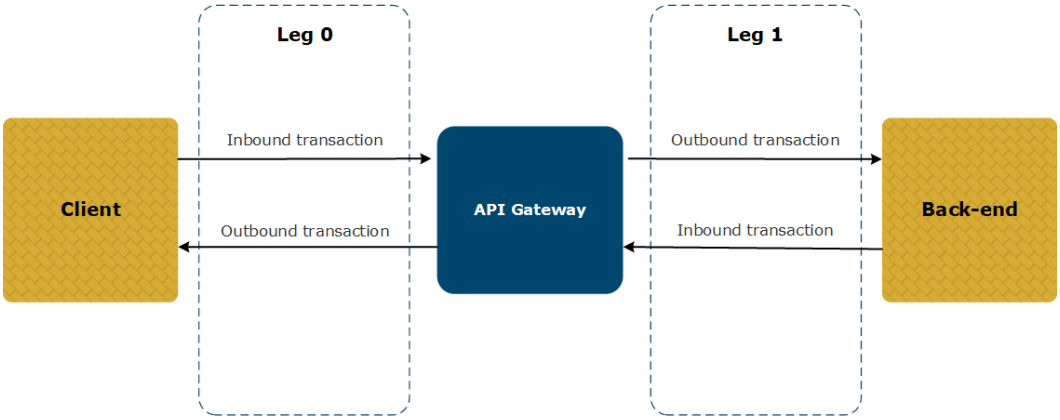
In addition, you can use the properties metadata associated with API Gateway configuration packages (`.fed`, `.pol`, and `.env` files) to associate version information with the packages that are provided to upstream environments. You can also use the API Gateway Compare/Merge features to determine changes between different configuration packages.

For more details on API Gateway configuration packages and their properties metadata, see the *API Gateway DevOps Deployment Guide*. For details on exporting API Gateway policies and general configuration, and on Compare/Merge, see the *API Gateway Policy Developer Guide*.

Introduction to transactions and legs in API Gateway

This topic explains inbound and outbound transactions and legs in the context of how a typical message transaction flows from a client to a back-end via an API Gateway.

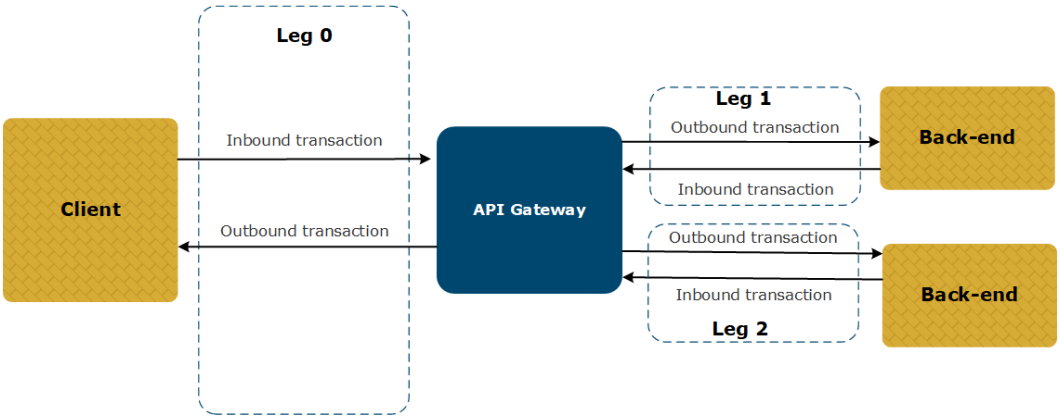
The following diagram shows a typical transaction flow through an API Gateway.



- An inbound transaction is one coming from a client or a back-end that is incoming to the API Gateway (received by the API Gateway).
- An outbound transaction is one that is outgoing from the API Gateway to a back-end or a client (sent by the API Gateway).
- Leg 0 is always the interaction between the client and the API Gateway.
- Leg 1 (and subsequent legs) are the interactions between API Gateway and the back-ends.
- The duration of leg 0 is the overall duration of the entire transaction (as seen by the client).
- The duration of each subsequent leg is the back-end transaction duration.

Tip The duration value for leg 0 minus the sum of the duration of all subsequent legs gives you the total time spent in the API Gateway for that transaction.

The following diagram shows a typical transaction flow when multiple back-ends are involved.



Manage an API Gateway domain 2

This part contains the following:

Configure an API Gateway domain

This topic describes how to use the `managedomain` command in interactive mode to configure a managed API Gateway. It shows how to register a host in a new domain, and create a new API Gateway instance. These are the minimum steps required to configure a domain.

Note This topic assumes that you have already installed the API Gateway (see the *API Gateway Installation Guide*). To use the API Gateway, you must have a domain configured in your installation.If you installed the QuickStart tutorial, an example domain was created automatically. If you did not install QuickStart, you must configure a domain using `managedomain`.

A single API Gateway installation supports a single API Gateway domain only. If you wish to run API Gateways in different domains on the same host, you need separate installations for each domain. For details API Gateway domains and groups, see the *API Gateway Concepts Guide*.

You can also use the topology view in the web-based API Gateway Manager tool to manage a newly created domain. For example, you can perform tasks such as create or delete API Gateway groups and instances,and start or stop API Gateway instances. For more details, see *Manage domain topology in API Gateway Manager* on page 43.

Managedomain script

When configuring a domain, the managedomain script enables you to perform tasks such as the following:

- Host management (registering and deleting hosts, or changing Admin Node Manager credentials)
- API Gateway management (creating and deleting API Gateway instances, or adding Linux services)
- Group management (editing or deleting API Gateway groups)

- Topology management (viewing topologies)
- Deployment (deploying to a group, listing deployments, creating or downloading deployment archives, and editing group passphrases)
- Domain SSL certificates (regenerating SSL certificates on localhost)

For example, you can use the `managedomain` script to register a host in a domain and create a new API Gateway instance. These are the minimum tasks required to create a new domain, and which are documented in this topic.

Managedomain options

For details on selecting specific options, enter the `managedomain` command in the following directory, and follow the instructions at the command prompt:

```
INSTALL_DIR/apigateway/posix/bin
```

Note To register an API Gateway instance as a service on Linux, you must run the `managedomain` command as `root`.

For more details on `managedomain` options, see *Managedomain command reference* on page 65. For details on how to use the `managedomain` command to configure SSL certificates and Admin Node Manager high availability, see *Configure Admin Node Manager high availability* on page 50.

Register a host in a domain

To register a host in a managed domain, perform the following steps:

1. Change to the following directory in your API Gateway installation:

   ```
   INSTALL_DIR/apigateway/posix/bin
   ```

2. Enter the following command:

   ```
   managedomain --menu
   ```

3. Enter **1** to register your host, and follow the instructions when prompted. For example, if this is the first host in the domain, enter **y** to configure an Admin Node Manager on the host. Alternatively, to add the host to an existing domain, enter **n** to configure a local Node Manager that connects to the Admin Node Manager in the existing domain.
4. Enter **q** to quit when finished.
5. Enter the following command to start the Admin Node Manager or local Node Manager on the registered host:

   ```
   nodemanager
   ```

Note Before registering multiple hosts in a domain, you must first ensure that a licensed API Gateway is installed on each host machine. Then to register each host, you must select option `1` on each host machine.

You must ensure the Admin Node Manager is running in the domain to enable monitoring and management of API Gateway instances.

Create an API Gateway instance

To create an API Gateway instance, perform the following steps:

1. Open a new command window.
2. Change to the following directory in your API Gateway installation:

```
INSTALL_DIR/apigateway/posix/bin
```

3. Enter the following command:

```
managedomain --menu
```

4. Enter **5** to create a new API Gateway instance, and follow the instructions when prompted. You can repeat to create multiple API Gateway instances on local or remote hosts.
5. Enter **q** to quit when finished.
6. Use the `startinstance` command to start API Gateway, for example:

```
startinstance -n "my_server" -g "my_group"
```

Note You can add an API Gateway instance on any registered host in the domain, not just the local host. However, if you are creating Linux services for API Gateway, you must run `managedomain` on same host.

You must run `startinstance` on the host on which you intend to start the instance. You must ensure that the `startinstance` file has execute permissions. Running `startinstance` without any arguments lists all API Gateway instances available on the host.

Test the health of an API Gateway instance

You can test the connection to the new API Gateway instance by connecting to the Health Check service. For example, enter the following default URL in your browser:

```
http://HOST:8080/healthcheck
```

This should display a simple `<status>ok</status>` message.

You can view the newly created API Gateway instance on the API Gateway Manager dashboard. For example, the default URL is as follows:

```
https://HOST:8090
```

The port numbers used to connect depend on those entered when configuring the domain using `managedomain`, and are available from the localhost only.

Alternatively, you can also connect to the new API Gateway instance in Policy Studio. For more details, see *Start and stop the API Gateway* on page 77.

Manage domain topology in API Gateway Manager

This topic describes how to use the topology view in the web-based API Gateway Manager tool to manage an existing API Gateway domain. For example, you can perform tasks such as the following:

- *Check the status of API Gateway groups and instances* on page 43
- *Check the installed version number of API Gateway instances* on page 45
- *Manage API Gateway groups* on page 47
- *Manage API Gateway instances* on page 48
- *Deploy API Gateway configuration* on page 49

Note When using API Gateway Manager to manage an existing domain, you must ensure that the host was first registered in the domain using the `managedomain` script. For more details, see *Configure an API Gateway domain* on page 40.

The API Gateway Manager web console is available from the following URL:

```
https://HOST:8090
```

For more details, see *Start the API Gateway tools* on page 81.

Check the status of API Gateway groups and instances

You can check the status of API Gateway groups and instances in the API Gateway Manager topology view.

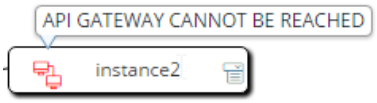
The possible status of an API Gateway group in the topology view is:

- Consistent – A group displayed with a blue icon is consistent. API Gateways in the group have the same configuration.

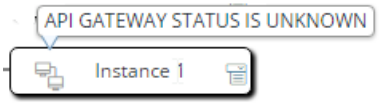
- Inconsistent – A group displayed with a black icon is inconsistent. API Gateways in the group do not have the same configuration (for example, during the phased roll out of configuration in a production environment). The tooltip displays `<GroupName> configuration is inconsistent`.
- Unlocked – API Gateways in the group are available for editing.
- Locked – API Gateways in the group are locked by a specific user, and are not available for editing. The tooltip displays `<GroupName> [locked by <UserName>]`.

The possible status of an API Gateway instance in the topology view is:

- Running – An API Gateway displayed with a green icon is running.
- Not running – An API Gateway displayed with a red icon is not running. The tooltip displays `API Gateway cannot be reached`.



- Unknown – An API Gateway displayed with a gray icon is not reachable. The Node Manager on the machine that the API Gateway is running on is down and the status cannot be determined. The tooltip displays `API Gateway status is unknown`.



The possible status of a host (Node Manager) in the topology view is:

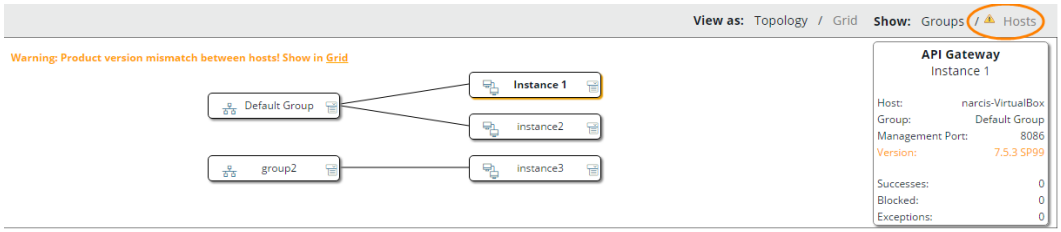
- Running – A host displayed with a blue icon is running.
- Not running – A host displayed with a red icon is not running. The tooltip displays `Node Manager cannot be reached`.

The topology view also displays the following messages and icons when there is a problem with the configuration:

- Version mismatch message – A warning message is displayed if there is a mismatch in product versions between hosts in a topology. The host status continues to show a blue icon.
- Warning icon – A warning icon is displayed if any of the following occur:
    - Node Manager is down

- API Gateway instance is down
- Version mismatch between hosts

For example:



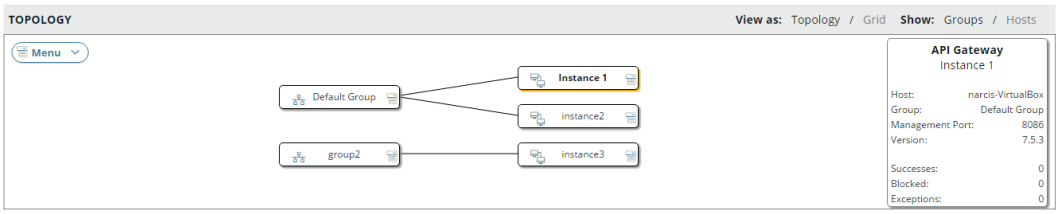
Check the installed version number of API Gateway instances

The installed product version number and any installed service pack are displayed in the API Gateway Manager topology and grid views.

When no service pack is installed on a system, API Gateway Manager shows the product version (for example, `7.6.2`). If a service pack is installed API Gateway Manager shows the product version and the service pack number (for example, `7.6.2 SP1`).

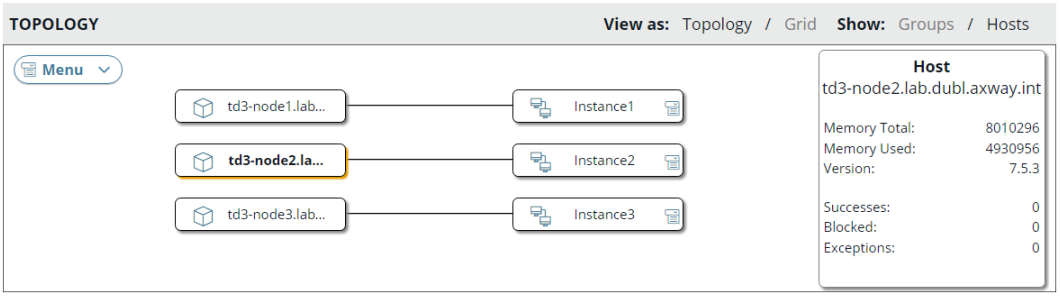
The version information is the same for all processes running on a host as they all use the same physical installation. Version information is shown at the host and API Gateway level. Version information is not shown at the group level as groups can span multiple hosts.

To view the version information for an API Gateway instance in the topology groups view, click **View as > Topology** and **Show > Groups** and click an API Gateway instance:



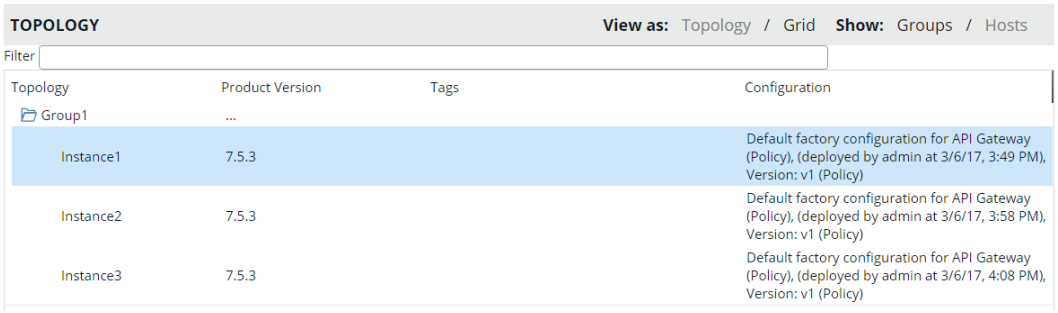
The version information of the API Gateway instance is shown in the Version field (for example, `7.5.3 SP1`).

To view the version information for a host in the topology hosts view, click **View as > Topology** and **Show > Hosts** and click a host:



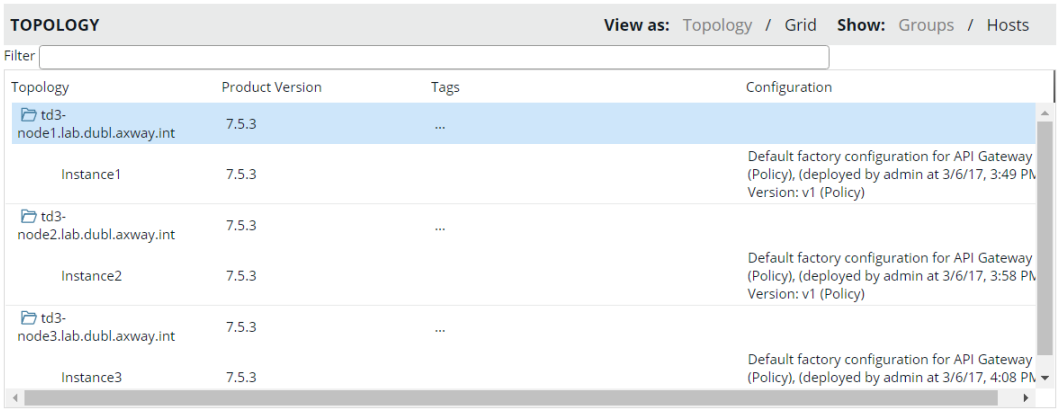
The version information of the host is shown in the Version field (for example, `7.5.3 SP1`).

To view the version information for an API Gateway instance in the grid groups view, click **View as > Grid** and **Show > Groups**:



The version information for each API Gateway instance in the group is shown in the Product Version column (for example, `7.5.3 SP1`).

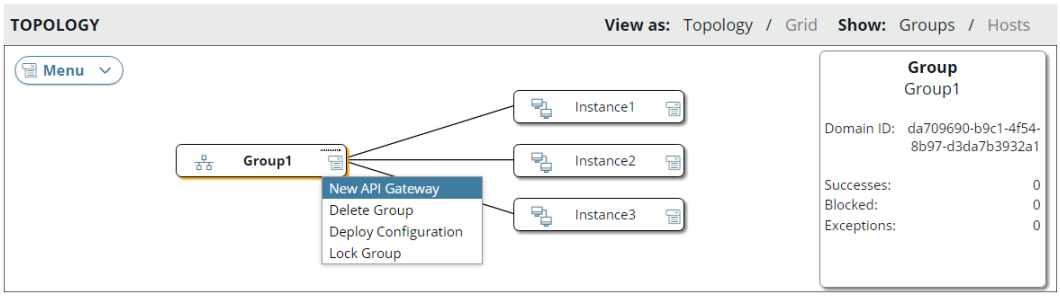
To view the version information for a host in the grid hosts view, click **View as > Grid** and **Show > Hosts**:



The version information for each host is shown in the Product Version column (for example, `7.5.3 SP1`).

Manage API Gateway groups

You can use the topology view in API Gateway Manager tool to create, delete, and lock API Gateway groups. The following example shows the options available at the group level:



Create an API Gateway group

To use the API Gateway Manager to create an API Gateway group, perform the following steps:

1. Click the **Menu** button in the topology view on the **Dashboard** tab.
2. Select **Create New Group**.
3. Enter a group name (for example, `Engineering`).
4. Click **OK**.

Delete an API Gateway group

To delete a group, perform the following steps:

1. Ensure that the API Gateway instances in the group have been stopped.
2. Hover over the group in the topology view, and click the edit button on the right.
3. Select **Delete Group**.
4. Click **OK**.

Lock an API Gateway group

To lock a group and prevent other users from editing its configuration, perform the following steps:

1. Hover over the group in the topology view, and click the edit button on the right.
2. Select **Lock Group**.
3. Click **OK**.

Similarly, to unlock an API Gateway group, select **Unlock Group**. Only administrator users can unlock groups locked by other users.

Manage API Gateway instances

You can use the topology view in API Gateway Manager to create, delete, start, and stop API Gateway instances.

Create an API Gateway instance

To use the API Gateway Manager to create an API Gateway instance, perform the following steps:

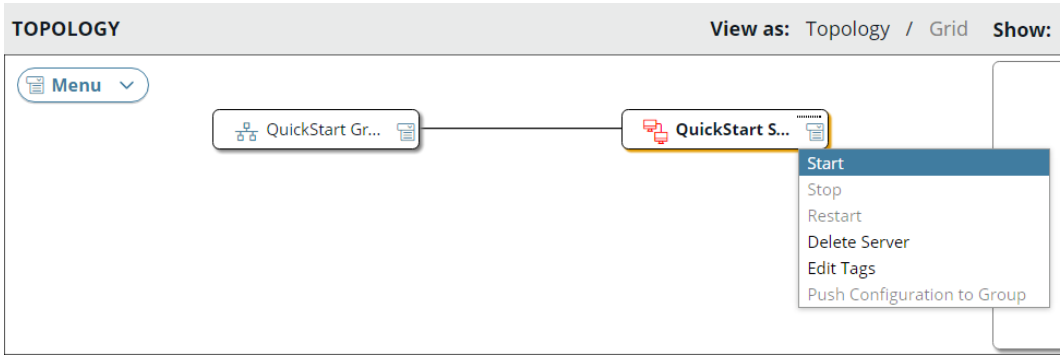
1. Hover over the API Gateway group in the topology view, and click the edit button on the right.
2. Select **New API Gateway**.
3. Configure the following fields:
    - **Name**: API Gateway instance name (for example, `Server2`).
    - **Management Port**: Local management port (for example, `8086`).
    - **Services Port**: External traffic port (for example, `8081`).
    - **Host**: Host address (for example, `127.0.0.1`).
    - **Domain CA Passphrase**: The passphrase for the domain SSL certificate authority.
4. Click **OK**.

Start an API Gateway instance

To start an API Gateway instance, perform the following steps:

1. Ensure that the API Gateway instance has been stopped.
2. Hover over the API Gateway instance in the topology view, and click the edit button on the right.
3. Select **Start**.
4. Click **OK**.

The following example shows how to start a stopped API Gateway instance:



Stop an API Gateway instance

To stop an API Gateway instance, perform the following steps:

1. Ensure that the API Gateway instance has been started.
2. Hover over the API Gateway instance in the topology view, and click the edit button on the right.
3. Select **Stop**.
4. Click **OK**.

Edit an API Gateway tag

API Gateway tags are name-value pairs that you can add to API Gateway instances, which then enable you to filter for API Gateway instances by tag in the API Gateway Manager**Dashboard**. To add an API Gateway tag, perform the following steps:

1. Hover over the API Gateway instance in the topology view, and click the edit button on the right.
2. Select **Edit Tags**.
3. In the dialog, click the green plus icon to add a tag.
4. Enter a **Tag** name (for example, `Department`), and a **Value** (for example, `Engineering`).
5. Click **Apply**.

To view the newly created tag in the in the API Gateway Manager topology view, click **View as** > **Grid**. To filter this view, enter a tag name or value in the **Filter** field.

Deploy API Gateway configuration

You can use the API Gateway Manager web console to deploy API Gateway configuration packages to a group of API Gateway instances. This includes the following configuration packages:

- deployment package (`.fed`)
- policy package (`.pol`)
- environment package (`.env`)

For more details on configuration packages, see the *API Gateway DevOps Deployment Guide*.

Deploy a deployment package

To deploy an existing deployment package to a group of API Gateways, perform the following steps:

1. In the **TOPOLOGY** view, right-click the API Gateway group to which to deploy the package, and select **Deploy Configuration**.
2. Select **I wish to deploy configuration contained in a single Deployment Package**.

3. Click **Browse for .fed**, and select the `.fed` file in the dialog.
4. Click **Next**.
5. Select **Deploy** in the wizard, and the deployment package is deployed to the API Gateway group.
6. Click **Finish**.

Deploy policy and environment packages

To deploy existing policy and environment packages to a group of API Gateways, perform the following steps:

1. In the **TOPOLOGY** view, right-click the API Gateway group to which to deploy the packages, and select **Deploy Configuration**.
2. Select **I wish to deploy configuration contained in Policy Package and Environment Package**.
3. Click **Browse for .pol**, and select the `.pol` file in the dialog.
4. Click **Browse for .env**, and select the `.env` file in the dialog.
5. Click **Next**.
6. Select **Deploy** in the wizard, and the packages are deployed to the API Gateway group.
7. Click **Finish**.

Tip You can also use Policy Studio to deploy API Gateway configuration. For details, see *Deploy API Gateway configuration* on page 141.

Configure Admin Node Manager high availability

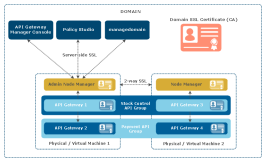
The Admin Node Manager (ANM) is the central administration server for an API Gateway domain, and is responsible for performing management operations across the domain. The Node Manager (NM) on each machine manages all the local API Gateways on that machine, regardless of the group they are in. This includes collecting monitoring information, managing dynamic settings, deploying configuration, and so on.

In addition to managing the local API Gateways on its host, the Admin Node Manager communicates with the Node Managers in the domain to perform management operations across the domain. In this architecture, the Node Managers only communicate with the Admin Node Manager, and the API Gateway Manager, Policy Studio, and `managedomain` tools connect to the Admin Node Manager. For more details on Admin Node Manager architecture, see the *API Gateway Concepts Guide*.

Note It is recommended that you configure at least two Admin Node Managers in an API Gateway domain for high availability (HA). This topic describes how to use the using the `managedomain` command to configure and secure multiple Admin Node Managers in a domain.

Hierarchy of SSL certificates in a domain

API Gateway uses Secure Sockets Layer (SSL) for communications between all processes in a domain. This includes internal management traffic between the Admin Node Manager, Node Manager, and API Gateway instances, as shown in the diagram.



This diagram is described as follows:

- Server-side SSL authentication is used for communication between the Admin Node Manager and client applications (Policy Studio, API Gateway Manager, `managedomain`).
- Mutual SSL is used for communication between Admin Node Managers and Node Managers, and also between Node Managers and API Gateway instances.
- Each Admin Node Manager, Node Manager, and API Gateway process has an associated SSL certificate used in the SSL session for inter-process communication. These certificates are also used to determine whether a process is a Node Manager or an Admin Node Manager.
- The domain SSL certificate is the Certificate Authority (CA) used to sign the SSL certificates generated for each Admin Node Manager, Node Manager, and API Gateway process.

Note The SSL inter-process communication discussed in this topic refers to system-level management communication only. This is not related to any SSL communication with back-end services, or any SSL ports that the API Gateway exposes for business traffic.

How SSL certificates are generated for domain processes

Each time you register a new Admin Node Manager or Node Manager, or create a new API Gateway instance, a new SSL certificate is generated for inter-process communication. Each SSL certificate associated with an Admin Node Manager, Node Manager, or API Gateway instance is signed by the same CA. You must use one of the following options to sign these SSL certificates:

Signing option	Description
System-generated CA key and certificate	By default, a system-generated CA key and certificate is used to sign all SSL certificates. The system creates a private key and self-signed certificate on the first Admin Node Manager to be registered. All subsequent requests for new SSL certificates for inter-process communication are processed by this Admin Node Manager which has the CA key.

Signing option	Description
User-provided CA key and certificate	Provide your own CA key by specifying a P12 file that contains a key and certificate. The system uses this CA key to sign all new SSL certificates for inter-process communication.
External CA	A private key and Certificate Signing Request (CSR) file are generated by the system, and then you can take this CSR to your chosen external CA. When you receive the certificate from the external CA, you must submit the certificate to `managedomain`, which then completes registering the new Admin Node Manager, Node Manager, or API Gateway instance associated with that certificate.

These options are available when using `managedomain` to register a new Admin Node Manager or Node Manager, or to create a new API Gateway instance, and when regenerating all SSL certificates for the domain. These options are also available when editing a host in `managedomain` (for example, when you change a Node Manager to an Admin Node Manager a new certificate is generated).

Note Private keys for the SSL certificates are always generated and stored on the host that uses them. This applies to a system-generated CA key, a key for an Admin Node Manager or Node Manager SSL certificate, or a key for an API Gateway SSL certificate for inter-process communication.

External Certificate Authority

You can use a third-party external CA to sign the SSL certificates (for example, Verisign). This is the typical approach that is recommended in a production environment. For more details, see your external CA user documentation.

In a development environment, you can install API Gateway on a locked down host to act as an external CA for testing purposes. No license is required because only `managedomain` is running. This external CA host stores a system-generated CA private key and certificate. You can use `managedomain` to process CSRs and output certificates.

In this scenario, the CSR file is generated by an API Gateway installation on another host (with a Node Manager and API Gateways running), and transported to the external CA installation out-of-band. The certificate file generated by the external CA is transported back out-of-band to the installation that created the CSR and where the new Node Manager or API Gateway is created. The certificate is specified to `managedomain` using the submit certificate option to complete Node Manager host registration or API Gateway creation.

Add the first Admin Node Manager to the domain

Note When registering the first Admin Node Manager, you must run `managedomain -i` on the host that will run the Admin Node Manager.

This generates the private key for the Admin Node Manager, and private keys must always be generated where they are used. No SSL certificates are provided out-of-the-box (when the QuickStart demo is not installed).

Sign Admin Node Manager certificate with system-generated CA key

To register the first Admin Node Manager in the domain and sign its SSL certificate with a system-generated CA key, run the following command:

```
managedomain -i
```

This command generates the CA private key, and a self-signed certificate. It also generates the Admin Node Manager private key and certificate signed by self-signed CA certificate. This command uses the `--sign_with_generated` option by default, which means that the SSL certificates are signed with a system-generated CA private key and certificate. There are no Node Managers running in the domain to call yet.

Sign Admin Node Manager certificate with user-provided CA key

To register the first Admin Node Manager in the domain and sign its SSL certificate with a user-provided CA key, run the following example command:

```
managedomain -i --sign_with_user_provided --ca=/home/keys/test.p12i
```

This command generates the Admin Node Manager private key and certificate signed by user-provided CA certificate.

Sign Admin Node Manager certificate with external CA

To register the first Admin Node Manager in the domain and sign its SSL certificate with an external CA, run the following example command:

```
managedomain -i --sign_with_external_ca
```

This command generates the Admin Node Manager private key and the CSR for the certificate. It then prompts you to take your CSR file to your external CA and return with the signed certificate.

When you have the signed certificate, use the following command to submit the certificate and finish creating the Admin Node Manager configuration:

```
managedomain --submit_cert --cert cert.pem
```

API Gateway as external CA

In a production environment, you can use the `--sign_with_external_ca` option, and take the generated CSR to an external CA (for example, Verisign). Alternatively, in a development environment, you can run the following command to generate the certificate on an isolated API Gateway installation:

```
managedomain --sign_csr --csr nodemanager.csr
```

This command creates a certificate signed using the system-generated CA key. It writes the certificate to a file, displays the file name so you can retrieve and copy to the host running `managedomain` , and complete registration of the new Node Manager.

Tip If you wish to avoid using a system-generated self-signed certificate, you can manually copy a user-provided CA private key and certificate to the following locations:

```
apigateway/groups/certs/private/domain.p12apigateway/groups/certs/privat
e/domainkey.pem apigateway/groups/certs/domaincert.pem
```

Additional certificate generation options

In addition to the example commands already shown, you can specify the following options when using `managedomain` to sign SSL certificates for inter-process communication:

Option	Description
`--domain_passphrase`	Encrypt a newly generated `domain.p12`, or unlock a user-provided `domain.p12`.
`--domain_name`	Specify the Distinguished Name (DName) for a system-generated CA certificate. Defaults to `CN=Domain`.
`--sign_alg`	Specify the certificate signing algorithm (for example, `sha1`, `sha256`, `sha384`, or `sha512`). Defaults to `sha256`.

Option	Description
`--subj_alt_name`	Specify a subject alternative name content for the Admin Node Manager certificate. This is used by web browsers during the SSL handshake exchange.
`--key_passphrase`	You can also encrypt temporary private key files stored on disk. For example, for the Admin Node Manager private key before it is consumed and written to the Node Manager configuration.

Note You can specify multiple subject alternative names, for example:

```
--subj_alt_name "DNS.1=name.axway.com" --subj_alt_name  "DNS.2=othername.axway.com"
   --subj_alt_name "IP.1=10.142.1.2"
```

Add a Node Manager to the domain

The steps for adding a Node Manager or Admin Node Manager are different because there is now an Admin Node Manager running in the domain that must be invoked to add the new Node Manager.

Sign Node Manager certificate with system-generated CA key

To add a Node Manager to the domain and sign its SSL certificate with a system-generated CA key, run the following command:

```
managedomain -a --sign_with_generated --anm_host my_anm.com
```

This command generates the Node Manager private key, obtains the Node Manager certificate from the Admin Node Manager, and completes registration of the new Node Manager in the domain.

Sign Node Manager certificate with user-provided CA key

To add a Node Manager to the domain and sign its SSL certificate with a user-provided CA key, run the following command:

```
managedomain -a --sign_with_user_provided --ca=/home/keys/test.p12
 --anm_host my_anm.com
```

This command signs the new Node Manager certificate with the user-provided CA key and certificate, and completes registration of the new Node Manager in the domain.

Sign Node Manager certificate with external CA

To add a Node Manager to the domain and sign its SSL certificate with an external CA, run the following command:

```
managedomain -a --sign_with_external_ca --anm_host my_anm.comm
```

This command performs similar steps to adding the first Admin Node Manager to the domain and signing with an external CA.

When you have the signed certificate, use the following command to submit the certificate and finish creating the Node Manager configuration:

```
managedomain --submit_cert --cert cert.pem
```

See also *Sign Admin Node Manager certificate with external CA* on page 53.

Additional options

You can use the `--is_admin` option to specify whether the process is an Admin Node Manager or Node Manager. For more details, see *Change a Node Manager to an Admin Node Manager* on page 57. See also *Additional certificate generation options* on page 54.

Add an API Gateway instance to the domain

This section describes the options for signing an API Gateway instance SSL certificate when adding an API Gateway instance to the domain.

Sign API Gateway certificate with system-generated CA key

To add an API Gateway instance to the domain and sign its SSL certificate with a system-generated CA key, run the following example command:

```
managedomain -c -n APIGateway1 -g Group1
```

This command uses the `--sign_with_generated` option by default.

Sign API Gateway certificate with user-provided CA key

To add an API Gateway instance to the domain and sign its SSL certificate with a user-provided CA key, run the following example command:

```
managedomain -c -n APIGateway1 -g Group1 --sign_with_user_provided --
ca=/home/keys/test.p12
```

Sign API Gateway certificate with external CA

To add an API Gateway instance to the domain and sign its SSL certificate with an external CA, run the following example command:

```
managedomain -c -n APIGateway1 -g Group1 --sign_with_external_ca
```

This command performs similar steps to adding the first Admin Node Manager to the domain and signing with an external CA.

When you have the signed certificate, use the following command to submit the certificate and finish creating the API Gateway configuration:

```
managedomain --submit_cert --cert cert.pem
```

See also *Sign Admin Node Manager certificate with external CA* on page 53 and *Additional certificate generation options* on page 54.

Change a Node Manager to an Admin Node Manager

Tip When you add a Node Manager to the domain using the `-a` option, you can add it as a Node Manager or Admin Node Manager. For more details, see *Add a Node Manager to the domain* on page 55. The section that follows describes how to change whether it is a Node Manager or Admin Node Manager at a later stage.

An existing Node Manager in a domain can only communicate with API Gateway instances on the same host machine, and only an Admin Node Manager can connect to a Node Manager. This means that if you have a single Admin Node Manager in a domain, this is a single point of failure for management capabilities. If the single Admin Node Manager fails, you cannot manage or monitor the domain or deploy configuration. However, you can change an existing Node Manager to act as an Admin Node Manager in the domain. A domain can run with any number of Admin Node Managers, but it is recommended that you have at least two Admin Node Managers in a domain.

You can change the admin capabilities of the Node Manager running on the host local to `managedomain` using the `--edit_host` option. This enables you to change a Node Manager to an Admin Node Manager, or change an Admin Node Manager to a Node Manager. Changing admin capabilities means that the tags on the process in the topology are updated, and a new SSL certificate is generated.

The same options for signing an SSL certificate for a Node Manager or API Gateway instance (described in the preceding sections) are available when changing a Node Manager to an Admin Node Manager, or an Admin Node Manager to a Node Manager.

Note To change admin capabilities of an Admin Node Manager or Node Manager, you must run `managedomain` on the same host that the Admin Node Manager or Node Manager you wish to change is running. You can edit other aspects of a host remote from `managedomain`.

You cannot remove admin capabilities when there is only one Admin Node Manager in a domain because there must be at least one Admin Node Manager in a domain.

Sign Node Manager certificate with system-generated CA key

When `host1.com` has an Admin Node Manager, to change admin capabilities from Admin Node Manager to Node Manager, run the following example command:

```
managedomain --edit_host --host host1.com
```

In this case, because `host1.com` is an Admin Node Manager and the `--is_admin` parameter is not passed, `managedomain` interprets this command as a change of capabilities from Admin Node Manager to Node Manager. This command uses the `--sign_with_generated` option by default.

When `host1.com` has a Node Manager, to change admin capabilities from Node Manager to Admin Node Manager, run the following example command:

```
managedomain --edit_host --host host1.com --is_admin
```

In both cases, the command generates and updates the private key and certificate in the local Node Manager configuration, and prompts you to reboot the Node Manager.

Note To remove admin capabilities from the first registered Admin Node Manager, you must first move the following directories to another Admin Node Manager

- `apigateway/groups/certs`
- `apigateway/groups/certs/private`

Otherwise, you will not be able to remove admin capabilities. This only applies to a system-generated CA key.

Sign Node Manager certificate with user-provided CA key

When `host1.com` has an Admin Node Manager, to change admin capabilities from Admin Node Manager to Node Manager, run the following example command:

```
managedomain --edit_host --host 10.142.12.31 --sign_with_user_provided --
ca=/home/keys/test.p12
```

When `host1.com` has a Node Manager, to change admin capabilities from Node Manager to Admin Node Manager, run the following example command:

```
managedomain --edit_host --host host1.com --sign_with_user_provided --
ca=/home/keys/test.p12 --is_admin
```

In both cases, the command performs the same steps described in *Sign Node Manager certificate with system-generated CA key* on page 58, and the same conditions apply.

Sign Node Manager certificate with external CA

When `host1.com` has an Admin Node Manager, to change admin capabilities from Admin Node Manager to Node Manager, run the following example command:

```
managedomain --edit_host --host host1.com --sign_with_external_ca
```

When `host1.com` has a Node Manager, to change admin capabilities from Node Manager to Admin Node Manager, run the following example command:

```
managedomain --edit_host --host host1.com --sign_with_external_ca --is_
admin
```

In both cases, this command performs similar steps to adding the first Admin Node Manager to the domain and signing with an external CA.

When you have the signed certificate, use the following command to submit the certificate and finish updating the Node Manager configuration:

```
managedomain --submit_cert --cert cert.pem
```

See also *Additional certificate generation options* on page 54.

Regenerate all SSL certificates in a domain

This section explains how to regenerate all SSL certificates used for inter-process communication in the domain. You can do this using a system-generated self-signed CA key and certificate, a user-provided CA key and certificate in a P12 file, or an external CA.

You can also change the domain certificate management option. For example, if the domain was first configured to use a system-generated CA key and certificate to sign all SSL certificates, you can change to a user-provided CA key and certificate, or an external CA. However, you must use the same option on all hosts in the domain.

Sign certificates in domain with system-generated CA key

Note You must first regenerate on the Admin Node Manager host that will hold the system-generated self-signed CA key. This may or may not be the same host that held the system-generated self-signed CA key before regeneration.

If you select a different host to hold the CA private key and certificate, you must delete the old CA private key and certificate from the other Admin Node Manager host to prevent certificates being signed by two different CA keys. This must be done before regeneration.

To regenerate all certificates on a host and sign with a system-generated key, run the following command on each host in the domain:

```
managedomain --regen_certs
```

This command uses the `--sign_with_generated` option by default. It regenerates the CA private key and self-signed certificate, and private keys and certificates for the Admin Node Manager, Node Manager, and API Gateway instances on the host that the command was run.

After regenerating certificates, you must reboot the Node Manager and API Gateway instances on the local machine.

Tip When you regenerate certificates on the first Admin Node Manager, a Node Manager or API Gateway does not need to be running. When you regenerate certificates on subsequent hosts, the Admin Node Manager holding the CA key must be running because it is used to sign the certificates. This applies only for system-generated CA keys.

Sign certificates in domain with user-provided CA key

To regenerate all certificates on a host and sign with a user-provided key, run the following example command on each host in the domain:

```
managedomain --regen_certs --sign_with_user_provided --
ca=/home/users/qa.p12 --domain_passphrase foo
```

This command generates the private keys for the Node Manager and API Gateway instances and signs their certificates. After regenerating the certificates, you must then reboot the Node Manager and API Gateway instances on the local machine.

In this case, `managedomain` does not communicate with an Admin Node Manager when regenerating certificates. The order of hosts selected for certificate regeneration does not matter. An Admin Node Manager does not need to be running when certificates are regenerated or submitted.

Sign certificates in domain with external CA

To regenerate all certificates on a host and sign with an external CA, run the following command on each host in the domain:

```
managedomain --regen_certs --sign_with_external_ca
```

This command generates the private keys for the Node Manager and API Gateway instances. You are prompted to take the CSR files to your CA and return with the signed certificates.

When you have the signed certificates, run the following command for each certificate:

```
managedomain --submit_cert --cert cert.pem
```

After regenerating certificates, you must reboot the Node Manager and API Gateway instances on the local machine. In this case, `managedomain` does not talk to an Admin Node Manager when regenerating certificates. The order of hosts selected for certificate regeneration does not matter. An Admin Node Manager does not need to be running when certificates are regenerated or submitted.

Tip You can use `managedomain` in interactive mode (option `24`) to regenerate a subset of certificates on the host. If you regenerate the CA key and certificate, the Node Manager and all API Gateway certificates are automatically generated.

If you do not regenerate the CA key and certificate, you can choose whether to regenerate the Node Manager certificate, and which API Gateway certificates to regenerate. For more details, see *Managedomain command reference* on page 65.

Reset passphrase for CA private key

If a system-generated CA private key is used, you can reset the passphrase on the command line as follows:

```
managedomain --change_domain_passphrase --old_passphrase "oldpass" --new_
passphrase "newpass"
```

Note You can only run this command on the host that has the `domain.p12` file. It does not affect any of the SSL certificates that have already been generated.

If a new SSL certificate is to be generated (due to a new Node Manager or new API Gateway instance), the new domain passphrase must be provided to unlock the CA private key. This command modifies local files only. It does not communicate with any Node Managers in the domain.

Change domain SSL certificate expiry date

By default, the SSL certificates are valid for 100 years. To change this value, edit the `apigateway/skel/openssl.cnf` file, and change the value of the `default_days` setting in the `CA_default` section.

```
[CA_default]
.
.
default_days = 36500
.
.
```

This setting applies to signing with a system-generated or user-generated CA key, or if you are using an API Gateway installation as an external CA. If you are using a third-party external CA, the CA decides on the expiry date.

When the `default_days` setting is changed, all newly signed certificates have the new expiry date. You do not need to reboot the Node Manager. However, you must regenerate certificates if you wish to modify the expiry date of existing certificates.

For more details, see *Location of SSL private keys and certificates* on page 63.

Admin Node Manager backup and disaster recovery

This section explains how to create a backup Admin Node Manager for signing certificates, and how to set up an Admin Node Manager for signing certificates from a backup `domain.p12` file.

Create backup Admin Node Manager for signing certificates

If you are using a system-generated CA private key and certificate, the CA private key and certificate are only on the first Admin Node Manager registered in the domain. You can manually copy this CA private key and certificate to other Admin Node Managers so that there is always a backup Admin Node Manager available to sign certificates. Alternatively, you can copy the CA private key and certificate only when there is a problem with the first Admin Node Manager (for example, it is no longer running).

To create a backup Admin Node Manager that can sign certificates, copy the following files from Admin Node Manager 1 to Admin Node Manager 2:

- `apigateway/groups/certs/domaincert.pem`
- `apigateway/groups/certs/index.txt`
- `apigateway/groups/certs/serial`
- `apigateway/groups/certs/private/domain.p12`
- `apigateway/groups/certs/private/domainkey.pem`

You do not need to reboot Admin Node Manager 2 to be able to sign certificates. You can test the Admin Node Manager's ability to sign certificates by taking down Admin Node Manager 1. Connect the client to Admin Node Manager 2, and add an API Gateway instance on any Node Manager that is running.

Note You can decide when and if to copy the CA private key to limit the copies of this sensitive data. You must always backup the `domain.p12` file after the registration of the first Admin Node Manager in some secure offline location.

Set up Admin Node Manager for signing certificates from a backup .p12

You can set up an Admin Node Manager for signing certificates from a backup `domain.p12` file only. Perform the following steps:

1. Copy the `domain.p12` file to `apigateway/groups/certs/private/` .
2. Create an `apigateway/groups/certs/serial` file with contents of `01` .
3. Create an empty `apigateway/groups/certs/index.txt` file.
4. Parse `domain.p12` into `domainkey.pem` and `domaincert.pem` (if these files were not backed up originally). For example, assuming a passphrase of `fred` for the CA private key:

```
cd apigateway/groups/certs/privateopenssl pkcs12 -in domain.p12 -out
../domaincert.pem -nokeys -passin pass:fredopenssl pkcs12 -in domain.p12
-out domainkey.pem -nocerts -passin pass:fred -passout pass:fred
```

For more details, see the OpenSSL user documentation.

Location of SSL private keys and certificates

This section describes the locations of the SSL private keys and certificates for the CA, Node Manager, and API Gateway instances.

Location of CA private key and certificate

The location of the CA private key and certificate depends on how they are signed.

System-generated CA private key and certificate :If SSL certificates are signed by a system-generated CA private key and certificate, the CA private key and certificate are written to disk when the first Admin Node Manager is created by `managedomain`:

- **CA private key**: `apigateway/groups/certs/private/domainkey.pem`
- **CA private key and certificate**: `apigateway/groups/certs/private/domain.p12`
- **CA certificate**: `apigateway/groups/certs/domaincert.pem`

Caution The files in `apigateway/groups/certs/private` are security sensitive because they contain the private key. This directory must have restricted access. These files are not copied to any other Node Manager, and must be backed up elsewhere.

If the CA private key and certificate are lost, you must recreate certificates for the entire domain before adding new Node Managers or API Gateways.

The CA private key files can be encrypted by a passphrase. You must enter this with `managedomain` when creating a new Node Manager or API Gateway. This is used to unlock the CA private key and sign the new SSL certificate.

User-provided CA private key and certificate: If SSL certificates are signed by a user-provided key and certificate, the key and certificate are in a location managed by the user. A `.p12` file containing the private key and certificate must be made available to `managedomain` when a new Node Manager or API Gateway is created. The CA certificate is copied into the API Gateway configuration because it is part of the SSL certificate chain.

External CA private key and certificate: If SSL certificates are signed by an external CA, the CA private key remains with the external third-party CA, or isolated API Gateway installation acting as an external CA. The CA certificate is copied to the API Gateway configuration because it is part of the SSL certificate chain.

Location of Node Manager private key and certificate

The Node Manager private key and certificate chain are stored in the Node Manager configuration in the following directory:

```
apigateway/conf/fed
```

The private key is encrypted using the entity store passphrase.

Location of API Gateway private key and certificate

The API Gateway private key and certificate chain are stored in the following directory:

```
apigateway/groups/group-2/instance-1/conf/certs.xml
```

The private key is encrypted with the group configuration passphrase.

Caution When Node Managers and API Gateway instances are being registered, private keys are written in `apigateway/groups/certs/private/temp/instance-id` and `nodemanager-id` .Temporary security sensitive files are removed on completion.

If an external CA is signing the certificate, sensitive files may exist for a considerable time, so this directory must have restricted access. You can encrypt this sensitive data using the `--key_passphrase` parameter.

Further information

For more details on Admin Node Manager architecture, see the *API Gateway Concepts Guide*.

For example Admin Node Manager deployment scenarios, see *Configure API Gateway high availability* on page 83.

Managedomain command reference

This topic describes how to run `managedomain` in the following modes:

- *Command interpreter mode* — enter commands from a list using tab completion (default mode)
- *Interactive mode* — follow instructions at the command prompt
- *Command mode* — enter specific commands and parameters

The `managedomain` command is available in the following directory:

```
INSTALL_DIR/apigateway/posix/bin
```

For an overview of the `managedomain` command, see *Configure an API Gateway domain* on page 40.

Managedomain command interpreter mode

To run in default command interpreter mode, enter `managedomain` , and press Tab to view and select options. For example:

```
Axway-7.6.2/apigateway/posix/bin>managedomain
Running in command interpreter mode. Enter 'quit' to exit.
Enter 'help' to view help topics. Enter 'help topic' to view help for a topic.
Press TAB to view and complete commands.
>
add                         add_service_only          add_tag
change_domain_passphrase    change_passphrase         create_archive
create_instance             delete_group              delete_host
delete_instance             delete_tag                deploy
download_archive            edit_group                edit_host
edit_instance               help                      initialize
list_deployments            print_topology            q
quit                        regen_certs               remove_service
reset                       sign_csr                  submit_cert
topology_check              topology_compare          topology_merge
topology_synch              version
>
```

Note You must first run `initialize` to register the first host in the domain in order to create and run API Gateways.

View help for a command

You can view detailed help for each command and its parameters by entering `help` followed by the command name. The following example shows the help for the `initialize` command:

```
> help initialize

Register the first Node Manager and host in a new domain. This Node Manager will
always be an Admin Node Manager.

*** Usage ***
initialize
initialize nm_name NodeManagerOne
initialize host host1.axway.com port 8090
initialize domain_name DomainOne domain_passphrase 12345678 sign_alg sha256
initialize sign_with_user_provided ca ca.p12 domain_passphrase password
initialize sign_with_external_ca
initialize subj_alt_name "DNS.1=host1.com" subj_alt_name "DNS.2=host2.com"

*** Command Parameters ***
nm_name : User-friendly name of new Node Manager. Defaults to 'Node Manager
on host name'.

host : Host name. Default host name for the localhost will be used if not
provided.

port : Port for new Node Manager. Default is env.PORT.MANAGEMENT from
envSettings.props (i.e., 8090).

sign_with_generated : Use a system-generated self-signed domain certificate
to sign the SSL certificates for Node Managers and API Gateways. When the
--initialize command is run, the domain private key and certificate will be
generated. This is the default cert management option.

sign_with_user_provided : Use a user-provided domain key and certificate to
sign SSL certificates for Node Managers and API Gateways.

sign_with_external_ca : Use an external CA to sign SSL certificates for Node
Managers and API Gateways. managedomain will output CSRs which must be sent to
the external CA. Use --submit_cert command to submit the certificate when returned
from the external CA.

domain_passphrase : Passphrase for the domain private key, use with
--sign_with_generated and --sign_with_user_provided for non-default passphrase.
Defaults to no passphrase.
```

```
sign_alg : Optional signing algorithm for domain certificates (e.g., 'sha1',
'sha256', 'sha384' or 'sha512'). 'sha256' by default.

ca : Name of PKCS12 file with user-provided domain key and certificate, use with
--sign_with_user_provided.

key_passphrase : Used to encrypt temporary key files. Defaults to no passphrase.
This
is optional, but recommended especially for --sign_with_external_ca when temporary
sensitive files may reside on disk for considerable time.

domain_name : Name of the domain, this will be used as the dname in the domain
certificate if it is system-generated, use with --sign_with_generated. Maximum
length
is 64 characters.

subj_alt_name : Subject alternative names for Node Manager SSL certificates, can
specify multiple names. Default subject alternative names will be added if not
provided.

add_service : create Linux service for the new server (Node Manager or API Gateway).

service_user : User to run service as.

nm_entitystore_passphrase : The entity store password

metrics_enabled : Controls whether metrics data collection is enabled or not.

metrics_dburl : Metrics database URL e.g. jdbc:mysql://127.0.0.1:3306/DefaultDB or
jdbc:mysql://127.0.0.1:3306/reports

metrics_dbuser : The database username e.g. root or metrics db user.

metrics_dbpass : The plaintext password associated with the database username.

debug : enable debugging.
```

Run a command

You can run a command using tab completion to specify parameters. The following example shows the available parameters for the `create_instance` command:

```
> create_instance <press TAB>
name                        group                       host
instance_management_port    instance_services_port      passphrase
sign_with_generated         sign_with_user_provided     sign_with_external_ca
domain_passphrase           sign_alg                    ca
key_passphrase              add_service                 service_user
```

```
debug                    anm_host                 anm_port
username                 password                 truststore_file
truststore_password
```

The following example creates a new API Gateway instance with a specific name and group:

```
> create_instance name APIServer1 group Group1

Requesting CSR from Admin Node Manager...
CSR received from Admin Node Manager.
Requesting signed certificate from Admin Node Manager...
Signed certificate received from Admin Node Manager.
Requesting Admin Node Manager to create new API Gateway...
New API Gateway SSL certificate details:
    dname:CN=instance-1,OU=group-2,DC=host-1
    expires:Fri Apr 27 16:36:37 BST 2114
    thumbprint:E6:C5:B5:6D:52:1D:74:3E:CD:3E:8B:B5:82:24:3A:78:1B:C9:27:F9
    issuer dname:CN=Domain
    issuer thumbprint:B5:68:73:7A:7B:ED:6D:B0:04:40:CF:E3:BC:36:6D:84:F7:49:29:12

The new API Gateway 'APIServer1' in group 'Group1' has been successfully created and
installed

Start the new API Gateway by executing the following command:
Axway-7.6.2/apigateway/posix/bin/startinstance -g "Group1 " -n "APIServer1"

You can alternatively add Axway-7.6.2/apigateway/posix/bin/ to your path and use
"startinstance -g "Group1" -n "APIServer1"".

You can test the connection by visiting the URL:
http://roadrunner:8080/healthcheck
```

Tab completion is also available for some parameter values (instance names, group names and host names). The following example shows available instances for the `delete_instance` command:

```
> delete_instance name <press TAB>

APIServer1   APIServer2

> delete_instance name APIServer
```

Managedomain interactive mode

To run in interactive mode, enter `managedomain --menu`, and follow the instructions at the command prompt. The following options are available:

Host management

The `managedomain --menu` options for host management are as follows:

Option	Description	Why use this option
1	`Register host`	Add a new host that runs an API Gateway to a domain topology. This is equivalent to the `initialize` command. You must ensure that the host is registered in order to create and run API Gateways. For example, you can specify the following: • If the host is an Admin Node Manager • Host name • Node Manager name • Node Manager port • Node Manager passphrase • Linux service for Node Manager • Trust store details When registering a Node Manager (Admin or not) in an existing domain, you must specify host details of a running Admin Node Manager in the domain. This is not required when registering the first Admin Node Manager in a new domain because this is always an Admin Node Manager.

Option	Description	Why use this option
2	`Edit a host`	Edit the details for a host registered in a domain topology (used occasionally). You can update the following: • Host name • Node Manager name • Node Manager port • Node Manager passphrase • Linux service for Node Manager • Change admin capabilities • Enable metrics Changing admin capabilities enables you to change a Node Manager to Admin Node Manager, or an Admin Node Manager to Node Manager. You can only change admin capabilities of the Node Manager running on the same machine. You cannot remove admin capabilities of the last Admin Node Manager in a domain, or from an Admin Node Manager that has the domain key and certificate used to sign CSRs. When you get a license for an evaluation mode API Gateway, you must use this option to change the host from `127.0.0.1` to a network reachable address or host name. You must also restart the Node Manager to pick up any changes.
3	`Delete a host`	Delete a registered host from a domain topology (used occasionally). You must first stop and delete all API Gateways running on the host. You can use this option to delete an Admin Node Manager or Node Manager. The Admin Node Manager that services this request is not allowed to delete itself from the domain, ensuring the domain always has at least one Admin Node Manager.

Option	Description	Why use this option
4	`Change Admin Node Manager and/or credentials, currently connecting as:user admin with truststore None`	By default, you connect to an Node Manager using `managedomain` with the credentials specified at installation time. You can override these at startup by passing the `--username --password` command line parameters, or reset while running `managedomain` with this option. This username/password refers to an `admin` user configured in Policy Studio. You can also use this option to select which Admin Node Manager `managedomain` connects to. `managedomain` must talk to an Admin Node Manager, which can be local or remote. By default, `managedomain` connects to the local running Admin Node Manager, otherwise it searches the topology and uses the first running Admin Node Manager that it finds.

API Gateway management

The `managedomain --menu` options for API Gateway management are as follows:

Option	Description	Why use this option
5	`Create API Gateway instance`	Create a new API Gateway instance. You can also do this in Policy Studio and API Gateway Manager. You can create API Gateway instances locally or on any host configured in the topology.
6	`Edit API Gateway (rename, change management port)`	Rename the API Gateway instance, or change the management port. This functionality is not available in Policy Studio and API Gateway Manager.
7	`Delete API Gateway instance`	Delete an API Gateway instance from the topology, and optionally delete the files on disk. You can also do this in Policy Studio and API Gateway Manager. You must ensure that the API Gateway instance has stopped.
8	`Add a tag to API Gateway`	Add a name-value tag to the API Gateway. The **Topology** view on the API Gateway Manager **Dashboard** displays tags and enables you to filter for API Gateway instances by tag.

Option	Description	Why use this option
9	`Delete a tag from API Gateway`	Delete a name-value tag from the API Gateway. The tag will no longer be displayed in the API Gateway Manager **Dashboard**.
10	`Add or remove a Linux service for existing local API Gateway`	Must be run by a user with permission to create a service on the host operating system (`root` on Linux). When run, adds an `init.d` script.

Group management

The `managedomain --menu` options for group management are as follows:

Option	Description	Why use this option
11	`Edit group (rename it)`	Rename an API Gateway group. This functionality is not available in Policy Studio and API Gateway Manager.
12	`Delete a group`	Delete all API Gateways in the group and the group itself. You must ensure that all API Gateways in the group have been stopped first.

Topology management

The `managedomain --menu` options for topology management are as follows:

Option	Description	Why use this option
13	`Print topology`	Output the contents of the deployed domain topology. This includes the following: • Topology version • Hosts • Admin Node Managers • Node Managers • Groups • API Gateway instances (tags)

Option	Description	Why use this option
14	`Check topologies are in sync`	For advanced users. Check that all Node Managers are running the same topology version. Useful only in multi-host environment. Topologies should be in sync if everything is running correctly.
15	`Check the Admin Node Manager topology against another topology`	For advanced users. Compare the two topologies and highlights differences. There should be no differences if everything is running correctly.
16	`Sync all topologies`	For advanced users. Forces a synchronization of all topologies.
17	`Reset the local topology`	For advanced users. Delete the contents of the `apigateway/groups` directory. This means that you must re-register the host and recreate a local API Gateway instance. Alternatively, you can manually delete the contents of this directory to prevent issues if the host has been registered with other node managers.

Deployment

The `managedomain --menu` options for deployment are as follows:

Option	Description	Why use this option
18	`Deploy to a group`	Deploy a configuration (`.fed` file) to API Gateways. This functionality is also available in Policy Studio and API Gateway Manager.
19	`List deployment information`	List the deployment information for all API Gateways in a topology. This functionality is also available in Policy Studio and API Gateway Manager.
20	`Create deployment archive`	Create a deployment archive from a directory that contains a federated API Gateway configuration.

Option	Description	Why use this option
21	`Download deployment archive`	Download the `.fed` file deployed to an API Gateway. This functionality is also available in Policy Studio.
22	`Update deployment archive properties`	Update the manifest properties relating to the deployed configuration only. This functionality is also available in Policy Studio. Enables you to update the properties without performing a new deployment.
23	`Change group configuration passphrase`	The default passphrase for the API Gateway configuration is "". Use this option to set a more secure password. This functionality is also available in Policy Studio.

Domain SSL certificates

The `managedomain --menu` options for group management are as follows:

Option	Description	Why use this option
24	`Regenerate SSL certificates on localhost`	Regenerate the SSL certificates used to secure API Gateway components in the domain (for example, Node Manager and the API Gateway instances that it manages). You must restart the Node Manager on the localhost after running this option. You must run this option on all hosts in the domain.
25	`Sign CSR`	Specify a Certificate Signing Request (CSR) to send to the Certificate Authority (CA) when applying for an SSL certificate for a Node Manager or API Gateway instance. You can use this option when `managedomain` acts as the CA, and is passed the CSR to create a signed certificate. You will most likely use an external CA in production. However, this option facilitates testing of certificates signed by an external CA. You can install the API Gateway on a locked-down host, and use this feature only (no license required). You would typically only do this when using a system-generated self-signed domain certificate, and do not wish to store the domain private key on an Admin Node Manager host, and do not wish to use an external CA.

Option	Description	Why use this option
26	Submit externally signed certificate	Specify an SSL certificate signed by an externally signed Certificate Authority (CA) to be used by a Node Manager or API Gateway instance. Use this option after registering a host or creating an API Gateway using a certificate signed by an external CA. Submitting the certificate with this option completes the host registration or API Gateway creation.

Managedomain command mode

You can also enter `managedomain` commands and parameters directly on the command line. For example, the following command creates an Admin Node Manager on the first host in the domain and signs with a user-provided domain key:

```
managedomain -i --sign_with_user_provided --ca=/home/keys/test.p12
```

Note You must run `managedomain -i` or `--initialize` to register the first host in the domain in order to create and run API Gateways.

For details on all available commands, enter

```
managedomain --help
```

For detailed examples of using `managedomain` in command mode, see the following:

- *Configure Admin Node Manager high availability* on page 50
- *Find your installed version and list patches using managedomain* on page 193

Provide credentials to managedomain

You can use the following properties file to automatically provide admin user name and password credentials to authenticate to the Admin Node Manager:

```
INSTALL_DIR/apigateway/conf/managedomain.props
```

Perform the following steps:

1. Open the `managedomain.props` file in an editor.
2. Uncomment the `password_exec` property.
3. Ensure that the path to `../apigateway/conf/execute.sh` is correct.
4. Change the password echoed in `../apigateway/conf/execute.sh`.
5. Save your changes to the file.

Note You must ensure that the appropriate read and execute privileges for your operating system have been set for the `execute` file. You must also ensure that the `execute` and `managedomain.props` files are protected.

Alternatively, you can provide credentials on the command line. The following example shows command mode:

```
managedomain.bat --print_toplogy --username <userName> --password <password>
```

The following example shows interactive mode:

```
>managedomain --menu
Enter username: my_admin_user
Enter password: **********
```

Manage API Gateway operation 3

This part contains the following:

Start and stop the API Gateway

This topic describes how to start and stop the Node Manager and API Gateway instance on the command line, on all platforms. It also describes how to start the Policy Studio graphical tool. For details on API Gateway components and concepts, see the *API Gateway Concepts Guide*.

You can also start and stop API Gateway instances using the API Gateway Manager web console. For more details, see *Manage API Gateway instances* on page 48.

Prerequisites

Before you can start API Gateway, you must first create a new domain that includes an API Gateway instance. If you installed the QuickStart tutorial, a sample API Gateway domain is automatically configured in your installation. Otherwise, you must create a new domain. For more details, see *Configure an API Gateway domain* on page 40.

If you are using Apache Cassandra, before starting API Gateway, you must first ensure that Cassandra is running. For details on installing and running Cassandra, see the *API Gateway Installation Guide*.

Set passphrases

By default, data is stored unencrypted in the API Gateway configuration store. However, you can encrypt certain sensitive information such as passwords and private keys using a passphrase. When the passphrase has been set, this encrypts the API Gateway configuration data.

You must enter the passphrase when connecting to the API Gateway configuration data (for example, using the Policy Studio, or when the API Gateway starts up). For more details on configuring this passphrase, see *Configure an API Gateway encryption passphrase* on page 102.

Start the Node Manager

To start the Node Manager on Linux, complete the following steps:

1. Open a shell at the `/posix/bin` directory of your API Gateway installation.
2. Run the `nodemanager.sh` file, for example:

```
prompt# ./nodemanager
```

3. If you are using an encryption passphrase, you are prompted for this passphrase. Enter the correct encryption passphrase and press Return. For more details, see *Configure an API Gateway encryption passphrase* on page 102.

Start the API Gateway instance

```
startinstance -n "my_server" -g "my_group"
```

To start the API Gateway instance and Policy Studio on Linux, perform the following steps:

1. Open a shell at the `/posix/bin` directory of your API Gateway installation.
2. Use the `startinstance` command to start API Gateway, for example:

```
startinstance -n "my_server" -g "my_group"
```

Note You must ensure that the `startinstance` file has execute permissions.

3. If you are using an encryption passphrase, you are prompted for this passphrase. Enter the correct encryption passphrase and press Return. For more details, see *Set passphrases* on page 77.
4. When API Gateway has successfully started up, run the`policystudio.sh` file in your Policy Studio installation directory. For example:

```
prompt# cd /usr/home/policystudioprompt# ./policystudio"
```

5. When Policy Studio is starting up, you are prompted for connection details for API Gateway.

Tip You can enter the `startinstance` command without any arguments to display the servers registered on the machine. For example:

```
INSTALL_DIR/apigateway/posix/bin>startinstanceusage:"startinstance
```

```
[[-n instance-name -g group-name [instance-args]] | [directory-location [instance-
args]]]"

The API Gateway instances listed below are available to run on this machine as
follows:

startinstance -n "server1" -g "group1"

startinstance -n "server2" -g "group2"
```

If you have a single API Gateway instance on the host on which you run `startinstance`, that instance starts when you specify no arguments.

Startup options

You can specify the following optional instance arguments to the `startinstance` command:

Option	Description
`-b <file>`	Specifies the boot-time trace destination.
`-d`	Runs as daemon/service.
`-h <directory>`	Specifies the service instance directory.
`-k`	Kills the instance .
`-P`	Prompts for a passphrase at startup.
`-q`	Runs in quiet mode (equivalent to `-Dquiet`).
`-v`	Changes to the installation instance directory on startup.
`-s`	Tests if the service is running .
`-e`	Specifies the end of arguments for `vshell`.
`-D prop[=value]`	Sets a configuration file property.

The `-d`, `-s` and `-k` options are designed for use with the service controller (for example, traditional SVR4 init, systemd, upstart, and so on). The `-d` option waits until the service is running before returning, and `-k` waits for the process to terminate. This means that when used in a script, the completion of the command indicates that the operation requested has completed.

If the service is running, `-s` will exit with a `0` status code, and with `1` otherwise. For example, you can use the following to print a message if the service is running:

```
startinstance -s -n InstanceName -g GroupName && echo Running
```

Connect to API Gateway in Policy Studio

When starting the Policy Studio, you are prompted for details on how to connect to the Admin Node Manager (for example, the server session, host, port, user name, and password). The default connection URL is:

```
https://HOST:8090/api
```

`HOST` is the IP address or host name of the machine on which the API Gateway runs. For more details, see the *API Gateway Policy Developer Guide*.

Stop API Gateway

To stop the API Gateway instance, you must specify the group and instance name to the `startinstance` command along with the `-k` option. For example:

```
./startinstance -g "my_group" -n "my_server" -k
```

Stop the Node Manager

To stop the Node Manager, you must specify the `nodemanager` command along with the `-k` option. For example:

```
./nodemanager -k
```

Perform zero downtime shutdown

This topic describes how to perform a zero downtime shutdown of API Gateway in a multi-node API Gateway environment with a load balancer.

When you need to shut down an API Gateway for any reason (for example, during an upgrade), zero downtime shutdown enables you to indicate this to the load balancer for a set amount of time before the shutdown begins, avoiding traffic loss.

Shut down an API Gateway using zero downtime shutdown

To perform a zero downtime shutdown, follow these steps:

1. Enable zero downtime shutdown in Policy Studio, and set the delay before shutdown. For more information, see *Zero downtime settings* on page 256.
2. Configure your load balancer to ping the Health Check LB policy periodically to determine if each API Gateway is healthy. This is available on the following default URL:

```
http://APIGATEWAY_HOST:8080/healthchecklb
```

3. Initiate shutdown of an API Gateway using the command line or API Gateway Manager. For more information, see *Start and stop the API Gateway* on page 77 or *Manage API Gateway instances* on page 48.
4. When shutdown is initiated on the API Gateway:
   - The Health Check LB policy returns a `503 Service Unavailable` response. This indicates to the load balancer that the API Gateway is not available for traffic and the load balancer stops routing to it.
   - After the specified delay before shutdown (for example, 10 seconds), the API Gateway is shut down.

Start the API Gateway tools

This topic describes the prerequisites and preliminary steps. It explains how to start the API Gateway Manager administrator tool and the Policy Studio developer tool.

Before you begin

Before you start the API Gateway tools, do the following:

Install the API Gateway and Policy Studio

If you have not already done so, see the *API Gateway Installation Guide*.

Configure a managed domain

If you have not already created a domain, use the `managedomain` script to configure a domain. You should ensure that the Admin Node Manager and an API Gateway instance are running.

Launch API Gateway Manager

To access the web-based API Gateway Manager administration tools, perform the following steps:

Note You must ensure that the Admin Node Manager is running before you can access the web-based API Gateway Manager tools.

1. Enter the following URL:

   ```
   https://HOST:8090/
   ```

   `HOST` refers to the host name or IP address of the machine on which API Gateway is running (for example, `https://localhost:8090/`).
2. Enter the administrator user name and password configured at installation time.
3. Click the appropriate button in the API Gateway Manager page in the browser. The **Dashboard** view is displayed by default.

The API Gateway Manager includes the following main views:

- **Dashboard**: System health, traffic summary, and topology (domain, hosts, API Gateways, and groups).
- **Traffic**: Message log and performance statistics on the traffic processed by API Gateway. For example, all HTTP, HTTPS, JMS, File Transfer, and Directory messages processed by API Gateway.
- **Monitoring**: Real-time monitoring of all the traffic processed by API Gateway. Includes statistics at the system level and for services invoked and remote hosts connected to.
- **Logs**: API Gateway trace log, transaction log, and access log files.
- **Events**: API Gateway transaction log points, alerts, and SLA alerts.
- **Settings**: Enables you to configure dynamic API Gateway logging, user roles, and credentials.

Start Policy Studio

To start the Policy Studio tool used to create and manage policies, perform the following steps:

1. In your Policy Studio installation directory, enter the `policystudio` command.
2. In Policy Studio, select **File > New Project**, and follow the steps in the wizard. For more details, see the *API Gateway Policy Developer Guide*.

   Alternatively, if a project has already been created, select **File > Open Project**, or click a link to the existing project on the Policy Studio landing page. For more details, see the *API Gateway Policy Developer Guide*.

Policy Studio enables you to perform the full range of API Gateway configuration and management tasks. This includes tasks such as develop and assign policies, import services, optimize API Gateway configuration settings, and manage API Gateway deployments.

For more details on using the Policy Studio to manage API Gateway processes and configurations, see *Manage API Gateway deployments* on page 139.

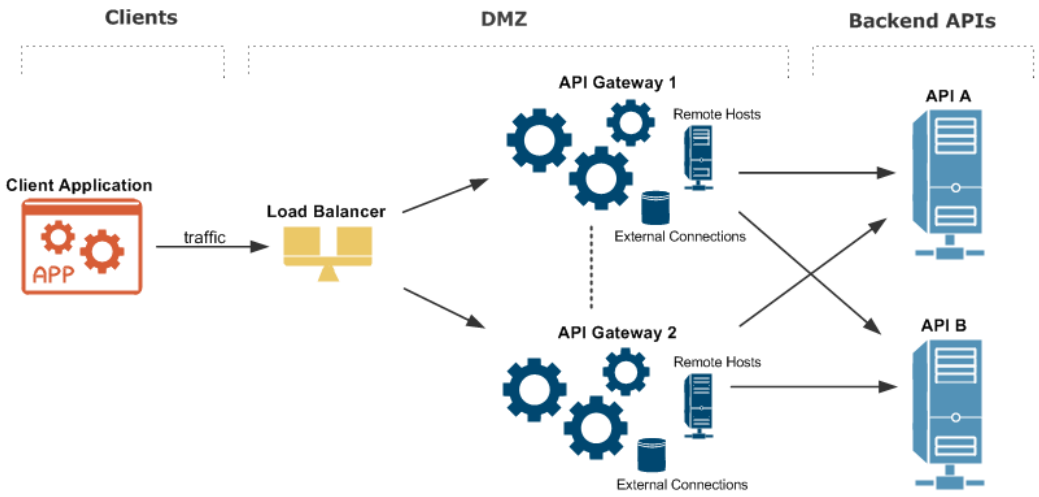
Configure API Gateway high availability

System administrators can configure High Availability (HA) in an API Gateway environment to ensure that there is no single of point of failure in the system. This helps to eliminate any potential system downtime in production environments. Typically, the API Gateway platform is deployed in the Demilitarized Zone (DMZ) to provide an additional layer of security for your back-end systems.

This topic describes the recommended API Gateway architecture in an HA production environment. It includes recommendations on topics such as load balancing, commonly used transport protocols, caching, persistence, and connections to external systems.

HA in production environments

The following diagram shows an overview of an API Gateway platform running in an HA production environment:



The architecture shown in the diagram is described as follows:

- An external client application makes inbound calls sending business traffic over a specific message transport protocol (for example, HTTP, JMS, or FTP) to a load balancer.
- A standard third-party load balancer performs a health check on each API Gateway instance, and distributes the message load to the listening port on each API Gateway instance (default is `8080`).
- Each API Gateway instance has External Connections to third-party systems. For example, these include databases such as Oracle and MySQL, and Authentication Repositories such as CA SiteMinder, Oracle Access Manager, Lightweight Directory Access Protocol (LDAP) servers, and so on.
- Caching is replicated between each API Gateway instance using a distributed caching system based on Ehcache.

- Each API Gateway instance has Remote Host interfaces that specify outbound connections to back-end API systems, and which can balance the message load based on specified priorities for Remote Hosts.
- Each API Gateway instance connects to an external Apache Cassandra database used by certain features for persistent data storage, and which has its own HA capabilities.
- Each API Gateway instance contains an embedded Apache ActiveMQ messaging system, which can be configured for HA in a shared file system.
- Each back-end API is also replicated to ensure there is no single point of failure at the server level.
- Management traffic used by the Admin Node Manager, API Gateway Manager, and Policy Studio is handled separately on different port (default is `8090`).

Note For simplicity, the diagram shows two API Gateway instances only. However, for a resilient HA system, a minimum of at least two active API Gateway instances at any time, with a third and fourth in passive mode, is recommended.

Load Balancing

In this HA production environment, the load balancer performs a health check and distributes message load between API Gateway instances. The API Gateway supports a wide range of standard third-party load balancing products (for example, F5) without any special requirements. Multiple API Gateway instances can be deployed active/active (hot standby) or active/passive (cold standby) modes as required.

The load balancer polls each API Gateway instance at regular intervals to perform a health check on the message traffic port (default `8080`). The load balancer calls the API Gateway Health Check policy, available on the following default URL:

```
http://GATEWAY_HOST:8080/healthcheck
```

The health check returns a simple `<status>ok</status>` message when successful. In this way, if one API Gateway instance becomes unavailable, the load balancer can route traffic to an alternative API Gateway instance.

Both transparent load balancing and non-transparent load balancing are supported. For example, in transparent load balancing, API Gateway can see that incoming messages are sent from specific client and load balancer IP addresses. API Gateway can also extract specific client details from the HTTP header as required (for example, the SSL certificate, user credentials, or IP address for Mutual or 2-Way SSL Authentication). In non-transparent load balancing, API Gateway sees only the virtual service address of the load balancer.

In addition, API Gateway can also act as load balancer on the outbound connection to the back-end APIs. For more details, see *Remote Hosts* on page 85.

Java Message System

API Gateway supports integration with a wide range of third-party Java Message System (JMS) products. For example, these include the following:

- IBM WebSphere MQ
- Progress SonicMQ
- Tibco Rendezvous
- Fiorano
- OpenJMS
- JBoss Messaging

API Gateway can act as a JMS client (for example, polling messages from third-party JMS products or sending message to them). For details on configuring API Gateway client connections to JMS systems, see the *API Gateway Policy Developer Guide*. For details on configuring HA in supported third-party JMS systems, see the user documentation available from your JMS provider.

API Gateway also provides an embedded Apache ActiveMQ server in each API Gateway instance. For more details, see *Embedded Apache ActiveMQ* on page 87.

File Transfer Protocol

API Gateway supports the following protocols:

- File Transfer Protocol (FTP)
- FTP over SSL (FTPS)
- Secure Shell FTP (SFTP)

When using FTP protocols, API Gateway writes to a specified directory in your filesystem. In HA environments, when the uploaded data is required for subsequent processing, you should ensure that the filesystem is shared across your API Gateway instances—for example, using Storage Area Network (SAN) or Network File System (NFSv4).

For more details on configuring FTP connections, see the *API Gateway Policy Developer Guide*.

Remote Hosts

You can use the **Remote Host Settings** in Policy Studio to configure how API Gateway connects to a specific external server or routing destination. For example, typical use cases for configuring Remote Hosts with API Gateway are as follows:

- Mapping a host name to a specific IP address or addresses (for example, if a DNS server is unreliable or unavailable), and ranking hosts in order of priority.
- Specify load balancing settings (for example, whether load balancing is performed on a simple round-robin basis or weighted by response time).

- Allowing the API Gateway to send HTTP 1.1 requests to a destination server when that server supports HTTP 1.1, or resolving inconsistencies in how the destination server supports HTTP.
- Setting timeouts, session cache size, input/output buffer size, and other connection-specific settings for a destination server. For example, if the destination server is particularly slow, you can set a longer timeout.

For details on how to configure **Remote Host Settings** in Policy Studio, see the *API Gateway Policy Developer Guide*.

Distributed caching

In an HA production environment, caching is replicated between each API Gateway instance using a distributed caching system. In this scenario, each API Gateway has its own local copy of the cache, but registers a cache event listener that replicates messages to the caches on other API Gateway instances. This enables the put, remove, expiry, and delete events on a single cache to be replicated across all other caches.

In the distributed cache, there is no master cache controlling all caches in the group. Instead, each cache is a peer in the group that needs to know where all the other peers in the group are located. Peer discovery and peer listeners are two essential parts of any distributed cache system.

For more details on configuring distributed cache settings, see the topic on Global Caches in the *API Gateway Policy Developer Guide*. API Gateway distributed caching system is based on Ehcache. For more details, see http://ehcache.org/.

External Connections

You can use **External Connections** settings in Policy Studio to configure how API Gateway connects to specific external third-party systems. For example, this includes connections such as the following:

- Authentication Repositories (LDAP, CA SiteMinder, Oracle Access Manager, and so on)
- Databases (Oracle, DB2, MySQL, and MS SQL Server)
- JMS Services
- SMTP Servers
- ICAP Servers
- Kerberos
- Tibco
- Tivoli
- Radius Clients

For details on how to configure **External Connections** in Policy Studio, see the *API Gateway Policy Developer Guide*. For details on how to configure HA for any external third-party systems, see the product documentation provided by your third-party vendor.

Note When configuring connections to server hosts, it is recommended that you specify server host names instead of IP addresses, which are less stable and are subject to change in a Domain Name System (DNS) environment.

External Apache Cassandra database

You can configure an API Gateway instance to use an external Apache Cassandra database for persistent data storage. This Cassandra database is required by the following API Gateway features:

- API Manager
- OAuth tokens and codes
- Client Application Registry
- API Keys
- KPS collections that you create

If your API Gateway system uses any of these features, you must configure an external Apache Cassandra database for HA. All API Gateways in a group can share the same external Cassandra data source. In a production environment, you must configure the API Gateway group to use the same Cassandra data source to provide HA.

Note All nodes in a Cassandra cluster must run on the same operating system.

For more details on installing and configuring an external Cassandra database for HA, see the *API Gateway Installation Guide*.

Embedded Apache ActiveMQ

API Gateway provides an embedded Apache ActiveMQ broker in each API Gateway instance. In a HA production environment, multiple ActiveMQ broker can work together as a network of brokers in a group of API Gateways. This requires setting up a shared directory that is accessible from all API Gateway instances—for example, using Storage Area Network (SAN) or Network File System (NFSv4).

When setting up a shared directory, do not use OCFS2, NFSv3, or other software where exclusive file locks do not work reliably. For details, see Apache ActiveMQ Shared File System Master Slave.

In this shared network of ActiveMQ brokers, each API Gateway can start a local embedded ActiveMQ broker, which listens on a configured TCP port (`61616` by default). This port is accessible from both the local API Gateway and remote clients using the load balancer.

For details on how to configure and manage embedded Apache ActiveMQ, see the following:

- *Embedded ActiveMQ settings* on page 276
- *Manage embedded ActiveMQ messaging* on page 239

For more details on Apache ActiveMQ, see http://activemq.apache.org/.

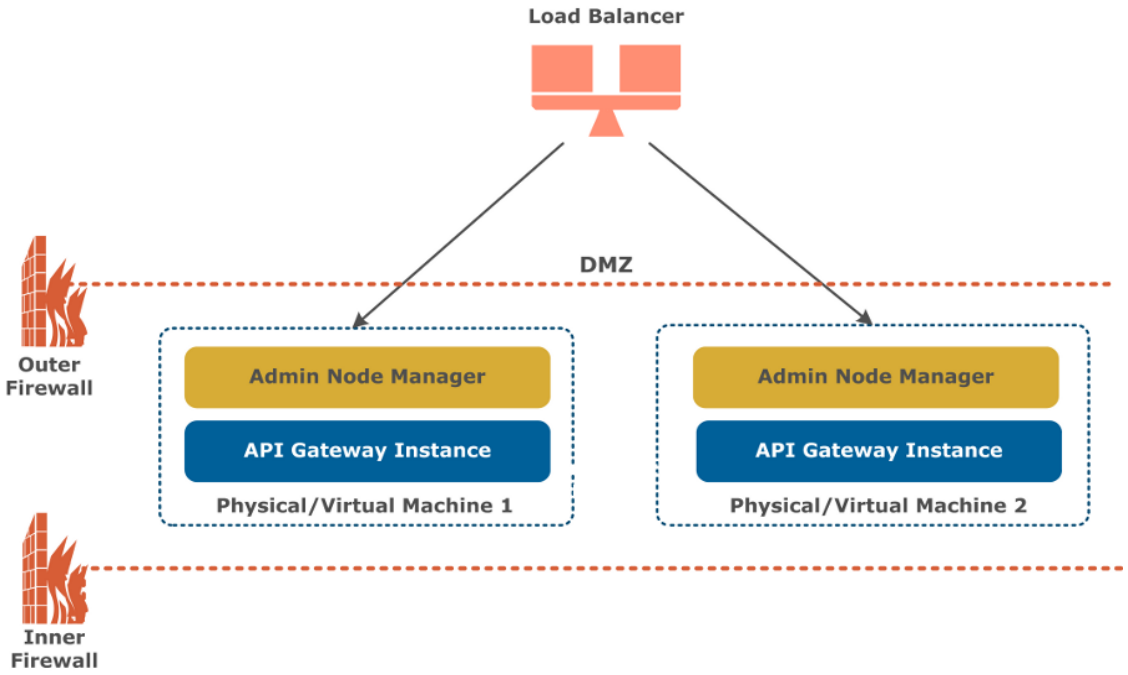
Admin Node Manager high availability

API Gateway provides an active/active high availability solution for the Admin Node Manager that supports multiple DMZ deployment patterns. Multiple Admin Node Managers in a domain are supported in an active/active configuration, in which each Admin Node Manager can perform management operations with the same shared topology and configuration. This supports the following DMZ deployment patterns:

- All nodes in the DMZ run Admin Node Managers
- Only nodes behind the internal firewall run Admin Node Managers
- DMZ has multiple zones, and only nodes behind the firewall run Admin Node Managers

Scenario 1–Admin Node Managers in DMZ

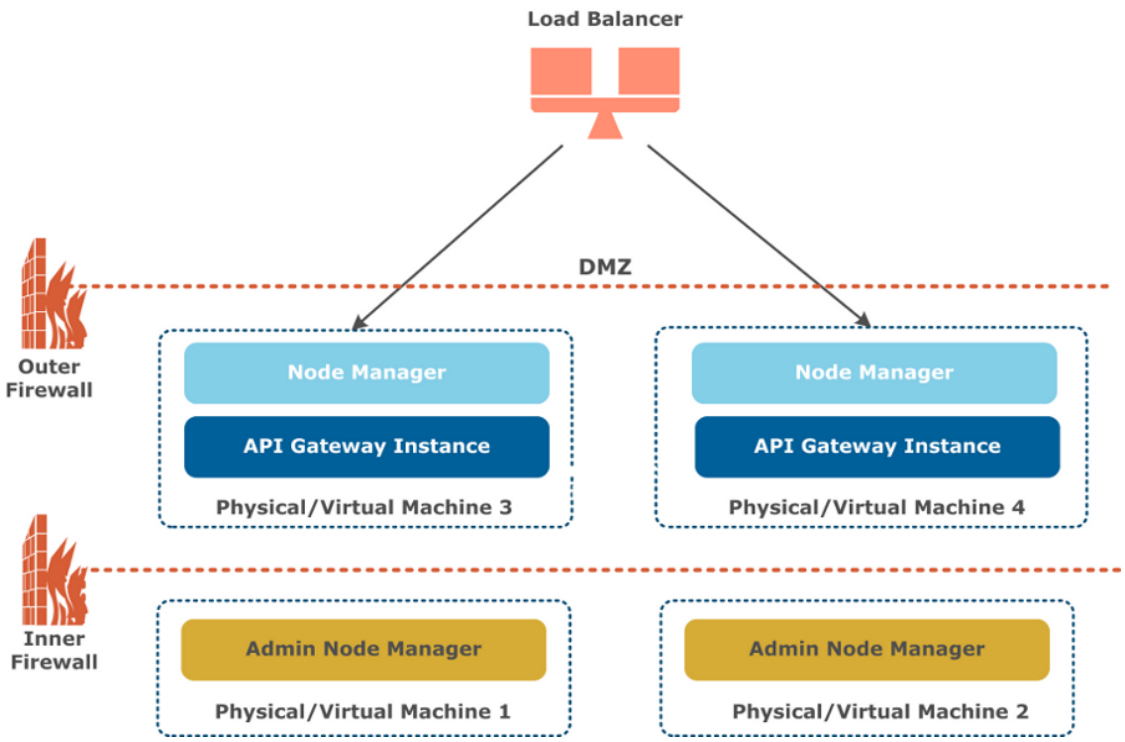
In this deployment pattern, all nodes in the DMZ run an Admin Node Manager in active/active mode. For example:



Multiple API Gateway instances are deployed on separate nodes in the DMZ, and all nodes can communicate with each other. Each node runs an Admin Node Manager in an active/active configuration, and there are no Node Managers. This means that the API Gateway management infrastructure is deployed in the DMZ.

Scenario 2–Admin Node Managers behind firewall

In this deployment pattern, only nodes behind the internal firewall run Admin Node Managers in active/active mode. For example:

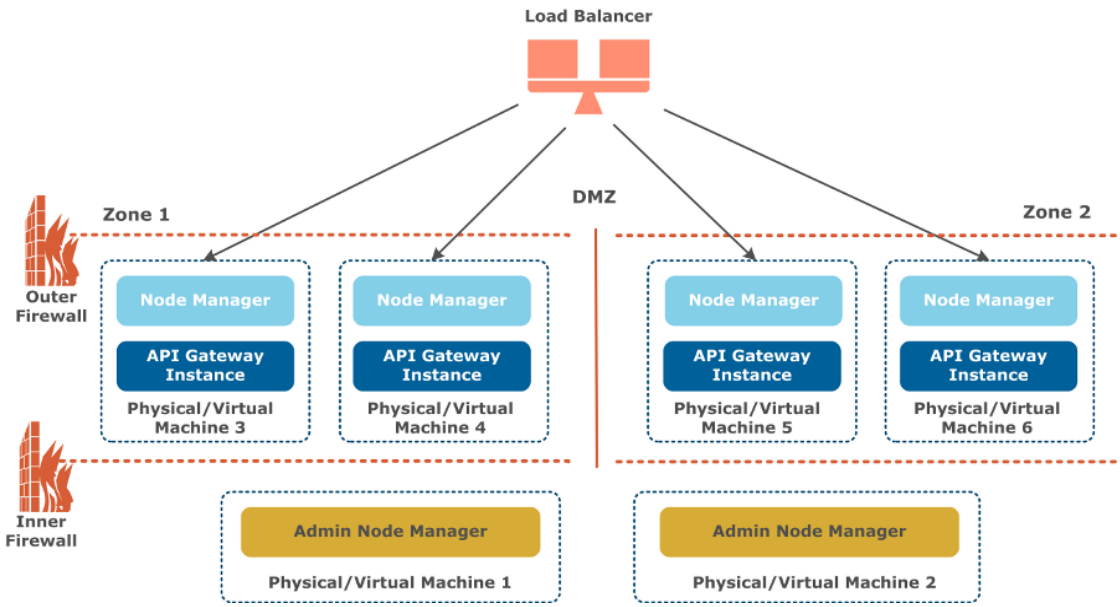


Similar to scenario 1, multiple API Gateway instances are deployed on separate nodes in the DMZ, and all nodes can communicate with each other. However, the management infrastructure cannot be deployed in the DMZ, and must be deployed behind the internal firewall.

All nodes in the DMZ run Node Managers instead. There are two nodes deployed behind the internal firewall running Admin Node Managers in an active/active configuration. Both Admin Node Managers can manage any Node Manager in the DMZ.

Scenario 3–Multi-zoned DMZ with Admin Node Managers behind firewall

In this deployment pattern, all nodes in the DMZ run an Admin Node Manager in active/active mode. For example:



This deployment pattern is a refinement of scenario 2. The DMZ is divided into multiple zones with no inter-zone communication. Multiple API Gateway instances are deployed on separate nodes in each zone, and all nodes in the zone can communicate with each other.

All nodes in the DMZ run Node Managers. There are two nodes deployed behind the internal firewall running Admin Node Managers in an active/active configuration. Both Admin Node Managers can manage any Node Manager in any zone.

Further details

You must configure at least two Admin Node Managers in a domain for high availability and security. For details on how to configure and secure multiple Admin Node Managers in a domain, see *Configure Admin Node Manager high availability* on page 50.

API Gateway backup and disaster recovery

You must ensure that your API Gateway system can recover from any natural disasters (for example, floods, hurricanes, or earthquakes) and human-induced disasters (for example, failures, fires, or explosions).

Many organizations have a mirrored backup and disaster recovery site with full capacity to recover from any major incidents. These systems are typically kept in a separate physical location on cold stand-by until they need to be brought into action. In this case, the backup and disaster site must be a mirrored image of production environment that replicates all resources and assets (for example, LDAP and databases), and with the same number of API Gateway instances. You should ensure that any required third-party systems are included in your backup and recovery solution.

Example approaches to keeping production and backup environments in sync include making backups to disk or tape, and sending these off-site at regular intervals, or cloud-based solutions that replicate on-site systems, and back up to off-site datacenters.

Tip For details on backing up and restoring an Admin Node Manager in a highly available environment, see *Configure Admin Node Manager high availability* on page 50.

Components that must be backed up

Whichever backup strategy you choose, in a production environment you must ensure that API Gateway installations on all API Gateway host nodes are backed up on a regular basis. For example, this includes hosts that run the following components:

- API Gateway instance
- Admin Node Manager
- Node Manager
- API Manager
- API Gateway Analytics

You must also back up all databases and third-party systems used with the API Gateway. For example, this includes the following:

- Databases used by API Gateway and API Manager to store metrics (for example, MySQL, Oracle, MS SQL, or DB2)
- Apache Cassandra database used by API Gateway and API Manager for internal data storage.
- Shared disks used by embedded ActiveMQ to store JMS messages
- Any databases or third-party systems that the API Gateway connects to in External Connections

Note You do not need to back up Policy Studio, Configuration Studio, or API Tester because these tools run in a temporary workspace when required.

However, if you have modified any third-party dependencies on the **Preferences** page (for example, to connect to a specific database), you must also add the relevant `.jar` on the **Runtime Dependencies** page in your disaster recovery environment.

Back up API Gateway

To back up an API Gateway installation, you must back up files that have changed in the following directory:

```
<install-dir>/apigateway
```

This backs up all Admin Node Manager, Node Manager, API Manager , and API Gateway instances in that installation.

For example, the following directories include API Gateway configuration, and will typically need to be backed up on a regular basis:

```
<install-dir>/apigateway/conf
```

```
<install-dir>/apigateway/groups
<install-dir>/apigateway/ext
<install-dir>/apigateway/system/conf/nodemanager.xml
```

Note Before backing up, you can remove the contents of the `apigateway/conf/opsdb.d` directory. This contains transient monitoring data, which can be quite large in some cases, and does not need to be backed up.

Back up API Gateway Analytics

Similarly, to back up an API Gateway Analytics installation, you must back up files that have changed in the following directory:

```
<install-dir>\analytics
```

For example, the following directories include API Gateway Analytics configuration, and will typically need to be backed up on a regular basis:

```
<install-dir>\analytics\conf
<install-dir>\analytics\ext
```

Note This backs up the API Gateway Analytics installation only. You must also back up the metrics database separately. For more details, see the next section.

Back up databases and third-party systems

You must back up all databases and third-party systems used with API Gateway. For example, you can back a MySQL database by creating a dump file of the tables in use:

```
mysqldump -u root temp_backup > db_tables.dump
```

For more details, see the user documentation for your third-party system.

Disaster recovery plan and tests

To ensure that your backup and disaster recovery processes are successful, you should conduct full restoration tests on a regular basis. You must ensure that you can restore all required files as quickly and easily as possible.

To ensure this, your backup and disaster recovery plan should include key metrics for recovery points and recovery times for your real-world business processes (for example, creating a purchase order, booking a reservation, and so on).

Example of creating an API Gateway disaster recovery site

This simple example shows how to create a disaster recovery site from a backup of an API Gateway production deployment. It assumes that both the disaster recovery site and the primary production site have the same version of API Gateway installed. In this scenario, the disaster recovery site is a cold standby, and the configuration from production is replicated using a backup of production configuration.

Back up the production environment

To back up the production environment, perform the following steps.

1. Browse to the directory where the API Gateway is installed (for example, `/opt/apigateway`).
2. Tar or zip the following:
   - `apigateway/groups`
   - `apigateway/conf`
   - `apigateway/system/conf/nodemanager.xml`
   - `apigateway/ext`

   If you want the API Gateway and Node Manager to start up automatically on the new host, you should also include `/etc/init.d/vshell-*`.

Note This includes separate startup scripts files for the Node Manager and API Gateway instances if an `init.d` script was created using `managedomain` during initial setup.

You can create these at any time using `managedomain`, and choosing option `2`, Edit a host, for a Node Manager, or option `10`, Add script or service for an existing local API Gateway. For moredetails, see *Managedomain command reference* on page 65.

For example, the following command creates a `.tar` file running from the root directory:

```
>tar -cvf apigateway_backup.tar /opt/apigateway/conf
/opt/apigateway/groups
 /opt/apigateway/system/conf/nodemanager.xml
```

The following example creates a `.tar`file containing the startup scripts running from the root directory:

```
>tar -cvf startup_scripts.tar /etc/init.d/vshell-*
```

Copy to the disaster recovery site

To replicate to the disaster recovery site:

1. Copy the created `.tar` file(s) from the primary production machine to the cold standby machine.

   Note Ensure the files are tarred before copying because this preserves the permissions and ownership of the files.

2. Extract the files so that they are extracted in the same directories, overwriting existing files if necessary. The following example extracts the files from the root directory:

```
>tar -C / -xvf apigateway_backup.tar
```

3. The following is optional:

```
>tar -C / -xvf startup_scripts.tar
```

4. When all the files are copied over and extracted successfully, you should be able to start the Admin Node Manager and API Gateway instances the same way as in primary production site running the same topology and configurations.

Note This example assumes that backups are collected on regular basis from the master node in the production site.

Further information

For details on how to back up and restore an Admin Node Manager for signing SSL certificates in an API Gateway domain, see *Configure Admin Node Manager high availability* on page 50.

For details on how to back and restore internal data stored in Apache Cassandra (for example, API Gateway KPS data or API Manager data), see the *API Gateway Apache Cassandra Administrator Guide*.

Manage API Gateway settings

You can configure the underlying settings for API Gateway using the **Server Settings** node in the Policy Studio tree. Alternatively, select the **Tasks > Manage Gateway Settings** menu option in the Policy Studio main menu.

This topic describes the settings available in the **Server Settings** window. Click or expand a tree node on this window to configure the appropriate settings. You can confirm changes to these settings by clicking the **Save** button at the bottom right of each window.

API Manager settings

The **API Manager** settings enable you to configure the API management features that are available when Axway API Manager product is installed with API Gateway. For example, this includes settings for API Manager alerts, directory service, metrics, policies, quotas, and SMTP server.

For more details, see the *API Manager User Guide*.

General settings

The top-level **General** settings are applied to all instances of the API Gateway that use this configuration. For example, you can change the tracing level, various timeouts and cache sizes, and other such global information. For more details, see *General settings* on page 249.

In addition, you can also configure the following settings under the **General** node:

Cache

If you have deployed several API Gateways throughout your network, you should configure a distributed cache. In a distributed cache, each cache is a peer in a group and needs to know where all the other peers in the group are located. The **Cache** settings enable you to configure settings for peer listeners and peer discovery. For more details, see the *API Gateway Policy Developer Guide*.

MIME

API Gateway can filter Multipurpose Internet Mail Extensions (MIME) messages based on the content types (or MIME types) of the individual parts of the message.

The MIME settings list the default MIME types that API Gateway can filter on. These types are then used by the **Content Types** filter to determine which MIME types to block or allow through to the back end Web service. For more details, see *MIME settings* on page 253.

Namespaces

The **Namespaces** settings are used to determine the versions of SOAP, Web Services Security (WSSE) and Web Services Utility (WSU) that API Gateway supports. For more details, see *Namespace settings* on page 254.

HTTP session

The **HTTP Session** settings enable you to configure HTTP session management settings for the selected cache. For example, you can configure the period of time before expired sessions are cleared from the default `HTTP Sessions` cache. For more details, see *HTTP Session settings* on page 256.

Zero downtime

The Zero Downtime settings enable you to configure zero downtime deployment and shutdown settings. For more details, see *Zero downtime settings* on page 256.

Logging settings

The **Logging** settings enable you to configure the following:

Transaction Audit Log

The **Transaction Audit Log** settings enable you to configure the default message transaction logging behavior of API Gateway. For example, you can configure API Gateway to log to a database, text or XML file, local or remote Linux syslog, or the system console. For more details, see the topic on *Transaction audit log settings* on page 258.

Transaction Access Log

The **Transaction Access Log** records a summary of all request and response messages that pass through API Gateway. For example, this includes details such as the remote hostname, username, date and time, first line of the request message, HTTP status code, and number of bytes. For details on configuring these settings per API Gateway, see *Transaction access log settings* on page 262.

Transaction Event Log

The **Transaction Event Log** provides a summary of each API Gateway message transaction, which is written to a log file, and used to generate metrics for API Gateway monitoring. For example, this information can be displayed in API Gateway Analytics, in the **Monitoring** view in API Manager, or in third-party monitoring tools. For details on configuring these settings per API Gateway, see *Transaction event log settings* on page 266.

Messaging settings

The **Messaging** settings enable you to configure settings for the embedded Apache ActiveMQ server that is available in each API Gateway instance. For example, these include the listening interface, port, shared directory, and so on. For more details, see *Embedded ActiveMQ settings* on page 276.

Monitoring settings

The **Monitoring** settings enable you to configure the following:

Real Time Monitoring

The **Real Time Monitoring** settings enable you to configure statistics about messages that API Gateway instances store in a local operations database. The API Gateway Manager monitoring tool can then poll this local database, and produce charts and graphs showing how API Gateway is performing. For more details, see *Real-time monitoring metrics* on page 281.

Traffic Monitor

The **Traffic Monitor** settings enable you to configure the web-based Traffic Monitor tool and its message traffic log. For example, you can configure where the data is stored and what message transaction details are recorded in the log. For more details, see *Traffic monitoring settings* on page 279.

Security settings

The **Security** settings enable you to configure the following:

Access Manager

The **Access Manager** settings enable you to configure how the Sun Access Manager Policy Agents that are embedded in API Gateway's Sun Access Manager filters connect to Access Manager. You can also specify how and where these agents trace and log runtime information. For more details, see the *API Gateway Policy Developer Guide*.

Security Service Module

You can configure API Gateway to act as an Oracle Security Service Module (SSM) to enable integration with Oracle Entitlements Server 10g. API Gateway acts as a Java SSM, which delegates to Oracle Entitlements Server 10g. For example, you can authenticate and authorize a user for a particular resource against an Oracle Entitlements Server 10g repository. For more details, see the *API Gateway Policy Developer Guide*.

Note Oracle SSM is required for integration with Oracle OES 10g only. Oracle SSM is not required for integration with Oracle OES 11g.

Kerberos

You can configure Kerberos settings such as the Kerberos configuration file to API Gateway, which contains information about the location of the Kerberos Key Distribution Center (KDC), encryption algorithms and keys, and domain realms.

You can also configure options for APIs used by the Kerberos system, such as the Generic Security Services (GSS) and Simple and Protected GSSAPI Negotiation (SPNEGO) APIs. For more details, see the *API Gateway Policy Developer Guide*.

Tivoli

You can configure how API Gateway instance connects to an instance of an IBM Tivoli Access Manager server. Each API Gateway instance can connect to a single Tivoli server. For more details, see the *API Gateway Policy Developer Guide*.

Manage API Gateway security 4

This part contains the following:

Run API Gateway on privileged ports

API Gateway is run as a non-root user to prevent any potential security issues with running as the `root` user. This topic describes the steps you must perform to grant the required privileges to API Gateway processes running as non-root.

Before you begin

The examples in this topic are for a non-root user named `admin` and for an API Gateway installation at `/opt/Axway-7.6.2/apigateway`. If you have a different non-root user name or API Gateway installation, you must modify the examples accordingly.

Step 1 - Set API Gateway file ownership to non-root user

The ownership of the API Gateway files must be set to the non-root user. You can change the user and group ownership of all files under the API Gateway installation directory, for example:

```
# chown -R admin:admin /opt/Axway-7.6.2/apigateway
```

SSL accelerators for HSM

When using a Hardware Security Module (HSM), the non-root user must have access to the device corresponding to the cryptographic accelerator or HSM card. For HSMs such as Cavium and Ultimaco, this means that you must have access to the following directories:

- Cavium: `/dev/pkp_nle_drv`
- Ultimaco: `/dev/cs2a`

For example, when using Cavium, an `admin` user using `/dev/pkp_nle_drv` should have the following permissions:

```
crw-rw-r-- 1 root admin 126, 0 /dev/pkp_nle_dev
```

Step 2 - Patch vshell binary with static search paths

When you enable processes running as non-root to listen on privileged ports using `setcap` (see *Step 4 – Enable API Gateway processes to listen on privileged ports* on page 101), certain environment variables are disabled as a security precaution. For this reason, you must make the locations of the API Gateway library files available to the operating system.

To patch the `vshell` binary with static search paths, perform the following steps:

1. Use `patchelf` to patch the `vshell` binary with the absolute paths of the API Gateway library files. For example, run the following command:

```
$ patchelf --force-rpath --set-rpath
"$VDISTDIR/platform/jre/lib/amd64/server:
$VDISTDIR/platform/jre/lib/amd64:
$VDISTDIR/platform/jre/lib/amd64/jli:
$VDISTDIR/platform/lib/engines:
$VDISTDIR/platform/lib:
$VDISTDIR/ext/lib"
$VDISTDIR/Linux.x86_64/bin/vshell
```

Alternatively you can use `chrpath`, for example:

```
$ chrpath -r
"$VDISTDIR/platform/jre/lib/amd64/server:
$VDISTDIR/platform/jre/lib/amd64:
$VDISTDIR/platform/jre/lib/amd64/jli:
$VDISTDIR/platform/lib/engines:
$VDISTDIR/platform/lib:
$VDISTDIR/ext/lib"
$VDISTDIR/Linux.x86_64/bin/vshell
```

The examples use `$VDISTDIR` to refer to the absolute path where API Gateway is installed. You must set `$VDISTDIR` or modify the example before using it on your own environment. For example, to set `VDISTDIR` to your API Gateway installation directory:

```
export VDISTDIR=/opt/Axway-7.6.2/apigateway
```

Note `patchelf` is a development and administration tool, and is not installed on systems by default. If you do not wish to install extra tools on your production environment, you can patch the `vshell` binary on another non-production system and move it to the production environment.

Step 3 - Add API Gateway library paths to jvm.xml

You also need to add the API Gateway library paths to `jvm.xml`. To modify your `jvm.xml` file, perform the following steps:

1. Open the `system/conf/jvm.xml` file in your API Gateway installation.
2. Near the top of the file, insert a new line after the following line:

```
<JVMSettings classloader="com.vordel.boot.ServiceClassLoader">
```

3. Enter the following:

```
<VMArg name="-Djava.library.path=
$VDISTDIR/$DISTRIBUTION/jre/lib/amd64/server:
$VDISTDIR/$DISTRIBUTION/jre/lib/amd64:
$VDISTDIR/$DISTRIBUTION/lib/engines:
$VDISTDIR/ext/$DISTRIBUTION/lib:
$VDISTDIR/ext/lib:
$VDISTDIR/$DISTRIBUTION/jre/lib:
system/lib:
$VDISTDIR/$DISTRIBUTION/lib"/>
```

Step 4 - Enable API Gateway processes to listen on privileged ports

API Gateway processes must be able to listen on Internet domain privileged ports (port numbers less than `1024`). You can use the `setcap` command to enable processes running as non-root to listen on privileged ports. In this case, you must set the `CAP_NET_BIND` capability on the `vshell` binary, which enables the following processes to listen on privileged ports:

- API Gateway instance
- Admin Node Manager

- API Gateway Analytics

Note Using `setcap` to set this capability is supported from kernel 2.6.24 onwards. If the kernel version is before 2.6.33, you must enable `CONFIG_SECURITY_FILE_CAPABILITIES`.

To set the capability on the `vshell` binary, run the following command:

```
# sudo setcap 'cap_net_bind_service=+ep' /opt/Axway-
7.6.2/apigateway/platform/bin/vshell
```

To verify that the permission has been set, run the following command:

```
# getcap /opt/Axway-7.6.2/apigateway/platform/bin/vshell
/opt/Axway-7.6.2/apigateway/platform/bin/vshell = cap_net_bind_service+ep
```

Caution If you set this capability, you must remove it again before applying a service pack or uninstalling, as it results in the product binaries being locked.

To remove the capability, run the following command:

```
# sudo setcap -r /opt/Axway-7.6.2/apigateway/platform/bin/vshell
```

Step 5 - Restart API Gateway

When you have completed the preceding steps, start API Gateway as the non-root user. For more information on starting API Gateway, see *Start and stop the API Gateway* on page 77.

Configure an API Gateway encryption passphrase

By default, the API Gateway configuration data is stored unencrypted. However, you can encrypt sensitive information such as passwords and private keys using an encryption passphrase. When the passphrase has been set (and the data has been encrypted with it), you must then enter the passphrase when connecting to the API Gateway with Policy Studio, or when the API Gateway is starting up, so that the encrypted data can be decrypted. You can enter an encryption passphrase at the level of a local Policy Studio project on the local file system, and at the level of a running API Gateway group instance.

All sensitive information in the API Gateway configuration data is encrypted when you set an encryption passphrase. For example, this sensitive information includes passwords that the API Gateway requires for connecting to external systems (databases, LDAP, and so on), or private keys that are not stored on a Hardware Security Module (HSM). All sensitive information is encrypted using the Password-Based Encryption (PBE) system specified by the Public-Key Cryptography Standard (PKCS#12). For more details, see Appendix B of the PKCS#12 documentation.

This topic describes how to specify an encryption passphrase for a local Policy Studio project or when connecting to an API Gateway in Policy Studio, in an API Gateway configuration file, or when the API Gateway is starting up. It also describes how to change the passphrase when it has been set initially.

Caution It is *crucial* that you remember the passphrase when you change it. Failure to remember the passphrase results in the loss of encrypted data, and may prevent the API Gateway from functioning correctly.

Configure the project passphrase using projchangepass

You can use the `projchangepass` command to change the encryption passphrase for a Policy Studio project. This example shows how to change the project passphrase on proj1 from `changeme` to `newpassPhrase`:

```
projchangepass --proj=/home/user1/apiprojects/proj1 --oldpass=changeme --
newpass=newpassPhrase --confirmpass=newpassPhrase
```

For more information on `projchangepass`, see the *API Gateway DevOps Deployment Guide*.

Configure the group passphrase using managedomain

You can use the `managedomain` command to change the encryption passphrase for an API Gateway group. The following example shows this using `managedomain` in command interpreter mode:

```
> change_passphrase group Group1 old_passphrase "" new_passphrase
"12345"
```

For more details on using `managedomain`, see *Managedomain command reference* on page 65.

Note You must also re-encrypt Key Property Store tables after an encryption passphrase for an API Gateway group has been changed. You can do this using the `kpsadmin` tool. For more details, see the *API Gateway Key Property Store User Guide*.

Enter the passphrase when editing configuration in Policy Studio

If you have set an encryption passphrase for the API Gateway configuration data, you must enter this passphrase every time you open a configuration for editing in Policy Studio. You can specify this in the **Enter Passphrase** dialog, which is displayed before editing an active server configuration.

Tip When you first open a connection to an API Gateway in Policy Studio, you specify a **Password**. The different roles of the **Passphrase** and the **Password** fields are as follows:

Passphrase	Used to decrypt sensitive data that has already been encrypted (for example, private keys and passwords) . Not required by default, and only needed if you have already set the encryption passphrase in Policy Studio.
Password	Used to authenticate to the API Gateway's management interface using HTTP basic authentication when opening a connection to a server. Required by default.

Provide the passphrase in a configuration file or at startup

For API Gateway to read (decrypt) encrypted data from its configuration, it must be primed with the passphrase key. You can enter the passphrase directly in a configuration file, prompt for it at startup, or obtain it automatically with a script.

Note Typically, the passphrase is only entered directly in the file if the server must be started as a Linux daemon. In this case, the administrator cannot enter the passphrase manually when the server is starting. To avoid this, you must enter the passphrase in the configuration file.

Enter the Node Manager passphrase in a configuration file

You can enter a passphrase directly in the Node Manager configuration file. Open the following file in your API Gateway installation:

```
INSTALL_DIR/apigateway/system/conf/nodemanager
```

This file contains values for general system settings, such as the server name and trace level, and also (if required) the passphrase key that the Node Manager uses to decrypt its own configuration data.

You should specify the passphrase as the value of the `secret` attribute as follows:

```
secret="myPassphrase"
```

Enter the API Gateway passphrase in a configuration file

You can also enter the passphrase for API Gateway instances created using the `managedomain` script. To do this, enter the `secret` attribute in the `group.xml` file for your API Gateway instance. For example:

```
INSTALL_DIR/apigateway/groups/GROUP/conf/group.xml
```

Prompt for the passphrase at server startup

If you do not wish to enter the passphrase directly in the Node Manager or API Gateway instance configuration file, and do not need to start as a Linuxdaemon, you can configure the Node Manager or API Gateway to prompt the administrator for the passphrase on the command line when starting up. To do this, enter the "`(prompt)`" special value for the `secret` attribute as follows:

```
secret="(prompt)"
```

To configure this for the Node Manager, update your `nodemanager.xml` file. To configure for an API Gateway group, update the relevant `group.xml` file.

Provide the passphrase automatically at startup using a script

Alternatively, you can use a script to automatically provide the passphrase when the API Gateway server starts up. Perform the following steps:

1. Open the following file in your API Gateway installation:

```
INSTALL_DIR/apigateway/groups/GROUP/INSTANCE/conf/service.xml
```

2. Add the following to your `service.xml` file:

```
<SystemSettings tracelevel="INFO"
       passphraseExec="$VINSTDIR/../passwd.sh"
       serviceID="${serviceID}" groupID="${groupID}"
       serviceName="${serviceName}" groupName="${groupName}"
      domainID="${domainID}" title="API Gateway"/>
```

Note The `passphraseExec` option is only used if it is present and the `secret`

option (described in the previous sections) is not used.

3. Create the passphrase script in the specified location.

   For example, the contents of the `passwd.sh` script is as follows:

```
#!/bin/sh
echo secret
```

Note The script must be secured by the operating system file permissions so that it is only accessible by, and can only be invoked by the API Gateway. The command should write the password to standard output.

The following files should also be protected:

- `../system/conf/nodemanager.xml`
- `../skel/service.xml`

Promotion between environments

When you promote and deploy passphrase-protected configuration between environments (for example, from testing to production), the passphrase for the target configuration (production) can be different from the passphrase in the source configuration (testing).

If you are using a different passphrase in each environment, when the deployment takes place, you can specify the correct passphrase for the target configuration. For more details, see *Deploy configuration in Policy Studio* on page 141.

For more details on promoting configuration between environments, see the *API Gateway DevOps Deployment Guide*.

Further information

For more details on supported security features, see the *API Management Security Guide*.

Manage X.509 certificates and keys

For API Gateway to trust X.509 certificates issued by a specific Certificate Authority (CA), you must import that CA's certificate into the API Gateway's trusted certificate store. For example, if API Gateway is to trust secure communications (SSL connections or XML Signature) from an external SAML Policy Decision Point (PDP), you must import the PDP certificate, or the issuing CA certificate into the API Gateway certificate store.

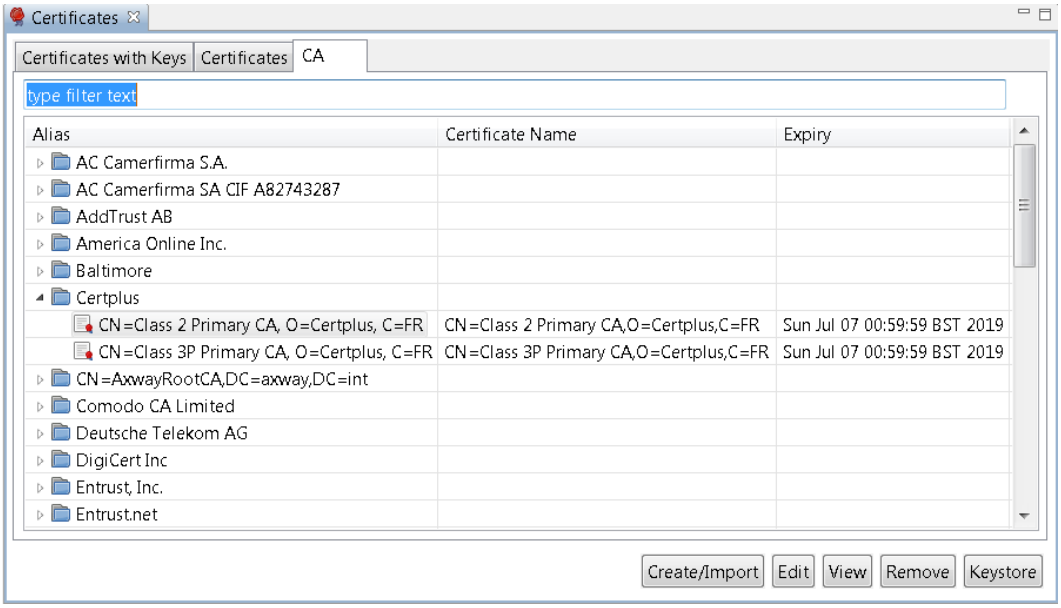
In addition to importing CA certificates, you can import and create server certificates and private keys in the certificate store. These can be stored locally or on an external Hardware Security Module (HSM). You can also import and create public-private key pairs. For example, these can be used with the Secure Shell (SSH) File Transfer Protocol (SFTP) or with Pretty Good Privacy (PGP).

View certificates and keys

To view the certificates and keys stored in the certificate store, select **Environment Configuration > Certificates and Keys > >Certificates** in the tree. Certificates and keys are listed on the following tabs in the **Certificates** window:

- **Certificates with Keys**: Server certificates with associated private keys
- **Certificates**: Server certificates without any associated private keys
- **CA**: Certificate Authority certificates with associated public keys

You can search for a specific certificate or key by entering a search string in the text box at the top of each tab, which automatically filters the tree.



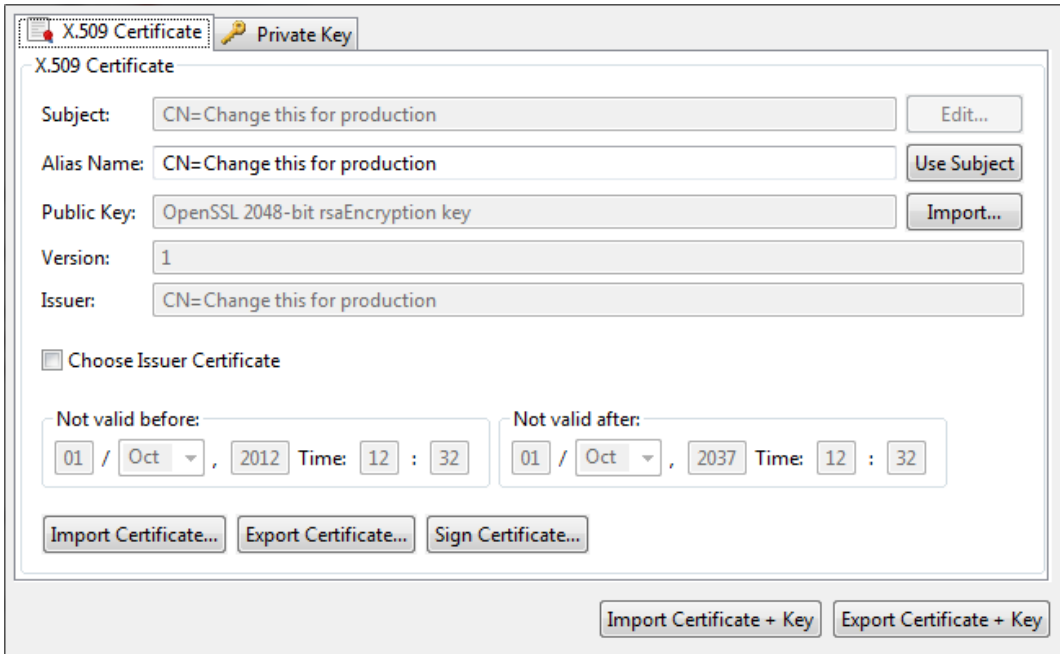
Certificate management options

The following options are available at the bottom right of the window:

- **Create/Import**: Click to create or import a new certificate and private key. For details, see *Configure an X.509 certificate* on page 108.
- **Edit**: Select a certificate, and click to edit its existing settings.
- **View**: Select a certificate, and click to view more detailed information.
- **Remove**: Select a certificate, and click to remove the certificate from the certificate store.
- **Keystore**: Click this to export or import certificates to or from a Java keystore. For details, see *Manage certificates in Java keystores* on page 118.

Configure an X.509 certificate

To create a certificate and private key, click **Create/Import**. The **Configure Certificate and Private Key** dialog is displayed. This section explains how to use the **X.509 Certificate** tab on this dialog.



Create a certificate

Configure the following settings to create a certificate:

- **Subject**:
  Click **Edit** to configure the *Distinguished Name* (DName) of the subject.
- **Alias Name**:
  This mandatory field enables you to specify a friendly name (or alias) for the certificate. Alternatively, you can click **Use Subject** to add the DName of the certificate in the text box instead of a certificate alias.
- **Public Key**:
  Click **Import** to import the subject's public key (usually from a PEM or DER-encoded file).
- **Version**:
  This read-only field displays the X.509 version of the certificate.
- **Issuer**:
  This read-only field displays the distinguished name of the CA that issued the certificate.
- **Choose Issuer Certificate**:
  Select to explicitly specify an issuer certificate for this certificate (for example, to avoid a

potential clash or expiry issue with another certificate using the same intermediary certificate). You can then click the browse button on the right to select an issuer certificate. This setting is not selected by default.

- **Not valid before**:
  Select a date to define the start of the validity period of the certificate.
- **Not valid after**:
  Select a date to define the end of the validity period of the certificate.
- **Sign Certificate**:
  You must click this button to sign the certificate. The certificate can be self-signed, or signed by the private key belonging to a trusted CA whose key pair is stored in the certificate store.

Import certificates

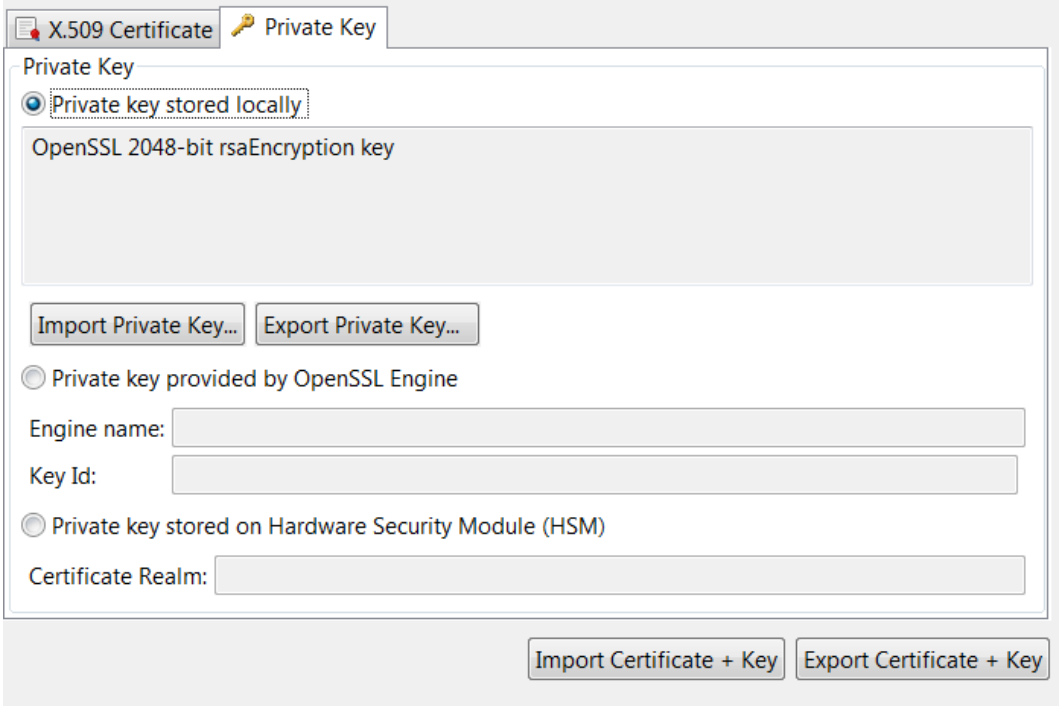
You can use the following buttons to import or export certificates into the certificate store:

- **Import Certificate**:
  Click to import a certificate (for example, from a `.pem` or `.der` file).
- **Export Certificate**:
  Click to export the certificate (for example, to a `.pem` or `.der` file).

Configure a private key

Use the **Private Key**tab to configure details of the private key. By default, private keys are stored locally (for example, in the API Gateway certificate store). They can also be provided by an OpenSSL engine, or stored on a Hardware Security Module (HSM) if required.

API Gateway supports PKCS#11-compatible HSM devices. For example, this includes Thales nShield Solo, SafeNet Luna SA, and so on.



Private key stored locally

If the private key is stored in the API Gateway certificate store, select **Private key stored locally**. The following options are available for keys stored locally:

- **Private key stored locally**:
  This read-only field displays details of the private key.
- **Import Private Key**:
  Click to import the subject's private key (usually from a PEM or DER-encoded file).
- **Export Private Key**:
  Click to export the subject's private key to a PEM or DER-encoded file.

Private key provided by OpenSSL engine

If the private key that corresponds to the public key in the certificate is provided by an OpenSSL engine, select **Private key provided by OpenSSL Engine**.

Configure the following fields to associate a key provided by the OpenSSL engine with the current certificate:

- **Engine name**:
  Enter the name of the OpenSSL engine to use to interface to an HSM. All vendor implementations of the OpenSSL Engine API are identified by a unique name. See your vendor's OpenSSL engine implementation or HSM documentation to find out the name of the engine.

- **Key Id**:
  Enter the key ID used to uniquely identify a specific private key from all others stored on an HSM. When you complete this dialog, the private key is associated with the certificate that you are currently editing. Private keys are identified by their key ID by default.

Private key stored on external HSM

If the private key that corresponds to the public key stored in the certificate resides on an external HSM, select **Private key stored on Hardware Security Module (HSM)**, and enter the name of the **Certificate Realm**.

Note To use the API Gateway's PKCS#11 engine to access objects in an external HSM, the corresponding HSM provider and certificate realms must also be configured. For more details, see *Configure HSMs and certificate realms* on page 111.

Configure HSMs and certificate realms

Certificate realms are abstractions of private keys and public key certificates, which mean that policy developers do not need to enter HSM-specific configuration such as slots and key labels. Instead, if a private key exists on an HSM, the developer can configure the certificate to show that its private key uses a specific certificate realm, which is simply an alias for a private key (for example, `JMS Keys`).

For example, on the host machine, an administrator could configure the `JMS Keys` certificate realm, and create a *keystore* for the realm, which requires specific knowledge about the HSM (for example, PIN, slot, and private key label). The certificate realm is the alias name, while the keystore is the actual private keystore for the realm.

Manage HSMs with keystoreadmin

The `keystoreadmin` script enables you to perform the following tasks:

- Register an HSM provider
- List registered HSM providers
- Create a certificate realm
- List certificate realms

For example, if a policy developer is using JMS, and wants to indicate that private keys exist on an HSM, they could indicate that the certificate is using the `JMS Keys` certificate realm. On each instance using the configuration, it is the responsibility of the administrator to create the `JMS Keys` certificate realm.

For more details, enter `keystoreadmin` in the following directory, and perform the instructions at the command prompt:

```
INSTALL_DIR/apigateway/posix/bin
```

Use keystoreadmin in interactive mode

When you enter `keystoreadmin` without arguments, this displays an interactive menu with the following options:

Option	Description	When to use
1	`Change group or instance`	When registering HSMs or configuring certificate realms, you must choose the local group and instance to configure.
2	`List registered HSM providers`	Display the HSMs that are currently registered.
3	`Register an HSM provider`	Before creating certificate realms, you must first register the HSM. This option guides you through the steps. The HSM must be installed, configured, and active, and you must know the full path to the HSM device driver (PKCS#11). You give the HSM an alias (for example, LunaSA), which you use later when registering certificate realms.
4	`List Certificate Realms`	List configured certificate realms and associated keystores.
5	`Create a Certificate Realm`	Create a keystore and assign it to a certificate realm.

Step 1–Register an HSM provider

You must first register an HSM provider as follows:

1. Open a command prompt in the API Gateway `bin` directory (for example, `INSTALL_DIR/apigateway/posix/bin`).
2. Enter the `keystoreadmin` command.
3. Select option `3) Register an HSM provider`.
4. If prompted, select the appropriate API Gateway group or instance.
5. You are prompted for a provider alias name. The alias is local only. For example, if registering a LunaSA HSM, you might enter the `LunaSA` alias.

6. For convenience, `keystoreadmin` searches for supported HSM drivers. If found, it shows the list of supported drivers. If none are found, this does not mean the driver does not exist. You must see your HSM documentation for the location of the drivers. For example:

```
Choose from one of the following:1) /LunaSA/cryptoki.dll o)
Other q) Quit
```

7. If successful, `keystoreadmin` loads the driver and displays its details. For example:

```
Registering HSM provider...
Initializing HSM...
Crypto Version:2.20
Manufacter Id:SafeNet, Inc.
Library Description:Chrystoki
Library Version:5.1 Device registered.
```

Step 2–Create a certificate realm and associated keystore

To create a certificate realm and associated keystore, perform the following steps:

1. Open a command prompt in the API Gateway `bin` directory (for example, `INSTALL_DIR/apigateway/posix/bin`).
2. Enter the `keystoreadmin` command.
3. Select option `5) Create a Certificate Realm`.
4. You are prompted to enter a certificate realm name. This certificate realm name is used in when configuring the private key of the corresponding X.509 certificate. The realm name is case sensitive (for example, `JMS Keys`).
5. The registered HSMs are listed. For example, select option `1) HSM`.
6. The command connects to the selected HSM, and a list of available slots is displayed. Select the slot containing the private key to use for the certificate realm (for example, select slot `1`).
7. You are prompted to input the PIN passphrase for the slot. The passphrase will not echo any output.
8. When you enter the correct PIN passphrase for the slot, this displays a list of private keys. Choose the key to use for the certificate realm. For example:

```
Choose from one of the following:
  1) server1_priv
  2) jms_priv
  q) Quit

Select option:2
```

9. You are prompted for a file name for the keystore. For example:

```
Certificate realm filename [jms keys.ks]:Successfully created the
certificate realm:JMS KeysPress any key to continue...
```

10. The keystore is output to the API Gateway instance directory. For example:

```
apigateway/groups/group-2/instance-1/conf/certrealms/jms keys.ks
```

Note Each API Gateway instance must have its certificate realm configured. When finished creating certificate realms, you must restart the API Gateway instance for the changes to take effect.

Step 3–Start API Gateway when using an HSM

When API Gateway is configured to use certificate realms, these realms are initialized on startup, and a connection to the corresponding HSM is established. This requires the PIN passphrase for the specific HSM slots. At startup, you can manually enter the required HSM slot PIN passphrase, or you can automate this instead.

Start API Gateway with manually entered PIN passphrase

When API Gateway is configured to use an HSM, API Gateway stops all processing, prompts for the HSM slot PIN passphrase, and waits indefinitely for input. For example:

```
INFO    07/Jan/2015:16:31:54 Initializing certificate realm 'JMS Keys'...
Enter passphrase for Certificate Realm, "JMS Keys":
```

API Gateway does not reprompt if the PIN passphrase is incorrect. It logs the error and continues, while any services that use the certificate realm cannot use the HSM.

Start API Gateway with automatic PIN passphrase

You can configure API Gateway to start and initialize the HSM by invoking a command script on the operating system to obtain the HSM slot PIN passphrase. This enables API Gateway for automatic startup without manually entering the PIN passphrase.

To configure an automatic PIN passphrase, perform the following steps:

1. Edit the API Gateway instance's `vpkcs11.xml` configuration file. For example:

```
apigateway/groups/group-2/instance-1/conf/vpkcs11.xml
```

2. Add a `PASSPHRASE_EXEC` command that contains the full path to the script that executes and obtains the passphrase. The script should write the passphrase to stdout, and should have

the necessary operating system file and execute protection settings to prevent unauthorized access to the PIN passphrase. The following example shows a `vpkcs11.xml` file that invokes the `hsmpin.sh` to echo the passphrase:

```
<?xml version="1.0" encoding="utf-8"?>
<ConfigurationFragment provider="cryptov">

    <Engine name="vpkcs11" defaultFor="">
        <EngineCommand when="preInit" name="REALMS_DIR"
             value="$VINSTDIR/conf/certrealms" />
        <EngineCommand when="preInit" name="PASSPHRASE_EXEC"
             value=""$VDISTDIR/hsmpin.sh"" />
    </Engine>
</ConfigurationFragment>
```

3. API Gateway provides the certificate realm as an argument to the script, so you can use the same script to initialize multiple realms. The following examples show scripts that write a PIN of `1234` to stdout when initializing the `JMS Keys` certificate realm:

```
#!/bin/shcase $1 in"JMS Keys")echo 1234;;esac
```

Configure SSH key pairs

To configure public-private key pairs in the certificate store, select **Environment Configuration > Certificates and Keys > Key Pairs**. The **Key Pairs** window enables you to add, edit, or delete OpenSSH public-private key pairs, which are required for the Secure Shell (SSH) File Transfer Protocol (SFTP).

Add a key pair

To add a public-private key pair, click **Add** on the right, and configure the following settings in the dialog:

- **Alias**:
  Enter a unique name for the key pair.
- **Algorithm**:
  Enter the algorithm used to generate the key pair. Defaults to `RSA`.
- **Load**:
  Click to select the public key or private key files to use. The **Fingerprint** field is auto-populated when you load a public key.

Note The keys must be OpenSSH keys. RSA keys are supported, but DSA keys are not supported. The keys must not be passphrase protected.

Edit a key pair

To edit a public-private key pair, select a key pair alias in the table, and click **Edit** on the right. For example, you can load a different public key and private key. Alternatively, double-click a key pair alias in the table to edit it.

You can delete a selected key pair from the certificate store by clicking **Remove** on the right. Alternatively, click **Remove All**.

Manage OpenSSH keys

You can use the `ssh-keygen` command provided on Linux to manage OpenSSH keys. For example:

- The following command creates an OpenSSH key:

```
ssh-keygen -t rsa
```

- The following command converts an `ssh.com` key to an OpenSSH key:

```
ssh-keygen -i -f ssh.com.key > open.ssh.key
```

- The following command removes a passphrase (enter the old passphrase, and enter nothing for the new passphrase):

```
ssh-keygen -p
```

- The following command outputs the key fingerprint:

```
ssh-keygen -lf ssh_host_rsa_key.pub
```

Configure PGP key pairs

To configure Pretty Good Privacy (PGP) key pairs in the certificate store, select **Environment Configuration > Certificates and Keys > PGP Key Pairs**. The **PGP Key Pairs** window enables you to add, edit, or delete PGP public-private key pairs.

Add a PGP key pair

To add a PGP public-private key pair, click the **Add** on the right, and configure the following settings in the dialog:

- **Alias**:
  Enter a unique name for the PGP key pair.

- **Load**:
  Click **Load** to select the public key and private key files to use.

Note The PGP keys added must not be passphrase protected.

Edit a PGP key pair

To edit a PGP key pair, select a key pair alias in the table, and click **Edit** on the right. For example, you can load a different public key and private key. Alternatively, double-click a key pair alias in the table to edit it.

You can delete a selected PGP key pair from the certificate store by clicking **Remove** on the right. Alternatively, click **Remove All**.

Manage PGP keys

You can use the freely available GNU Privacy Guard (GnuPG) tool to manage PGP key files (available from http://www.gnupg.org/). For example:

- The following command creates a PGP key:

  ```
  gpg --gen-key
  ```

  For more details, see http://www.seas.upenn.edu/cets/answers/pgp_keys.html
- The following command enables you to view the PGP key:

  ```
  gpg -a --export
  ```
- The following command exports a public key to a file:

  ```
  gpg --export -u 'UserName ' -a -o public.key
  ```
- The following command exports a private key to a file:

  ```
  gpg --export-secret-keys -u 'UserName ' -a -o
  private.key
  ```
- The following command lists the private keys:

  ```
  gpg --list-secret-keys
  ```

Global import and export options

This section describes global import and export options available when managing certificates and keys.

Import and export certificates and keys

The following global configuration options apply to both the **X.509 Certificate** and **Private Key** tabs:

- **Import Certificate + Key**:
  Use this option to import a certificate and a key (for example, from a `.p12` file).
- **Export Certificate + Key**:
  Use this option to export a certificate and a key (for example, to a `.p12` file).

Click **OK** when you have finished configuring the certificate and private key.

Manage certificates in Java keystores

You can also export a certificate to a Java keystore. You can do this by clicking **Keystore** on the main **Certificates** window. Click the browse button at beside the **Keystore** field at the top right to open an existing keystore, or click **New Keystore** to create a new keystore. Choose the name and location of the keystore file, and enter a passphrase for this keystore when prompted. Click **Export to Keystore**, and select a certificate to export.

Similarly, you can import certificates and keys from a Java keystore into the certificate store. To do this, click **Keystore** on the main **Certificates** window. On the **Keystore** window, browse to the location of the keystore by clicking the browse button beside the **Keystore** field. The certificates/keys in the keystore are listed in the table.

To import any of these keys to the certificate store, select the box next to the certificate or key to import, and click **Import to Trusted certificate store**. If the key is protected by a password, you are prompted for this password.

You can also use the **Keystore** window to view and remove existing entries in the keystore. You can also add keys to the keystore and to create a new keystore. Use the appropriate button to perform any of these tasks.

Further information

For more details on supported security features, see the *API Management Security Guide*.

Generate a CSR and import the certificate and key

You can use a Certificate Signing Request (CSR) from a Certificate Authority (CA) to obtain certificates used by API Gateway. Policy Studio can generate both X.509 certificates and associated private keys. However, it cannot generate a CSR. You may need to generate a CSR to request a CA to issue a certificate for use in the API Gateway.

This topic explains how to generate a CSR using the open source OpenSSL tool. It explains how to import the resulting certificate and corresponding private key into Policy Studio to be used in API Gateway configuration (for example, for SSL, signing, and encryption).

How are certificates and keys stored in API Gateway?

The API Gateway runtime incorporates X.509 certificate and private key material in its configuration store. However, this is not a Java Key Store (JKS).

A common misunderstanding is that certificates are trusted because they are imported into API Gateway configuration, and displayed in the **Certificates** view in Policy Studio. However, imported certificates are not trusted by default, and must be configured in Policy Studio.

For more details, see *Manage X.509 certificates and keys* on page 106.

What is OpenSSL?

OpenSSL is a free, popular and robust open source toolkit implementing Secure Sockets Layer (SSL) v2 and v3, Transport Layer Security (TLS) v1, and a full-strength general purpose cryptography library. You can use OpenSSL to easily create CSRs. You can also use OpenSSL to create self-signed certificates for use in SSL or message-level security scenarios in a development environment for testing purposes. However, if required, it is considerably faster and easier to create self-signed developer certificates directly in Policy Studio, and there is no need to use an external CA.

Step 1: Create a private key and CSR

You can use Policy Studio to generate certificates and private keys. However, certificates created in this way must be signed (self-signed or by a private key already configured in the tool). In most cases, this is not appropriate, so you should create the certificate and private key using a 3rd party tool such as OpenSSL.

The private key is required to generate the X.509 certificate and corresponding CSR. For example, the following command creates a private key file named `mycompanyca.key` with a key length of 2048 bits (strong) and a corresponding CSR in the file `mycompany.csr`:

```
openssl req -out mycompany.csr -new -newkey rsa:2048 -nodes \
  -keyout mycompany.key
Country Name (2 letter code) []:US
State or Province Name (full name) [Some-State]:Massachusetts
Locality Name (e.g., city) []:Boston
Organization Name (e.g., company) [My Company Ltd]:Acme
Organizational Unit Name (e.g., section) []:
Common Name (e.g., server's hostname) [myserver]:api.acme.comEmail
Address []:api-admin@acme.com

Please enter the following 'extra' attributes to be sent with your certificate
request
A challenge password []:
An optional company name []:Acme
```

You can now submit the CSR to a public or private CA, and the CA will issue the certificate to be imported into API Gateway for use in SSL and message signing operations.

Step 2: Submit the CSR to the CA

Each CA will have a slightly different process for submitting a CSR. However, most CAs provide a web-based interface for this purpose. After you submit the CSR to the appropriate CA, the CA will provide a signed version of the certificate, which you can then import into Policy Studio.

Step 3: Import the certificate and key into Policy Studio

When you receive a signed certificate from the CA after submitting the CSR generated earlier, you must import both the certificate and associated private key into the appropriate key store in Policy Studio to enable SSL and message-level security. Depending on the CA used and how the certificate and key were generated, the certificate an key can be in separate files or combined in a single file.

Perform the following steps:

1. In Policy Studio, connect to an API Gateway instance.
2. In the tree on the left, select **Certificates**. The certificates are displayed in the pane on the right.
3. Depending on your use case, click the appropriate import button at the bottom right:
    - Design-time: Click **Keystore**, click **Add to keystore** on the subsequent dialog box. This imports the certificate and private key into the key store for Policy Studio.
    - Run-time: Click **Create/Import**. This import the certificate and private key into the runtime key store for API Gateway.

4. Configure the **X.509 Certificate** tab as follows:
    i. Click **Import Certificate** if the certificate is in a separate file. If the certificate and key are in the same file, click **Import Certificate + Key**.
    ii. Browse to `mycompany.crt` or the file that contains both the certificate and private key. Ensure that the correct file type is set in the file selector at the bottom right (usually `.pem`).
5. Configure the **Private Key** tab as follows:
    i. Select the **Private Key** tab (you must import both the certificate and the associated private key).
    ii. Click **Import Private Key**.
    iii. Browse to `mycompany.key`. Ensure that the correct file type is set in the file selector at the bottom right.
6. Click **OK** to complete importing the key and certificate.

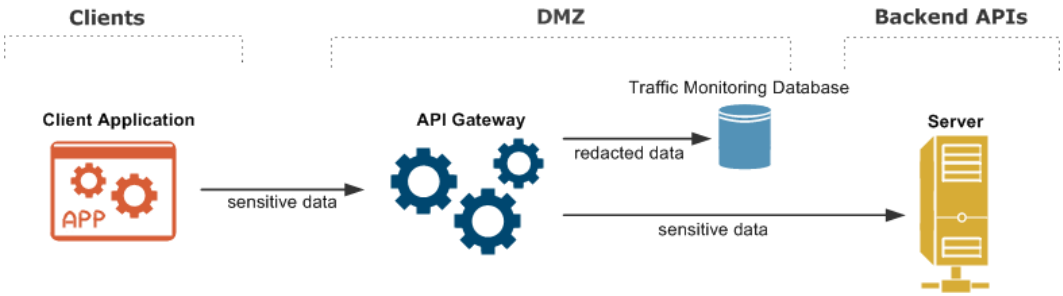
Further information

For more details on supported security features, see the *API Management Security Guide*.

Hide sensitive data in API Gateway Manager

API Gateway enables you to remove sensitive content from messages monitored in the API Gateway Manager web console and traffic monitoring database. You can redact sensitive content message content types such as HTTP headers, JSON, XML, HTML form, and plain text.

For example, sensitive data such as user passwords or credit card details can be redacted from both request and response messages. This means that such sensitive data is only ever present in the API Gateway memory during message processing, and is never displayed onscreen or persisted to disk. This is shown in the following architecture diagram:



API Gateway redaction configuration

In the API Gateway configuration, message redaction rules are configured in the following XML file:

```
apigateway/system/conf/redaction.xml
```

When the API Gateway configuration is loaded, this creates redactors for the specified message protocol and content. This XML-based configuration uses the following model:

```
<Redaction enabled=”true” provider="redactors">
    <JSONRedactor>...</JSONRedactor>
    <RawRedactor>...</RawRedactor>
    <XMLRedactor>...</XMLRedactor>
    <HTTPRedactor>...</HTTPRedactor>
    <FormRedactor>...</FormRedactor>
</Redaction >
```

During the transaction processing, for each traffic monitoring stream, a chain of redactors is created for redacting the received and sent data. Each redactor removes any sensitive data that it finds and passes the data for the next redactor for processing. The redacted content is then written to the traffic monitoring database.

Each redactor defines its supported content types in `RedactMime` child elements. For example, the following shows content types for a JSON redactor:

```
<JSONRedactor>
    <RedactMime mimeType="application/json"/>
    <RedactMime mimeType="text/json"/>
     ...
</JSONRedactor>
```

Enable redaction for an API Gateway

To enable redaction for an API Gateway instance, perform the following steps:

1. Copy the sample redaction configuration file from the following directory:

```
apigateway/samples/redaction/sample_redaction.xml
```

2. Copy to the following directory:

```
apigateway/groups/GROUP/INSTANCE/conf/redaction.xml
```

3. Ensure that redaction is enabled in `redaction.xml` as follows:

```
<ConfigurationFragment>
  <Redaction enabled="true" provider="redactors">
  ...
</ConfigurationFragment>
```

4. You can customize this file to configure redactors for different message payloads (HTTP, JSON, HTML form, and plain text). This is described in the next sections.
5. Edit the following file:

```
apigateway/groups/GROUP/INSTANCE/conf/service.xml
```

6. And add the following line at the end of the file:

```
<NetService provider="NetService">
   ...
   <include file="$VINSTDIR/conf/redaction.xml"/>
</NetService>
```

7. Restart the API Gateway instance.

Note For all message content (HTTP, JSON, HTML form, and plain text), you must first ensure that the appropriate URL is defined in an `HTTPRedactor`. For more details, see *Redact HTTP message content* on page 123.

Redact HTTP message content

You can redact any HTTP header or parameter value from the API Gateway message stream based on HTTP URLs specified in configuration. This applies to both HTTP requests and responses. The following shows a simple example configured in `redaction.xml`:

```
<HTTPRedactor>
   <HTTPURL value="/payment"/>
   <HTTPParam value="credit_card"/>
   <HTTPParam value="password"/>
   <HTTPHeader value="Authorization"/>
</HTTPRedactor>
```

This example specifies to remove the `credit_card` and `password` query string parameters and `Authorization` header from messages sent to and from the `/payment` URL.

URL path matching

Each `HTTPURL` value is used to match URL paths, and to determine if the redaction applies to the transaction. You can use the `match` attribute to specify a match for an exact URL path or for a URL prefix. The following example shows an exact URL path match:

```
<HTTPURL value="/secure_folder" match="exact"/>
```

In this exact match example:

- `/secure_folder` matches
- `/secure_folder/` does not match

- `/secure_folder/123` does not match

The following example shows a URL prefix match:

```
<HTTPURL value="/creditcard/" match="prefix"/>
```

In this prefix match example:

- `/creditcard/` matches
- `/creditcard/charge` matches
- `/creditcard/charge/1234` matches
- `/creditcard` does not match

`HTTPURL` values are also case sensitive. For example:

```
<HTTPURL value="/ORDER/shiptoaddress"/>
```

This is different from:

```
<HTTPURL value="/order/shiptoaddress"/>
```

Supported HTTP features

HTTP features such as the following are supported:

- Chunked transfer encoding
- Multipart body entities (`Content-Type:multipart/`)

Example: Redact an HTTP Basic authorization header

This section shows an end-to-end example of redacting an HTTP Basic authorization header. Given the following HTTP request message:

```
GET /securefiles/ HTTP/1.1
Host:www.httpwatch.com
Authorization:Basic aHR0cHdhdGNoOmY=
```

And the following HTTP redactor configuration:

```
<HTTPRedactor>
   <HTTPURL value="/securefiles/" match="exact"/>
   <HTTPHeader value="Authorization"/>
</HTTPRedactor>
```

The HTTP message is redacted and stored in the traffic monitoring database as follows:

```
GET /securefiles/ HTTP/1.1Host:www.httpwatch.com
```

Redact JSON message content

You can redact JSON content from a message by configuring a specific path to be removed. You can define a relative or absolute location for elements inside a JSON document. When you configure a specific path in the JSON redactor configuration, all elements found in that element are removed.The following general syntax is used to remove JSON content:

```
rule = path_seg [“.”path_seg]*
path_seg = wildcard/rel_wildcard/identifier/array_elem
array_elem = identifier ”[”wildcard/number“]”
identifier = char*
wildcard = “*”
rel_wildcard = “**”
```

The following simple examples show how this syntax works:

```
ca.b.c, a.*.c, a.**.c, **.b.c,**.b[0].c,**.b[*].c, *.b[0].*
```

This results in the following configuration model:

```
<JSONRedactor>
   <RedactMime mimeType="text/json"/>
    ...
   <JSONPath path="a.b.c"/>
   <JSONPath path="**.b[0].c"/>
</JSONRedactor>
```

JSON redactor configuration

The following shows a simple example from `redaction.xml`:

```
<JSONRedactor>
   <RedactMime mimeType="application/json"/>
   <JSONPath path="**.subject[0].id"/>
</JSONRedactor>
```

This example removes JSON content such as the following:

```
authentication.subject[0].id
cert.subject[0].id
attribute.subject[0].id
```

Example: Redact OAuth message tokens from a JSON message

This section shows an end-to-end example of redacting an OAuth message token. Given the following JSON request message:

```
{
   "access_token":"2YotnFZFEjr1zCsicMWpAA",
   "token_type":"example",
   "expires_in":3600,
   "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA",
   "example_parameter":"example_value"
}
```

And the following JSON redactor configuration:

```
<JSONRedactor>
    <RedactMime mimeType="application/json"/>
    <JSONPath path="**.access_token"/>
    <JSONPath path="**.refresh_token"/>
</JSONRedactor>
```

The JSON message is redacted and stored in the traffic monitoring database as follows:

```
{
  "access_token":null,
  "token_type":"example",
  "expires_in":3600,
  "refresh_token":null,
  "example_parameter":"example_value"
}
```

For more details on OAuth, see the *API Gateway OAuth User Guide*.

Redact XML message content

You can redact specific XML content from a message by configuring XML elements or attributes to be removed. The XML redactor removes sensitive data based on the document location. You can define the locations to be removed using the fully qualified name of the redacted element.

For example, to redact all the children of an element named `axway:sensitive_data`, where `xmlns:axway` is `axway.com/`, you can use the following syntax:

```
<XMLRedactedElement localname="sensitive_data" namespace="http://axway.com"
  redactionDisposition="redactChildren"/>
```

You can specify the following XML redaction directives:

`redactChildren`	Removes all children of a specified element
`redactElement`	Redacts the specified element and all its descendants
`redactText`	Removes all text nodes from the specified element
`redactDescendants`	Redacts children and text descendants of the specified node

If you need to redact attributes of the specified node, you can configure this using `XMLRedactedAttribute` (child of `XMLRedactedElement`). `XMLRedactedElement` has two mandatory attributes, `localname` and `namespace`, which have the same meaning for `XMLRedactedAttribute`.

Note An empty XML namespace name is the same as the default document namespace.

XML redactor configuration

The following example from `redaction.xml` removes all children from`a_namespace:a_name`. It also removes the `an_attribute_name` and `another_attribute_name` attributes:

```
<XMLRedactor>
   <RedactMime mimeType="application/xml"/>
   <RedactMime mimeType="text/xml"/>
   <!--Remove children of a_namespace:a_name and some attributtes-->
   <XMLRedactedElement localname="a_name" namespace="a_namespace"
     redactionDisposition="redactChildren">
     <XMLRedactedAttribute localname="an_attribute_name" namespace="an_attribute_
namespace"/>
     <XMLRedactedAttribute localname="another_attribute_name" namespace="o"/>
   </XMLRedactedElement>
</XMLRedactor>
```

The following example removes the `b:a` element and all its children:

```
<XMLRedactor>
   <RedactMime mimeType="application/xml"/>
   <RedactMime mimeType="text/xml"/>
   <!---Remove element b:a and all its descendants-->
   <XMLRedactedElement localname="a" namespace="b"
redactionDisposition="redactElement"/>
</XMLRedactor>
```

Example: Redact a WS-Security username token from an XML message

This section shows an end-to-end example of redacting a WS-Security user name token. Given the following XML request message:

```
<?xml version="1.0" encoding="UTF-8"?>
<Envelope xmlns="http://www.w3.org/2003/05/soap-envelope">
 <Header>
  <Security xmlns="http://docs.oasis-open.org/wss/2004/01/
    oasis-200401-wss-wssecurity-secext-1.0.xsd">
    <UsernameToken>
      <Username>root</Username>
      <Password Type="http://docs.oasis-open.org/wss/2004/01/
         oasis-200401-wss-username-token-profile-1.0#PasswordDigest">EVfkjdkljla=
      </Password>
      <Nonce>tKUH8ab3Rokm4t6IAlgcdg9yaEw=</Nonce>
      <Created xmlns="http://docs.oasis-open.org/wss/2004/01/
         oasis-200401-wss-wssecurity-utility-1.0.xsd">2010-08-10T10:52:42Z
      </Created>
    </UsernameToken>
  </Security>
 </Header>
 <Body>
  <SomeRequest xmlns="http://example.ns.com/foo/bar" />
 </Body>
</Envelope>
```

And the following XML redactor configuration:

```
<XMLRedactor>
    <RedactMime mimeType="text/xml" />
    <XMLRedactedElement localname="UsernameToken"
        namespace="http://docs.oasis-open.org/wss/2004/01/
          oasis-200401-wss-wssecurity-secext-1.0.xsd
          "redactionDisposition="redactChildren">
    </XMLRedactedElement>
</XMLRedactor>
```

The XML message is redacted and stored in the traffic monitoring database as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<Envelope xmlns="http://www.w3.org/2003/05/soap-envelope">
 <Header>
  <Security xmlns="http://docs.oasis-open.org/wss/2004/01/
    oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <UsernameToken></UsernameToken>
  </Security>
 </Header>
```

```
 <Body>
  <SomeRequest xmlns="http://example.ns.com/foo/bar" />
 </Body>
</Envelope>
```

Redact HTML form message content

You can redact the content of specific HTML form fields by configuring the fields to be removed. The following shows an example from `redaction.xml`:

```
<FormRedactor>
    <RedactMime mimeType="application/x-www-form-urlencoded"/>
    <FormField value="credit_card"/>
    <FormField value="phone_number"/>
</FormRedactor>
```

This example removes the contents of the `credit_card` and `phone_number` form fields from the message.

Supported HTML form content types are as follows:

- `application/x-www-form-urlencoded`
- `multipart/formdata`

Redact raw message content

You can redact specific plain text by configuring regular expressions to define content to be removed. The following shows a configuration example:

```
<RawRedactor>
     <RedactMime mimeType="text/plain"/>
     <Regex exp="creditcard\s*=\s*(\d{16})" redact="1" icase="true"/>
     <Regex exp="source:\b(\d{1,3})\.(\d{1,3})\.(\d{1,3})\.(\d{1,3})\b" redact="1,2"
icase="true"/>
     <Regex exp="\d{16}" redact="0" icase="false" />
</RawRedactor>
```

In this configuration model, the `Regex` element includes the following attributes to define the redactor behavior:

Attribute	Description
`exp`	Regular expression used to match the desired content. Possible values are valid regular expressions.

Attribute	Description
`redact`	Specifies which groups in the match are redacted. Possible values are comma-separated lists of group indexes (for example, `1` or `1,2` or `4,6,7`, and so on). You can specify `0` to redact the entire match.
`icase`	Specifies whether the match is case insensitive. Possible values are `true` (case insensitive) and `false` (case sensitive).

Example: Redact credit card details from raw text

This section shows some configured regular expressions and the behavior with specific raw message content. The following expression specifies to redact a defined group:

```
<Regex exp="creditcard\s*=\s*(\d{16})" redact="1" icase="false"/>
```

The following shows example message content and the behavior with this expression:

Message content	Behavior
`&creditcard=1234123412341234`	Content matches expression. Defined group 1 `(\d{16})` is redacted (in this case, `1234123412341234`).

The following expression specifies to redact multiple defined groups:

```
<Regex exp="ccdigits:(\d{1,4})\.(\d{1,4})\.(\d{1,4})\.(\d{1,4})" redact="1,2,3"
icase="false"/>
```

The following shows example message content and the behavior with this expression:

Message content	Behavior
`ccdigits:1234.2345.3456.4567`	Content matches expression. Defined groups 1 `(\d{1,4})`, 2 `(\d{1,4}))`, and 3 `(\d{1,4})` are redacted (in this case `1234`, `2345`, and `3456`. Defined group 4 `(\d{1,4})` is left intact (in this case `4567`).

The following expression specifies to redact content using case insensitivity:

```
<Regex exp="creditcard\s*=\s*(\d{16})" redact="1" icase="true"/>
```

The following shows example message content and the behavior with this expression:

Message content	Behavior
`credit card 123456781234567`	Content matches expression. Entire match (`credit card 1234567812345678`) is redacted.
`Credit Card 1234567812345678`	Content matches expression because of the `icase` attribute. Entire match (`Credit Card 1234567812345678`) is redacted.

Redact sensitive data from log files

For details on how to redact sensitive data from domain audit log and access log files, see the following topics:

- *Configure API Gateway logging and events* on page 162
- *Transaction access log settings* on page 262

Configure an advisory banner

You can configure API Gateway to display an advisory warning message about unauthorized use of API Gateway when establishing a successful user session from Policy Studio or API Gateway Manager.

Configure an advisory banner in API Gateway Manager

To enable an advisory banner, perform the following steps:

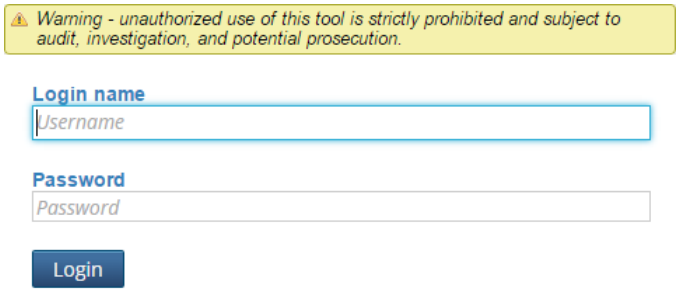
1. Select **Settings > Advisory Banner** in the API Gateway Manager web console.
2. Configure the following settings:
   - **Advisory banner enabled**:
     Select whether the banner is enabled. The default is disabled.
   - **Advisory banner text**:
     Enter the text to display on the advisory banner. The default text is:

```
Warning - unauthorized use of this tool is strictly prohibited and subject to audit, investigation, and potential prosecution.
```

3. Click **Apply** to save the changes.

Advisory banner in API Gateway Manager

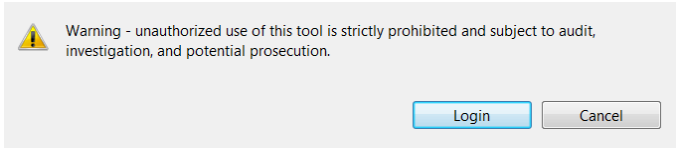
When the banner is enabled, it is displayed on the API Gateway Manager login dialog:



For more details on API Gateway Manager, see *Monitor services in API Gateway Manager* on page 147.

Advisory banner in Policy Studio

The advisory banner is also displayed when you log in to Policy Studio:



For more details on Policy Studio, see the *API Gateway Policy Developer Guide*.

Manage API firewalling

API Gateway provides API firewalling capabilities by embedding Apache ModSecurity. This is a toolkit for real-time HTTP traffic monitoring, logging, and access control. This helps companies to mitigate application-level threats to their APIs. For example, this includes cross-site scripting, SQL injection, command injection, cross-site request forgery, and many others.

API Gateway administrators can configure the embedded ModSecurity engine to protect API Gateway HTTP traffic against threats and monitor reported exceptions. This topic explains how to enable API firewalling on an API Gateway interface in Policy Studio, and how to monitor API firewalling in the API Gateway Manager web console.

For more details on ModSecurity, see Apache ModSecurity documentation.

Configure API firewalling

ModSecurity provides very little protection on its own. However, you can configure the required protection by configuring The ModSecurity rules engine with a threat protection profile. Protecting against specific threats requires specific rules, and different vendors provide rules for specific threat protection capabilities.

The Open Web Application Security Project (OWASP) ModSecurity Core Rule Set (CRS) project provides a popular rule set. For more details on OWASP, see OWASP web page.

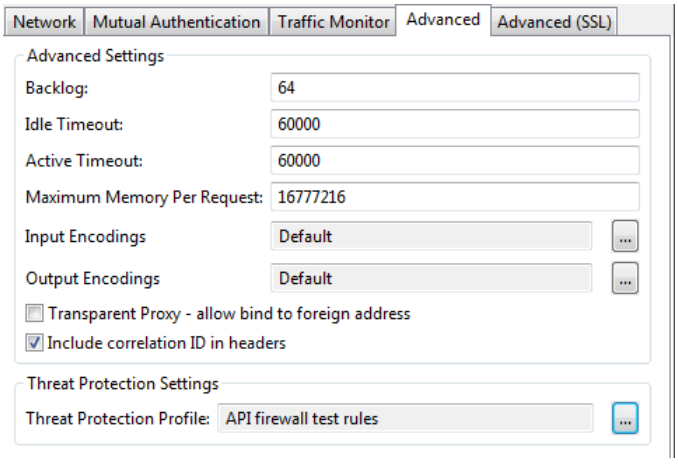
For details on how to write security rules yourself, see, for example, How To Write A WAF Rule - Modsecurity Rule Writing.

This section explains how to configure API firewalling by enabling threat protection and configuring threat protection rules.

Enable threat protection

By default, the embedded ModSecurity engine is disabled. To enable ModSecurity on an API Gateway interface, perform the following steps:

1. In the Policy Studio node tree, click **Environment Configuration > Listeners**, and select the interface you want to enable (for example, **API Gateway > Default Services > Ports**).
2. Right-click the HTTP or HTTPS interface in the window on the right, and select **Edit**.
3. Go to the **Advanced** tab.
4. Under **Threat Protection Settings**, browse to the **Threat Protection Profile** you want to use to protect this interface with ModSecurity rules. For example:



When a profile is selected, all traffic is processed by the ModSecurity engine, and threats are rejected based on the selected security rules.

Configure a threat protection profile

If no threat protection profiles have been configured, do the following:

1. In the **Select WAF Profile** dialog, right-click the **Threat Protection Profiles**, and select **Add a Threat Protection Profile**.
2. Enter a profile name, and configure the following settings:

Field	Description
Configuration directory	Enter the name of the directory that stores the threat protection configuration file. When threat protection has been enabled, the embedded ModSecurity engine looks for threat protection rules configuration in this directory. API Gateway uses the OWASP ModSecurity CRS directory structure. The default is `${environment.VDISTDIR}/system/conf/threat-protection/default`.
Configuration file	Enter the threat protection configuration file name. The default value is `modsecurity.conf`. This file contains the engine global settings. For details on the file format and recommended settings, see Recommended Base Configuration in the ModSecurity documentation.
Rules directory	Enter the name of the subdirectory that stores the threat protection rules. When you download or create ModSecurity security rules, you must put them in this subdirectory. The embedded ModSecurity engine loads all `.conf` files in this directory. The default is `${environment.VDISTDIR}/system/conf/threat-protection/default/activated_rules`.
Alert policy	Select an API Gateway policy you have configured that is executed when a threat protection rule is triggered. The policy can, for example, include an Alert filter to send alert notifications to monitoring systems, or call an API. For details on creating policies, see the *API Gateway Policy Developer Guide*. This setting is optional.

3. Deploy the updated configuration to API Gateway after changing any threat protection settings.

The recommended ModSecurity default configuration sets the engine mode to `SecRuleEngine DetectionOnly`, which applies the activated rules. This does not interfere with the traffic (does not block any requests).

You can also add threat protection profiles in **Environment Configuration > Libraries > Threat Protection Profiles > Add a Threat Protection Profile** in the Policy Studio node tree.

Configure modsecurity.conf file

Depending on your environment, you may need to configure the settings in the `modsecurity.conf` file. For example:

- To handle an `application/xml` content type instead of `text/xml`, add the following line to `modsecurity.conf`:

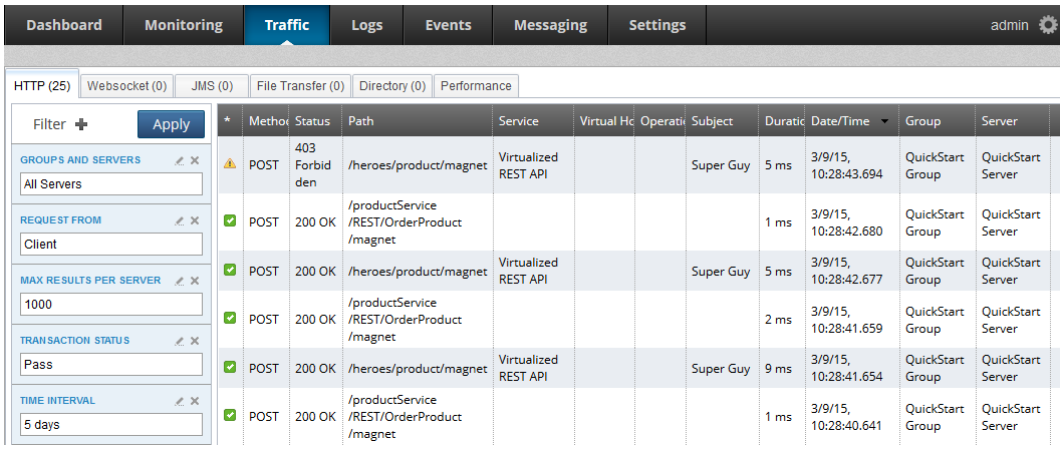
```
SecRule REQUEST_HEADERS:Content-Type "application/xml" \

"id:'200100',phase:1,t:none,t:lowercase,pass,nolog,ctl:requestBodyProcess
or=XML"
```

- To configure ModSecurity to start denying requests with threatening content, in `modsecurity.conf`, change the value of `SecRuleEngine` from `DetectionOnly` to `On`.
- If you have not included the security action in your security rules, you may need to set `SecDefaultAction` in `modsecurity.conf`. See *Configure API firewalling* on page 133. For more details on the `SecDefaultAction` parameter, see ModSecurity Reference Manual.

For more details on the `modsecurity.conf` file format and recommended settings, see Recommended Base Configuration in the ModSecurity documentation.

Monitor API firewalling

An API Gateway administrator or operator can use the **Traffic > HTTP** tab in the API Gateway Manager web console to monitor API firewalling. You can use this tab to show how threat protection affects the HTTP traffic API Gateway serves.



You can filter this tab to display by **Threat Protection** to quickly locate all passed or failed transactions.

1. Click **Filter +** in the search pane.
2. Select **Threat Protection** in the list.
3. Select a threat protection status in the dialog:
    - **Pass**:
      The ModSecurity engine marks all transactions that pass its rules with this status.
    - **Fail**:
      Transactions that violate any active ModSecurity engine rules are marked with this status. These transactions should be monitored because they represent a false positive (the protection rules might need to be adjusted), or malicious client traffic. You can view more details about the failure reason and specific rule violation by drilling down a specific transaction and looking at the trace details.
    - **Exception**:
      Transactions that cause a rule processing or other unknown error are marked with this status. These should not occur and probably indicate some rule configuration problem.
4. Click **Apply**.

For example, the following shows detailed trace output from drilling down a failed transaction:

```
Message:Access denied with code 403 (phase 2).
Pattern match "(?i:(?:\\b(?:(?:s(?:ys\\.(?:user_(?:(?:t(?:ab(?:_column|le)|rigger)
|object|view)s|c(?:onstraints|atalog))|all_tables|tab)|elect\\b.{0,40}\\b
(?:substring|users?|ascii))|m(?:sys(?:(?:queri|ac)e|relationship|column|object)
s|ysql\\.(db|user))|c(?:onstraint ..." at ARGS:q. [file "C:\Axway-7.6.2\
apigateway\system\conf\threat-protection\default\activated_rules\
modsecurity_crs_41_sql_injection_attacks.conf"] [line "116"] [id "950007"]
[rev "2"] [msg "Blind SQL Injection Attack"] [data "Matched Data:SELECT *
FROM USERS found within ARGS:q:SELECT * FROM USERS"] [severity "CRITICAL"]
[ver "OWASP_CRS/2.2.9"] [maturity "9"] [accuracy "8"]
[tag "OWASP_CRS/WEB_ATTACK/SQL_INJECTION"] [tag "WASCTC/WASC-19"]
[tag "OWASP_TOP_10/A1"] [tag "OWASP_AppSensor/CIE1"] [tag "PCI/6.5.2"]
```

In addition of being written to trace files, ModSecurity report is also stored in the message attribute `modsec.error.message`. You can configure an alert policy that, for example, uses an Alert filter with a selector for this message attribute in the default message to pass the threat report to third-party monitoring systems. For more details how to configure the Alert filter, see "Alert" in the *API Gateway Policy Developer Filter Reference*.

Further information

For more details on supported security features, see the *API Management Security Guide*.

Run API Gateway in FIPS mode

API Gateway supports Federal Information Processing Standards (FIPS). When running an API Gateway instance or a Policy Studio client in FIPS mode, the following FIPS-certified cryptographic modules are enabled and invoked for all FIPS-compliant cryptographic algorithms:

Cryptographic Module	FIPS 140-2 Certificate Number
Entrust Authority Security Toolkit for the Java Platform v8.0	1839
OpenSSL FIPS Object Module	1747

Note Running API Gateway in FIPS mode is a separately licensed option that must be specifically ordered. For more details, contact your Axway sales representative.

This topic explains how to enable FIPS for an API Gateway instance and a Policy Studio client, and describes restrictions that apply when running in FIPS mode.

Enable FIPS mode for an API Gateway

You can use the `togglefips` script to enable or disable FIPS mode for an API Gateway instance.

Note

- You can run this script only with a FIPS-enabled API Gateway license.
- You must restart any Node Manager or API Gateway instances after running the `togglefips` script.
- To enable FIPS in a multi-node domain you must run `togglefips --enable` on all nodes in the domain, and all API Gateway processes must be restarted.

Run the following commands:

```
> cd apigateway/posix/bin
> ./togglefips --enable | -e
> ./togglefips --disable | -d
```

Enable FIPS mode for Policy Studio

You can also enable FIPS mode for a Policy Studio client application only. To enable or disable FIPS mode in Policy Studio, perform the following steps:

1. Select **Preferences > IPS Mode**.
2. Select **Enable FIPS Mode in Axway Policy Studio**.

3. Restart Policy Studio with the `clean` option as follows:

```
>policystudio -clean
```

Tip You can use the same instructions to enable FIPS in Configuration Studio.

Restrictions when running in FIPS mode

When running in FIPS mode, certain API Gateway features are not enabled because they depend on non-FIPS compliant algorithms.

Note For a complete list of non-FIPS compliant algorithms and cipher suites configured in all crypto-related filters and interfaces in Policy Studio, select **Tools > Check Security Constraints > FIPS**, and view the output on the pane on the right.

The following features cannot be run when the API Gateway is running in FIPS mode:

- HTTP digest authentication filter
- Kerberos authentication where MD5, DES, and other non-FIPS compliant algorithms are used

Further information

For more details on FIPS, see http://www.nist.gov/itl/fips.cfm.

For more details on supported security features, see the *API Management Security Guide*.

Deploy API Gateway configuration

5

This part contains the following:

Manage API Gateway deployments

You can use Policy Studio to deploy configuration to API Gateway instances running in groups in an API Gateway domain. Policy Studio enables you to edit API Gateway configuration and then deploy it to the server instance, where it can be reloaded later. You can deploy modified configuration to multiple API Gateway instances in a group managed by an Admin Node Manager.

The API Gateway Manager web console also enables you to deploy configuration packages to API Gateway instances running in groups in a domain, to create groups and API Gateway instances, and to manage administrator users. In this way, Policy Studio and the API Gateway Manager enable policy developers and administrators to centrally manage the policies that are enforced at all nodes throughout the network.

In addition, Policy Studio enables you to compare and merge differences between versions of the same policy. Policies can be merged, and deployed to any running instance that is managed by Policy Studio. One of the most powerful uses of this centralized management capability is in transitioning from a staging environment to a production environment. For example, policies can be developed and tested on the staging environment, and when ready, they can be deployed to all instances deployed in the production environment.

Create a project in Policy Studio

Note Before starting Policy Studio, you should first ensure that the Admin Node Manager and the server instance that you wish to deploy to have been started.

To create a new Policy Studio project, select **File** > **New Project**, and follow the steps in the wizard. For more details, see the *API Gateway Policy Developer Guide*.

Alternatively, if a project has already been created, select **File** > **Open Project** in the main menu, or click **Open Project** on the landing page. For more details, see the *API Gateway Policy Developer Guide*.

Edit a project configuration in Policy Studio

When you create or open a Policy Studio project and make a server connection, this loads the project configuration and displays it in the following format:

ProjectName [ServerInstanceType]

For example:

MyDevProject [API Gateway]

When a project configuration is loaded, its services are displayed in the Policy Studio tree on the left. Expand one of the top-level nodes in the tree to display additional details (for example, **APIs**, **Policies**, **Resources**, or **Environment Configuration**).

When editing a project configuration, you can deploy updates using the **Deploy** button in the toolbar (alternatively, press **F6**). For more details, see *Deploy API Gateway configuration* on page 141.

Deploy to a server in Policy Studio

To deploy to a running API Gateway instance in a group, click **Deploy** in the toolbar, and follow the steps in the wizard. For more details, see *Deploy API Gateway configuration* on page 141.

Tip You must connect to the Admin Node Manager server to deploy API Gateway configuration or manage multiple API Gateway instances in your network.

Manage deployments in API Gateway Manager

In the web-based API Gateway Manager tool, the **TOPOLOGY** section on the **Dashboard** tab enables you to create API Gateway groups and instances, and to deploy configuration packages to running servers in API Gateway groups.

For details on how to access the API Gateway Manager, see *Start the API Gateway tools* on page 81.

Compare and merge configurations in Policy Studio

You can compare and merge differences between the currently loaded API Gateway configuration with a configuration stored in a deployment package (.fed file). Click the **Compare** button on the Policy Studio toolbar to select a `.fed` file to compare the current configuration against. The results are displayed in the **Compare/Merge** tab.

For example, you can view the differences made to particular fields in an Authentication filter that occurs in both configurations. When a difference is located, you can merge the differences, and thereby update the fields in the Authentication filter in the current configuration with the field values for the same Authentication filter in the deployment package.

For more details, see the *API Gateway Policy Developer Guide*.

Manage administrator users in API Gateway Manager

You can add new administrator users to enable role-based access to the API Gateway configuration managed by Policy Studio and API Gateway Manager. The default administrator user has access to all API Gateway features in Policy Studio and API Gateway Manager, and can view and modify all API Gateway configurations.

To add or remove administrator users, click the **Settings > Admin Users** tab in the API Gateway Manager. For more details, see *Manage admin users* on page 198.

For more details on role-based access, see *Configure Role-Based Access Control (RBAC)* on page 202.

Configure policies in Policy Studio

You can use Policy Studio to manage the configuration of your policies, which can then be deployed to running instances of Axway API Gateways.

For details on configuring the full range of message filters (for example, for Authentication, Authorization, or Content Filtering), see the *API Gateway Policy Developer Guide*.

Deploy API Gateway configuration

You can edit API Gateway configuration in a Policy Studio project, and deploy to specified API Gateway instances running in an API Gateway group. You can deploy projects based on existing configuration, configuration packages, factory configuration, or a running API Gateway instance.

Policy Studio also enables you to create configuration packages (`.fed`, `.pol`, or `.env` files), and to deploy projects based on configuration packages to API Gateway instances.

You can also deploy API Gateway configuration packages in the API Gateway Manager web console. Alternatively, you can use the `managedomain` script to create and deploy deployment packages (`.fed` files) on the command line.

Deploy configuration in Policy Studio

You can deploy updates to a currently loaded configuration when editing the configuration in Policy Studio. To deploy a currently loaded configuration, perform the following steps:

1. Click the **Deploy** button on the right in the toolbar.
2. In the **Open Connection** dialog, in the **Saved Session**s section, select the server session to use from the list. You can edit a session name by entering a new name and clicking **Save**. You can also click the appropriate button to **Add**, **Clone**, or **Remove** saved sessions.
3. In the **Connection Details** section, configure the following:
   - **Host**:
     Enter the server host to connect to. The default is `localhost`.
   - **Port**:
     Enter the port to connect on. The default Admin Node Manager port is `8090`.
   - **User name**:
     The deployment service is protected by HTTP basic authentication. Enter the administrator user name to use to authenticate to the server. For more details, see *Manage admin users* on page 198.
   - **Password**:
     Enter the password for the administrator user.
4. Click **Advanced** to enter the **URL** of the deployment service exposed by the server. This setting is optional. The default Admin Node Manager URL is `https://localhost:8090/api`.
5. Click **Next** to configure deployment options.

   If an advisory warning has been configured, you must click **Next** again. For more details, see *Configure an advisory banner* on page 131.
6. In the **Select the servers(s) you wish to deploy to** section, select an API Gateway group from the **Group** list, and select the server instance(s) in the box below.

   If the server uses a different API Gateway encryption passphrase for its environment, click **Advanced**, select **The target server uses a different passphrase**, and enter the **Passphrase** used by the target server.
7. Click **Next**, and wait for the deployment to complete.
8. Click **Finish**.

Note You must connect to the Admin Node Manager server to deploy API Gateway configuration or manage multiple API Gateway instances in your network.

View deployment results in Policy Studio

When you click **Deploy**, the **Deployment Results** screen is displayed, and deployment to each server occurs sequentially. Feedback is provided using icons in the **Task** column, and text in the **Status** column. When the configuration has deployed, click **Finish**.

Cancel deployments

You can cancel deployments by clicking the **Cancel** button. Feedback is provided in the **Status** column. You cannot cancel a deployment when it has started. The wizard performs the cancellation at the end of the current deployment, with all remaining deployments being canceled.

Deployment errors

Client-side and server-side errors can occur. Client-side errors are displayed in the **System Trace** in the **Console** view. If any server-side deployment errors occur during the deployment process, you can review these in the **Deployment Error Log** view. This is displayed at the bottom of the screen when you click **Finish**, and lists any errors that occur for each instance. The corresponding Console **Deployment Log** is also available in the **Console** view.

Redeploy

When you have deployed a configuration to one or more instances, you can click back through the wizard to change your selections and redeploy, without needing to exit and relaunch the wizard.

API Gateway configuration packages

You can deploy configuration based on API Gateway configuration packages in Policy Studio and in the API Gateway Manager web console. API Gateway includes the following types of configuration package:

- A *deployment package* is a `.fed` file that contains all API Gateway configuration. This includes policies, listeners, external connections, users, certificates, and environment settings.
- A *policy package* is a `.pol` file that contains policies, listeners, external connections, and environment settings.
- An *environment package* is an `.env` file that contains users, certificates, and environment settings. The content of the `.fed` file is equivalent to the combined contents of the `.pol` and `.env` files.
- A *package property* is a name-value pair that applies to a specific configuration package (`.fed`, `.pol`, or `.env`). Specifying a property associates metadata with the configuration in that package. For example, the **Name** property with a value of `Default Factory Configuration` is associated with a default installation.

For more details on configuration packages and properties, see the *API Gateway DevOps Deployment Guide*.

Create a configuration package in Policy Studio

You can create an API Gateway configuration package for a currently loaded project configuration. To create a package (`.fed`, `.pol`, or `.env`), perform the following steps:

1. In the main menu, select **File > Save Package** followed by the appropriate option:
   - **Deployment Package** (`.fed`)
   - **Policy Package** (`.pol`)
   - **Environment Package** (`.env`)
2. Enter a file name, and click **Save**.

Configure package properties in Policy Studio

You can view or modify API Gateway configuration package properties for a currently loaded project configuration. To view and modify configuration properties, perform the following steps:

1. In the Policy Studio tree, and select **Environment Configuration > Package Properties > Policy** or **Environment**.
2. If you wish to create any additional properties (for example, **Department**), click the green (+) button on the right, and enter a property value (for example, `Engineering`).
3. If you wish to remove a property, click the red (x) button on the right of the property.
4. Click **Save** at the top right of the screen.

Deploy packages in Policy Studio

You can deploy the configuration as normal using the **Deploy** button in the toolbar. For more details, see *Deploy configuration in Policy Studio* on page 141.

Deploy packages in API Gateway Manager

You can also use the API Gateway Manager web console to deploy configuration packages to a group of API Gateway instances. This functionality is available on the default **Dashboard** tab.

For more details, see *Manage domain topology in API Gateway Manager* on page 43.

Deploy packages on the command line

You can create and deploy a deployment package (`.fed`) using the `managedomain --menu` command in the following directory:

```
INSTALL_DIR/apigateway/posix/bin
```

The deployment options for the `managedomain --menu` command are as follows:

```
18) Deploy to a group
19) List deployment information
```

```
20) Create deployment archive
21) Download deployment archive
22) Update deployment archive properties
```

For more details, see *Managedomain command reference* on page 65.

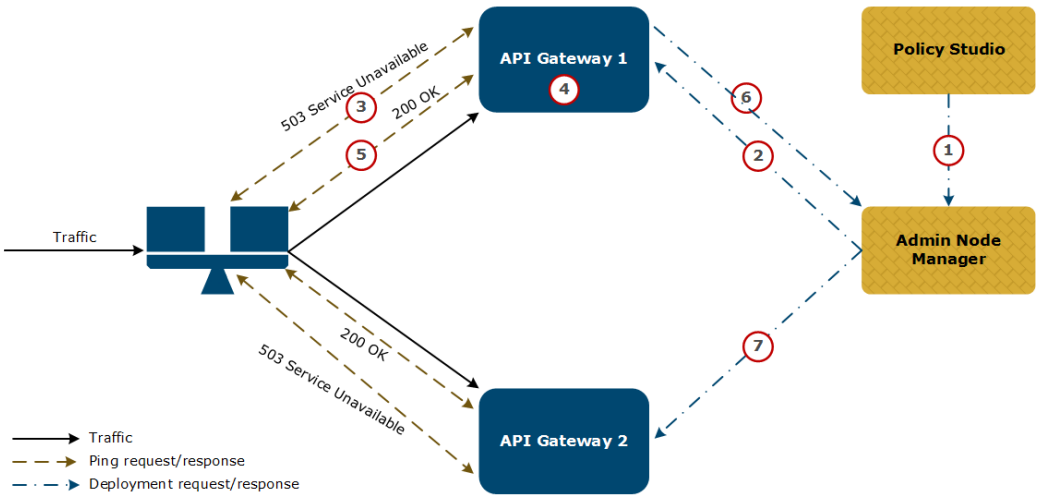
Perform zero downtime deployment

This topic describes how to perform a zero downtime policy deployment to API Gateway in a multi-node API Gateway environment with a load balancer.

When you deploy configuration (for example, a `.fed` file) to a group of API Gateways, the configuration is deployed sequentially to each API Gateway in the group. While the configuration is being deployed to a given API Gateway there is a service interruption during which that API Gateway cannot process traffic.

Zero downtime deployment enables you to orchestrate deployment to a load-balanced set of API Gateways, ensuring that a subset can always process traffic.

The following diagram illustrates the process:



1. A user initiates deployment of new configuration from Policy Studio.
2. The Admin Node Manager starts deployment in `API Gateway 1`.
3. `API Gateway 1` starts responding to a ping from the load balancer with `503 Service Unavailable`. The load balancer stops routing traffic to `API Gateway 1`.
4. `API Gateway 1` performs the deployment.
5. `API Gateway 1` starts responding to the load balancer ping with `200 OK`. The load balancer starts routing traffic to `API Gateway 1` again.
6. `API Gateway 1` informs the Admin Node Manager that deployment is complete.
7. The Admin Node Manager repeats step 2 through step 6 for `API Gateway 2`.

Deploy configuration using zero downtime deployment

To perform a zero downtime policy deployment, follow these steps:

1. Enable zero downtime deployment in Policy Studio, and set the delays before and after deployment. For more information, see *Zero downtime settings* on page 256.
2. Configure your load balancer to ping the Health Check LB policy periodically to determine if each API Gateway is healthy. This is available on the following default URL:

```
http://APIGATEWAY_HOST:8080/healthchecklb
```

3. Initiate deployment to a group of API Gateways using API Gateway Manager, Policy Studio, or managedomain. For more information, see *Deploy API Gateway configuration* on page 141. The configuration is deployed sequentially to each API Gateway in the group.
4. When deployment is initiated on each API Gateway:
    - The Health Check LB policy returns a `503 Service Unavailable` response. This indicates to the load balancer that this API Gateway is not available for traffic and the load balancer stops routing to it.
    - After the specified delay before deployment (for example, 10 seconds), the configuration is deployed to the API Gateway.
    - When the deployment is complete, the Health Check LB policy returns a `200 OK` response. This indicates to the load balancer that this API Gateway is available for traffic again.
    - After the specified delay after deployment (for example, 10 seconds), a response is sent to the deployment request. Deployment can now be initiated to the next API Gateway in the group.

Monitoring and metrics 6

This part contains the following:

Monitor services in API Gateway Manager

This topic explains how to monitor example services using the API Gateway Manager monitoring tools. For example, real-time monitoring metrics, message traffic monitoring, and performance statistics.

Note API Gateway Manager is designed as an operational diagnostics tool only. API Gateway Analytics is recommended for monitoring and reporting of large volumes of historical data. For more details, see the *API Gateway Analytics User Guide*.

Ensure the API Gateway is running

You must first ensure that the API Gateway, Admin Node Manager, and API Gateway tools are running. For more details, see the following:

- *Start and stop the API Gateway* on page 77
- *Start the API Gateway tools* on page 81

Ensure monitoring is enabled

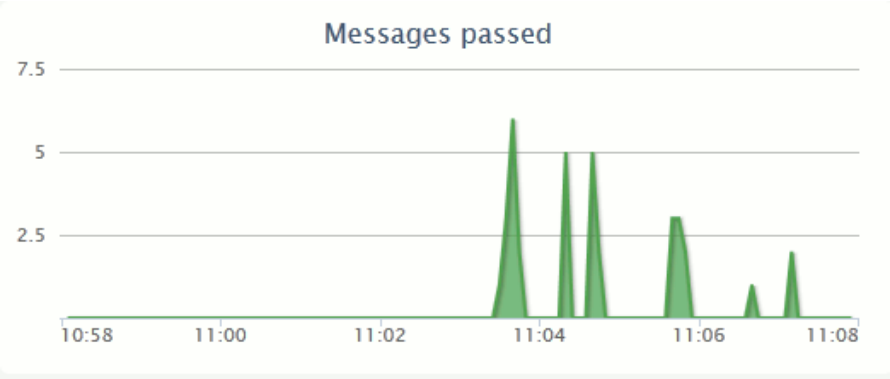
Real-time monitoring and traffic monitoring are enabled by default. If necessary, you can enable these settings as follows:

1. In the Policy Studio tree, select the **Server Settings** node, and select **Monitoring** > **Traffic Monitor**.
2. In the **Traffic Monitor** settings, ensure **Enable Traffic Monitor** is selected.
3. Select **Monitoring** > **Real Time Monitoring**, and ensure **Enable Real Time Monitoring** is selected.
4. Click the **Deploy** icon in the Policy Studio toolbar to deploy these settings to the API Gateway. Alternatively, press F6.

Note Enabling traffic monitoring may have a negative impact on performance. If you wish to maximize performance, you can disable these settings. For more details, see *Traffic monitoring settings* on page 279.

View real-time monitoring

You can view a wide range of monitoring data in the API Gateway Manager. For example, this includes message status, message traffic, filter execution path, message content, system, services, and remote hosts. You can view real-time traffic monitoring summary data on the main **Dashboard** tab in the **TRAFFIC** section. The following example shows the number of messages that have been passed by the API Gateway on to a service:



Each time you send test messages through the API Gateway to an example service (for example, using API Tester or the Send Request (SR) tool), the message status is displayed in the **TRAFFIC** section.

View traffic monitoring

You can use the traffic monitoring tools in API Gateway Manager for operational diagnostics and root cause analysis. The **Traffic** view provides a web-based message log of the HTTP, HTTPS, JMS, and FTP traffic processed by the API Gateway. You can perform tasks such as the following:

- Filter messages on a range of criteria (for example, transaction ID, service name, or remote host)
- Drill down to view message contents
- View performance statistics (for example, number of requests, average bytes sent, or average duration)

For example, you can click the **Traffic** button in the API Gateway Manager to view summary information for each message sent to the API Gateway. Alternatively, you can click one of the summary charts displayed on the **Dashboard** (for example, **Messages passed** or **Messages failed**). This displays the message traffic automatically filtered according to your selection.

The following simple example shows the details displayed on the **Traffic** tab for **Messages passed** by the API Gateway:

*	Method	Status	Path	Service	Virtual Hos	Operat	Subject	Duratio	Date/Time	Group	Server
☑	POST	200 OK	/productService /REST/OrderProduct /magnet					1 ms	1/30/15, 15:18:56.823	QuickStart Group	QuickStart Server
☑	POST	200 OK	/heroes/product/magnet	Virtualized REST API			Super Guy	5 ms	1/30/15, 15:18:56.821	QuickStart Group	QuickStart Server
☑	POST	200 OK	/productService /REST/OrderProduct /magnet					1 ms	1/30/15, 15:18:55.495	QuickStart Group	QuickStart Server
☑	POST	200 OK	/productService /REST/OrderProduct /magnet	Virtualized REST API			Super Guy	3 ms	1/30/15, 15:18:55.494	QuickStart Group	QuickStart Server
☑	POST	200 OK	/heroes/product/magnet	Virtualized REST API			Super Guy	7 ms	1/30/15, 15:18:55.491	QuickStart Group	QuickStart Server
☑	POST	200 OK	/productService /REST/OrderProduct /magnet					1 ms	1/30/15, 15:18:54.433	QuickStart Group	QuickStart Server

Filter message traffic

In the **SELECTION** pane on the left of the **Traffic** tab, you can click the **Apply** button to filter the messages displayed based on a range of criteria. For example, the default filters include **REQUEST FROM** (Client or API Gateway), **MAX RESULTS PER SERVER**, **TRANSACTION STATUS**, and **TIME INTERVAL**.

You can click **Add Filter** to search on different criteria (for example, Service Name, Remote Host, Authentication Subject, Transaction ID, and Operation). The API Gateway inserts a transaction ID in all HTTP and HTTPS traffic in a header named `X-CorrelationID`. When you have selected your search criteria, click the **Apply** button.

View message content

When you click a selected message listed on the **Traffic** tab, this displays the message filter execution path and the contents of each request message and response message. The following example displays the message path for a simple Google Search message:

FILTER EXECUTION PATH

Filter	Status	Audit	Execution Time	Time
Main Policy Chain for API Service 'GoogleSearch'				
Set service context	✓		0	Sep 26, 2013, 09:37:49.494
Set destinationURL	✓		0	Sep 26, 2013, 09:37:49.494
Request	✓		1	Sep 26, 2013, 09:37:49.495
Request				
End Policy Package Chain	✓		0	Sep 26, 2013, 09:37:49.495
Routing	✓		49	Sep 26, 2013, 09:37:49.544
Routing				
Google Search	✓		49	Sep 26, 2013, 09:37:49.544
Google Search				
Connect to URL Google Search	✓		49	Sep 26, 2013, 09:37:49.544
End Policy Package Chain	✓		0	Sep 26, 2013, 09:37:49.544
Response	✓		0	Sep 26, 2013, 09:37:49.544
Response				
End Policy Package Chain	✓		0	Sep 26, 2013, 09:37:49.544
End Policy Package Chain	✓		0	Sep 26, 2013, 09:37:49.544

The following example shows the corresponding message content for the selected message displayed below:

REQUEST FROM CLIENT 127.0.0.1 (127.0.0.1) AND RESPONSE FROM API SERVER

```
GET /ajax/services/search/web GoogleSearch 200 OK 84ms
```

Host	localhost:8080	**Server**	Gateway
User-Agent	Mozilla/5.0 (Windows NT 6.1; WOW64; rv:13.0) Gecko/20100101 Firefox/13.0.1	**Transfer-Encoding**	chunked
Accept	text/html,application /xhtml+xml,application/xml;q=0.9,* /*;q=0.8	**Connection**	keep-alive
		X-Correlation ID	Id-d79540a54fec7d5401266f72 0
Accept-Language	en-us,en;q=0.5	**Cache-Control**	no-cache, no-store, max-age=0, must-revalidate
Accept-Encoding	gzip, deflate	**Date**	Thu, 28 Jun 2012 15:50:43 GMT
Connection	keep-alive	**Expires**	Fri, 01 Jan 1990 00:00:00 GMT
Cookie	liqpw=1135; RT=s=13396745668398 u=&r=http%3A//localhost%3A8080 /services/api/	**Pragma**	no-cache
		Server	GSE
		X-Content-Type-Options	nosniff
Cache-Control	max-age=0	**X-Embedded-Status**	400
		X-Frame-Options	SAMEORIGIN
		X-XSS-Protection	1; mode=block
		Content-Type	text/javascript; charset=utf-8
Save Request		Save Response	

You can click **Save Request** or **Save Response** to download the message contents and save them to a file.

View performance statistics

The **Performance** tab displays performance statistics for the HTTP and HTTPS traffic processed by the API Gateway. For example, these include the number of requests, average bytes sent, and average duration. For example, the **Performance** page is displayed as follows:

Request from	Path	Num. requests	Avg Bytes Received	Avg Bytes Sent	Avg Duration	Min Duration	Max Duration
Client	//ajax/services/search/web	4	436	552	14	3	29
Client	/healthcheck	7	389	531	104	5	620
Client	/services/api/	2	457	602	175	18	333
Gateway	/ajax/services/search /web?v=1.0	5	469	372	255	154	371
Client	/ajax/services/search/web	5	432	578	9064	493	21802

Filter performance statistics

You can click the **Apply** in the left pane to filter the performance statistics displayed based on different criteria. By default, the statistics are grouped by path name, with a time interval of 1 day. You can select different criteria from the **GROUP BY** and **TIME INTERVAL** lists. When you have selected your search criteria, click the **Apply** button.

Detect malformed messages

Messages with malformed content or an incorrect relative path are blocked by the API Gateway and displayed on the **Dashboard** tab in the **TRAFFIC** section as follows:

You can click the chart to display the list of blocked messages automatically filtered on the **Traffic** tab. Click a message in the list to display the filter execution path and message content. The following example shows the execution path of a malformed message that has been blocked by the API Gateway:

Filter	Status	Audit Trail Message
Return HTTP Error 403: Access Denied (Forbidden)		
Make the "Forbidden" Message	✓	
Pass the "Forbidden" message back	✓	Successfully echoed back the message to IP Address /192.168.0.52:51959
Flag as blocked message	✓	

Note When a blocked message has failed in API Gateway, this means that a filter executed in a policy has returned a failure status. When a blocked message generates an exception, this means that a filter executed in a policy has aborted (thrown an exception). For more details on filters and policies, see the *API Gateway Policy Developer Guide*.

Monitor real-time metrics

The **Monitoring** view enables you to monitor successes, failures, exceptions, and real-time metrics for the following:

- **System**: Metrics for memory, disk space, and CPU.
- **API Services**: Metrics for messages and processing times.
- **API Methods**: Metrics for messages and processing times.
- **Clients**: Metrics for messages.
- **Remote Hosts**: Metrics for transactions, bytes sent and received, and response times.

For example, on the **System** tab, when you click a panel in the **ALL SYSTEMS** section at the top, a graph for the selected setting is displayed below. The following example shows the graph displayed for the **System CPU Avg (Max)** setting selected on the right:

You can drill down to view metrics for specific components at the bottom (for example, for a specific API Gateway group or instance, service, client, method, or remote host). You can also configure the metrics time window on the right (for example, 10 minutes, 10 hours, or 5 days).

Configure dynamic trace, logging, and monitoring

You can click the **Settings** > **Dynamic** tab to configure trace, logging, and monitoring settings on-the-fly. These are dynamic settings, which means that you do not need to refresh or deploy to the API Gateway. For example, you can specify these settings for an API Gateway system, instance, service, interface, or path. For more details, see *Configure API Gateway logging and events* on page 162.

Configure API Gateway with the metrics database

This topic explains how to configure an API Gateway instance and Node Manager to store metrics on historic traffic in a relational database used to store metrics. For example, you can configure monitoring in API Gateway Analytics or API Manager to view data stored in the metrics database, or write custom SQL queries to retrieve metrics data as required.

Note This topic explains how to configure API Gateway with a metrics database. This topic assumes that you have already created your metrics database using the steps described in the *API Gateway Installation Guide*.

This topic explains how to perform the following tasks:

- Use Policy Studio to configure an API Gateway instance to write audit logging events to the metrics database, and to write metrics data to the transaction event log.
- Use the `managedomain` command to configure the Node Manager to process event logs and update the metrics database.

API Gateway metrics data streams

The following data streams are used to populate the metrics database:

- **Transaction and system data**: Transaction data includes clients, services, remote hosts, and protocols. System data includes CPU, memory and disk usage, and SLA breaches. The API Gateway writes this data to a transaction event log, with a new log file automatically created every 5 minutes. The Node Manager parses completed event logs and updates the metrics database.
- **Transaction audit log events**: These are written directly to the metrics database by the API Gateway instance.

Connect to the API Gateway in Policy Studio

To connect to the API Gateway in Policy Studio, perform the following steps:

1. Ensure the Admin Node Manager and API Gateway are running. For more details, see *Start and stop the API Gateway* on page 77.
2. Create a new project or open an existing project based on a running API Gateway instance. For more details, see the *API Gateway Policy Developer Guide*.

Configure the metrics database connection

To configure the API Gateway connection to the metrics database, perform the following steps:

1. Expand the **Environment Configuration** > **External Connections** > **Database Connections** node in the Policy Studio tree.
2. Right-click the **Default Database Connection** tree node, and select **Edit**.
3. Configure the database connection to point to your metrics database. For details on connection settings, see the *API Gateway Policy Developer Guide*.
4. Verify that your database connection is configured correctly by clicking the **Test Connection** button on the **Configure Database Connection** dialog.

Tip You can troubleshoot your database connection by viewing the contents of your server `.trc` file in the `INSTALL_DIR/apigateway/trace` directory. For more details, see *Configure API Gateway diagnostic trace* on page 176.

Configure transaction audit logging to the metrics database

To configure the API Gateway instance to write transaction audit log data to the metrics database, perform the following steps:

1. In the Policy Studio tree, select the **Server Settings** node, and select **Logging** > **Transaction Audit Log** in the window on the right.
2. Select the **Database** tab, and select **Enable logging to database**.
3. Select the **Default Database Connection** from the drop-down list if appropriate. Alternatively, select a database connection that you have configured. You must ensure that your database connection points to your metrics database.

For more details, see *Transaction audit log settings* on page 258.

Tip To write the content of message transactions to the database, you must also configure the **Log Message Payload** filter in your policies (for example, at the start and end of the policy). For more details, see the *API Gateway Policy Developer Guide*.

Configure the API Gateway to write to the transaction event log

To configure the API Gateway instance to write transaction data to the transaction event log, perform the following steps:

1. In the Policy Studio tree, select the **Environment Configuration** > **Server Settings** node, and select **Logging** > **Transaction Event Log** in the window on the right.
2. Ensure **Writing to Transaction Event Log** is selected.
3. To enable monitoring of protocol and remote host metrics, select the **Monitoring** > **Traffic Monitor** node, and ensure the following settings are selected:
    - **Enable Traffic Monitor**
    - **Record inbound transactions**
    - **Record outbound transactions**

For more details, see *Transaction event log settings* on page 266.

Deploy the updated configuration to the API Gateway

You must deploy these configuration changes to the API Gateway. Click the **Deploy** button in the toolbar, or press F6.

The API Gateway now sends transaction audit logging to the metrics database, and writes transaction data to the transaction event log. The final step is to configure the Node Manager to read the transaction event logs and write system and transaction metrics to the metrics database.

Configure the Node Manager to process event logs and update the metrics database

If you have not already done so, you must use the `managedomain` tool to enable the Node Manager to process event logs from your API Gateway host, and to write metrics data to the metrics database.

All API Gateway instances running on the host node generate transaction event log files. These files are all written to the same folder, and are collectively processed and aggregated by the Node Manager on the host, and then written to the metrics database. The metrics database provides the data for the graphical charts in the monitoring views in API Gateway Analytics and API Manager.

Note The Node Manager on each host in the domain must be configured to write metrics data to the same database that API Gateway Analytics reads from. The API Gateway can write to the same database for transaction audit logging if required.

Use the managedomain interactive menu

You can enable metrics using the interactive `managedomain --menu` command. The following shows an example:

```
Select option:2
Select a host:
   1) LinuxMint01
   2) Enter host nameEnter selection from 1-2 [2]:1
Enter a new host name [LinuxMint01]:
Enter a new Node Manager name [Node Manager on LinuxMint01]:
Enter a new Node Manager port [8090]:
There is only one Node Manager in this domain so it must remain as an Admin Node
Manager
Do you want to create an init.d script for this Node Manager [n]:
Do you want to reset the passphrase for the Node Manager on this host ? [n]:
Do you wish to edit metrics configuration (y or n) ? [n]:y
Do you wish to enable metrics (y or n) ? [y]:y
Enter metrics database URL [jdbc:mysql://127.0.0.1:3306/reports]:
Enter metrics database username [root]:
Enter metrics database plaintext password [*******]:
Testing Database connectivity for :jdbc:mysql://127.0.0.1:3306/reports, user :root
Metrics database connectivity succeeded
Metrics generation enabled. All other specified metrics settings updated.
Metrics settings updated successfully. Please reboot Node Manager on completion of
this program.
Completed successfully.
Hit enter to continue...
```

Use the managedomain command options

Alternatively, you can use `managedomain` command options to enable metrics when initializing a host, adding a host other than the Admin Node Manager, or editing a host.

The following example shows enabling metrics when initializing a host machine:

```
./managedomain --initialize --metrics_enabled y
  --metrics_dburl jdbc:mysql://127.0.0.1:3306/reports
  --metrics_dbuser root --metrics_dbpass MY_DB_PWD
```

The following example shows enabling metrics when adding a host machine other than the Admin Node Manager:

```
./managedomain --add --anm_host MY_HOSTNAME --nm_entitystore_passphrase MY_CONFIG_PWD
  --metrics_enabled y --metrics_dbuser root --metrics_dbpass MY_DB_PWD
  --metrics_dburl jdbc:mysql://1.2.3.4:3306/reports --nm_name MY_NODE_MNGR --port
8055
```

The following example shows enabling metrics when editing a host machine in the domain:

```
./managedomain --edit_host --nm_entitystore_passphrase bonjour
   --metrics_enabled y --metrics_dburl jdbc:mysql://127.0.0.1:3306/reports
   --metrics_dbuser root --metrics_dbpass MY_DB_PWD
```

The `managedomain` metrics options are described as follows:

Option	Description
`--nm_entitystore_pass`	Specifies the encryption passphrase used to access the API Gateway instance configuration. If no passphrase has been set, omit this argument. For more details, see *Configure an API Gateway encryption passphrase* on page 102.
`--metrics_enabled`	Specifies whether writing of metrics data is enabled. Enter `y` or `n`.
`--metrics_dburl`	Specifies the JDBC URL for the metrics database (for example, `jdbc:mysql://127.0.0.1:3306/reports`).
`--metrics_dbuser`	Specifies the metrics database user (for example, `root` or metrics DB user name).
`--metrics_dbpass`	Specifies the password for the metrics database user.

Note When the `managedomain` command has finished, you must restart the Node Manager.

For more details on `managedomain`, see *Managedomain command reference* on page 65.

Configure additional options for event log processing in the Node Manager

The parameters described in this section specify how transaction event logs are processed in the Node Manager. You can configure these optional settings by editing the Node Manager configuration using the `esexplorer` tool.

For example, perform the following steps:

1. Change to the following directory:
   `INSTALL_DIR/apigateway/posix/bin`
2. Enter the `esexplorer` command.
3. Select **Store** > **Connect**.

4. Browse to `INSTALL_DIR/apigateway/conf/fed/configs.xml`.
5. Select **System Components** > **Metrics Generation Configuration** in the tree on the left.
6. Configure the appropriate fields in the window on the right:

Option	Description
`sourceEventLogDir`	Specifies the folder in which the Node Manager looks for event log files. This should match the API Gateway transaction event log directory set in Policy Studio (see *Transaction event log settings* on page 266). Defaults to `${environment.VDISTDIR}/events`.
`retainProcessedEventLogs`	Specifies whether processed event logs should be deleted or retained in a separate directory. By default, event logs are deleted when their contents are written to the metrics database. Logs can be retained if they are needed for audit purposes or as input to a custom analytics process. Defaults to `false`.
`processedEventLogDir`	When `retainProcessedEventLogs` is `true`, specifies the directory to which event files are moved after being processed by the Node Manager. Defaults to `${environment.VDISTDIR}/events/processed`.
`dirSizeMb`	If `retainProcessedEventLogs` is `true`, specifies the maximum size of the `processedEventLogDir`. When the configured size is reached, the oldest log files in the directory are deleted. Defaults to 1024 MB.
`processCustomMessageAttributes`	Specifies whether message attributes contained in the transaction event log, are written to the database `transaction_data` table. Defaults to `true`. For more details, see *Transaction event log settings* on page 266.

Option	Description
`processCustomMetrics`	Specifies whether custom metrics generated by the API Gateway Java Metrics API and written to the transaction event log are written to the database. Defaults to `true`. For more details, see the following: API Gateway Javadoc available from Axway Support at https://support.axway.com *Transaction event log settings* on page 266

Note When making changes using `esexplorer` , ensure that you open the latest configuration. For example, you could overwrite changes made using `managedomain` if an old version of the configuration was loaded into `esexplorer` and then updated.

7. Stop and restart the Node Manager after editing its configuration using `esexplorer`.

Further information

For details on viewing metrics in API Manager, see the *API Manager User Guide*. For details on viewing metrics in API Gateway Analytics, see the *API Gateway Analytics User Guide*.

Purge the metrics database

You can use the `dbpurger` command to connect to your metrics database and to purge old data (for example, from API Manager or third-party tools). This command also enables you to retain a specified amount of data, and to archive all data.

This topic assumes that you have already configured the connection to your metrics database. For more details, see:

- *Configure API Gateway with the metrics database* on page 153
- *API Manager User Guide*

Run the dbpurger command

For API Gateway and API Manager metrics, you can run the `dbpurger` command from the following directory:

```
INSTALL_DIR/apigateway/posix/bin
```

dbpurger options

You can specify the following options to the `dbpurger` command:

Option	Description
`-h, --help`	Displays help message and exits.
`-p PASSPHRASE, --passphrase=PASSPHRASE`	Specifies the configuration passphrase (leave blank for zero length).
`--dbname=DBNAME`	Specifies the database name (mutually exclusive with `dburl`, `dbuser`, and `dbpass` options).
`--dburl=DBURL`	Specifies the database URL.
`--dbuser=DBUSER`	Specifies the database user.
`--dbpass=DBPASS`	Specifies the database passphrase.
`--archive`	Archive all data.
`--out=OUT`	Archive all data in the specified directory.
`--purge`	Purge data from the database. You must also specify the `--retain` option.
`--retain=RETAIN`	Specifies the amount of data to retain (for example, `30days`, `1month`, or `1year`). You must specify this option with the `--retain` option.

Example dbpurger commands

This section shows examples of running `dbpurger` in default interactive mode and of specifying command-line options.

Run dbpurger in interactive mode

The following example shows the output when running the `dbpurger` command in interactive mode. This example archives all data, retains three months of data, and purges older data from the database:

```
>dbpurger
Choosing:Default Database Connection
Archive database (Y, N) [N]:y
Archive path [./archive]:Purge an amount of data from the database (Y, N) [N]:y
Amount of data to retain (e.g. 1year, 3months, 7days) [3months]:
Wrote archive:./archive/process_groups.xml
Wrote archive:./archive/processes.xml
Wrote archive:./archive/metric_types.xml
Wrote archive:./archive/audit_log_sign.xml
Wrote archive:./archive/time_window_types.xml
Wrote archive:./archive/audit_log_points.xml
Wrote archive:./archive/audit_message_payload.xml
Wrote archive:./archive/transaction_data.xml
Wrote archive:./archive/metric_groups.xml
Wrote archive:./archive/metric_group_types.xml
Wrote archive:./archive/metrics_alerts.xml
Wrote archive:./archive/metrics_data.xml
Purging data older than:Wed Jun 27 15:26:00 BST 2012
Purging table:audit_log_sign... deleted 0 rows
Purging table:transaction_data... deleted 0 rows
Purging table:audit_message_payload... deleted 7 rows
Purging table:audit_log_points... deleted 16 rows
Purging table:metrics_alerts... deleted 4 rows
Purging table:metrics_data... deleted 703 rows
```

Specify dbpurger command options

The following example shows the output when specifying options the `dbpurger` command. This example retains 30 days of data, and purges older data from the database:

```
dbpurger --dburl=jdbc:mysql://127.0.0.1:3306/reports --dbuser=root --dbpass=fred
   --purge --retain=30days
```

Note You can run `dbpurger` without a password by specifying the name of the database connection. For example:

```
dbpurger --dbname="Default Database Connection" --archive --out=archive.dat
```

Troubleshoot your API Gateway installation 7

This part contains the following:

Configure API Gateway logging and events

This topic describes how to configure API Gateway logging and events. It includes the following:

API Gateway logs and events

This section first provides an overview of the different types of logs and events that are displayed in API Gateway Manager, and how to configure them.

You can use the **Logs** and **Events** views in the API Gateway Manager web console to view and search API Gateway log and event files, and you can use the **Settings > Dynamic** tab to configure the log settings at runtime.

You can also configure API Gateway log and event settings in **Server Settings** in Policy Studio.

Domain management and diagnostics

The **Logs** view displays the following API Gateway logs for domain management and runtime diagnostics:

- **Domain Audit**: Displays management changes at the API Gateway domain level (such as updates to API Gateway configuration, topology, login, or deployment). The domain audit log is configured by default. For more details, see *Configure audit logs per domain* on page 163.
- **Trace**: Records detailed diagnostic and debugging information on API Gateway instance execution (such as services starting or stopping, or messages sent through the API Gateway). The trace log is configured by default. Include trace log files in your Support query when raising issues with Axway Support. For more details, see *Configure API Gateway diagnostic trace* on page 176.

Message transactions

The **Logs** view also displays the following API Gateway logs for message transactions:

- **Transaction Audit**: Records policy-level message transaction log entries generated by each filter as the message passes through the filter. You can define the transaction audit log output for each filter in Policy Studio (for example, success, failure, or abort filters), and configure different log output destinations. For more details, see *Transaction audit log settings* on page 258.
- **Transaction Access**: Provides a summary of HTTP request and response message transactions in Apache HTTP Server format. You can configure the access log per API Gateway path. For more details, see *Transaction access log settings* on page 262.

Events and alerts

The **Events** view displays in-memory events for transaction audit logs, system alerts, and SLA alerts. For more details, see *Manage API Gateway events and alerts* on page 170.

Configure audit logs per domain

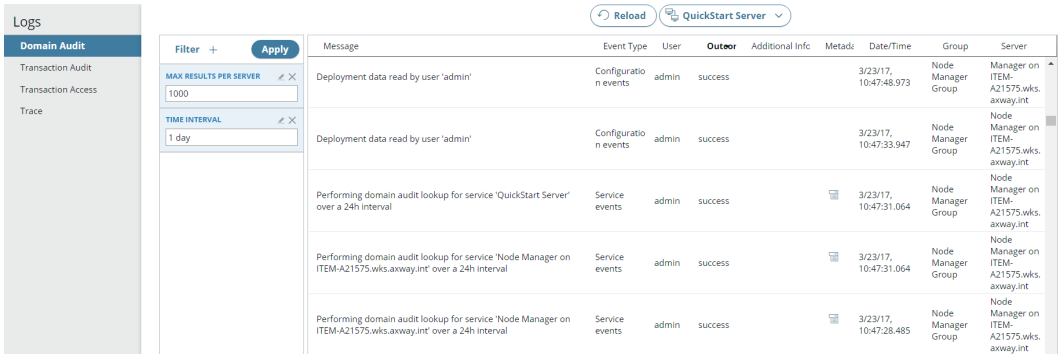
The domain audit log captures management changes in the API Gateway domain that are written by the Admin Node Manager and by API Gateway instances. This includes details such as API Gateway configuration changes, log in/log out, deployments, user, or topology changes. For example, user Joe deployed a new configuration, admin user created a new group, or user Jane has read deployment data.

The domain audit log is enabled by default. You can configure options such as the number of events displayed, time interval, and event type.

View domain audit log events in API Gateway Manager

To view domain audit log events in the API Gateway Manager web console, perform the following steps:

1. In the API Gateway Manager, select **Logs > Domain Audit**.
2. Configure the number of events displayed in the **Max results per server** field on the left. Defaults to `1000`.
3. Configure **Time Interval** for events. Defaults to `1 day`.
4. Click the **Filter** button to add more viewing options (**Event Type** or **Groups and Servers**).
5. Click **Apply** when finished.



View the domain audit log file

Alternatively, you can view contents of the domain audit log file. For example, the following shows the file for the Admin Node Manager:

```
<install-dir>/apigateway/logs/audit.log
```

For example:

```
{"timestamp":1397724538713,"message":"User 'admin' connected with 3 defined user
roles","eventId":107,"metadata":{"userID":"admin"}}
{"timestamp":1397724539638,"message":"Deployment data read by user
'admin'","eventId":1037,"metadata":{}}
{"timestamp":1397726232992,"message":"Performing domain audit lookup for service
'Node Manager on cayote.acme.com'
over a 24h interval","eventId":9,"metadata":
{"userID":"admin","serviceID":"nodemanager-1"}}
{"timestamp":1397726235233,"message":"Performing domain audit lookup for service
'Node Manager on cayote.acme.com'
over a 24h interval","eventId":9,"metadata":
{"userID":"admin","serviceID":"nodemanager-1"}}
```

The default maximum size for the audit log file is 5 MB. A new file is created when the server instance restarts. The maximum of files stored in the `logs` directory is `50`. When this maximum number of log files is reached, the files roll over, and the oldest files are deleted. See also *Offload audit log files to an external audit server* on page 166.

Configure events displayed in domain audit log

To configure the set of events that are displayed in the domain audit log, perform the following steps:

1. In API Gateway Manager, select **Settings > Domain Audit Events**.
2. Select the event categories that you wish to display in the domain audit log. You can drill down in each category to select individual events. If events are not selected, they are not written to the domain audit log. For example, the available events include the following:

Event category	Event types
API Manager admin	API Manager web server error or update.
Application events	Application and associated credentials, quota, or client created, deleted, or disabled in API Manager.
Communication events	Communication between the Admin Node Manager and API Gateway (connection or failure).
Configuration events	Configuration deployment started, completed, error, or rollback. Passphrase or archive update, and so on.
KPS events	Key Property Store (KPS) object created, query read, and so on.
Management events	API publish request, approval, or rejection in API Manager.
Organization events	Organization or registration token created, deleted, or updated in API Manager.
Service events	Service started, stopped, or shutdown failed, and audit log offload or event configuration.
Session events	HTTPS and TLS session established, failed, or terminated.
Topology events	Host, group, or instance added, removed, or updated, and so on.
User events	API Manager user created, updated, deleted, logged in/out in. Password updated, reset, and so on.

Event category	Event types
User store events	Admin user, role, and password policy created, updated, deleted, and so on.

3. Click **Apply** when finished.

The list of configured domain audit log events is stored on disk in the following JSON file:

```
<install-dir>/system/conf/events.json
```

Offload audit log files to an external audit server

You can periodically offload the following audit log files to an external audit server using an HTTP `POST`:

- Domain audit log
- Transaction audit log

When you enable this feature, these log files are offloaded every 5 minutes. The files are rolled over when the scheduler runs to ensure that the records audited up to that point are offloaded. This guarantees a greater degree of synchronicity between the local and remote audit records.

To configure how the scheduler connects to the remote audit server, perform the following steps:

1. In API Gateway Manager, select **Settings > Domain Audit Settings**.
2. Configure the following settings:
    - **Enabled**: Select whether the external audit offload scheduler is enabled. This is disabled by default.
    - **Destination URL**: Enter the required HTTP URL of the external audit server. The application at this URL must be capable of processing the audit files.
    - **Username**: If the audit server requires HTTP Basic authentication, enter the user name.
    - **Password**: If HTTP Basic authentication is required, enter the password.
    - **Trusted Certificates**: Enter the list of PEM-encoded certificates that are considered trusted for the TLS connection to the remote audit server.
3. Click **Apply** when finished.

Redact domain audit log output

You can also customize and redact the contents of the output in the `logs/audit.log` file. For example, for security purposes, you can redact sensitive information, such as specific query parameters that contain customer details, passwords, or credit card information. Alternatively, you can prevent the file from becoming flooded with specific messages, such as `GET` API calls for metrics.

You can use the following file to customize the output of the domain audit log file:

```
<install-dir>/apigateway/conf/apiaudit.xml
```

This file enables you to specify rules to filter out sensitive details or noisy API calls. The default file contains some predefined rules (for example, filtering out metrics). You can use this file to specify whether an entry is made to the domain audit log file, and to specify the contents of the text in the output message.

For example, the following entry specifies an `outputMessage` for all `GET` messages on the `ops/setserviceconfig` path:

```
<apiauditrule>
   <method>GET</method>
   <path>^ops/getserviceconfig$</path>
   <pathMatch>MATCHES</pathMatch>
       <queryArgs>*</queryArgs>
       <outputMessage>Update configuration for service '${serviceName}'
        :${queryArgs}</outputMessage>
</apiauditrule>
```

The following example specifies no `outputMessage` for `GET` messages on the `api/monitoring/metrics` path:

```
<apiauditrule>
   <method>GET</method>
   <path>api/monitoring/metrics</path>
   <pathMatch>BEGINS_WITH</pathMatch>
</apiauditrule>
```

Domain audit rule syntax

The rules in the `apiaudit.xml` file analyze the traffic passing through the API Gateway router service, and control the entries in the domain audit log. These rules are checked in the order specified in the file. The `method`, `path` and `pathMatch` elements determine whether a rule is triggered. If a rule is triggered, all subsequent rules are ignored. You should specify all rules in order of priority (for example, most sensitive or noisy first).

The domain audit rule elements are described as follows:

Element	Description
`method`	Required comma-separated list of HTTP methods (`GET`, `PUT`, and so on). Use the `*` wildcard to specify all methods.
`path`	Required regular expression that specifies a URL path (for example, `^api/domainaudit/search$`). Use the `*` wildcard to specify all paths.

Element	Description
`pathMatch`	Required path matching statement (one of the following: `MATCHES`, `BEGINS_WITH`, `ENDS_WITH`, `CONTAINS`, `DOES_NOT_CONTAIN`, `IS`, `IS_NOT`, `DOES_NOT_MATCH`).
`queryArgs`	Option to specify query string arguments output in the log. To redact certain arguments, you must explicitly list only the arguments you wish to show in a comma-separated list. Leaving this blank or omitting the element specifies that no query arguments are displayed. The `*` wildcard specifies that all query arguments are available for printing.
`outputMessage`	Option to specify the message output printed in the log. Leaving this blank or omitting the element means that no entry is made in the domain audit log for this rule.

For more details and example rules, see the contents of the `conf/apiaudit.xml` file.

Configure transaction audit log destinations

The API Gateway provides detailed transaction audit logging for specific message filters (for example, the request, time of the request, where the request was routed to, and response returned to the client). You can configure transaction logging output to a number of different destinations:

- Text file
- XML file
- Database
- Local syslog
- Remote syslog
- System console

Transaction audit logging is not configured by default. To configure where transaction audit log information is sent, perform the following steps:

1. In the Policy Studio tree, select **Server Settings > Logging > Transaction Audit Log**.
2. Specify the required settings on the appropriate tabs (for example, **Text File**, **Database**, or **XML File**).
3. When finished, click **Save** at the bottom right.
4. Click the **Deploy** button in the toolbar to deploy your settings to the API Gateway.

When a transaction audit log **Text File** destination is enabled, a text log file is written and displayed in API Gateway Manager in the **Logs > Transaction Audit** view. For details on configuring all available options, see *Transaction audit log settings* on page 258.

Configure transaction audit logs per filter

You can configure the transaction audit log level and log message for a specific filter as follows:

1. In the Policy Studio tree, click any policy to display it in the canvas on the right (for example, **QuickStart > Virtualized Services > REST > GetProducts**).
2. Double-click a filter on the canvas to edit (for example, **Connect to Heroes REST Service**).
3. Click **Next** to display the **Transaction Audit Logging Level and Message** window.
4. Select **Override Logging Level for this filter**.
5. Select the log levels required for troubleshooting (for example, **Fatal** and **Failure**).
6. Enter any non-default log messages if required.
7. Click **Finish**.
8. Click the **Deploy** button in the toolbar to deploy your settings to the API Gateway.

Message payload logging

You can also enable logging of the message payload in the transaction using a **Log Message Payload** filter in your policy, or by enabling dynamic payload logging in API Gateway Manager (see *Configure dynamic trace and log settings* on page 171). When message payload logging is enabled, the transaction payload is logged to the transaction audit log destinations that are configured.

For more details on transaction audit logging for specific message filters, see the following in the *API Gateway Policy Developer Filter Reference*:

- **Transaction Audit Logging Level and Message** monitoring in each filter
- **Log Message Payload** filter

Configure transaction event logs per API Gateway

The transaction event log provides a summary of each API Gateway transaction. These logs are persisted, and when configured, are used to generate metrics on the **Monitoring** tab in API Manager, or in third-party monitoring tools such as Splunk.

Transaction event logging is enabled by default. To configure the transaction event log output, perform the following steps:

1. In the Policy Studio tree, select **Server Settings > Logging > Transaction Event Log**.
2. Specify the required settings (for example, directory name, max disk space, and so on).
3. When finished, click **Save** at the bottom right.
4. Click **Deploy** in the toolbar to deploy your settings to the API Gateway.

For details on configuring all the available options, see *Transaction event log settings* on page 266.

Configure transaction access logs per path

The access log provides summary of the HTTP request and response messages that are written to an access log file in the format used by Apache HTTP Server. For example, this includes details such as the remote hostname, user login name, and authenticated user name.

Access logging is not configured by default. To configure the access log output, perform the following steps:

1. In the Policy Studio tree, select **Server Settings > Logging > Transaction Access Log**.
2. Specify the required settings (for example, file name, directory name, and so on).
3. When finished, click **Save** at the bottom right.
4. You must also configure the access log at the service level on a specific relative path. In the Policy Studio tree, select the relative path, right-click it in the **Resolvers** pane, and select **Edit**.
5. Click the **Logging Settings** tab, and select **Include in server access log records**.
6. Click the **Deploy** button in the toolbar to deploy your settings to the API Gateway.

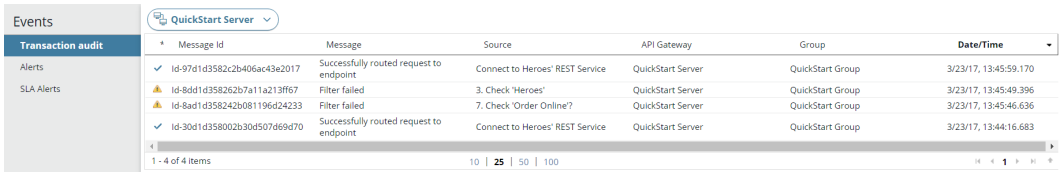
For details on configuring all the available options, see *Transaction access log settings* on page 262.

Manage API Gateway events and alerts

The **Events** view in API Gateway Manager enables you to view and search the contents of the following in-memory API Gateway events and alerts:

- **Transaction Audit**: When policy-level transaction audit logging is configured in API Gateway filters, this displays an in-memory list of transaction audit log events. A transaction audit log destination does not need to be enabled for the in-memory list of events to be updated. For more details, see *Transaction audit log settings* on page 258.
- **Alerts**: When system **Alert** filters are configured in your policies, this displays an in-memory list of alert events. An alert destination does not need to be enabled for in-memory list of events to be updated. For more details, see the *API Gateway Policy Developer Guide*.
- **SLA Alerts**: When **SLA Alert** filters are configured in your policies, this displays an in-memory list of Service Level Agreement (SLA) alert events. An alert destination does not need to be enabled for in-memory list of events to be updated. For more details, see the *API Gateway Policy Developer Guide*.

The following example shows some **Transaction Audit** events:



Configure dynamic trace and log settings

You can click the **Settings > Dynamic** tab in API Gateway Manager to configure dynamic trace, logging, and monitoring settings on-the-fly at runtime. These settings are dynamic because you do not need to refresh or deploy updates to the API Gateway. You can specify these settings for an API Gateway system, instance, service, interface, or path.

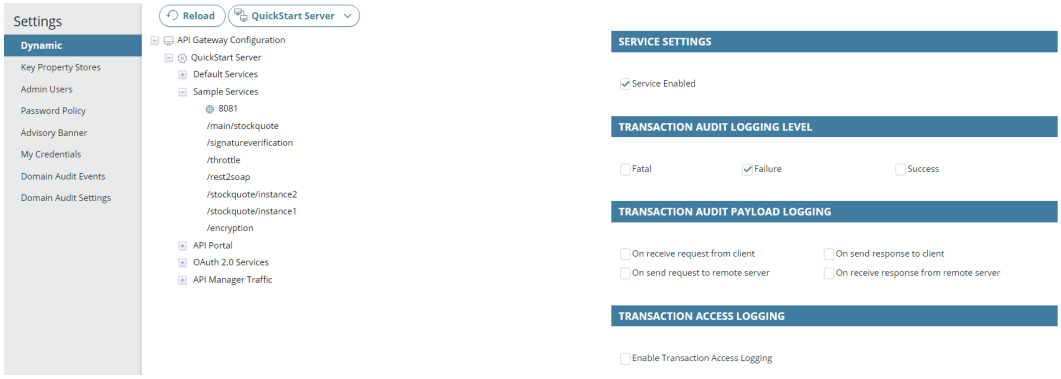
For example, the top-level **SYSTEM SETTINGS** allow you to configure logging of inbound and outbound transactions, policy paths, and message trace. You can select an HTTP interface in the tree on the left to configure the **INTERFACE SETTINGS**, **TRAFFIC MONITORING SETTINGS**, and **Trace level**. You can select the API Gateway instance in the tree to configure its trace level.

You can also select a relative path or service in the tree, and configure the following options:

- **SERVICE SETTINGS**: Select whether the service is enabled.
- **TRANSACTION AUDIT LOGGING LEVEL**: You can select from **Fatal**, **Failure**, and **Success**. See the **Transaction Audit Logging Level and Message** monitoring setting in the *API Gateway Policy Developer Guide*.
- **TRANSACTION AUDIT PAYLOAD LOGGING**: Disabled by default. See the **Log Message Payload** filter in the *API Gateway Policy Developer Filter Reference*.
- **TRANSACTION ACCESS LOGGING**: See *Transaction access log settings* on page 262.

When finished, click **Apply Changes**at the bottom of the page.

The following example shows the dynamic settings available for a service:



Further information

For more details on real-time monitoring and traffic monitoring, see *Monitor services in API Gateway Manager* on page 147.

Configure open logging

Open logging enables you to consolidate all of the transaction event data (metadata and payload), traces, and system metrics stored by API Gateway and visualize and analyze them in external observability systems (for example, a third-party system, such as Splunk).

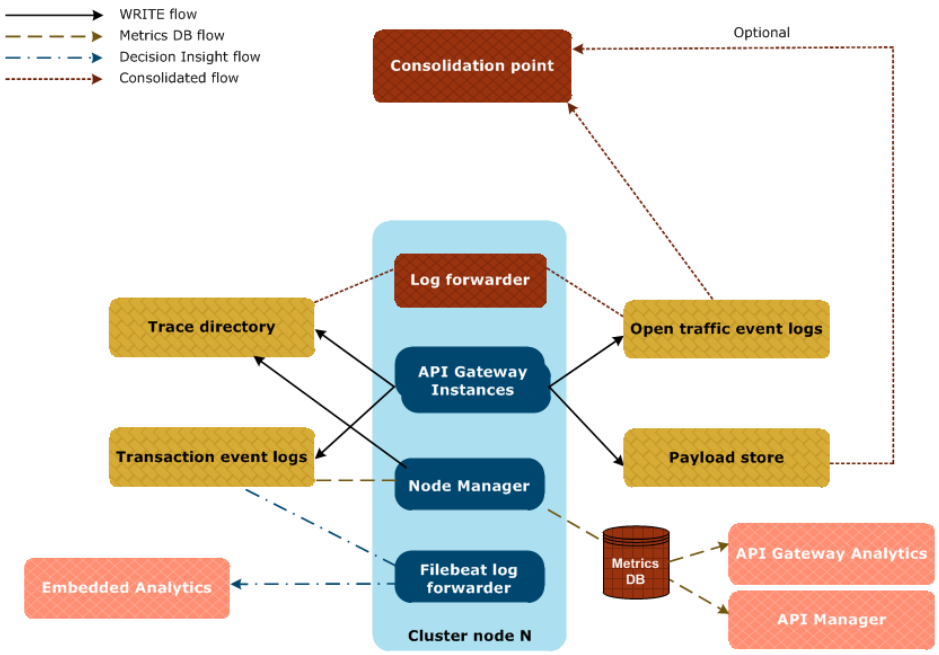
When open logging is enabled, API Gateway generates open traffic event logs containing traffic event data in a JSON format that is easily consumed by external systems.

This topic describes the supported open logging flows, and explains how you can use open logging to forward API Gateway log and trace information to an external system. It also describes the open traffic event log file format and configuration options.

Note Events are logged after a transaction is processed by API Gateway. A catastrophic API Gateway failure might result in one or more processed transactions not being written to the log.

API Gateway logging flows

Some typical logging flows are shown in the following diagram:



Log and trace information stored by API Gateway

Each API Gateway instance writes:

- transaction event information to the transaction event log
- traffic monitoring information to an OpsDB database (not shown in above diagram)
- trace information to trace files
- transaction event and traffic monitoring information to the open traffic event log
- transaction payload information to a payload store (if payload storage is enabled)

The Node Manager also writes trace information to trace files.

Metrics database flow

The Node Manager aggregates the transaction event logs from the API Gateway instances in a domain and writes them to a metrics database for monitoring in API Gateway Analytics, API Manager, or in third-party tools. For more details, see:

- *Configure API Gateway with the metrics database* on page 153
- *API Gateway Analytics User Guide*
- *API Manager User Guide*

Decision Insight flow

A Filebeat log forwarder is used to forward the transaction event logs from the API Gateway instances to Axway Decision Insight for use by Embedded Analytics.

Filebeat is included in your API Gateway installation in the following directory:

```
INSTALL_DIR/apigateway/tools/
```

For more information on configuring Filebeat, see the *Embedded Analytics for AMPLIFY API Management documentation*.

Note In a future release, Embedded Analytics will use the open traffic event log.

Consolidated flow for third-party system

To consolidate the flow of transaction event data, traces, and system metrics stored by API Gateway, you can perform the following steps:

1. Configure open traffic event logging as detailed in *Configure open traffic event logging* on page 175.
2. Set up a log forwarder to read the trace files and the open traffic event log files from API Gateway and forward them to a consolidation point for use by a third-party observability system.

Open traffic event log formats

Open traffic event log files are located in the `logs` directory of your API Gateway installation by default. For example:

```
INSTALL_DIR/apigateway/logs/group-2_instance-1_traffic.log
```

In a distributed system with multiple API Gateway instances running, the data is written to separate open traffic event log files for each API Gateway instance.

The open traffic event log combines the data from:

- Transaction event log – This log contains a summary of transactions in JSON format. For more details on the transaction event log, see *Transaction event log settings* on page 266.
- Traffic monitor – This log contains detailed traffic monitoring information including transaction payloads, policy paths, and so on. This data is stored in a proprietary OpsDB format. For more details on traffic monitor, see *Traffic monitoring settings* on page 279.

The combined data is written to an open traffic event log file in JSON format. The JSON schema is described in *Open logging schema* on page 286 and is available in your installation at `INSTALL_DIR/apigateway/system/schemas/logging/openLogging.json`.

Transaction payloads are stored separately (if payload storage is enabled) and are linked to the open traffic event log files by ID. Payload files are named as follows:

```
<transaction ID>-<Leg>-<sent|received>
```

For more information on transaction and legs, see *Introduction to transactions and legs in API Gateway* on page 39.

The following are some example payload file names:

```
48e59a594900238c49037036-0-received
48e59a594900238c49037036-1-sent
48e59a594900238c49037036-1-received
48e59a594900238c49037036-0-sent
```

Payload files are only generated by open logging if the message includes a payload. For example, for a POST request in a 2-legged flow (single call-out to a back-end), leg 0 is the interaction between the client and the API Gateway and leg 1 is the interaction between the API Gateway and the back-end, giving you:

- `<transaction id>-0-received`: incoming request body from client (leg 0)
- `<transaction id>-1-sent`: outgoing request body to back-end (leg 1)
- `<transaction id>-1-received`: incoming response body from back-end (leg 1)
- `<transaction id>-0-sent`: outgoing response body to client (leg 0)

However, for a GET request with no body in the same situation, only the following payload files are generated:

- `<transaction id>-1-received`: incoming response body from back-end (leg 1)
- `<transaction id>-0-sent`: outgoing response body to client (leg 0)

Configure open traffic event logging

Open traffic event logging is disabled by default. To enable it, perform the following steps:

1. In the Policy Studio tree, select **Server Settings** > **Logging** > **Open Traffic Event Log**.
2. Select **Enable Open Traffic Event Log**.
3. Specify the required settings (for example, directory, max disk space, payload storage, and so on).
4. When finished, click **Save** at the bottom right.
5. Click **Deploy** in the toolbar to deploy your settings to the API Gateway.

For details on configuring all the available options, see *Open traffic event log settings* on page 275.

Environment variables

These variables override the open logging settings configured in Policy Studio, which enables the logging behavior to be defined at runtime.

```
APIGW_LOG_OPENTRAFFIC_OUTPUT=[none | file | stdout]
```

- none = Do not log open traffic data
- file = Log open traffic data to a file
- stdout = Log open traffic data to `stdout`

```
APIGW_LOG_OPENTRAFFIC_PAYLOAD=[true | false]
```

- true = Store copies of message payloads on disk
- false = Do not store copies of message payloads on disk

Best practices

When using the open logging feature, consider the following:

- Enabling the open traffic event log will have a performance impact:
    - We recommend that you test the performance impact before using this feature in a production system.
    - Enabling payload storage will consume additional system resources and slow down your overall TPS.
- Ensure that you have enough disk space available for storing open traffic event log files.
- Ensure that you implement a file retention policy for open traffic event log files, as API Gateway does not delete files automatically.
- If you enable payload storage, we recommend using fast disks (for example, SSD).

Configure API Gateway diagnostic trace

By default, API Gateway produces diagnostic trace and debugging information to record details about its runtime execution. For example, this includes services starting or stopping, exceptions, and messages sent through API Gateway. This information can then be used by API Gateway administrators and developers for diagnostics and debugging purposes, and is useful when contacting Axway Support.

You can view and search the contents of API Gateway tracing in the following locations:

- **Logs > Trace** view in API Gateway Manager
- Trace files in the following locations:
    - Admin Node Manager: `INSTALL_DIR/trace`
    - API Gateway instance: `INSTALL_DIR/groups/<group-id>/<instance-id>/trace`
    - API Gateway Analytics: `INSTALL_DIR/trace`
- A console window for the running server

You can view and search the contents of the API Gateway trace log, domain audit log, and transaction logs in the **Logs** view in API Gateway Manager.

This topic explains how to configure the trace log only. For more details, see *Configure API Gateway logging and events* on page 162.

For details on how to redact sensitive data from trace files (for example, user passwords or credit card details), see *Hide sensitive data in API Gateway Manager* on page 121. The trace level you set impacts the redaction.

View API Gateway trace files

Each time API Gateway starts up, by default, it writes a trace file to the `trace` directory in your API Gateway installation (for example, `INSTALL_DIR/groups/group-2/server1/trace`).

The following example shows an extract from a default API Gateway trace file:

```
INFO    15/Jun/2012:09:54:01.047 [1b10] Realtime monitoring enabled
INFO    15/Jun/2012:09:54:01.060 [1b10] Storing metrics in database disabled
INFO    15/Jun/2012:09:54:03.229 [1b10] cert store configured
INFO    15/Jun/2012:09:54:03.248 [1b10] keypairs configured...
```

The trace file output takes the following format:

```
TraceLevel   Timestamp [thread-id] TraceMessage
```

For example, the first line in the above extract is described as follows:

TraceLevel	INFO
Timestamp	15/Jun/2012:09:54:01.047 (*day*:*hours*:*minutes*:*seconds*:*milliseconds*)
Thread-id	[1b10]
TraceMessage	Realtime monitoring enabled

Set API Gateway trace levels

The possible trace levels in order of least to most verbose output are as follows:

- FATAL
- ERROR
- INFO
- DEBUG
- DATA

FATAL is the least verbose and DATA the most verbose trace level. The default trace level is INFO.

Set the trace level

You can set the trace level using the following different approaches:

Startup trace	When Admin Node Manager is starting up, it gets its trace level from the tracelevel attribute of the SystemSettings element in /system/conf/nodemanager. You can set the trace level in this file if you need to diagnose boot up issues.

Default Settings trace	When API Gateway has started, it reads its trace level from the Default Settings for the API Gateway instance. To set this trace level in the Policy Studio, click the **Server Settings** node in the Policy Studio tree, then click the **General** option, then select a **Tracing level** from the drop-down list.
Dynamic trace	You can also change dynamic API Gateway trace levels on-the-fly in API Gateway Manager. For more details, see *Configure API Gateway logging and events* on page 162.

Configure API Gateway trace files

By default, trace files are named `servername_timestamp.trc` (for example, `server1_20130118160212.trc`). You can configure the settings for trace file output in `INSTALL_DIR/system/conf/trace.xml`, which is included by `INSTALL_DIR/system/conf/nodemanager`. By default, `trace.xml` contains the following setting:

```
<FileRolloverTrace maxfiles="500" filename="%s_%Y%m%d%H%M%S.trc"/>
```

This setting means that API Gateway writes Node Manager trace output to `nodemanager;on`*`hostname_timestamp`* `.trc` (for example, `nodemanager;on127.0.0.1_20130118160212.trc`) in the `trace` directory of the API Gateway installation. And, the maximum number of files that the `trace` directory can contain is `500`.

Configure rollover settings

The `FileRolloverTrace` element can contain the following attributes:

`filename`	File name used for trace output. Defaults to the `tracecomponent` attribute read from the `SystemSettings` element.
`directory`	Directory where the trace file is written. Defaults to `INSTALL_DIR/trace` when not specified. **Note** If you change the trace directory, you will not be able to view the trace files in API Gateway Manager. For the recommended way to change the trace directory, see the following knowledge base article on Axway Support at https://support.axway.com.
`maxlen`	Maximum size of the trace file in bytes before it rolls over to a new file. Defaults to `16777216` (`16` MB).
`maxfiles`	Maximum number of files that the `trace` directory contains for this `filename`. Defaults to `500`.
`rollDaily`	Whether the trace file is rolled at the start of the day. Defaults to `true`.

The following setting shows example attributes:

```
<FileRolloverTrace maxfiles="5" maxlen="10485760" rollDaily="true
    directory="/mydir/log/trace" filename="myserver.trc"/>
```

This setting means that API Gateway writes trace output to `myserver.trc` in the `/mydir/log/trace` directory, and rolls the trace files over at the start of each day. The maximum number of files that this directory can contain is `5`, and the maximum trace file size is 10 MB.

Write output to syslog

On Linux, you can send API Gateway trace output to `syslog`. In your `INSTALL_DIR/system/conf/trace.xml` file, add a `SyslogTrace` element, and specify a `facility`. For example:

```
<SyslogTrace facility="local0"/>
```

Run trace at DEBUG level

When troubleshooting, it can be useful to set to the trace level to `DEBUG` for more verbose output. When running a trace at `DEBUG` level, API Gateway writes the status of every policy and filter that it processes into the trace file.

Debug a filter

The trace output for a specific filter takes the following format:

```
Filter name {
     Trace for the filter is indented to identify output from the filter
} status, in x milliseconds
```

The status is `0`, `1`, or `2`, depending if the filter failed, succeeded, or aborted. For example, the result of an **WS-Security Username Token** filter running successfully is as follows:

```
DEBUG 12:43:59:093 [11a4] run filter [WS-Security Username Token] {
DEBUG 12:43:59:093 [11a4]   WsUsernameTokenFilter.invoke:Verify username and
password
DEBUG 12:43:59:093 [11a4]   WsAuthN.getWSUsernameTokenDetails:
                            Get token from actor=current actor
DEBUG 12:43:59:093 [11a4]   Version handler -  creating a new ws block
DEBUG 12:43:59:108 [11a4]   Version handler - adding the ws element to the
wsnodelist
```

```
DEBUG 12:43:59:108 [11a4]   Version handler -  number of ws blocks found:1
DEBUG 12:43:59:124 [11a4]   No timestamp passed in WS block, no need to check
                            timestamp
DEBUG 12:43:59:139 [11a4]   WsAuthN.getWSUsernameTokenDetails: Check <Created>
element
                            in token. Value=2010-08-06T11:43:43Z
DEBUG 12:43:59:139 [11a4]   WS Nonce TimeStamp Max Size is 1000 and wsNonces cache 4
DEBUG 12:43:59:139 [11a4]   Add WS nonce for key [joe:2010-08-06T11:43:43Z].
                            New cache size [5].
DEBUG 12:43:59:155 [11a4]   WsBasicAuthN.getUsername:Getting username
DEBUG 12:43:59:171 [11a4]   WS-Security UsernameToken authN via CLEAR password
DEBUG 12:43:59:171 [11a4]   VordelRepository.checkCredentials:username=joe
DEBUG 12:43:59:186 [11a4] } = 1, in 62 milliseconds
```

Debug a policy

The trace output for a policy shows it running with all its contained filters, and takes the following format:

```
policy Name {
     Filter 1 {
         Trace for the filter
     } status, in x milliseconds
     Filter 2 {
        Trace for the filter
    } status, in x milliseconds
}
```

For example, the following extract shows a policy called when running a simple service:

```
DEBUG ... run circuit "/axis/services/urn:cominfo"...
DEBUG ... run filter [Service Handler for 'ComInfoServiceService'] {
DEBUG ...   Set the service name to be ComInfoServiceService
DEBUG ...   Web Service context already set to ComInfoServiceService
DEBUG ...   close content stream
DEBUG ...   Calling the Operation Processor Chain [1. Request from Client]...
DEBUG ...   run filter [1. Request from Client] {
DEBUG ...      run filter [Before Operation-specific Policy] {
DEBUG ...         run circuit "WS-Security UsernameToken AuthN"...
DEBUG ...           run filter [WS-Security Username Token] {...
DEBUG ...           } = 1, in 62 milliseconds
DEBUG ...           ..."WS-Security UsernameToken AuthN" complete.
DEBUG ...       } = 1, in 74 milliseconds  ...
```

Debug at startup

When running a startup trace with a `DEBUG` level set in the `SystemSettings`, API Gateway logs the configuration that it is loading. This can often help to debug any incorrectly configured items at start up, for example:

```
DEBUG 14:38:54:206 [1ee0] configure loadable module type RemoteHost, load order =
500000
DEBUG 14:38:54:206 [1ee0] RemoteHost {
DEBUG 14:38:54:206 [1ee0]     ESPK:1035
DEBUG 14:38:54:206 [1ee0]     ParentPK:113
DEBUG 14:38:54:206 [1ee0]     Key Fields:
DEBUG 14:38:54:206 [1ee0]         name:{csdwmp3308.wellsfargo.com}
DEBUG 14:38:54:206 [1ee0]         port:{7010}
DEBUG 14:38:54:221 [1ee0]     Fields:
DEBUG 14:38:54:221 [1ee0]         maxConnections:{128}
DEBUG 14:38:54:268 [1ee0]         turnMode:{off}
DEBUG 14:38:54:268 [1ee0]         inputBufSize:{8192}
DEBUG 14:38:54:268 [1ee0]         includeContentLengthRequest:{0}
DEBUG 14:38:54:268 [1ee0]         idletimeout:{15000}
DEBUG 14:38:54:268 [1ee0]         activetimeout:{30000}
DEBUG 14:38:54:268 [1ee0]         forceHTTP10:{0}
DEBUG 14:38:54:268 [1ee0]         turnProtocol:{http}
DEBUG 14:38:54:268 [1ee0]         includeContentLengthResponse:{0}
DEBUG 14:38:54:268 [1ee0]         addressCacheTime:{300000}
DEBUG 14:38:54:268 [1ee0]         outputBufSize:{8192}
DEBUG 14:38:54:268 [1ee0]         sessionCacheSize:{32}
DEBUG 14:38:54:268 [1ee0]         _version:{1}
DEBUG 14:38:54:268 [1ee0]         loadorder:{500000}
DEBUG 14:38:54:268 [1ee0]         class:{com.vordel.dwe.NativeModule}
DEBUG 14:38:54:268 [1ee0] }
```

For details on setting trace levels, and running a startup trace, see *Set API Gateway trace levels* on page 177.

Run trace at DATA level

When the trace level is set to `DATA`, API Gateway writes the contents of the messages that it receives and sends to the trace file. This enables you to see what messages API Gateway has received and sent (for example, to reassemble a received or sent message).

Note When the trace level is set to `DATA`, passwords provided during login are logged in plain text.

Search by thread ID

Every HTTP request handled by API Gateway is processed in its own thread, and threads can be reused when an HTTP transaction is complete. You can see what has happened to a message in API Gateway by following the trace by thread ID. Because multiple messages can be processed by API Gateway at the same time, it is useful to eliminate threads that you are not interested in by searching for items that only match the thread you want.

You can do this using the search feature in the API Gateway Manager **Logs** view. Enter the thread you wish to search for in the **Only show lines with text** text box, and click **Refresh**. Alternatively, you can do this on the command line using vi by specifying the thread ID as a pattern to search for in the trace file. In this example, the thread ID is `145c`:

```
:g!/145c/d
```

The following example shows the `DATA` trace when a message is sent by API Gateway (message starts with `snd`):

```
DATA 17:45:35:718 [145c]  snd 1495:
    <POST /axis/services/urn:cominfo HTTP/1.1Connection:closeContent-Length:
    1295User-Agent:VordelSOAPAction:""Via:1.0 devsupport2 (Vordel)
    Host:devsupport2:7070Content-Type:text/xml<?xml version="1.0"
    encoding="UTF-8" standalone="no"?>
    <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <soap:Header>
       <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/
          oasis-200401-wss-wssecurity-secext-1.0.xsd">
          <wsse:UsernameToken xmlns:wsu="http://docs.oasis-open.org/wss/
            2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
              wsu:Id="Id-00000128d05aca81-00000000009d04dc-10">
              <wsse:Username>joeuser</wsse:Username>
              <wsse:Nonce EncodingType="utf-8">
                  gmP9GCjoe+YIuK1einlENA==
               </wsse:Nonce>
              <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/
                 oasis-200401-wss-username-token-profile-1.0#PasswordText">
                  joepwd
              </wsse:Password>
              <wsu:Created>2010-05-25T16:45:30Z</wsu:Created>
          </wsse:UsernameToken>
       </wsse:Security>
      </soap:Header>
      <soap:Body>
         <ns1:getInfo xmlns:ns1="http://stock.samples">
              <symbol xsi:type="xsd:string">CSCO</symbol>
              <info xsi:type="xsd:string">address</info>
          </ns1:getInfo>
      </soap:Body>
```

```
    </soap:Envelope>
```

The following example shows the `DATA` trace when a message is received by API Gateway (message starts with `rcv`):

```
DATA 17:45:35:734 [145c]  rcv 557:<HTTP/1.0 200 OKSet-Cookie:8Set-Cookie2:
    8Content-Type:text/xml; charset=utf-8<?xml version="1.0" encoding="UTF-8"?>
    <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
       <soapenv:Body>
         <ns1:getInfoResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/
             soap/encoding/" xmlns:ns1="http://stock.samples">
             <getInfoReturn xsi:type="xsd:string">
               San Jose, CA
             </getInfoReturn>
         </ns1:getInfoResponse>
       </soapenv:Body>
    </soapenv:Envelope>
```

If you want to see what has been received by API Gateway on this thread, run the following command:

```
:g!/145c] rcv/d
```

All `snd` and `rcv` trace statements start and end with < and > respectively. If you are assembling a message by hand from the `DATA` trace, remember to remove characters. In addition, the sending and/or receiving of a single message may span multiple trace statements.

Integrate trace output with Apache log4J

Apache log4j is included on API Gateway classpath. This is because some third-party products that API Gateway interoperates with require log4j. The configuration for log4j is found in API Gateway `INSTALL_DIR/system/conf` directory in the `log4j2.xml` file.

For example, to specify that the log4j appender sends output to API Gateway trace file, add the following setting to your `log4j2.xml` file:

```
<Root level="debug">
<AppenderRef ref="STDOUT" />
<AppenderRef ref="VordelTrace" />
</Root>
```

Environment variables

These variables override the `trace.xml` file settings, which enables the logging behavior to be defined at runtime.

```
APIGW_LOG_TRACE_TO_FILE=[true | false]
```

- true = Write trace files to disk
- false = Do not write trace files to disk

```
APIGW_LOG_TRACE_JSON_TO_STDOUT=[true | false]
```

- true = Output JSON formatted trace to `stdout`
- false = Do not output JSON formatted trace to `stdout`

API Gateway performance tuning

This topic explains how to optimize API Gateway performance using various configuration options. For example, general performance tuning options include tracing, monitoring, and logging. More advanced performance tuning options include database pooling, HTTP keep alive, chunked encoding, and client threads.

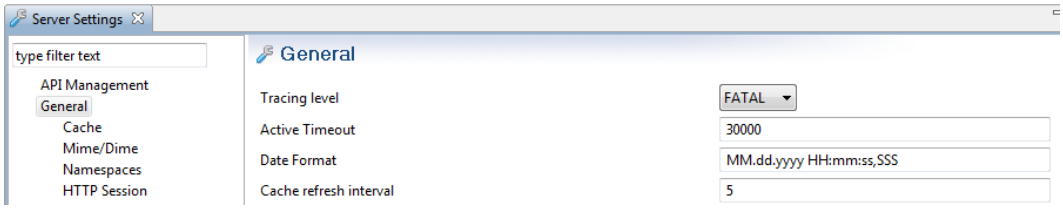
General performance tuning

You can optimize API Gateway performance by using Policy Studio to configure the general settings described in this section.

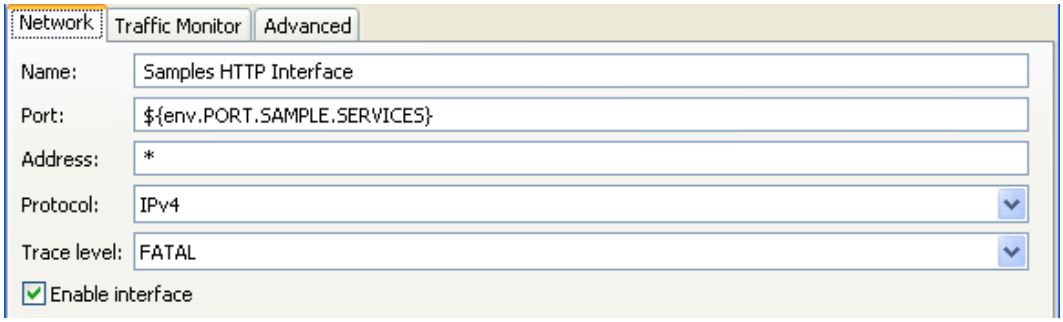
Minimize tracing

The **Trace Log** is displayed in the **Logs** view in the API Gateway Manager web console. When tracing is running at a verbose level (for example, `DEBUG`), this means that API Gateway is doing more work and is very dependent on disk input/output. You can set a less verbose trace level for an API Gateway instance or API Gateway port interface (for example, `ERROR` or `FATAL`).

To set the tracing for an API Gateway instance, select **Environment Configuration > Server Settings > General** in the Policy Studio tree, and select the **Trace Level** (for example, `FATAL`):



You can also override the trace level for an API Gateway port interface, and set it to a quieter level. For example, in the Policy Studio tree, select **Environment Configuration > Listeners > API Gateway > Sample Services > Ports**. Right-click an interface in the list on the right, select **Edit**, and set the **Trace level** (for example, `FATAL`):

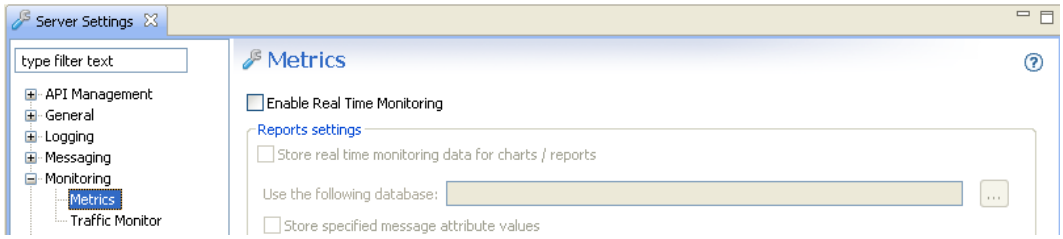


For more details, see *Configure API Gateway diagnostic trace* on page 176.

Disable real-time monitoring

Real-time monitoring is displayed in the **Monitoring** view in the API Gateway Manager web console. This caches recent message transactions in the memory of API Gateway. You can remove this overhead by disabling real-time monitoring.

To disable in Policy Studio, select **Environment Configuration > Server Settings > Monitoring > Real Time Monitoring**, and deselect **Enable Real Time Monitoring**:

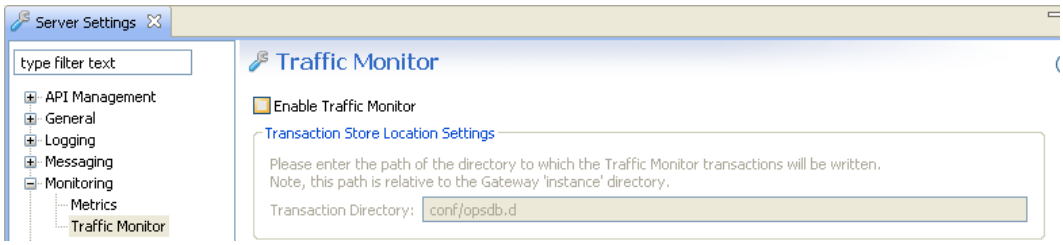


For more details, see *Real-time monitoring metrics* on page 281.

Disable traffic monitoring

Traffic monitoring is displayed in the **Traffic** view in the API Gateway Manager web console. By default, the API Gateway stores recent HTTP traffic summaries to the API Gateway disk for use in API Gateway Manager. You can remove this overhead by disabling traffic monitoring.

To disable in Policy Studio, select **Environment Configuration > Server Settings > Monitoring > Traffic**, and deselect **Enable Traffic Monitor**:

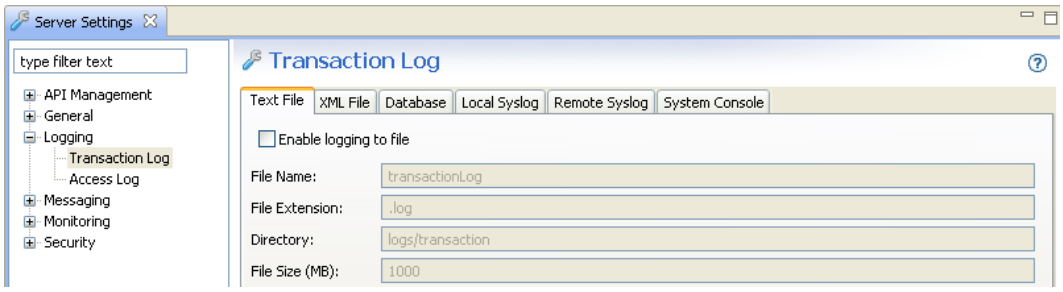


For more details, see *Traffic monitoring settings* on page 279.

Disable transaction logging

The **Transaction Log** is displayed in the **Logs** view in the API Gateway Manager web console. You should ensure that API Gateway is not sending transaction log messages or events to transaction log destinations. This is because the performance of API Gateway will be determined by the log destination.

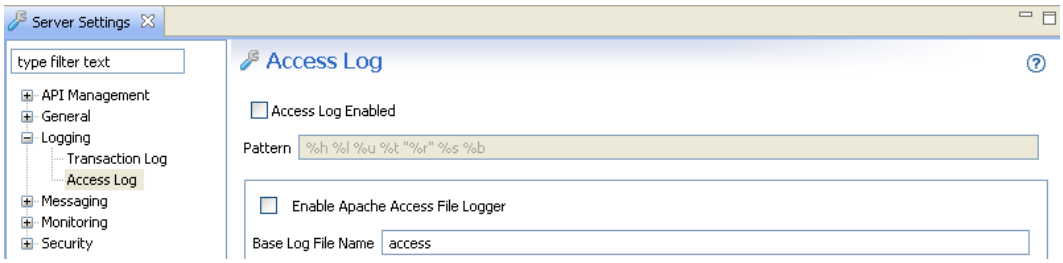
To disable transaction logging in the API Gateway, you must disable all log destinations in Policy Studio. For example, select **Environment Configuration > Server Settings > Logging > Transaction Log**, and deselect **Enable logging to a file**. The following example shows disabling logging to file, you must perform this step in all tabs on this screen:



For more details, see *Transaction audit log settings* on page 258.

Disable access logging

You should also ensure that the API Gateway is not sending log messages to the access log. To disable access logging in the API Gateway, select **Environment Configuration > Server Settings > Logging > Transaction Access Log**, and deselect **Transaction Access Log Enabled**:



For more details, see *Transaction access log settings* on page 262.

Advanced performance tuning

You can also use the advanced configuration settings described in this section to optimize API Gateway performance.

Configure spilling of data to disk

When stress testing with large messages (greater than 4 MB), the API Gateway spills data to disk instead of holding it in memory. By default, the `spilltodisk` option is triggered with payload sizes of 4 MB or more. For example, you can configure this in the `service.xml` file in the following directory by adding the `spilltodisk` option configured in bytes:

```
<install-dir>/apigateway/groups/<group-id>/<instance-id>/conf/service.xml
```

For example:

```
<NetService provider="NetService">
    <SystemSettings
            tracelevel="&server.tracelevel;"
            tracecomponent="&server.title;"
            title="&server.title;"
            homedir="$VINSTDIR"
            secret="&server.entitystore.secret;"
            servicename="&server.servicename;
            "spilltodisk="10485760"
             ...
```

This setting specifies the limit of what is considered a reasonably-sized incoming request message to hold in memory. After this limit is exceeded, to preserve memory, the system writes the content of the incoming request to disk when it arrives.

For example, if API Gateway receives a 200 MB message from an HTTP/1.1 server to forward to an HTTP/1.0 client, it may need to calculate the content length of that message before transmitting the first byte to the client. API Gateway can do this by holding it in memory, or storing it to disk. After a certain point, the benefits of holding it in memory are outweighed by the cost of the memory footprint, and API Gateway stores it on disk.

When the message is entirely received, API Gateway knows the content length, and can generate it before reading the data from the disk block-by-block, and forwarding to the client. The entire 200 MB does not need to reside in API Gateway at the same time.

Note The `spilltodisk` option is purely used to reduce pressure on virtual address space, and does not impact on monitoring or metrics. This option applies to HTTP and SMTP only.

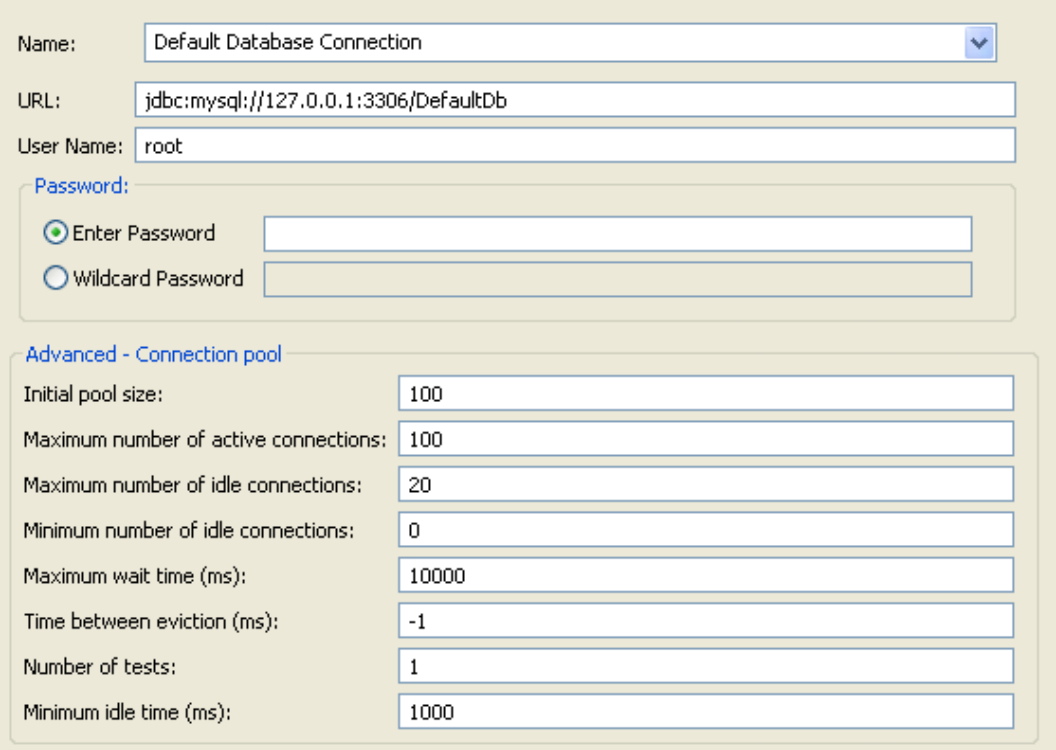
Configure database pooling

The API Gateway uses Apache Commons Database Connection Pools (DBCP) for pooling database connections. For details, see http://commons.apache.org/dbcp/.

Note Axway recommends that if your policy interacts heavily with the database, the pool size for the database connection should be at least as big as the expected client population. This assumes the database can cope with this number of parallel connections.

For example, if you are providing load from 100 parallel clients, the pool settings shown in the following example are recommended.

To configure database pooling in Policy Studio, select **Environment Configuration > External Connections > Database Connections > Add Database Connection**. For example:



Configure API Gateway for HTTP persistent connections

By default, API Gateway uses HTTP 1.0 for better interoperability. To enable HTTP persistent connections (`HTTP Keep-alive`), you must configure API Gateway to use HTTP 1.1.

In HTTP/1.1, the connection between a client and a server is maintained unless otherwise declared, so that further client requests can avoid the overhead of setting up a new connection. This may or may not model the client population of a particular scenario very well. If it is acceptable to reuse TCP connections (and SSL connections on top of these), ensure your client uses HTTP/1.1, and does not opt out of the HTTP persistent connection.

For the `sr` command, this means you should use the `-V1.1` and `-U1000` arguments to enable the connection be used a number of times before closing it. For details on `sr`, see the *API Gateway Policy Developer Guide*.

Note For conformance with the HTTP/1.1 specification, the client must send a `Host` header in this configuration, so you must pass a further `-aHost:localhost` argument to `sr`. If the persistent connection is working correctly, `sr` reports a larger number of transactions to connections in its periodic output.

Configure HTTP 1.1 for outgoing connections

You can configure a remote host for a destination server supporting HTTP 1.1.

In the Policy Studio node tree, click **Environment Configuration > Listeners > API Gateway**. Select the remote host you want and click **Edit**, or add a new host, and select **Allow HTTP 1.1**.

Remote host only enforces settings to a specific configured endpoint, not globally. You must edit the settings separately for each destination server you want to configure for HTTP 1.1.

Configure HTTP 1.1 for incoming connections

You can enable HTTP 1.1 globally for incoming connections in API Gateway by editing the following file:

```
INSTALL_DIR/apigateway/groups/<group>/<instance>/conf/service.xml
```

To change the default behavior of the HTTP 1.1 settings, set `allowHTTP11` to `true`:

```
<?xml version="1.0" encoding="UTF-8"?>
<NetService provider="NetService">
<!-- Configuration file for service. Note that if you wish for the user to enter a
passphrase at
start up then give the "secret" attribute a value "(prompt)", for example: secret="
(prompt)" -->
<include file="serviceids.xml"/>
<include file="../../conf/group.xml"/>
<SystemSettings tracelevel="INFO" secret="${secret}"
        serviceID="${serviceID}" groupID="${groupID}"
        serviceName="${serviceName}"
        groupName="${groupName}"
        domainID="${domainID}" title="API Server"
        allowHTTP11="true"/>
<set property="headless" value="true"/>
<include file="$VDISTDIR/system/conf/platform.xml"/>
<include file="$VDISTDIR/system/conf/trace.xml"/>
<include file="$VDISTDIR/system/conf/libxml.xml"/>
<include file="$VDISTDIR/system/conf/jvm.xml"/>
<include file="$VDISTDIR/system/conf/nativeJAXP.xml"/>
```

```
<include file="esconnection.xml"/>
<include file="mgmt.xml"/>
<includes dir="extensions" pattern="*.xml"/>
</NetService>
```

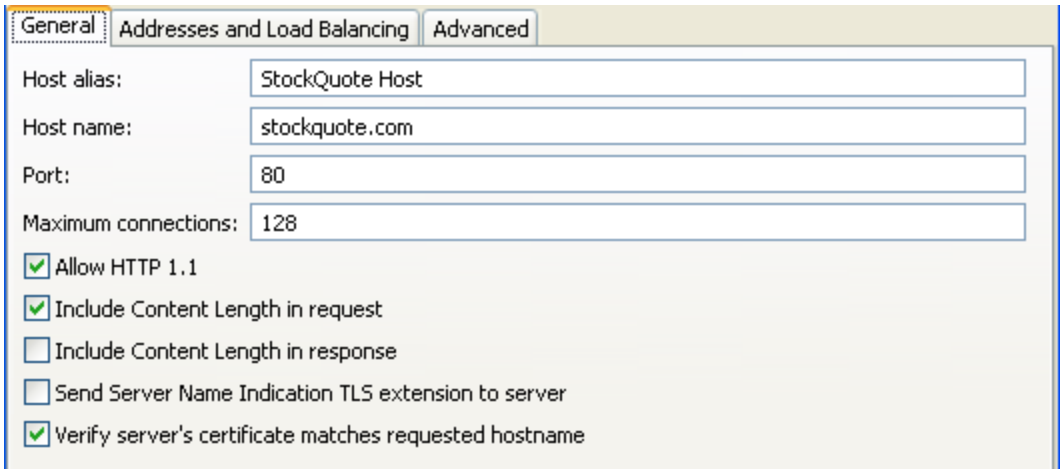
Changing this setting enables HTTP 1.1 globally for this particular API Gateway instance. You must restart the instance before the setting takes effect.

Configure chunked encoding

For interoperability reasons, API Gateway normally does not use chunked encoding when talking to a remote server. Because of the HTTP protocol, API Gateway must send the `Content-Length` header to the server, and so must precompute the exact size of the content to be transmitted. This may be expensive.

For example, when relaying data directly from client to server, or when the message exists as an abstract XML document in API Gateway, the size may not be immediately available. This means that the entire content from the client must be buffered, or the internal body structure must be serialized an extra time just to measure its size. By configuring a remote host for the destination server to allow HTTP 1.1, when a server is known to be advertising HTTP 1.1, chunked encoding can be used when transmitting to that server where appropriate.

For example, in the Policy Studio tree, select **Environment Configuration > Listeners > API Gateway**. Select the remote host Right-click the remote host, select **Edit**, and select **Allow HTTP 1.1**:



Number of threads for concurrent connections

You may want to consider reducing the number of threads that API Gateway uses for processing incoming messages. Reducing the thread count may result in less resources being consumed. API Gateway handles less load because of the decreased thread count, depending on volume and number of concurrent requests.

For example, you can reduce the number of threads by adding a `maxThreads="64"` attribute in the `SystemSettings` element in the `service.xml` file for the API Gateway instance (for example, in `groups/group-2/conf/instance-1`). This setting also helps in configuring API Gateway to back off from the target service (if API Gateway can process more load than the target service).

By default, the maximum thread count for an API Gateway instance on Linux is 1024.

Note You must restart API Gateway for these settings to take effect.

Number of client threads on Linux

If there are more than 200 client threads connecting, you must change the following setting in the `venv` script in the `posix/lib` directory of your API Gateway installation:

```
limit=unlimited
```

For example, change this setting to the following:

```
limit=131072
```

This prevents the too many open files error that you might see in a 200 client thread test.

Multiple connection filters

This applies if the performance test involves more than one connection filter, where the filter is routing to the same host and port with the same SSL credentials. You should create a policy containing the single connection filter, and delegate to this policy from where you normally do the routing, so you delegate to a single filter instead of a connection filter per policy.

Each connection processor caches connections independently. This is because two connection processors using different SSL certificates cannot pool their connections. They are not interchangeable from an authentication point of view. Therefore, when using multiple connection filters, there is potential to soak the target machine with too many connections.

Note This applies to both API Gateway **Connection** and **Connect to URL** filters in Policy Studio. For more details, see *API Gateway Policy Developer Filter Reference*.

Get help with API Gateway

This topic describes how you can access help and documentation for API Gateway, how to find your installed version (including any installed service pack and patches), and what information you need if you contact Axway Support for assistance.

Access help and documentation

Context-sensitive help is available in Policy Studio. Click the **Help** button on any window to display the relevant help page for that window.

You can access all of the API Gateway documentation on the Axway Documentation portal at https://docs.axway.com.

Find your installed version

You can find your installed version of API Gateway in several ways.

Policy Studio

To view the installed version number in Policy Studio, select **Help > About Policy Studio**.

API Gateway Manager

You can view the installed version number, including any installed service pack, for API Gateway instances in API Gateway Manager. For more information, see *Check the installed version number of API Gateway instances* on page 45.

Process listing

You can list the API Gateway processes to view the installed version number, including any installed service pack. For example:

```
$ ps -eaf | grep vshell
user1 13696 13687  0 Nov14 pts/24   00:11:50 APIGateway1 (Group1) (7.6.2 SP1)
(vshell)
user1 19595 13643  0 Nov14 pts/23   00:06:05 NodeManager on Host1 (Node Manager
Group) (7.6.2 SP1) (vshell)
```

Trace file

You can find the installed version number, including any installed service pack, in the header of the trace files for the Node Manager, API Gateway instance, or API Gateway Analytics server. For more information on where to find the trace files, see *Configure API Gateway diagnostic trace* on page 176.

For example, the following trace file (for a Node Manager) shows the installed version and service pack:

```
# ProductName=nodemanageronhost1.axway.com 7.6.2 SP1-2016-12-01 rev. 39e69a0
(Linux.x86_64)
# CurrentDate=Fri, 02 Dec 2016 09:26:14 +0000
# CurrentDateUTC=1480670774
# TZ=GMT
```

managedomain

For more information on using `managedomain` to find your installed version number, see *Find your installed version and list patches using managedomain* on page 193.

Find your installed version and list patches using managedomain

You can use the `managedomain` command with the `-v` or `--version` options to find the installed version number, including any installed service pack or patches, and build information.

The following example shows the output for a system with API Gateway 7.6.2 installed, with service pack `SP1` installed, and no patches installed:

```
$ ./managedomain --version
Version:    7.6.2 SP1
Build Date: 2016-11-04 08:34:37 UTC
Commit Id:  05a6440fc3128d8d2f3dfe105904d04fffae67ea
Patch:      None
```

The `Build Date` and `Commit Id` fields in the output relate to the main product version (or the service pack, if one is installed). These fields are updated when a service pack is installed, but they are not updated when a patch is installed.

When patches are installed, `managedomain --version` displays the names of the patches installed. Patches are named after the internal ticket number and related commit ID (for example, `RDAPI-4563_3y7shb87635bhsaf7864nckzj7r9mp8744n8asa`). It also validates the installed patches and displays messages about any problems with the patch installation.

The following example shows the output for a system with patches installed and no patch validation issues:

```
$ ./managedomain --version
Version:    7.6.2
Build Date: 2016-12-06 14:56:44 UTC
Commit Id:  ba35aba7bf84a165dda0df81470f4e9a87d73778
Patch:      OpenSSL_1_0_2j-fips
Patch:      RDAPI-5885_44dc0cde1d94f4dd4744327728aeeac5b2c4a802

$ ./managedomain --version
```

```
Version:    7.5.3
Build Date: 2016-12-06 14:56:44 UTC
Commit Id:  ba35aba7bf84a165dda0df81470f4e9a87d73778
Patch:      RDAPI-5693_etwe2346zdg567fhfg4ue7645846856766700a32
Patch:      RDAPI-5885_44dc0cde1d94f4dd4744327728aeeac5b2c4a802
```

The following example shows the output for a system with patches installed and various patch validation issues:

```
$ ./managedomain --version
Version:    7.5.3
Build Date: 2016-12-06 14:56:44 UTC
Commit Id:  ba35aba7bf84a165dda0df81470f4e9a87d73778
Patch:      RDAPI-4563_3y7shb87635bhsaf7864nckzj7r9mp8744n8asa
               Info: ./Linux.x86_64/jre: Cannot validate, no checksum available
Patch:      RDAPI-1783_mnc785jnsi84sni6a65l09vxgm93kmi78zd985q
               Warning: ./Linux.x86_64/lib/libssl.so.1.0.0:        Content changed
               Warning: ./Linux.x86_64/lib/libcrypto.so.1.0.0:     Content changed
               Warning: ./Linux.x86_64/lib/engines/libgost.so:     Content changed
               Warning: ./Linux.x86_64/lib/engines/libatalla.so:   Content changed
               Warning: ./Linux.x86_64/lib/engines/lib4758cca.so:  Content changed
               Warning: ./Linux.x86_64/lib/engines/libchil.so:     Content changed
               Warning: ./Linux.x86_64/lib/engines/libcswift.so:   Content changed
               Warning: ./Linux.x86_64/lib/engines/libaep.so:      Content changed
               Warning: ./Linux.x86_64/lib/engines/libubsec.so:    Content changed
               Warning: ./Linux.x86_64/lib/engines/libsureware.so: Content changed
Patch:      RDAPI-5885_44dc0cde1d94f4dd4744327728aeeac5b2c4a802
               Error: ext/lib/com.vordel.circuit.net.QaReflectProcessor.jar: File
not found
               Error: ext/lib/com.vordel.circuit.net.ReflectProcessor.jar:   File
not found
Patch:      RDAPI-5693_etwe2346zdg567fhfg4ue7645846856766700a32
               Error: META-INF/RDAPI-6666_ba35aba.id: Malformed file
Patch:      ext/lib/com.vordel.circuit.net.UnexpectedProcessor.jar: Unexpected file
```

For more information on the patch validation messages and how to resolve them, see "Update API Gateway" in the *API Gateway Installation Guide*.

The `managedomain --version` command checks the version on the local machine only. You do not need to have an Admin Node Manager or API Gateway running to run this command.

This command lists version information relating to what is installed on disk. You must stop your Node Manager and API Gateways before installing service packs and patches, you must restart them after installation, or the output of this command might not reflect what is loaded for the runtime of the Node Manager and API Gateways.

This command can also be run in command interpreter mode.

Information to include when you contact Axway Support

It is important to include as much information as possible when contacting the Axway Support team. This helps to diagnose and solve the problem in a more efficient manner. The following information should be included with any Support query:

- Name and version of the product (for example, AxwayAPI Gateway7.6.2).
- Details of any service pack or patches that were applied to the product, if any.
- Operating system on which the product is running.
- A clear (step-by-step) description of the problem or query.
- If you have encountered an error, the error message should be included in the email. It is also useful to include any relevant trace files from the `/trace` directory of your product installation, preferably with the trace level set to `DEBUG`.

Manage user access 8

This part contains the following:

Manage API Gateway users

By default, the API Gateway user store contains the configuration data for managing API Gateway user information. The API Gateway user store is typically used in a development environment, and is useful for demonstration purposes.

In a production environment, user information may be stored in existing user Identity Management repositories such as Microsoft Active Directory, Oracle Access Manager, CA SiteMinder, and so on. For more details, see the relevant *API Gateway Integration Guide*.

Note API Gateway users provide access to the messages and services protected by API Gateway. However, *admin users* provide access to the API Gateway configuration management features available in Policy Studio, Configuration Studio, and API Gateway Manager.

API Gateway users

API Gateway users specify the user identity in the API Gateway user store. This includes details such as the user name, password, and X.509 certificate. API Gateway users must be a member of at least one user group. In addition, users can specify optional attributes, and inherit attributes at the group level.

To view all existing users, select the **Environment Configuration > Users and Groups > Users** node in the tree. The users are listed in the table on the main panel. You can find a specific user by entering a search string in the **Filter** field.

Add API Gateway users

You can create API Gateway users on the **Users** page. Click the **Add** button on the right.

To specify the new user details, complete the following fields on the **General** tab:

- **User Name**:
  Enter a name for the new user.

- **Password**:
  Enter a password for the new user.
- **Confirm Password**:
  Re-enter the user's password to confirm.
- **Signing Key**:
  Click to load the user certificate from the **Certificate Store**. For details on how to create and import certificates, see *Manage X.509 certificates and keys* on page 106.

You can also specify optional user attributes on the **Attributes** tab, which is explained in the next section.

API Gateway user attributes

You can specify attributes at the user level and at the group level on the **Attributes** tab. Attributes specify user configuration data (for example, attributes used to generate SAML attribute assertions).

The **Attributes** tab enables you to configure user attributes as simple name-value pairs. The following are examples of user attributes:

- `role=admin`
- `email=steve@axway.com`
- `dept=eng`
- `company=axway`

You can add user attributes by clicking the **Add** button. Enter the attribute name, type, and value in the fields provided. The `Encrypted` type refers to a string value that is encrypted using a well-known encryption algorithm or cipher.

API Gateway user groups

API Gateway user groups are containers that encapsulate one or more users. You can specify attributes at the group level, which are inherited by all group members. If a user is a member of more than one group, that user inherits attributes from all groups (the superset of attributes across the groups of which the user is a member).

To view all existing groups, select the **Environment Configuration > Users and Groups > Groups** node in the tree. The user groups are listed in the table on the main panel. You can find a specific group by entering a search string the **Filter** field.

Add API Gateway user groups

You can create user groups on the **Groups** page. Click the **Add** button on the right to view the **Add Group** dialog.

To specify the new group details, complete the following fields on the **General** tab:

- **Group Name**:
  Enter a name for the new group.
- **Members**:
  Click the **Add** button to display the **Add Group Member** dialog, and select the members to add to the group.

You can also specify optional attributes at the group level on the **Attributes** tab. For more details, see *API Gateway user attributes* on page 197.

Update API Gateway users or groups

To edit details for a specific user or group, select it in the list, and click the **Edit** button on the right. Enter the updated details in the **Edit User** or **Edit Group** dialog.

To delete a specific user or group, select it in the list, and click the **Remove** button on the right. Alternatively, to delete all users or Groups, click the **Remove All** button. You are prompted to confirm all deletions.

Manage admin users

When logging into the Policy Studio or API Gateway Manager, you must enter the user credentials stored in the local admin user store to connect to the API Gateway server instance. Admin users are responsible for managing API Gateway instances using the API Gateway management APIs. To manage admin users, click the **Settings** > **Admin Users** tab in the API Gateway Manager.

Note Admin users provide access to the API Gateway configuration management features available in the Policy Studio and API Gateway Manager. However, *API Gateway users* provide access to the messages and services protected by the API Gateway. For more details, see *Manage API Gateway users* on page 196.

Admin user privileges

After installation, a single admin user is defined in the API Gateway Manager with a user name of `admin`. Admin user rights in the system include the following:

- Add another admin user
- Delete another admin user
- Update an admin user
- Reset admin user passwords

Note An admin user *cannot* delete itself.

Remove the default admin user

If you need to remove the default admin user, perform the following steps:

1. Add another admin user.
2. Log in as the new admin user.
3. Delete the default admin user.

The **Admin Users** tab displays all existing admin users. You can use this tab to add, update, and delete admin users. These tasks are explained in the sections that follow.

Admin user roles

The API Gateway uses Role-Based Access Control (RBAC) to restrict access to authorized users based on their assigned roles in a domain. Using this model, permissions to perform specific system operations are assigned to specific roles only. This simplifies system administration because users do not need to be assigned permissions directly, but instead acquire them through their assigned roles.

For example, the default admin user (`admin`) has the following user roles:

- `Policy Developer`
- `API Server Administrator`
- `KPS Administrator`

API Gateway user roles and privileges

User roles have specific tools and privileges assigned to them. These define who can use which tools to perform what tasks. The user roles provided with the API Gateway assign the following privileges to admin users with these roles:

Role	Tool	Privileges
`API Server Administrator`	API Gateway Manager	Read/write access to API Gateway Manager.
`API Server Operator`	API Gateway Manager	Read-only access to API Gateway Manager.
`Deployer`	Deployment scripts	Deploy a new configuration.
`KPS Administrator`	API Gateway Manager	Perform create, read, update, delete (CRUD) operations on data in a Key Property Store (KPS).
`Policy Developer`	Policy Studio	Download, edit, deploy, version, and tag a configuration.

Note A single admin user typically has multiple roles. For example, in a development environment, a policy developer admin user would typically have the following roles:

- `Policy Developer`
- `API Server Administrator`

Add a new admin user

Complete the following steps to add a new admin user to the system:

1. Click the **Settings** > **Admin Users** tab in API Gateway Manager.
2. Click the **Create** button.
3. In the **Create New Admin User** dialog, enter a name for the user in the **Username** field.
4. Enter a user password in the **Password** field.
5. Re-enter the user password in the **Confirm Password** field.
6. Select roles for the user from the list of available roles (for example, `Policy Developer` and `API Server Administrator`).
7. Click **Create**.

Remove an admin user

To remove an admin user, select it in the **Username** list, and click **Delete**. The admin user is removed from the list and from the local admin user store.

Reset an admin user password

You can reset an admin user password as follows:

1. Select the admin user in the **Username** list.
2. Click the **Edit** button.
3. Enter and confirm the new password in the **Password** and **Confirm Password** fields.
4. Click **OK**.

Manage admin user roles

You can manage the roles that are assigned to specific admin users as follows:

1. Select the admin user in the **Username** list.
2. Click the **Edit** button.

3. Select the user roles to enable for this admin user in the dialog (for example, `Policy Developer` and/or `API Server Administrator`).
4. Click **OK**.

Edit API Gateway user roles

To add or delete specific API Gateway roles, you must edit the available roles in the `adminUsers.json` and `acl.json` files in the `conf` directory of your API Gateway installation.

For more details, see *Configure Role-Based Access Control (RBAC)* on page 202.

Configure a password policy for admin users

To configure the password policy that applies to admin user passwords, perform the following steps:

1. Click the **Settings** > **Admin Users** tab in API Gateway Manager.
2. Select **Password Policy enabled** to enable the password policy rules on this page. This is not selected by default.
3. Configure the following in **PASSWORD RULES**:
    - **Password must not be equal to the account name**: The password cannot be identical to the admin user name. This is selected by default.
    - **Password must not be the reverse of the account name**: The password cannot be the reverse of the admin user name. This is selected by default.
    - **Password must not contain the account name**: The password cannot contain the admin user name. This is selected by default.
    - **Minimum password length**: The password must be the specified minimum length. Defaults to 4 characters. If no value is specified, this rule is disabled.
    - **Password history length**: Enter the number of previous passwords to be compared. Leave this field empty to disable this rule.
    - **Minimum character differences from last password**: Enter the minimum number of different characters from the last password. Leave this field empty to disable this rule.
    - **Password lifetime (days)**: Enter how long the password is valid for in days. Leave this field empty to disable password expiry.
4. Configure the following in **PASSWORD COMPOSITION RULES**:
    - **Minimum uppercase characters**: Defaults to 1 uppercase alphabetic character.
    - **Minimum lowercase characters**: Defaults to 1 lowercase alphabetic character.
    - **Minimum numeric characters**: Defaults to 1 numeric character.
    - **Minimum special characters**: Defaults to 1 special character (`~!@#$%^&*()-_`

=+\[{}];:"",< >/?).

If no value is specified in these fields, these rules are disabled.

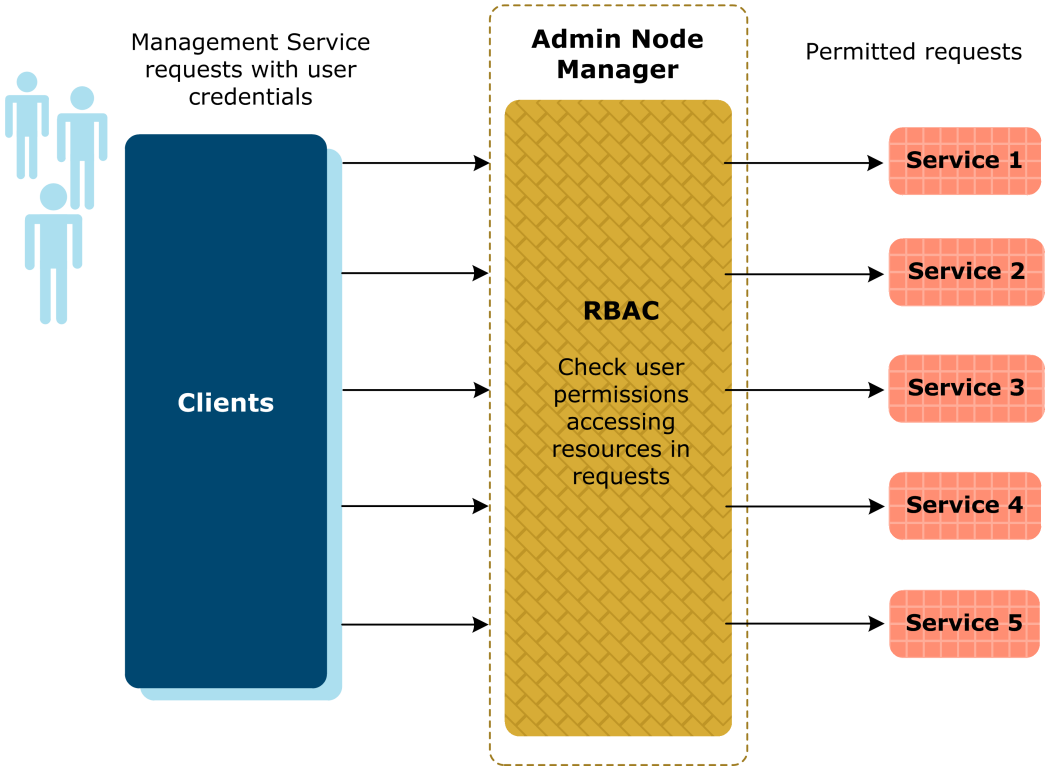
5. Click **Apply** when finished.

Configure Role-Based Access Control (RBAC)

Note This topic applies to both API Gateway and API Gateway Analytics.

Role-Based Access Control (RBAC) enables you to restrict system access to authorized users based on their assigned roles. Using the RBAC model, permissions to perform specific system operations are assigned to specific roles, and system users are granted permission to perform specific operations only through their assigned roles. This simplifies system administration because users do not need to be assigned permissions directly, and instead acquire them through their assigned roles.

The API Gateway uses the RBAC permissions model to ensure that only users with the assigned role can access parts of the management services exposed by the Admin Node Manager. For example, this includes access to traffic monitoring data or making a configuration change by deploying to a group of API Gateways. The following diagram shows an overview of the RBAC model in the API Gateway:



API Gateway Manager

The web-based API Gateway Manager tool (`https://localhost:8090`) is a centralized dashboard for managing and monitoring the API Gateway, and is controlled by RBAC. Users connecting to this URL with different roles results in different features being displayed.

For example, users with the `API Server Administrator` role has read/write access in the API Gateway Manager tool. However, users in the `API Server Operator` have only read access in the API Gateway Manager tool.

For more details on the tools and privileges assigned to specific user roles, see *Manage admin users* on page 198.

Protected management services

The Admin Node Manager exposes a number of REST management services, which are all protected by RBAC. For example, the exposed services and the associated tools that use them include the following:

Protected Service	Tool	Description
Traffic Monitoring Service	API Gateway Manager	Displays HTTP, HTTPS, JMS, and FTP message traffic processed by the API Gateway.
Configuration Service	API Gateway Manager	Adds and removes tags on the API Gateway.
Topology API	API Gateway Manager	Accesses and configures API Gateway domains.
Static Content Resources	API Gateway Manager	Manages UI elements in a browser.
Deployment API	Policy Studio	Deploys configurations to the API Gateway.
KPS Service	Policy Studio	Manages a Key Property Store.

RBAC user roles

User access to management services is determined by their role(s). Each role has a defined set of management services that it can access. A Management Service is defined by the URI used to access it, for example:

Role Name	Service Name	API Type	Example URI
API Server Operator	Topology API	REST	/api/topology/hosts

For full details on the default roles that have access to each Management Service, see *Management service roles and permissions* on page 209.

Local admin user store

By default, all the user credentials are stored in a local admin user store in the following file:

```
INSTALL_DIR/conf/adminUsers.json
```

INSTALL_DIR is the directory where the API Gateway is installed as Admin Node Manager.

The following shows an example file:

```
{
   "productVersion" :"7.6.2",
   "version" :1,
   "timestamp" :0,
   "adminUsers" :[ {
     "id" :"user-1",
     "name" :"admin",
     "roles" :[ "role-1", "role-6", "role-7" ]
   } ],
   "credentials" :{
     "user-1"
:"$AAGQAAAAAQAC$DxlhcXQ2FTZ3mjErhGDEEQ==$0/hQdegidtOyrX2QuUnmEWIzknx4bsf0iPr3HydpbyY
="
   },
   "adminUserRoles" :[ {
     "id" :"role-1",
     "name" :"API Server Administrator"
   }, {
     "id" :"role-2",
     "name" :"API Server Operator"
   }, {
     "id" :"role-5",
     "name" :"Deployer"
   }, {
     "id" :"role-6",
     "name" :"KPS Administrator"
   }, {
     "id" :"role-7",
     "name" :"Policy Developer"
   } ],
   "uniqueIdCounters" :{
```

```
      "Role" :8,
      "User" :2
    },
    "adminUsersVersion" :{
      "version" :1,
      "timestamp" :0
    }
}
```

The credentials from this file are used to authenticate and perform RBAC on all accesses to the management services. This store holds the user credentials, so their passwords can be verified, and also holds their roles. Credentials and associated roles can also be retrieved from an LDAP Directory Server (for example, Microsoft Active Directory or OpenLDAP).

Admin user password storage algorithm

Admin user passwords are in stored in the `adminUsers.json` file as a base64-encoded salted hash of the plaintext password. To authenticate a user password, the incoming password value is digested in the same way, and compared against the value in the file. If they match, the user is authenticated.

The salt is a 16 byte value generated using the SHA1PRNG pseudo-random number generator algorithm. This algorithm is provided by the Java Cryptography Extension (JCE) and is PBKDF2 with HMAC SHA2. The algorithm repeats the digest of the password along with the salt for 102400 iterations. The salt is stored with the digest allowing for password verification in the following format:

```
$<version-value>$<salt-value>$<password-digest>
```

For example:

```
$AAGQAAAAAQAC$DxlhcXQ2FTZ3mjErhGDEEQ==$0/hQdegidtOyrX2QuUnmEWIzknx4bsf0iPr3HydpbyY=
```

Because the salt is generated each time a password is stored, the same plaintext password will have a different salt and digest value. For example, the following stored passwords are all digests of the default password:

```
$AAGQAAAAAQAC$qqlNVeQ6eAGnBDK0VcU1oQ==$uzK6wvSLjVTdO0/qP9I4d72P6yRG1XbI3KRQIPpZjwA=
$AAGQAAAAAQAC$mJbacNzCfoeJ1Ukixyj/xw==$hw7uwm5/D/7HIKKH4mUoM76xnYvqOSDfucKI0U9yndk=
$AAGQAAAAAQAC$a3EP3mKkWpJm4flYacTtnQ==$Czke56xcmk9xbjw+KZhi26jpPoXV7fgE0CWp4WBz9kg=
```

Configure an LDAP repository to store credentials

Alternatively, for details on how to configure an LDAP repository to store user credentials, see the following topics:

- *Authentication and RBAC with Active Directory* on page 211
- *Authentication and RBAC with OpenLDAP* on page 222

RBAC access control list

The access control list file (`acl.json`) is located in the `conf` directory of your API Gateway installation. This file lists each role and the management services that each role may access. By default, this file defines the following roles:

- `API Server Administrator`
- `API Server Operator`
- `Deployer`
- `KPS Administrator`
- `Policy Developer`

The default `admin` user is assigned the `API Server Administrator`, `KPS Administrator`, and `Policy Developer` roles by default, which together allow access to everything. For full details on the management services that each role has access to, and the permissions that must be listed in the `acl.json` file to have access to them, see *Management service roles and permissions* on page 209.

Note The roles defined in the `acl.json` file should exist in the user store used to authenticate the users and load their roles and groups. The default roles are defined in the local Admin User store, which is used to control access to the management services using the **Protect Management Interfaces** policy.

If a different user store is used (for example, an LDAP repository), the LDAP groups should be listed in the `acl.json` and `adminUsers.json` files.

Access control list file format

Each role entry in the `acl.json` file has the following format:

```
"role-name" :[ <list_of_permission_names>]
```

The permissions consist of operations that are defined by HTTP methods and URIs:

```
“permission-name” : { <list_of_operation_names }
“operation-name” : {
     "methods" :[ <list of HTTP Methods>],
     "paths" :[ <list of path-names>]
}
“path-name” : {
     "path" :<URI>
}
```

This file entry format is described as follows:

- The permissions line is repeated for each permission the role has. To determine which permissions should be listed for each Management Service, see *Management service roles and permissions* on page 209.
- You can place a wildcard (`*`) at the end of the `path` field. For example, see the path for `dojo resources` in the example that follows. This means the role has access to all URIs that start with the URI content that precedes the `*`.
- In some cases, you must protect a Management Service by specifying a query string after the URI. Exact matches only are supported for query strings.

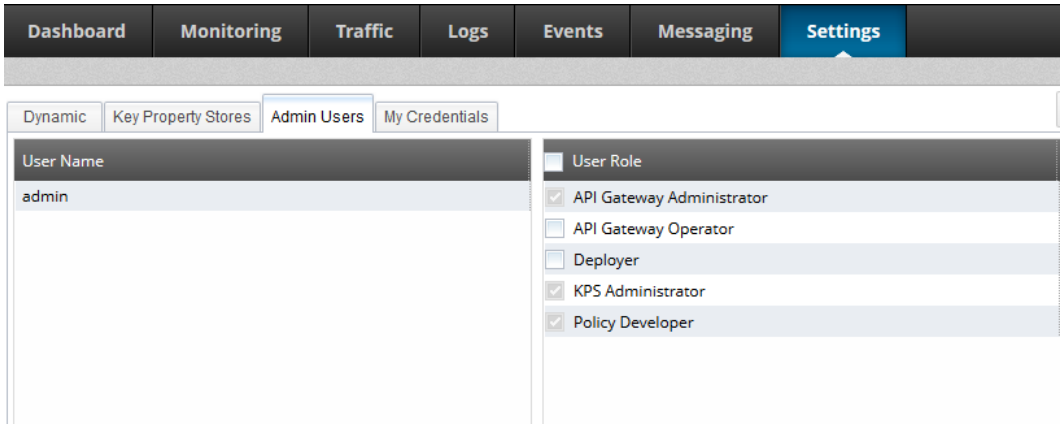
Example access control list file

The following example shows roles and permissions to URIs:

```
"paths" : {
      "root" :{ "path" :"/" },
      "emc pages" :{ "path" :"/emc/*" },
      "site images" :{ "path" :"/images/*" },
      "dojo resources" :{ "path" :"/dojo/*" },
      ....
      }
},
"operations" : {
       "emc_read_web" : {
       "methods" :[ "GET" ],
       "paths" :[ "emc pages", "dojo resources" ]
       },
       "common_read_web" : {
       "methods" :[ "GET" ],
       "paths" :[ "root", "site images" ]},
       ....
},"permissions" : {
       "emc" :[ "common_read_web", "emc_read_web" ],
       "config" :[ "configuration" ],
       "deploy" :[ "deployment", "management" ],
       ...
},
"roles" : {
       "Policy Developer" :[ "deploy", "config"]
       ...
}
```

Configure RBAC users and roles

You can use the API Gateway Manager to configure the users and roles in the local Admin User store. Click the **Settings** > **Admin Users** to view and modify user roles (assuming you have a role that allows this). This screen is displayed as follows:



Manage RBAC user roles

When you click **Create** to create a new user, you can select the roles to assignto the that new user. New users are not assigned a default role. While users that are replicated from an LDAP repository do not require a role to be assigned to them. You can click **Edit** to changed the roles assigned to a selected user.

Add a new role to the user store

When you add a new role to the Admin User store, you must modify the available roles in the `adminUsers.json` and `acl.json` files in the `conf` directory of your Admin Node Manager installation. You must add the new role to the `roles` section of the `acl.json` file, which lists all the permissions that the new role may have.

Note You must update the `acl.json` before you add the roles to the Admin User store. The RBAC policy object automatically reloads the `acl.json` file each time you add or remove a role in the Policy Studio.

When you update the `acl.json` file, you must restart the Admin Node Manager to reload the `acl.json` file. However, the Admin Node Manager does not need to be rebooted or refreshed if a user's roles change.

For more details on managing user roles, see *Manage admin users* on page 198.

Management service roles and permissions

You can use the following table for reference purposes when making changes to the `acl.json` file. It defines each Management Service, and the default roles that have access to them. It also lists the permissions that must be listed in the `acl.json` file to have access to the Management Service.

Management Service	Default Roles	Permissions
API Gateway Manager (`https://localhost:8090`)	API Server Administrator API Server Operator	`emc` `mgmt`

API Gateway Manager Dashboard	API Server Administrator	`emc`
		`mgmt`
		`mgmt_modify`
		`dashboard`
		`dashboard_modify`
		`deploy`
		`config`

API Gateway Manager Dashboard (read-only access)	API Server Operator	`emc`
		`mgmt`
		`dashboard`
		`dashboard_modify`

API Gateway Manager Monitoring	API Server Administrator
	API Server Operator
		`emc`
		`mgmt`
		`monitoring`
		`events`
		`traffic_monitor`
		`settings`
		`settings_modify`
		`logs`

Management Service	Default Roles	Permissions

API Gateway Manager Traffic	API Server Administrator
	API Server Operator
		`emc`
		`mgmt`
		`traffic_monitor`

API Gateway Manager Logs	API Server Administrator
	API Server Operator
		`emc`
		`mgmt`
		`logs`

API Gateway Manager Events	API Server Administrator
	API Server Operator
		`emc`
		`mgmt`
		`monitoring`
		`events`

API Gateway Manager Settings	API Server Administrator	`emc`
		`mgmt`
		`mgmt_modify`
		`settings`
		`settings_modify`

API Gateway Manager Settings (read-only access)	API Server Operator	`emc`
		`mgmt`
		`settings`

Documentation	API Server Administrator API Server Operator	`emc` `mgmt`
KPS	KPS Administrator	`mgmt` `kps`

Policy Studio	Policy Developer	`mgmt`
		`deploy`
		`config`

API Gateway Configuration Deployment	API Server Administrator
	Policy Developer
	Deployer
		`mgmt`
		`deploy`
		`config`

Authentication and RBAC with Active Directory

This topic explains how to use Lightweight Directory Access Protocol (LDAP) to authenticate and perform Role-Based Access Control (RBAC) of API Gateway management services. It describes how to reconfigure API Gateway to use a Microsoft Active Directory LDAP repository.

This topic uses the sample **Protect Management Interfaces (LDAP)** policy instead of the **Protect Management Interfaces** policy. This means that the API Gateway uses an LDAP repository instead of the local Admin User store for authentication and RBAC of users attempting to access API Gateway management services.

This topic contains the following sections:

Note If you have multiple Admin Node Managers in your topology, you must ensure that you apply the configuration changes to each Admin Node Manager.

Step 1: Create an Active Directory group

To create a new user group in Active Directory, perform the following example steps:

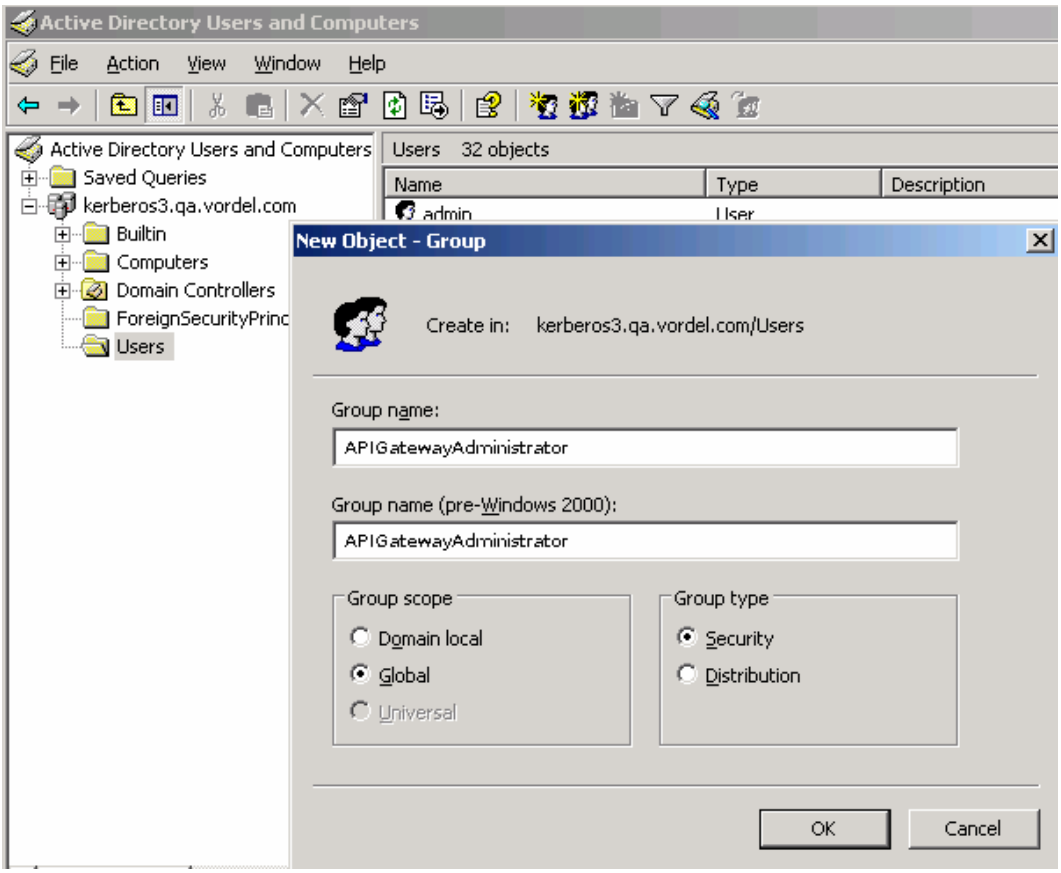
1. Click **Start** > **Administrative Tools** > **Active Directory Users and Computers**.
2. On the **Users** directory, right-click, and select **New** > **Group**.
3. Enter the Group name (for example, `APIGatewayAdministrator`).

You should add groups for the following default RBAC roles to give the LDAP users appropriate access to the API Gateway management services:

- `API Gateway Administrator`
- `API Gateway Operator`
- `Deployer`
- `KPS Administrator`
- `Policy Developer`

These RBAC roles are located in the `roles` section of the following file:

```
INSTALL_DIR\apigateway\conf\acl.json
```

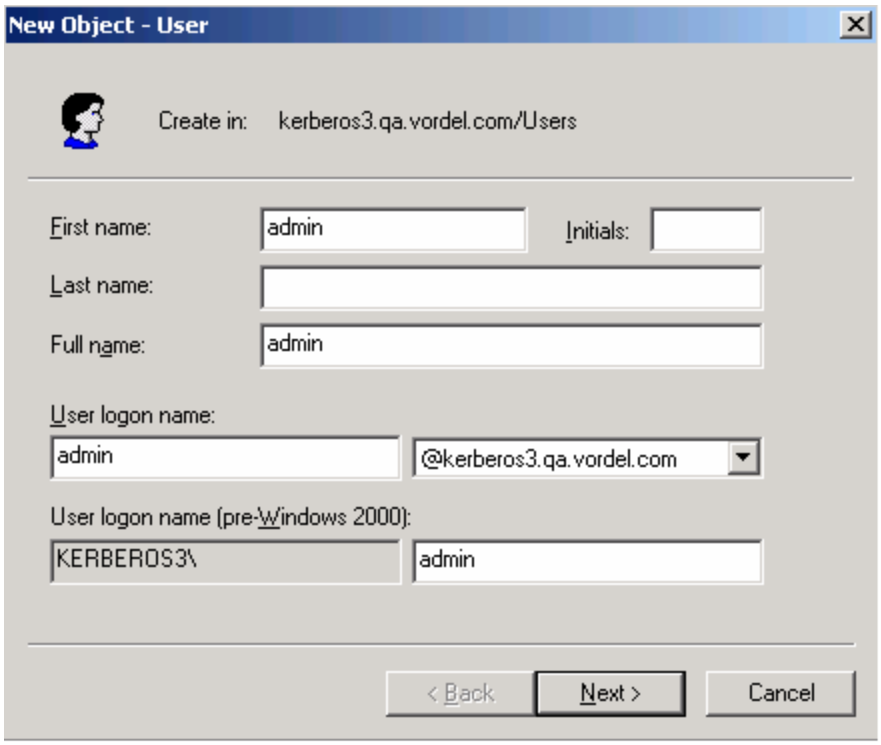


You can view the newly created groups using an LDAP browser.

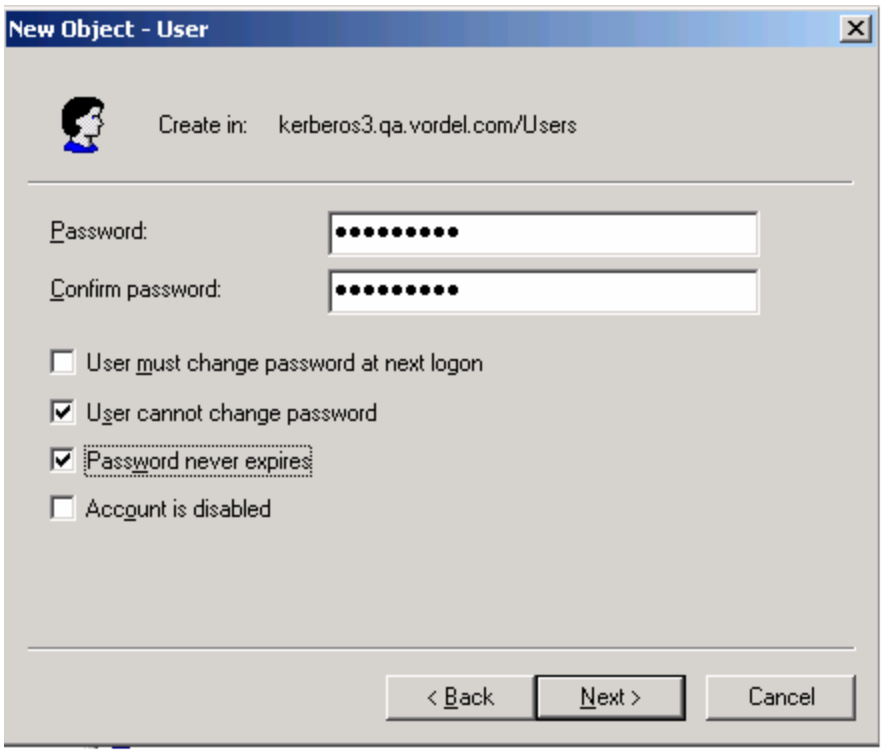
Step 2: Create an Active Directory user

You will most likely be unable to create an `admin` user with a default password because the default password is not strong enough to be accepted by Active Directory. Using **Active Directory Users and Computers**, perform the following steps:

1. On the **Users** directory, right-click, and select **New** > **User**.
2. Enter a user name (for example, `admin`).



3. Click **Next**.
4. Enter a password (for example, `Axway123`).
5. Select **User cannot change password** and **Password never expires**.
6. Ensure **User must change password at next logon** is not selected.
7. Click **Next**.
8. Click **Finish**.

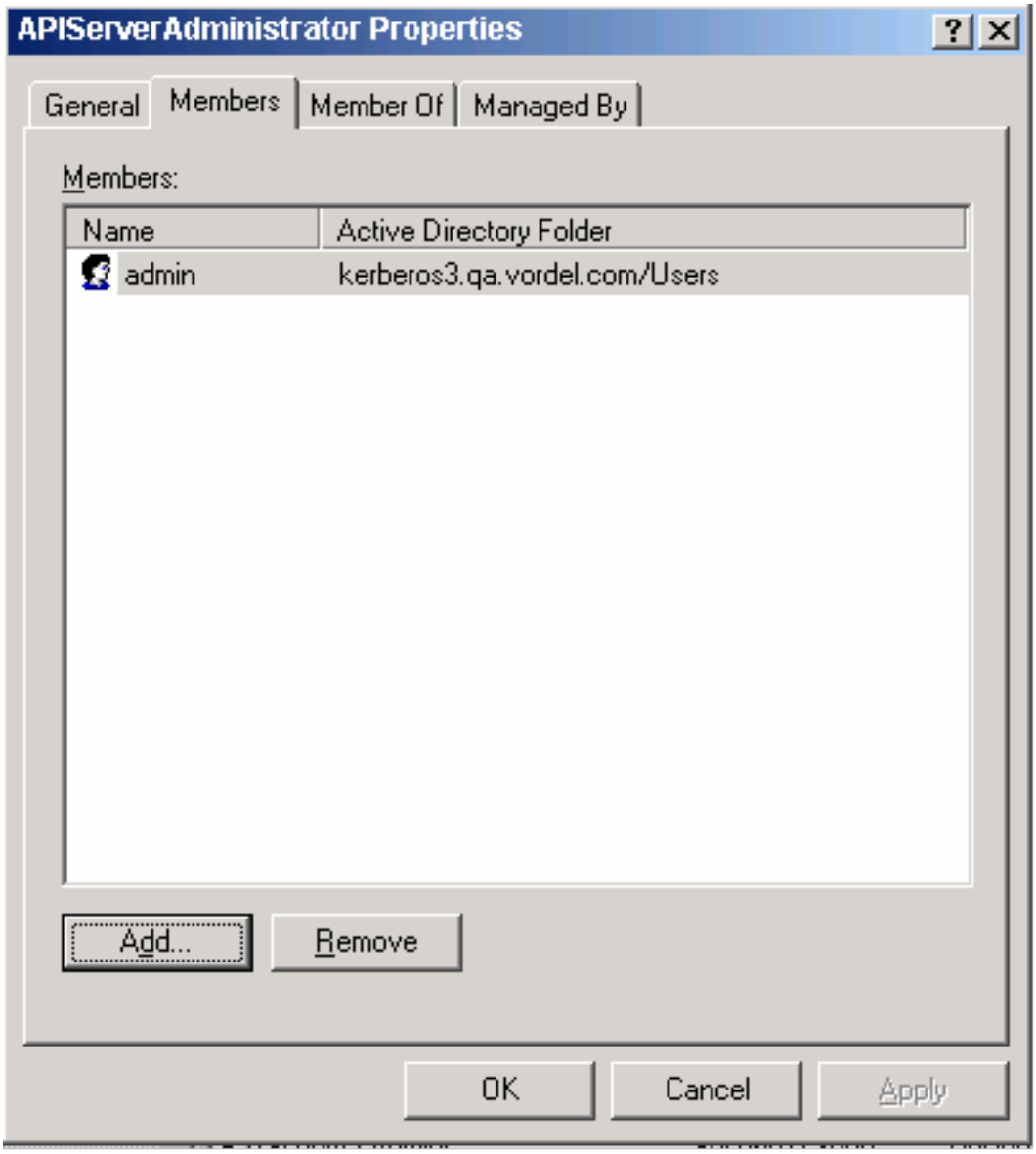


Add the user to the group

To make the user a member of the group using **Active Directory Users and Computers** , perform the following steps:

1. Select the **APIGatewayAdministrator** group, right-click, and select **Properties**.
2. Click the **Members** tab.
3. Click **Add**.
4. Click **Advanced**.
5. Click **Find Now**.
6. Select the `admin` user.

7. Click **OK**.



You can view the newly created user using an LDAP browser.

Note The `memberOf` attribute points to the Active Directory group. The user has an instance of this attribute for each group they are a member of.

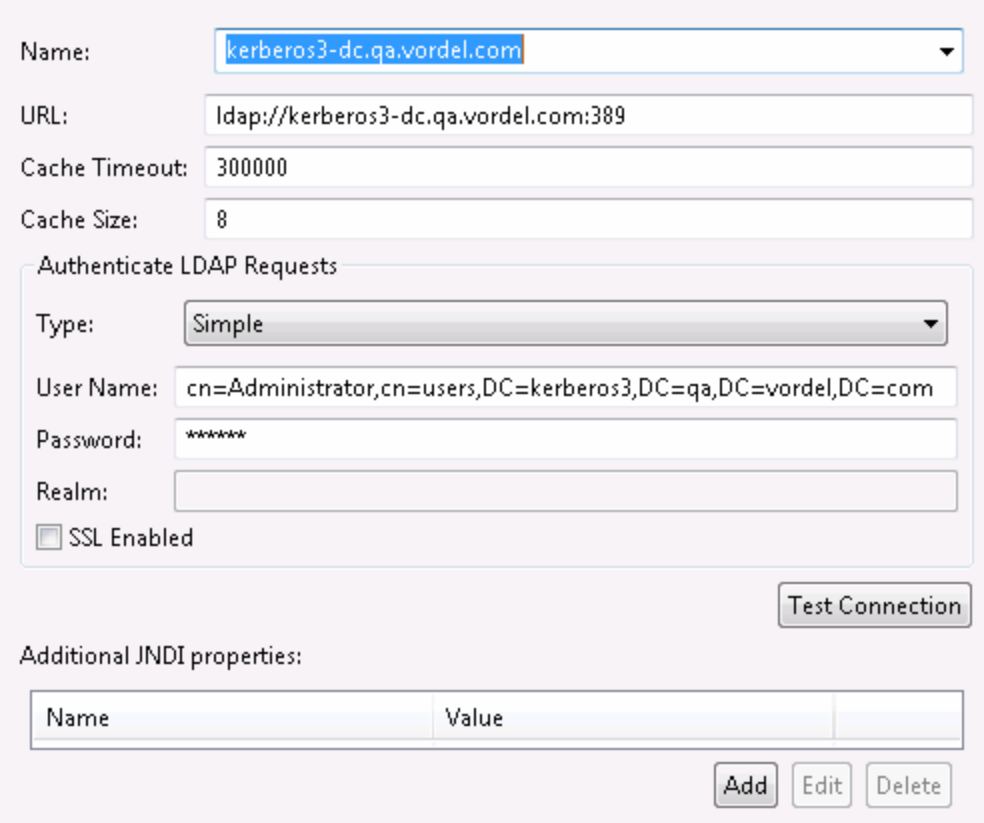
Step 3: Create an LDAP connection

To create an new LDAP Connection, perform the following steps:

1. In Policy Studio, create a new project based on the Admin Node Manager configuration. For example:

```
INSTALL_DIR\apigateway\conf\fed
```

2. In Policy Studio tree, select **Environment Configuration** > **External Connections** > **LDAP Connections**.
3. Right-click, and select **Create an LDAP Connection**.
4. Complete the fields in the dialog as appropriate. The specified **User Name** should be an LDAP administrator that has access to search the full directory for users. For example:



5. Click **Test Connection** to ensure the connection details are correct.

Step 4: Create an LDAP repository

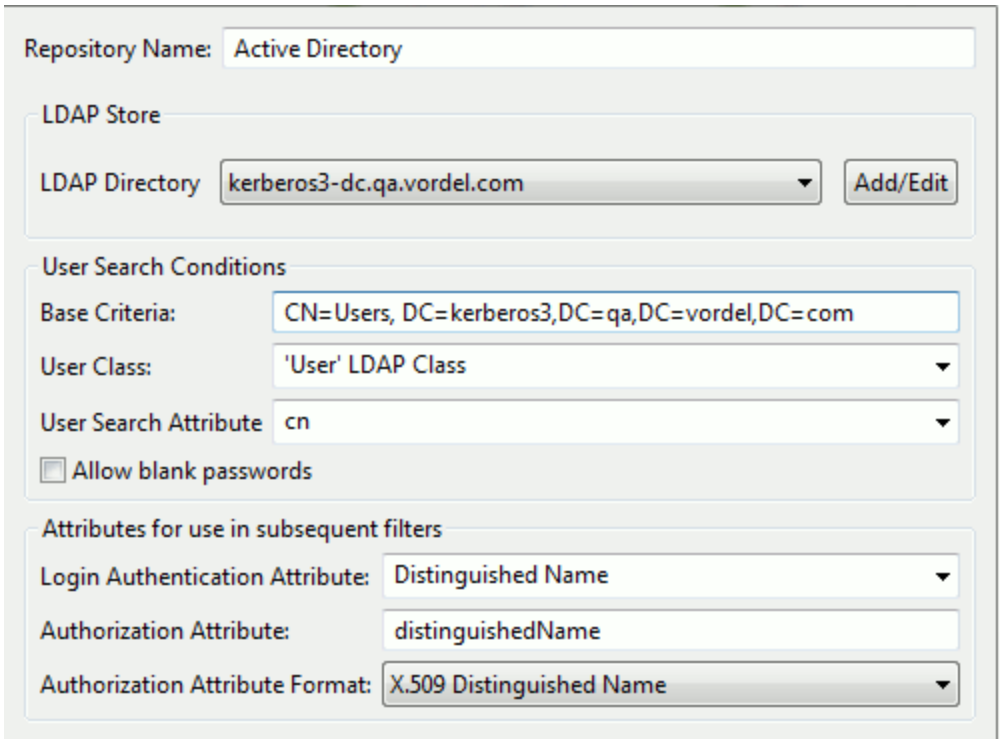
To create an new LDAP Repository, perform the following steps:

1. In the Policy Studio tree, select **Environment Configuration** > **External Connections** > **Authentication Repository Profiles** > **LDAP Repositories**.
2. Right-click, and select **Add a new Repository**.
3. Complete the following fields in the dialog:

Repository Name	Enter an appropriate name for the repository.
LDAP Directory	Use the LDAP directory created in *Step 3: Create an LDAP connection* on page 215.
Base Criteria	Enter the LDAP node that contains the users.
User Search Attribute	Enter `cn`. This is the username entered at login time (in this case, `admin`).
Authorization Attribute	Enter `distinguishedName`. This is the username entered at login time (`admin`). The `authentication.subject.id` message attribute is set to the value of this LDAP attribute (see example below). The `authentication.subject.id` is used as the base criteria in the filter that loads the LDAP groups (the user's roles). This enables you to narrow the search to a particular user node in the LDAP tree. For more details, see the **Retrieve Attributes from Directory Server** filter in *Step 5: Configure a test policy for LDAP authentication and RBAC* on page 218.

An example value of the `authentication.subject.id` message attribute is as follows:

```
CN=admin, CN=Users,DC=kerberos3,DC=qa,DC=vordel,DC=comn
```



Connect to other LDAP repositories

This topic uses Microsoft Active Directory as an example LDAP repository. Other LDAP repositories such as Oracle Directory Server (formerly iPlanet and Sun Directory Server) and OpenLDAP are also supported.

For an example of querying an Oracle Directory Server repository, see the **Retrieve Attributes from Directory Server** filter in *Step 5: Configure a test policy for LDAP authentication and RBAC* on page 218. For details on using OpenLDAP, see *Authentication and RBAC with OpenLDAP* on page 222.

Step 5: Configure a test policy for LDAP authentication and RBAC

To avoid locking yourself out of Policy Studio, you can configure an example test policy for LDAP authentication and RBAC, which is invoked when a test URI is called on the server (and not a management services URI). Policy Studio provides an example policy named **Protect Management Interfaces (LDAP)** when the Admin Node Manager configuration is loaded.

You must edit this policy to change it from using the sample LDAP connection to the one that you created in *Step 3: Create an LDAP connection* on page 215. You must also change it from using the sample authentication repository to the authentication repository that you created in *Step 4: Create an LDAP repository* on page 216. Then in *Step 6: Use the LDAP policy to protect management services* on page 220, you must hook up this new LDAP policy to the `/` path.

Configure the example test policy

Perform the following steps:

1. In Policy Studio, create a new project based on the Admin Node Manager configuration. For example:

   ```
   INSTALL_DIR\apigateway\conf\fed
   ```

2. For the example policy, select **Policies** > **Management Services** > **Sample LDAP Policies** > **Protect Management Interfaces (LDAP)** when the Admin Node Manager configuration is loaded. This policy is summarized at a high-level as follows:

- **Scripting Language** filter returns `true` if the Node Manager is the Admin Node Manager. This enables subsequent HTTP authentication and RBAC, and updates the HTTP headers based on the user role. Otherwise, this filter calls the **Call Internal Service (no RBAC)** filter without updating the HTTP headers.
- **HTTP Basic Authentication** filter verifies the user name and password against the LDAP repository configured in *Step 4: Create an LDAP repository* on page 216.
- **Retrieve Attributes from Directory Server** filter finds the LDAP groups that the user belongs to using the LDAP directory connection configured in *Step 3: Create an LDAP connection* on page 215.

- **Management Services RBAC** filter reads the user roles from the configured message attribute (`authentication.subject.role`). This returns `true` if one of the roles has access to the management service currently being invoked, as defined in the `acl.json` file. Otherwise, this returns `false` and the **Return HTTP Error 403:Access Denied (Forbidden)** policy is called because the user does not have the correct role.

3. You must edit some HTTP-based filters and change them from using the **Sample Active Directory Repository** to using the repository that you created in *Step 4: Create an LDAP repository* on page 216. This repository is referenced in the following filters in the example policy:
    - **Authenticate login attempt**
    - **HTTP Basic**

Tip You can view all referenced filters by selecting **Environment Configuration** > **External Connections** > **LDAP Repositories** > **Sample Active Directory Repository** > **Show all References**.

4. You must edit the LDAP-based filters and change them from using the **Sample Active Directory Connection** to using the LDAP connection that you created in *Step 3: Create an LDAP connection* on page 215. This repository is referenced in the following components in the example policy:
    - **Read Roles from Directory Server**
    - **Re-Read Roles from Directory Server**
    - **Sample Active Directory Repository**

    Similarly, you can view all referenced components by selecting **Environment Configuration** > **External Connections** > **LDAP Repositories** > **Sample Active Directory Connection** > **Show all References**.

For more details on creating policies, see the *API Gateway Policy Developer Guide*.

Test the policy configuration

To test this policy configuration, perform the following steps:

1. Update the `acl.json` file with the new LDAP group, for example:

```
"CN=APIGatewayAdministrator,CN=Users,DC=kerberos3,DC=qa,DC=vordel,DC=com" :[
   "emc", "mgmt", "mgmt_modify", "dashboard", "dashboard_modify", "deploy" "config",
   "monitoring", "events", "traffic_monitor", "settings", settings_modify", "logs" ]
```

2. Update the `adminUsers.json` file with the new role, for example:

```
{
   "name" :"CN=APIGatewayAdministrator,CN=Users,DC=kerberos3,DC=qa,DC=vordel,
    DC=com","id" :"role-8"
}
```

3. And increase the number of roles, for example:

```
"uniqueIdCounters" :{"Role" :9,"User" :2},
```

4. In the Policy Studio tree, select **Environment Configuration** > **Listeners** > **Node Manager** > **Add HTTP Services**, and enter a service name (for example, `LDAP Test`).
5. Right-click the HTTP service, and select **Add Interface** > **HTTP**.
6. Enter an available port to test the created policy (for example, `8888`), and click **OK**.
7. Right-click the HTTP service, and select **Add Relative Path**.
8. Enter a relative path (for example, `/test`).
9. Set the **Path Specify Policy** to the **Protect Management Interfaces (LDAP)** policy, and click **OK**.
10. Close the connection to the Admin Node Manager file, and restart the Admin Node Manager so it loads the updated configuration.
11. Use an API test client such as Postman or `curl` to call `http://localhost:8888/test`.
12. Enter the HTTP Basic credentials (for example, username `admin` and password `Axway123`). If authentication is passed, the Admin Node Manager should return an HTTP 404 code (not found).

Note Do not use the **Admin Users** tab in the API Gateway Manager to manage user roles because these are managed in LDAP.

Step 6: Use the LDAP policy to protect management services

If the authentication and RBAC filters pass, you can now use this policy to protect the management interfaces. To ensure that you do not lock yourself out of the server, perform the following steps:

1. Make a copy of the `conf/fed` directory contents from the server installation, and put it into a directory accessible from the Policy Studio.
2. Make another backup copy of the `conf/fed` directory, which will remain unmodified.
3. In the Policy Studio, select **File** > **New project**, enter a name, and click **Next**.
4. Select **From existing configuration**, and click **Next**.
5. Browse to `INSTALL_DIR/apigateway/conf/fed`, and click **Finish**.
6. Under the **Environment Configuration** > **Listeners** > **Node Manager** > **Management Services** node, select the `/` and the `/configuration/deployments` relative paths, and set the **Path Specify Policy** to the **Protect Management Interfaces (LDAP)** policy.
7. Remove the previously created `LDAP Test` HTTP Services, and close the connection to the file.
8. Copy the `fed` directory back to the Admin Node Manager's `conf` directory.
9. Reboot the Admin Node Manager.

10. Start the Policy Studio, and connect to the Admin Node Manager using `admin` and password `Axway123` (the LDAP user credentials). You should now be able to edit API Gateway configurations as usual.

Add an LDAP user with limited access to management services

You can add an LDAP user with limited access to management services. For example, assume there is already a user named `Fred` defined in Active Directory. `Fred` has the following DName:

```
CN=Fred,CN=Users,DC=kerberos3,DC=qa,DC=vordel,DC=com
```

`Fred` belongs to an existing LDAP group called `TraceAnalyzers` . He can also belong to other LDAP groups that have no meaning for RBAC in the API Gateway. The `TraceAnalyzers` LDAP group has the following DName:

```
CN=TraceAnalyzers,CN=Users,DC=kerberos3,DC=qa,DC=vordel,DC=com
```

The user `Fred` should be able to read server trace files in a browser. No other access to management services should be given to `Fred`.

Add limited access rights

You must perform the following steps to allow `Fred` to view the trace files:

1. Add the following entry in the `roles` section in the `acl.json` file:

```
"CN=TraceAnalyzers,CN=Users,DC=kerberos3,DC=qa,DC=vordel,DC=com" :
   [ "emc", "mgmt", "logs" ]
```

2. Update the `adminUsers.json` file with the new role as follows:

```
{
   "name" :"CN=TraceAnalyzers,CN=Users,DC=kerberos3,DC=qa,DC=vordel,
    DC=com","id" :"role-8"
}]
```

   And increase the number of roles, for example:

```
"uniqueIdCounters" : {
   "Role" :9,
   "User" :2
},
```

3. Restart the Admin Node Manager so that the `acl.json` and `adminUsers.json` file updates are picked up.
4. Enter the following URL in your browser:

```
http://localhost:8090/
```

5. Enter user credentials for `Fred` when prompted in the browser.
6. The API Gateway Manager displays a **Logs** tab enabling access to the trace files that `Fred` can view.

Note `Fred` is not allowed to access the server APIs used by the Policy Studio. If an attempt is made to connect to the server using the Policy Studio with his credentials, an `Access denied` error is displayed.

No other configuration is required to give user `Fred` the above access to the management services. Other users in the same LDAP group can also view trace files without further configuration changes because the LDAP group is already defined in the `acl.json` file.

Authentication and RBAC with OpenLDAP

This topic explains how to use Lightweight Directory Access Protocol (LDAP) to authenticate and perform Role-Based Access Control (RBAC) of API Gateway management services. It describes how to reconfigure API Gateway to use OpenLDAP as the LDAP repository, and to use the Apache Directory Studio as an LDAP browser.

This topic uses the sample **Protect Management and Interfaces (LDAP)** policy instead of the **Protect Management Interfaces** policy. This means that the API Gateway uses an LDAP repository instead of the local Admin User store for authentication and RBAC of users attempting to access the API Gateway management services.

This topic contains the following sections:

Note If you have multiple Admin Node Managers in your topology, you must ensure that you apply the configuration changes to each Admin Node Manager.

Prerequisites

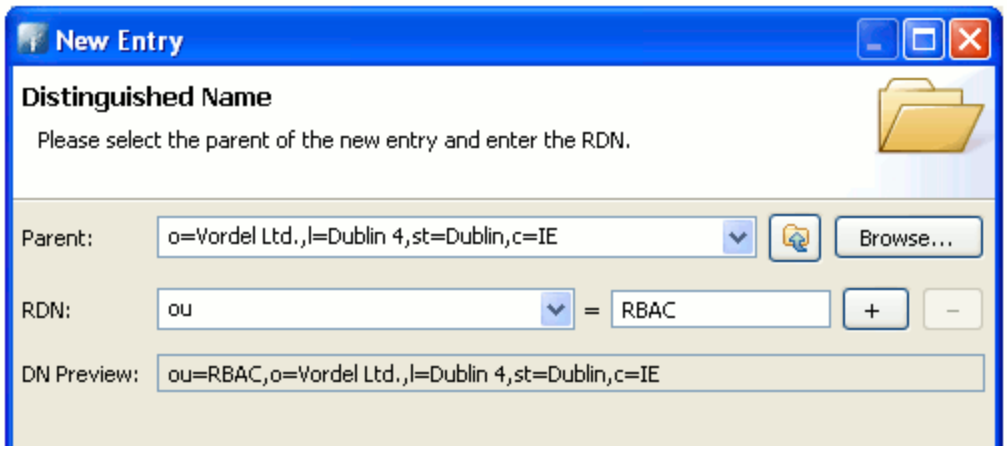
This example assumes that you already have configured a connection to the OpenLDAP server and set up your organization groups and users that you wish to use to perform RBAC. For example:

- **LDAP URL**: `ldap://openldap.qa.axway.com:389`
- **User**: `cn=admin,o=Vordel Ltd.,l=Dublin 4,st=Dublin,C=IE`
- **Password**: `axway`

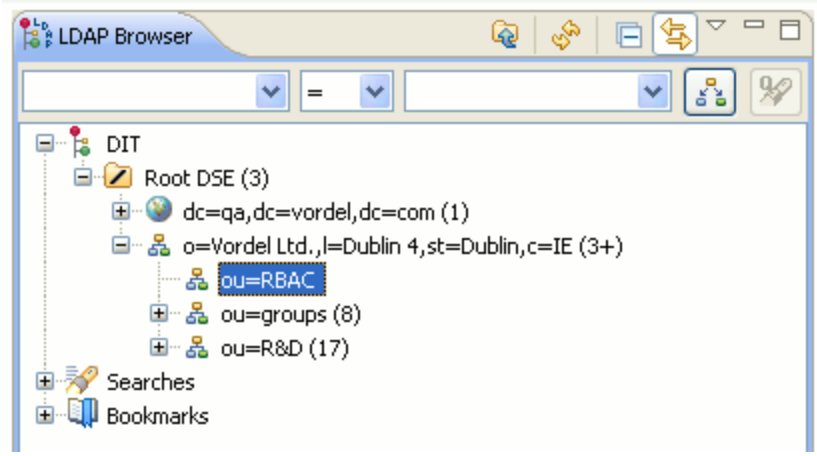
Step 1: Create an OpenLDAP group for RBAC roles

To create a new user group in OpenLDAP, perform the following steps:

1. Select the `cn=admin,o=Vordel Ltd.,l=Dublin 4,st=Dublin,C=IE` directory.
2. Right-click, and select **New** > **New Entry**.
3. Select **Create entry from scratch**.
4. Click **Next**.
5. Add an `organizationalUnit` object class.
6. Click **Next**.
7. Set the **Parent** to `o=Vordel Ltd.,l=Dublin 4,st=Dublin,C=IE`.
8. Set the **RDN** to `ou = RBAC`.
9. Click **Next**.
10. Click **Finish**.



You can view the new group using an LDAP Browser. For example:



Step 2: Add RBAC roles to the OpenLDAP RBAC group

You must add the following default RBAC roles to the `ou=RBAC,o=Vordel Ltd.,l=Dublin 4,st=Dublin,C=IE` group to give the LDAP users appropriate access to the API Gateway management services: These RBAC roles are located in the `roles` section of the `acl.json` file.

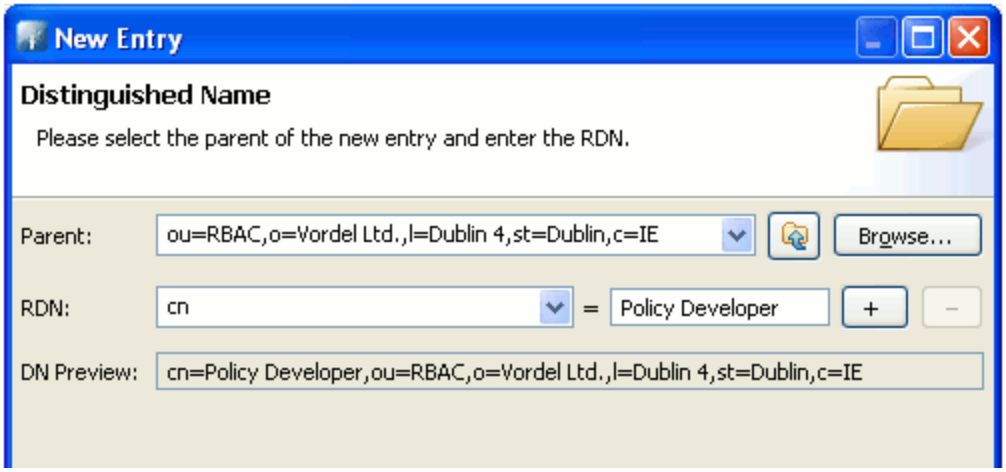
- `API Server Administrator`
- `API Server Operator`
- `KPS Administrator`
- `Policy Developer`
- `Deployer`

Add roles to the RBAC directory

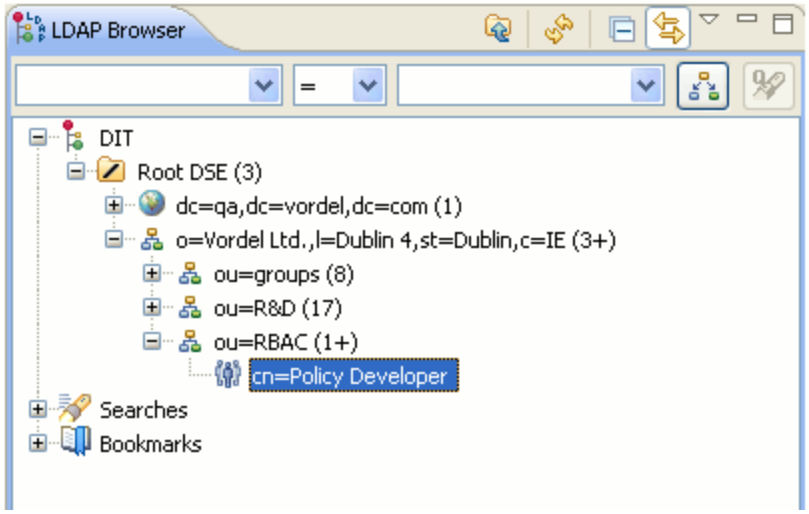
To add these RBAC roles to the OpenLDAP RBAC group, perform the following steps:

1. Select the `cn=admin,o=Vordel Ltd.,l=Dublin 4,st=Dublin,C=IE` directory.
2. Right-click, and select **New** > **New Entry**.
3. Select **Create entry from scratch**.
4. Click **Next**.
5. Add a `groupOfNames` object class.
6. Click **Next**.
7. Set the **Parent** to `ou=RBAC,o=Vordel Ltd.,l=Dublin 4,st=Dublin,C=IE`.
8. Set the **RDN** to `cn = Policy Developer`.
9. Click **Next**.

10. In the **DN Editor** dialog, set `admin` as first member of the following group:`cn=admin,ou=R&D,o=Vordel Ltd.,l=Dublin 4,st=Dublin,c=IE`. You can change the member Distinguished Name at any time.
11. Click **OK**.
12. Click **Finish**.

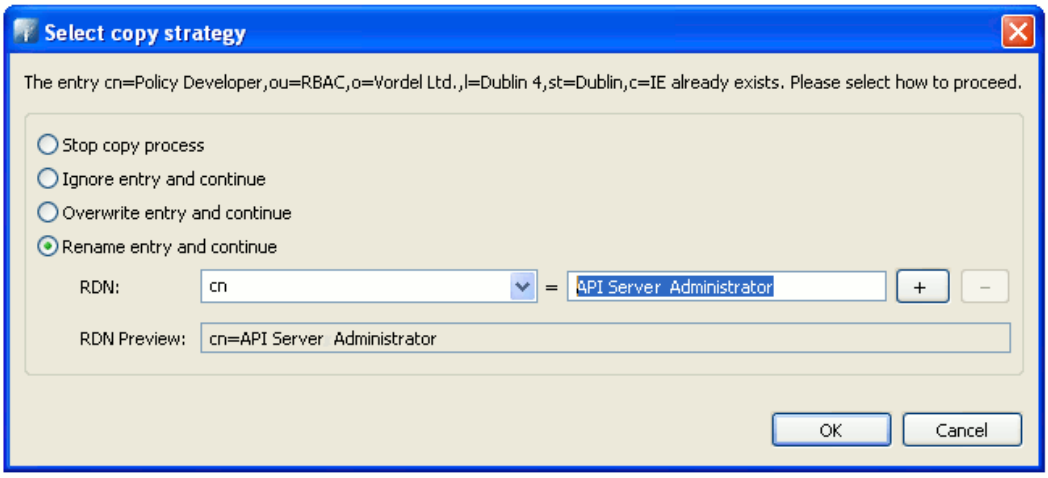


You can view the role in the OpenLDAP group in an LDAP Browser. For example:



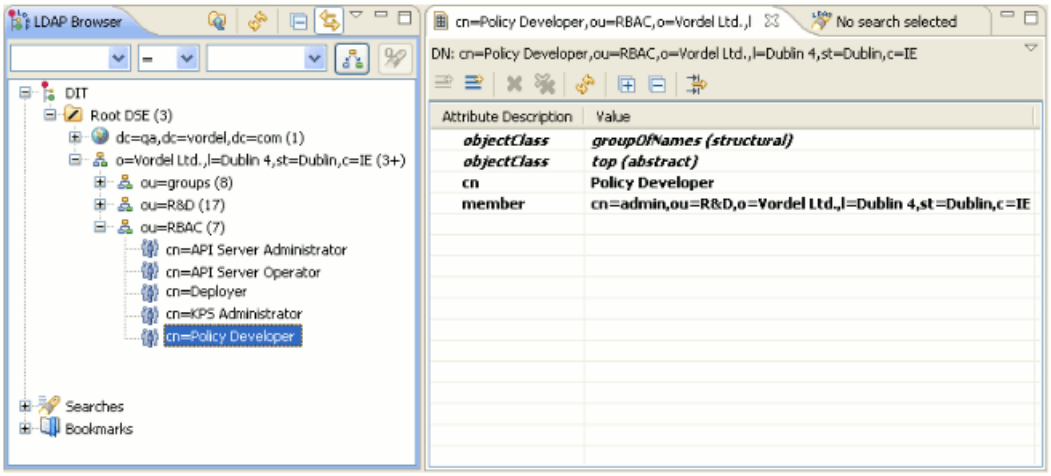
Add other roles to the RBAC directory

You can repeat these steps to add other roles to the RBAC directory. Alternatively, you can copy the `Policy Developer` entry, and paste it into the `RBAC` directory, renaming the entry with required RBAC role name. For example:



Note You should have the RBAC directory ready to add members to the role entries. By default, the `admin` user (“`cn=admin,ou=R&D,o=Vordel Ltd.,l=Dublin 4,st=Dublin,c=IE`”) is already a member of the role entries.

The following example shows the added roles:

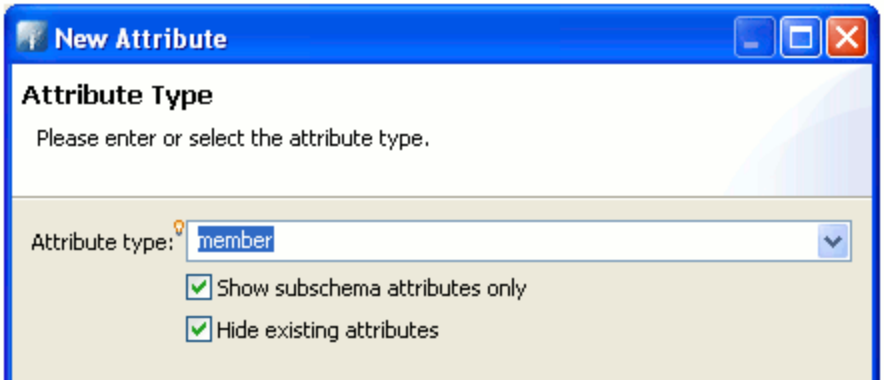


Now you can add new users to the RBAC role entries. The member attribute value should contain the user Distinguished Name. This is explained in the next section.

Step 3: Add users to the OpenLDAP RBAC group

To add a user to the OpenLDAP RBAC group, perform the following steps:

1. Select the required RBAC group (for example, `cn=API Gateway Administrator`) to view the group details.
2. Right-click the list of group attributes, and select **New Attribute**.
3. Enter `member` in the attribute type.



4. Click **Finish**.
5. In the **DN Editor** dialog, enter the user Distinguished Name (for example, `cn=joe.bloggs,o=Vordel Ltd.,l=Dublin 4,st=Dublin,c=IE`).
6. Click **OK**.

The `cn=joe.bloggs,o=Vordel Ltd.,l=Dublin 4,st=Dublin,c=IE` new user has been added to the RBAC `API Server Administrator` role.

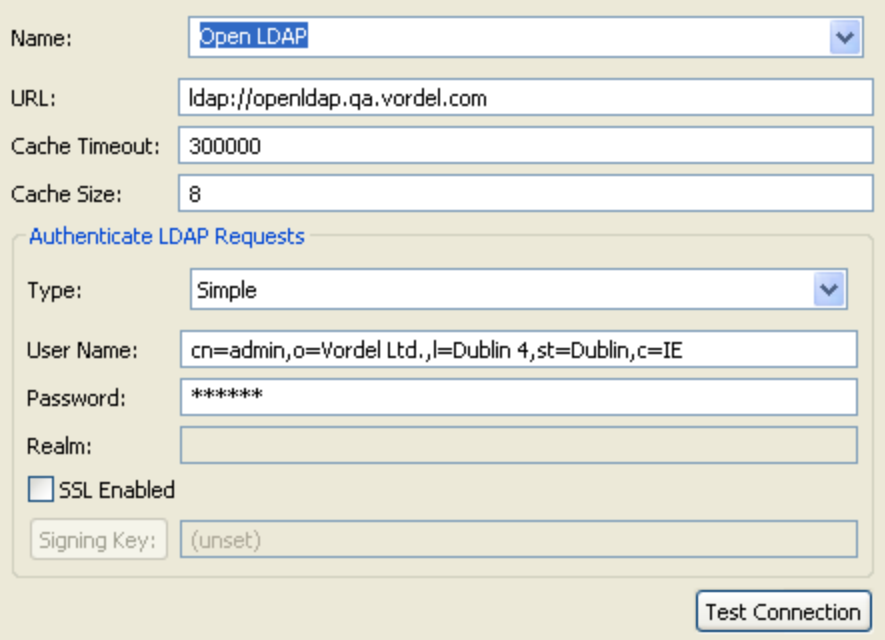
Step 4: Create an LDAP connection

To create an new LDAP Connection, perform the following steps:

1. In Policy Studio, create a new project based on the Admin Node Manager configuration. For example:

   ```
   INSTALL_DIR\apigateway\conf\fed
   ```

2. In the Policy Studio tree, select **Environment Configuration** > **External Connections** > **LDAP Connections**.
3. Right-click, and select **Create an LDAP Connection**.
4. Complete the fields in the dialog as appropriate. For example:



Note The specified **User Name** should be an LDAP administrator that has access to search the full directory for users.

5. Click **Test Connection** to ensure the connection details are correct.

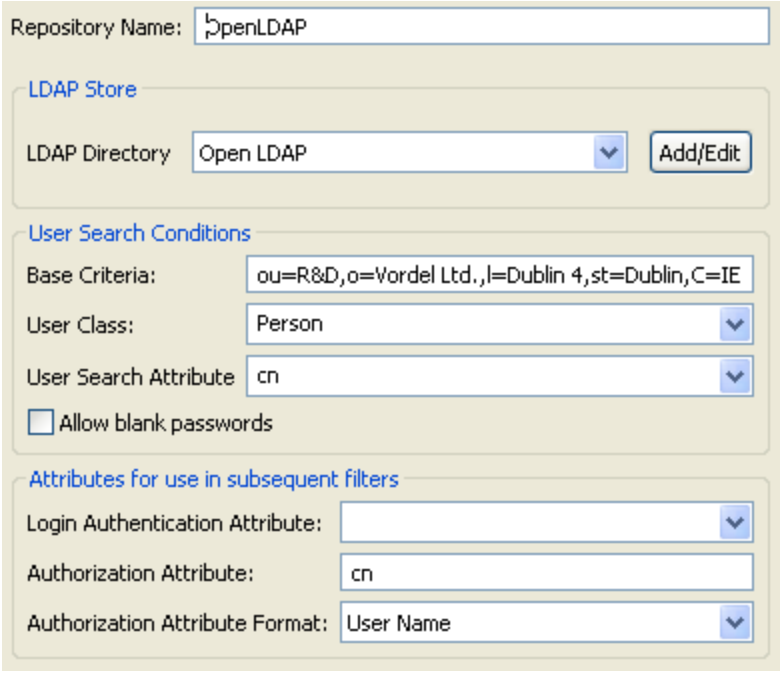
Step 5: Create an OpenLDAP repository

To create an new OpenLDAP Repository, perform the following steps:

1. In the Policy Studio tree, select **External Connections** > **Authentication Repository Profiles** > **LDAP Repositories**.
2. Right-click, and select **Add a new Repository**.
3. Complete the following fields in the dialog:

Repository Name	Enter an appropriate name for the repository.
LDAP Directory	Use the LDAP directory created in *Step 3: Add users to the OpenLDAP RBAC group* on page 226.
Base Criteria	Enter the LDAP node that contains the users (for example, see the LDAP Browser screen in *Step 3: Add users to the OpenLDAP RBAC group* on page 226).
User Search Attribute	Enter `cn`. This is the username entered at login time (in this case, `admin`).

Authorization Attribute	Enter `cn` . The `authentication.subject.id` message attribute is set to the value of this LDAP attribute (for example, `cn=admin,ou=R&D,o=Vordel Ltd.,l=Dublin 4,st=Dublin,c=IE` .`authentication.subject.id` is used as the base criteria in the filter used to load the LDAP groups (the user's roles). This allows you to narrow the search to a particular user node in the LDAP tree. For more details, see the **Retrieve Attributes from Directory Server** filter in *Step 6: Configure a test policy for LDAP authentication and RBAC* on page 230.



Connect to other LDAP repositories

This topic uses OpenLDAP as an example LDAP repository. Other LDAP repositories such as Oracle Directory Server (formerly iPlanet and Sun Directory Server) and Microsoft Active Directory are also supported. For details on using a Microsoft Active Directory repository, see *Authentication and RBAC with Active Directory* on page 211.

For an example of querying an Oracle Directory Server repository, see the **Retrieve Attributes from Directory Server** filter in *Step 6: Configure a test policy for LDAP authentication and RBAC* on page 230.

Step 6: Configure a test policy for LDAP authentication and RBAC

To avoid locking yourself out of Policy Studio, you can configure an example test policy for LDAP authentication and RBAC, which is invoked when a test URI is called on the server (and not a management services URI). Policy Studio provides an example policy named **Protect Management Interfaces (LDAP)** when the Admin Node Manager configuration is loaded.

You must edit this policy to change it from using the sample LDAP connection to the one that you created in *Step 4: Create an LDAP connection* on page 227. You must also change it from using the sample authentication repository to the authentication repository that you created in *Step 5: Create an OpenLDAP repository* on page 228. Then in *Step 7: Use the OpenLDAP policy to protect management services* on page 232, you must hook up this new LDAP policy to the `/` path.

Configure the example test policy

Perform the following steps:

1. In Policy Studio, create a new project based on the Admin Node Manager configuration. For example:

   ```
   INSTALL_DIR\apigateway\conf\fed
   ```

2. For the example policy, select **Policies** > **Management Services** > **Sample LDAP Policies** > **Protect Management Interfaces (LDAP)** when the Admin Node Manager configuration is loaded. This policy is summarized at a high-level as follows:

- **Scripting Language** filter returns `true` if the Node Manager is the Admin Node Manager. This enables subsequent HTTP authentication and RBAC, and updates the HTTP headers based on the user role. Otherwise, this filter calls the **Call Internal Service (no RBAC)** filter without updating the HTTP headers.
- **HTTP Basic Authentication** filter verifies the user name and password against the LDAP repository configured in *Step 5: Create an OpenLDAP repository* on page 228.
- **Retrieve Attributes from Directory Server** filter finds the LDAP groups that the user belongs to using the LDAP directory connection configured in *Step 4: Create an LDAP connection* on page 227.
- **Management Services RBAC** filter reads the user roles from the configured message attribute (`authentication.subject.role`). This returns `true` if one of the roles has access to the management service currently being invoked, as defined in the `acl.json` file. Otherwise, this returns `false` and the **Return HTTP Error 403:Access Denied (Forbidden)** policy is called because the user does not have the correct role.

3. You must edit some HTTP-based filters and change them from using the **Sample Active Directory Repository** to the repository that you created in *Step 3: Add users to the OpenLDAP RBAC group* on page 226. This repository is referenced in the following filters:

- **Authenticate login attempt**
- **HTTP Basic**

Tip You can view all referenced filters by selecting **Environment Configuration** > **External Connections** > **LDAP Repositories** > **Sample Active Directory Repository** > **Show all References**.

4. You must edit the LDAP-based filters and change them from using the **Sample Active Directory Connection** to using the LDAP connection that you created in *Step 4: Create an LDAP connection* on page 227. This repository is referenced in the following components:
   - **Read Roles from Directory Server**
   - **Re-Read Roles from Directory Server**
   - **Sample Active Directory Repository**

   Similarly, you can view all referenced components by selecting **Environment Configuration** > **External Connections** > **LDAP Repositories** > **Sample Active Directory Connection** > **Show all References**.

For more details on creating policies, see the *API Gateway Policy Developer Guide*.

Test the policy configuration

To test this policy configuration, perform the following steps:

1. In the Policy Studio tree, select **Environment Configuration** > **Listeners** > **Node Manager** > **Add HTTP Services** , and enter a service name (for example, `LDAP Test`).
2. Right-click the HTTP service, and select **Add Interface** > **HTTP** .
3. Enter an available port to test the created policy (for example, `8888`), and click **OK**.
4. Right-click the HTTP service, and select **Add Relative Path**.
5. Enter a relative path (for example, `/test`).
6. Set the **Path Specify Policy** to the **Protect Management Interfaces (LDAP)** policy, and click **OK**.
7. Close the connection to the Admin Node Manager file and reboot the Admin Node Manager so it loads the updated configuration.
8. Use API Tester to call `http://localhost:8888/test`.
9. Enter the HTTP Basic credentials (for example, username `admin` and password `Axway123`). If authentication is passed, the Admin Node Manager should return an HTTP 404 code (not found).

Step 7: Use the OpenLDAP policy to protect management services

If the authentication and RBAC filters pass, you can now use this policy to protect the management interfaces. To ensure that you do not lock yourself out of the server, perform the following steps:

1. Make a copy of the `conf/fed` directory contents from the server installation, and put it into a directory accessible from the Policy Studio.
2. Make another backup copy of the `conf/fed` directory, which will remain unmodified.
3. In the Policy Studio, select **File** > **New project**, enter a name, and click **Next**.
4. Select **From existing configuration**, and click **Next**.
5. Browse to `INSTALL_DIR/apigateway/conf/fed`, and click **Finish**.
6. Under the **Environment Configuration** > **Listeners** > **Node Manager** > **Management Services** node, select the `/` and the `/configuration/deployments` relative paths, and set the **Path Specify Policy** to the **Protect Management Interfaces (LDAP)** policy.
7. Remove the previously created `LDAP Test` HTTP Services, and close the connection to the file.
8. Copy the `fed` directory back to the Admin Node Manager's `conf` directory.
9. Reboot the Admin Node Manager.
10. Start the Policy Studio, and connect to the Admin Node Manager using `admin` with its LDAP password (for example, `Axway123`). You should now be able to edit API Gateway configurations as usual.

Manage network-level settings 9

This part contains the following:

Configure a DNS service with wildcards for virtual hosting

The Domain Name System (DNS) is a hierarchical distributed naming system for resources on the Internet or a private network. It translates domain names to numerical IP addresses used to locate services and devices worldwide, and associates details with domain names assigned to each resource. You can use wildcard DNS records to specify multiple domain names by using an asterisk in the domain name (such as `*.example.com`).

A virtual host is a server, or pool of servers, that can host multiple domain names (for example, `company1.api.example.com` and `company2.api.example.com`). To use virtual hosting for the API Gateway or API Manager, you need to be able to use different host names to refer to the same destination server. A convenient way to do this is by using a DNS service to specify wildcard entries for physical host names (for example, by setting `*.example.com` to `192.0.2.11`).

This setting enables you to run more than one website, or set of REST APIs, on a single host machine (in this case, `192.0.2.11`). Each domain name can have its own host name, paths, APIs, and so on. For example:

```
https://company1.api.example.com:8080/api/v1/test
https://company2.api.example.com:8080/api/v2/test
```

Note This topic explains how to set up DNS wildcards for virtual hosts. This is a prerequisite for configuring the API Gateway or API Manager for virtual hosts. For more details, see the *API Gateway Policy Developer Guide*.

DNS workflow

When a client makes an HTTP request to a specific URL, such as `http://www.example.com`, the client must decide which IP address to connect to. In this case, the client makes a request on the local name service for the address of the `www.example.com` machine. This usually involves the following workflow:

1. Check the `/etc/hosts` file for the specified name, and if that fails, make a DNS request.
2. Send the DNS request to the default DNS server for the client system, which refers the client to the server for `example.com` (referral), or makes the request on the client's behalf (recursion).
3. The DNS server that manages the `www.example.com` domain responds with a record that specifies the IP address of `www.example.com`.
4. The client contacts that IP address, and includes a `Host` header when making its request (`Host:www.example.com`).
5. The server can use this `Host` header to distinguish requests to different sites (virtual hosts) that use the same physical hardware and HTTP server process.

You can divide a parent domain such as `example.com` into subdomains, one for each hosted site. This provides each site with its own distinct namespace (for example, `company1.example.com`, `company2.example.com`, `company3.example.com`, and so on).

When using API Manager, you can divide the parent domain (for example, `apiportal.io`) into subdomains, one for each hosted organization. This provides each organization with its own distinct namespace (for example, `company1.apiportal.io`, `company2.apiportal.io`, `company3.apiportal.com`, and so on).

BIND DNS software

Internet Systems Consortium (ISC) BIND is the de facto standard DNS server used on the Internet. It works on a wide variety of Linux systems, and on Microsoft Windows. Windows Server operating systems can also use the Windows DNS service. The examples in this topic describe the configuration for BIND only. For more details, see http://www.isc.org/downloads/bind/.

Linux systems generally enable the configuration of BIND to be installed using a package manager. For example, for Ubuntu systems, you can use the following command:

```
$ sudo apt-get install bind9
```

The service and packages are generally called `bind`, `bind9`, or `named`. For example, on Ubuntu, you can restart BIND using the following command:

```
$ sudo /etc/init.d/bind9 start
$ sudo /etc/init.d/bind9 stop
$ sudo /etc/init.d/bind9 restart
```

Configure a wildcard domain

You can configure BIND using the `named.conf` configuration file, which is typically installed in one of the following locations:

```
/etc/named.conf /etc/bind/named.conf
```

This configuration file is typically set up with `include` entries to make configuration and upgrade easier, and which should be easy to follow. The simple example described in this topic uses a flat file only. There are two main parts to the configuration:

- `options` control the behavior of the service
- `zone` indicates how each autonomous part of the domain name tree should behave

The example shown in this topic uses the simplest options. This example also shows a single domain used for API Manager.

Configure DNS options

The following are some example DNS options that you can configure for your installation in the `named.conf` file:

```
options {
   directory "."; // e.g., /var/named
   listen-on port 8866 { any; }; // remove this for production
   pid-file "named.pid";
   allow-recursion { any; };
   allow-transfer { any; };
   allow-update { any; };
   forwarders { 10.253.253.253; 10.252.252.252; };
};
```

The example options are described as follows:

Option	Description
`directory`	This is normally set to a directory on a writeable filesystem, such as `/var/bind`. This example is set to the current directory (`"."`).
`listen-on port`	This example is set to `8866` for testing, but you should leave this blank for real DNS use in a production environment. Most DNS clients such as Web browsers always request on the standard DNS port `53`. You can also configure debugging tools such as `dig` and `nslookup` to try other ports.
`pid-file`	This example sets up `named` to run in the current directory (or `/var/named` if configured), and gives it a name to store its lock file.

Option	Description
`allow-`	The example `allow-` options are permissive, and allow any external host to use this name server, update it dynamically, and transfer a dump of its domains. This makes it unsuitable for external exposure. To restrict these `allow-` options to the local system, change the `any` settings to `127.0.0.1`.
`forwarders`	Requests that cannot be serviced locally are forwarded to the specified servers, which are the default DNS servers for the site. Specifying `forwarders` enables you to use this name server as your local DNS. It can forward requests for sites outside your domain (for example, `example.com`or `apiportal.io`) to the forwarding name servers. This enables your test environment to coexist with your normal name servers.

Configure default zones

The next step is to configure default zones for the `127.0.0.1` address. For example:

```
zone "localhost" IN {
   type master;
   file "localhost.zone";
   allow-transfer { any; };
};

zone "0.0.127.in-addr.arpa" IN {
   type master;
   file "127.0.0.zone";
   allow-transfer { any; };
};
```

These settings configure addresses for mapping `localhost` to `127.0.0.1`, and mapping `127.0.0.1` back to `localhost` when required. These example options are described as follows:

Option	Description
`type`	Specifies that this is the `master` server for `localhost`.
`file`	Specifies the domain zone file that contains the records for the domain (for more details,see *Configure domain zone files* on page 237).
`allow-transfer`	Specifies addresses that are allowed to transfer zone information from the zone server. The default `any` setting means that the contents of the domain can be transferred to anyone.

Configure logging

The following settings provide some default logging configuration:

```
logging {
   channel default_syslog {
     file "named.log";
     severity debug 10;
   };
};
```

For example, you can change the `file` setting to `/var/log/named.log`, or change the severity level.

Configure a wildcard domain

You must now configure your wildcard domain. Here you specify the name of the domain for which to serve wildcard addresses. For example:

```
zone "example.com." IN {
   type master;
   file "example.zone";
   allow-transfer { any;
   };
};
```

This is almost identical to the `localhost` zone already configured.

Similarly, for an API Manager domain:

```
zone "apiportal.io." IN {
   type master;
   file "apiportal.zone";
   allow-transfer { any;
   };
};
```

Configure domain zone files

Finally, your `zone` configuration now includes references to two separate domain zone files (in this case, `localhost.zone` and your wildcard zone (`example.zone` or `apiportal.zone`). These domain zone files use a standard format, which is defined in RFC 1035: http://www.ietf.org/rfc/rfc1035.txt. For more details, see also Wikipedia:http://en.wikipedia.org/wiki/Zone_file.

The domain zone files dictate how the server responds to requests for data in the specified zone. For example, your basic `localhost.zone` domain is as follows:

```
@  IN SOA  root.acmecorp.com. (
           20131021  ; serial number of zone file (yyyymmdd##)
           3H        ; refresh time
           15M       ; retry time in case of problem
           1W        ; expiry time
           1D )      ; maximum caching time in case of failed lookups
   IN NS   @
   IN A    127.0.0.1
```

In this file, `@` is shorthand for the domain, and describes the first (and only) record in the file. `@` specifies the name of the zone, and has the following associated records:

- `SOA` specifies details about the zone, including various serial and timer settings
- `NS` specifies that the name server for this domain is `localhost`
- `A` specifies that the address for localhost is `127.0.0.1`

Your wildcard domain is similar. For example, the contents of `example.zone` or `apiportal.zone` are as follows:

```
@  IN SOA  . root.acmecorp.com. (
             20130903  ; serial number of zone file (yyyymmdd##)
             604800    ; refresh time
             86400     ; retry time in case of problem
             2419200   ; expiry time
             604800)   ; maximum caching time in case of failed lookups
   IN NS     .
   IN A      192.168.0.10
*  IN A      192.168.0.10
```

This domain has the `SOA`, `NS`, and `A` records like the `localhost` zone, but also adds a `*` record. This matches any subdomain of `example.com` or `apiportal.io` to resolve to the specified IP address.

Manage ActiveMQ messaging

10

This part contains the following:

Manage embedded ActiveMQ messaging

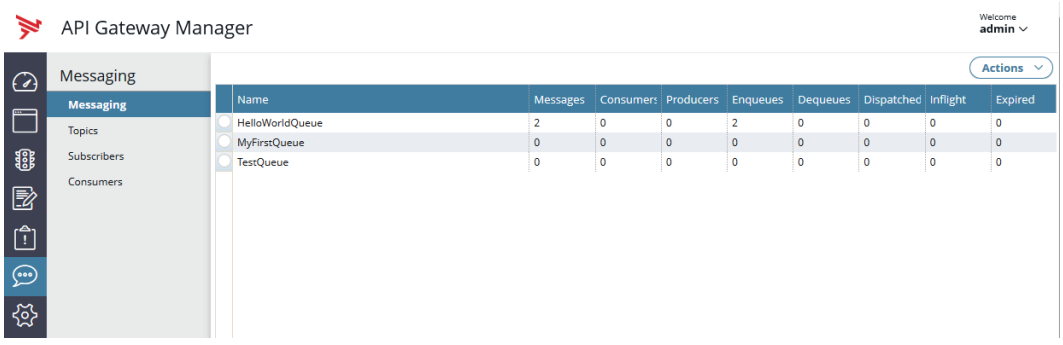
The **Messaging** view in API Gateway Manager enables you to manage the Apache ActiveMQ messaging broker that is embedded in the API Gateway instance. For example, this includes managing JMS message queues, topics, subscribers, and consumers, monitoring server connections, and so on.

For more details on Apache ActiveMQ, see http://activemq.apache.org/. For details on the embedded ActiveMQ architecture, see the *API Gateway Concepts Guide*.

Note The **Messaging** view is disabled by default. Before you can use the features in this view, you must first enable the ActiveMQ messaging broker and configure a shared directory in Policy Studio. For more details, see *Embedded ActiveMQ settings* on page 276.

Manage messaging queues

The **Queues** tab displays the messaging queues that exist in the selected API Gateway instance, along with queue statistics. For example, these include the number of messages in the queue, the number of consumers, and so on:



You can use the **Actions** drop-down menu on the right of the screen to perform the following tasks:

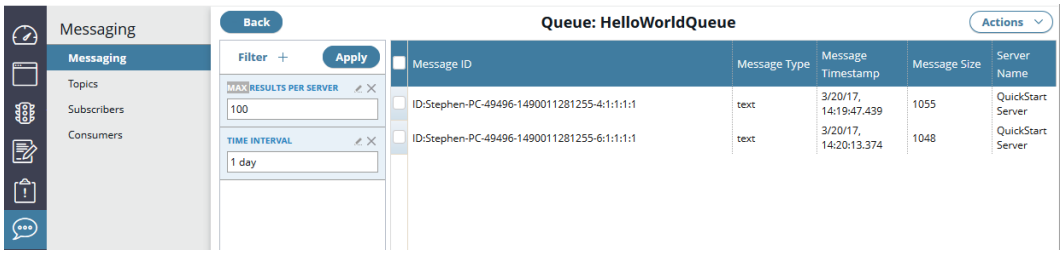
- Create a new queue
- Delete an existing queue

- Purge a queue (remove all its messages)
- Refresh the list of queues

For example, to create a new queue, select **Actions > Create Queue**, and enter a name for the queue in the dialog. The newly created queue is displayed in the list of queues.

Manage messages in a queue

When you select a queue name on the **Queues** tab, this displays the messages contained in the queue. For example:



You can filter the messages displayed in this screen using the filter panel on the left. By default, you can configure the number of messages displayed, or the time interval for messages. Click the **Filter** button to add more viewing options (for example, **JMS Message ID** or **JMS Type**).

You can use the **Actions** drop-down menu on the right of the screen to perform the following tasks:

- Create a message in the queue
- Copy selected messages to a different queue
- Move selected messages to a different queue
- Delete selected messages from the queue

Create a new message

To create a new message in the selected queue, perform the following tasks:

1. Select **Actions > Create Message**, and complete the following **Message Details**:

Setting	Description
Destination API Gateway	Specifies the required destination API Gateway for the message (for example `APIGateway1`).
Message Type	Specifies the required type of the message (`Text` or `Binary`). Defaults to `Text`.
JMS Reply To	Specifies the optional destination to send a reply message to (for example, JMS queue or topic). It is the responsibility of the application that consumes the message from the queue (JMS consumer) to send the message back to the specified destination.
Time to live	By default, the message never expires. However, if a message becomes obsolete after a certain period, you can set an optional expiration time (in milliseconds). If the specified value is `0`, the message never expires.
JMS Delivery Mode	Select one of the following delivery modes: • **Persistent**: Instructs the JMS provider to ensure that a message is not lost in transit if the JMS provider fails. A message sent with this delivery mode is logged to persistent storage when it is sent. This is the default mode. • **Non-persistent**: Does not require the JMS provider to store the message. With this mode the message may be lost if the JMS provider fails.
JMS Correlation ID	Specifies an optional identifier used to correlate response messages with the corresponding request messages. For example, you could enter the `${id}` message attribute selector to correlate request messages with their correct response messages.
JMS Priority	Specifies an optional message priority level to deliver urgent messages first. Priority levels are from `0` (lowest) to `9` (highest). If you do not specify a priority level, the default level is 4.
JMS Type	Specifies an optional user-defined message header that is defined as an arbitrary string. This is used to identify the message structure or payload (for example, the XML schema that the payload conforms to).

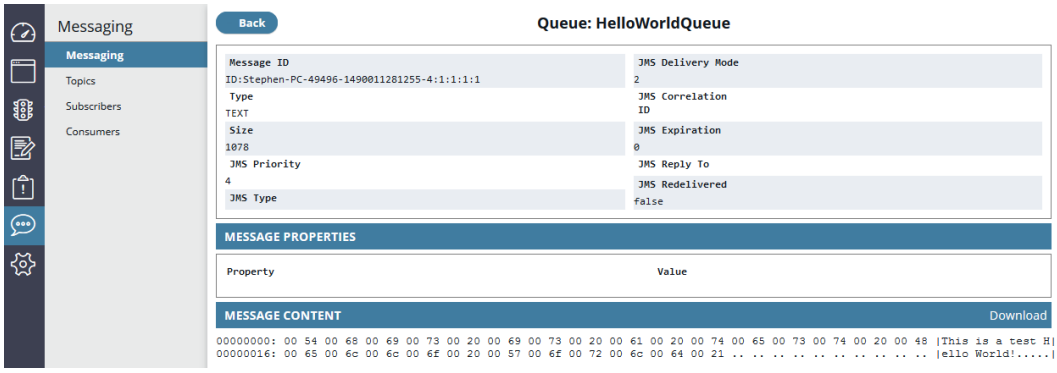
2. Click next to specify **User-defined properties**. Custom properties may be required for compatibility with other messaging systems. You can use message attribute selectors as property values. For example, you can create a property called `AuthNUser`, and set its value to `${authenticated.subject.id}`. Other applications can then filter on this

property—such as only consume messages where `AuthNUser` equals `admin`. To add a new property, enter the property name and property value in the text box. You can click the green plus sign to add more properties.

3. Click next to specify the message payload content. If you selected a **Message Type** of `Text`, you can enter a **Text Payload** (for example, `Hello World!`). Alternatively, if you selected a **Message Type** of `Binary`, click **Select File**, and browse to the file that you wish to send.
4. Click **Send**.

View message contents

When you have created a message in a queue, the message is displayed in the list of messages on the **Queues** tab. You can click a message in the list to display a detailed view of the message properties and message body contents. To save the message body to a file, click the **Download** link on the bottom right of the screen.



Manage messaging topics

The **Topics** tab displays the messaging topics that exist in the selected API Gateway instance, or across the selected API Gateway group. The **Actions** drop-down menu on the right enables you to perform the following tasks:

- Create a new topic
- Delete a topic
- Refresh the list of topics

To create a new topic, select **Actions > Create Topic**, and enter the required topic name in the dialog.

Unlike queues, where messages posted to the queue are kept until someone reads them, messages posted to the topic are broadcasted to all topic subscribers, and then immediately removed from the topic. This means that when you click a newly created topic in the list, the **Actions** menu on the right includes only the **Create Message** action to enable you to post a new message to the topic. Delete or move actions are not required because the topic is always empty.

Manage messaging subscribers

The **Subscribers** tab displays the list of durable subscribers that have registered to receive messages from a specified publisher (for example, messaging topic). In this model, the subscriber and the publisher are not aware of each other. The **Actions** drop-down menu on the right enables you to perform the following tasks:

- Create a new subscriber
- Delete a subscriber
- Refresh the list of subscribers

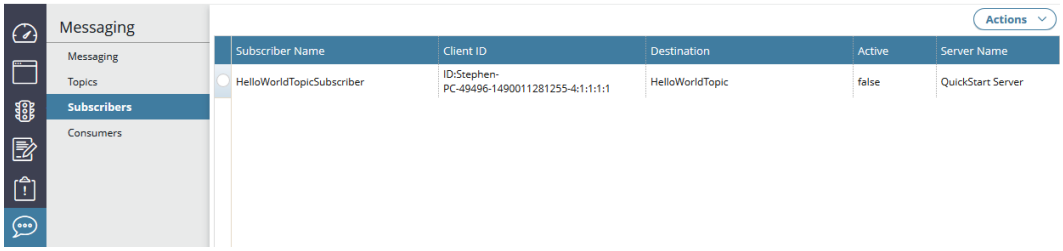
Create a new subscriber

To create a new subscriber, select **Actions > Create Subscriber**, and complete the following details:

Setting	Description
Subscriber Name	Specifies a name for this subscriber (for example `HelloWorldTopicSubscriber`).
Client ID	Specifies a unique client ID used for the subscriber connection. If you do not specify a client ID, one is generated by default (for example `ID:<hostname>-60862-1392656015480-28:1`). The client ID is required for durable subscriptions, which enable a client to disconnect or fall over while consuming a topic, and retrieve any missed messages when it reconnects. To achieve this, the broker needs the client ID to identify which messages are pending consumption.
Destination	Specifies the JMS destination being subscribed to (for example, topic name).
Selector	Specifies a JMS selector used to attach a filter to a subscription and perform content-based routing. JMS selectors are defined using SQL 92 syntax, and typically apply to message headers. For example: `JMSType = 'car' AND color = 'blue' AND weight >250` For more details, see http://activemq.apache.org/selectors.html.

Note JMS selectors and filters in ActiveMQ are in no way related to API Gateway selectors and filters. For more details on API Gateway selectors and filters, see the *API Gateway Policy Developer Guide*.

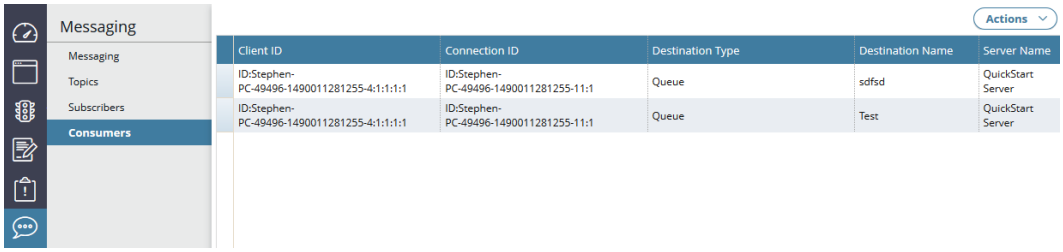
The following screen shows a newly created subscriber:



Manage messaging consumers

The **Consumers** tab displays the list of JMS consumers that are currently connected to this embedded ActiveMQ server. This includes details such as client and connection IDs, destination type and name, and server name.

Then when you click a consumer in the list, this displays more detailed information on that consumer. For example:



For details on creating JMS consumers in Policy Studio, see the *API Gateway Policy Developer Guide*.

API Gateway settings reference 11

This part contains the following:

Cassandra settings

The **Cassandra** settings in Policy Studio enable you to configure settings for the external Apache Cassandra database used to store API Gateway and API Manager data. For example, you can use a Cassandra database to store data used by the following components:

- API Manager: Client registry, API catalog, and quota management
- Key Property Store: Custom table definitions and data
- OAuth token store
- Client registry: API key and OAuth solutions using API Gateway only
- Throttling: Smooth rate limiting

Note You do not need to configure **Cassandra** settings if you do not use Apache Cassandra for any of these features.

To configure **Cassandra** settings, select the **Server Settings > Cassandra** node in the Policy Studio tree. To apply updates to these settings, click **Apply changes** at the bottom right of each page.

For more details on installing and configuring an Apache Cassandra database cluster, see the *API Gateway Installation Guide*.

Cassandra Keyspace settings

Configure the following in the **Keyspace** settings:

Keyspace name:
Specifies the Cassandra keyspace used to store tables and data. This is a unique namespace used to define data replication on cluster nodes. Defaults to `x${DOMAIN_ID}_${GROUP_ID}`. You can enter text, environment variables, and the following API Gateway-specific variables:

- `${DOMAINID}`: Topology ID of the API Gateway system
- `${GROUPID}`: API Gateway group ID
- `${GROUPNAME}`: API Gateway group name

Note The keyspace name must be unique and less than 48 permitted characters. These include underscore (_) and alphanumeric characters (`a-z, A-Z, 0-9`).

Initial replication strategy:
Specifies the initial Cassandra replication strategy used to determine the nodes where replicas are placed:

- **Simple Strategy**: Use this setting for a single data center and one rack only. This is the default setting. If you ever intend more than one datacenter, use the Network Topology Strategy instead.
- **Network Topology Strategy**: This setting is recommended for most deployment because it is much easier to expand to multiple datacenters when required by future expansion.

Initial replication:
Specifies the initial Cassandra replication factor used to create the keyspace only. Defaults to `1`. Then to modify the replication factor after the keyspace is created, use the `cqlsh` command in `CASSANDRA_INSTALL_DIR/cassandra/bin`.

Note On startup, if the keyspace does not already exist, the keyspace is updated with this value. This is the only time this value is used. If you need to update the replication factor, you must use `cqlsh` (located in `cassandra/bin`). When this value is changed, you must run the `node repair` command on all nodes in the Cassandra cluster.

For more details, see your Cassandra documentation.

Cassandra Hosts settings

To add a Cassandra server host in the **Hosts** settings, click **Add** at the bottom right, and configure the following required settings in the dialog:

- **Name**:
  Enter a user-friendly name for the Cassandra server host. Defaults to `Local Cassandra server`.

- **Host**:
  Enter the host name or IP address of the Cassandra server host. Defaults to `localhost`.
- **Port**:
  Enter the port on which the API Gateway client listens on the Cassandra native protocol. Defaults to `9042`.

Cassandra Authentication settings

If Cassandra authentication is enabled, configure the following in the **Authentication** settings:

- **Username**:
  Enter the user name to used to establish a connection with Cassandra. This field can be environmentalized. The user name can be formatted as follows:
  - Unquoted string containing only alphanumeric characters
  - Single-quoted string containing any printable character (including non-alphanumeric)
- **Password**:
  Enter the password to establish a connection with Cassandra. This field can be environmentalized.

Note Cassandra authentication is disabled by default. For details on enabling Cassandra authentication, see Internal authentication documentation.

Cassandra Security settings

Configure the following in the **Security** settings:

- **Enable SSL**:
  Select whether to use Secure Sockets Layer (SSL) to establish secured connections to the Cassandra server. This option is deselected by default. When **Enable SSL** is selected, you can configure the following:
  - **Trusted certificates**:
    Click to select the list of certificates or certificate authorities trusted when validating Cassandra server certificates. This is required when SSL is enabled. For details on importing certificates, see *Manage X.509 certificates and keys* on page 106.
  - **Client certificate**:
    Click to select the client certificate and key to use if client authentication is required by the Cassandra server (also known as SSL mutual authentication). For details on importing certificates, see *Manage X.509 certificates and keys* on page 106.
  - **Accepted cipher suites**:
    Select which cipher suites are only allowed when establishing SSL connections to the Cassandra server. This setting is optional. For details on the default cipher suites, see the SunJSSE Provider documentation.

- **Do not use the SSLv3 protocol**:
  Specifies not to use SSL v3 to avoid any weaknesses in this protocol. This option is selected by default (SSLv3 is not supported by Cassandra 2.x and above). This option is provided for backward compatibility when connecting to legacy Cassandra installations.
- **Do not use the TLSv1 protocol**:
  Specifies not to use TLS v1 to avoid any weaknesses in this protocol. TLSv1 is supported by Cassandra 2.x.
- **Do not use the TLSv1.1 protocol**:
  Specifies not to use TLS v1.1 to avoid any weaknesses in this protocol. TLSv1.1 is supported by Cassandra 2.x.
- **Do not use the TLSv1.2 protocol**:
  Specifies not to use TLS v1.2 to avoid any weaknesses in this protocol. TLSv1.2 is supported by Cassandra 2.x. The default protocol is TLS v1.2.

Throttling settings

Configure the following in the **Throttling** settings:

Keyspace name:

Specifies the Cassandra keyspace used to store tables for throttling and smooth rate limit data. This is a unique namespace used to define data replication on cluster nodes. Defaults to `t${DOMAIN_ID}_throttling`. You can enter text, environment variables, and the following API Gateway-specific variables:

- `${DOMAINID}`: Topology ID of the API Gateway system
- `${GROUPID}`: API Gateway group ID
- `${GROUPNAME}`: API Gateway group name

Note The keyspace name must be unique and less than 48 permitted characters. These include underscore (_) and alphanumeric characters (`a-z, A-Z, 0-9`). In addition, one throttling keyspace per group is recommended in a multi-group domain that shares the same Cassandra cluster.

Initial replication strategy:
Specifies the initial Cassandra replication strategy used to determine the nodes where replicas are placed:

- **Simple Strategy**: Use this setting for a single data center and one rack only. This is the default setting. If you ever intend more than one datacenter, use the Network Topology Strategy instead.
- **Network Topology Strategy**: This setting is recommended for most deployments because it is much easier to expand to multiple datacenters when required by future expansion.

Initial replication:
Specifies the initial Cassandra replication factor used to create the throttling keyspace only. Defaults to `1`. Then to modify the replication factor after the keyspace is created, use the `cqlsh` command in `CASSANDRA_INSTALL_DIR/cassandra/bin`.

Note On startup, if the throttling keyspace does not already exist, the keyspace is updated with this value. This is the only time this value is used. If you need to update the replication factor, you must use `cqlsh` (located in `cassandra/bin`). When this value is changed, you must run the `node repair` command on all nodes in the Cassandra cluster. For more details, see your Cassandra documentation.

Read consistency level:
Select the consistency level for Cassandra read operations from the list. Defaults to `ONE`.

Write consistency level:
Select the consistency level for Cassandra write operations from the list. Defaults to `ONE`.

For more details on Cassandra consistency levels, see the Consistency level documentation.

Further information

For more details on throttling and rate limiting, see the following:

- "Configure rate limiting" in the *API Gateway Policy Developer Guide*
- "Throttling" in the *API Gateway Policy Developer Filter Reference*

General settings

The top-level **General** settings screen in Policy Studio enables you to set global configuration settings to optimize the behavior of API Gateway for your environment.

To configure these settings, in the Policy Studio tree, select the **Server Settings** node, and click **General**. To confirm updates to these settings, click **Apply changes** at the bottom right of the screen.

After changing any settings, you must deploy to API Gateway for the changes to be enforced. You can do this in the Policy Studio main menu by selecting **Server > Deploy**. Alternatively, click the **Deploy** button in the toolbar, or press F6.

Settings

You can configure the following settings in the **General** screen:

Setting	Description
Tracing level	Enables you to set the trace level for API Gateway at runtime. Select the appropriate option from the drop-down list. Defaults to `INFO`.

Setting	Description
Active timeout	When the API Gateway receives a large HTTP request, it reads the request off the network when it becomes available. If the time between reading successive blocks of data exceeds the **Active Timeout** specified in milliseconds, API Gateway closes the connection. This guards against a host closing the connection while sending data. For example, if the host's network connection is pulled out of the machine while in the middle of sending data to API Gateway. When API Gateway has read all the available data off the network, it waits the **Active Timeout** period before closing the connection. Defaults to `30000` milliseconds. The **Active Timeout** value is also used as a wait time when the maximum number of connections for a host is reached. For example, when a host reaches the **Maximum connections** value, API Gateway waits the active timeout period before giving up on trying to make a new connection. The global default value for **Maximum connections** is `128` and cannot be changed. However, you can configure **Maximum connections** and **Active Timeout** on a per-host basis using the **Remote Hosts** interface. For more details, see the *API Gateway Policy Developer Guide*.
Date format	Configures the format of the date for the purposes of transaction audit logging and historic metrics. Defaults to `MM.dd.yyyy HH:mm:ss,SSS`. For more details on this format, see http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html. See also *Transaction audit log settings* on page 258.
Cache refresh interval	Configures the number of seconds that the server caches data loaded from an external source before refreshing data from that source. Defaults to `5` seconds. To disable the cache, set this to `0`. This cache applies to attributes retrieved from external databases, LDAP directories, internal user stores, and IBM Tivoli. It also applies to query results for authentication against LDAP or databases, and to certificate revocation lists for certificate validation (CRL and XKMS only).
Transaction timeout	A configurable transaction timeout that detects slow HTTP attacks (slow header write, slow body write, slow read) and rejects any transaction that keeps the worker threads occupied for an excessive amount of time. The default value is `240000` milliseconds.
Maximum sent bytes per transaction	The maximum number of bytes sent in a transaction. This is the maximum length for the transmitted data on transactions that API Gateway can handle. This helps to prevent denial-of-service (DoS) attacks. This setting limits the entire amount of data sent over the link, regardless of whether it consists of body, headers, or request line. The default value is `20` MB (`20971520` bytes).

Setting	Description
Maximum received bytes per transaction	The maximum number of bytes received in a transaction. This is the maximum length for the received data on transactions that API Gateway can handle. This helps to prevent denial-of-service (DoS) attacks. This setting limits the entire amount of data received over the link, regardless of whether it consists of the body, headers, or request. The default value is `20` MB (`20971520` bytes). **Note** In a multi-datacenter environment with a large amount of APIs and data, you may need to increase this value to optimize the performance of the API Manager web console. For more details, see "Configure API Management in multiple datacenters" in the *API Gateway Installation Guide*.
Idle timeout	API Gateway supports HTTP 1.1 persistent connections. The **Idle Timeout** specified in milliseconds is the time that API Gateway waits after sending a message over a persistent connection before it closes the connection. Typically, the host tells API Gateway that it wants to use a persistent connection. API Gateway acknowledges this instruction and decides to keep the connection open for a certain amount of time after sending the message to the host. If the connection is not reused within the **Idle Timeout** period, API Gateway closes the connection. Defaults to `15000` milliseconds. You can configure this setting on a per-host basis using the **Remote Hosts** interface. For more details, see the *API Gateway Policy Developer Guide*.
LDAP service provider	Specifies the service provider used for looking up an LDAP server (for example, `com.sun.jndi.ldap.LdapCtxFactory`). The provider is typically used to connect to LDAP directories for certificate and attribute retrieval.
Maximum memory per request	The maximum amount of memory in bytes that API Gateway can allocate to each request. This setting helps protect against denial of service caused by undue pressure on memory. You also can configure this setting at the HTTP/S interface level. For more details, see the *API Gateway Policy Developer Guide*. **Note** As a general rule for XML messages, if you need to process a message of size N, you should allocate 6–7 times N amount of memory.
Realm	Specifies the realm for authentication purposes.
Schema pool size	Sets the size of the Schema Parser pool.
Server brand	Specifies the branding to be used in API Gateway.

Setting	Description
SSL session cache size	Specifies the number of idle SSL sessions that can be kept in memory. You can use this setting to improve performance because the slowest part of establishing the SSL connection is cached. A new connection does not need to go through full authentication if it finds its target in the cache. Defaults to `32`. If there are more than 32 simultaneous SSL sessions, this does not prevent another SSL connection from being established, but means that no more SSL sessions are cached. A cache size of `0` means no cache, and no outbound SSL connections are cached.
Token drift time	Specifies the number of seconds drift allowed for WS-Security tokens. This is important in cases where API Gateway is checking the date on incoming WS-Security tokens. It is likely that the machine on which the token was created is out-of-sync with the machine on which API Gateway is running. The drift time allows for differences in the respective machine clock times.
Allowed number of operations to limit XPath transforms	Specifies the total number of node operations permitted in XPath transformations. Complex XPath expressions (or those constructed together with content to produce expensive processing) might lead to a denial-of-service risk. Defaults to `4096`.
Input encodings	Click the browse button to specify the HTTP content encodings that API Gateway instance can accept from peers. The available content encodings include `gzip` and `deflate`. Defaults to no context encodings. For more details, see the *API Gateway Policy Developer Guide*.
Output encodings	Click the browse button to specify the HTTP content encodings that API Gateway instancecan apply to outgoing messages. The available content encodings include `gzip` and `deflate`. Defaults to no context encodings. For more details, see the *API Gateway Policy Developer Guide*.
Server's SSL cert's name must match name of requested server	Ensures that the certificate presented by the server matches the name of the host address being connected to. This prevents host spoofing and man-in-the-middle attacks. This setting is enabled by default.

Setting	Description
Send desired server name to server during TLS negotiation	Specifies whether to add a field to outbound TLS/SSL calls that shows the name that the client used to connect. For example, this can be useful if the server handles several different domains, and needs to present different certificates depending on the name that the client used to connect. This setting is not selected by default.
Add correlation ID to outbound headers	Specifies whether to insert the correlation ID in outbound messages. For the HTTP transport, this means that an `X-CorrelationID` header is added to the outbound message. This is a transaction ID that is tagged to each message transaction that passes through API Gateway, and which is used for traffic monitoring in the API Gateway Manager web console. You can use the correlation ID to search for messages in the console. You can also access the its value using the `id` message attribute in an API Gateway policy. An example correlation ID value is `Id-54bbc74f515d52d71a4c0000`. This setting is selected by default.

MIME settings

The MIME settings list a number of default common content types that are used when transmitting Multipurpose Internet Mail Extensions (MIME) messages. You can configure API Gateway's **Content Type** filter to accept or block messages containing specific MIME types. Therefore, the contents of the MIME types library act as the set of all MIME types that API Gateway can filter messages with.

All of the MIME types listed in the table are available for selection in the **Content Type** filter. For example, you can configure this filter to accept only XML-based types, such as `application/xml`, `application/*+xml`, `text/xml`, and so on. Similarly, you can block certain MIME types (for example, `application/zip`, `application/octet-stream`, and `video/mpeg`).

For more details on configuring the **Content Type** filter, see the *API Gateway Policy Developer Guide*.

Configuration

To configure the MIME settings, in the Policy Studio main menu, select **Tasks > Manage Gateway Settings > General > MIME**. Alternatively, in the Policy Studio tree, select the **Environment Configuration > Server Settings** node, and click **General > MIME**. To confirm updates to these settings, click **Apply changes** at the bottom right of the screen.

The MIME settings screen lists the actual MIME types on the left column of the table, together with their corresponding file extensions (where applicable) in the right column.

To add a new MIME type, click the **Add** button. In the **Configure MIME Type** dialog, enter the new content type in the **MIME Type** field. If the new type has a corresponding file extension, enter this extension in the **Extension** field. Click the **OK** button when finished.

Similarly, you can edit or delete existing types using the **Edit** and **Delete** buttons.

Namespace settings

API Gateway exposes global settings that enable you to configure which versions of the SOAP and WSSE specifications it supports. You can also specify which attribute is used to identify the XML Signature referenced in a SOAP message.

To configure the namespace settings, in the Policy Studio tree, select the **Environment Configuration > Server Settings** node, and click **General > Namespaces**. Alternatively, in the Policy Studio main menu, select **Tasks > Manage Gateway Settings > General > Namespaces**. To confirm updates to these settings, click **Apply changes** at the bottom right of the screen.

SOAP Namespace

The **SOAP Namespace** tab can be used to configure the SOAP namespaces that are supported by API Gateway. In a similar manner to the way in which API Gateway handles WSSE namespaces, API Gateway will attempt to identify SOAP messages belonging to the listed namespaces in the order given in the table.

The default behavior is to attempt to identify SOAP 1.1 messages first, and for this reason, the SOAP 1.1 namespace is listed first in the table. API Gateway will only attempt to identify the message as a SOAP 1.2 message if it can't be categorized as a SOAP 1.1 message first.

Signature ID Attribute

The **Signature ID Attribute** tab allows you to list the supported attributes that can be used by API Gateway to identify a Signature reference within an XML message.

An XML-signature `<signedInfo>` section may reference signed data via the `URI` attribute. The `URI` value may contain an id that identifies data in the message. The referenced data will hold the "URI" field value in one of its attributes.

By default, the server uses the `Id` attribute for each of the WSSE namespaces listed above to locate referenced signed data. The following sample XML Signature illustrates the use of the `Id` attribute:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Header>
      <dsig:Signature id="Sample" xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
       <dsig:SignedInfo>
```

```
        ...
        <dsig:Reference URI="#Axway:sLmDCph3tGZ10">
         ...
        </dsig:Reference>
      </dsig:SignedInfo>
      ....
     </dsig:Signature>
  </soap:Header>
  <soap:Body>
      <getProduct wsu:Id="Axway:sLmDCph3tGZ10"
          xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
          <Name>SOA Test Client</Name>
          <Company>Company</Company>
      </getProduct>
  </soap:Body>
</soap:Envelope>
```

It is clear from this example that the Signature reference identified by the `URI` attribute of the `<Reference>` element refers to the nodeset identified with the `Id` attribute (the `<getProduct>` block).

Because different toolkits and implementations of the XML-Signature specification can use attributes other than the `Id` attribute, API Gateway allows the user to specify other attributes that should be supported in this manner. By default, API Gateway supports the `Id`, `ID`, and `AssertionID` attributes for the purposes of identifying the signed content within an XML Signature.

However you can add more attributes by clicking the **Add** button and adding the attribute in the interface provided. The priorities of attributes can be altered by clicking the **Up** and **Down** buttons. For example, if most of the XML Signatures processed by API Gateway use the `ID` attribute, this attribute should be given the highest priority.

WSSE Namespace

The **WSSE Namespace** tab is used to specify the WSSE (and corresponding WSSU) namespaces that are supported by API Gateway.

API Gateway attempts to identify WS Security blocks belongingto the WSSE namespaces listed in this table. It first attempts to locate Security blocks belonging to the first listed namespace, followed by the second, then the third, and so on until all namespaces have been utilized. If no Security blocks can be found for any of the listed namespaces, the message will be rejected on the grounds that API Gateway does not support the namespace specified in the message.To add a new namespace, click the add button.

Note Every WSSE namespace has a corresponding WSSU namespace. For example, the following WSSE and WSSU namespaces are inextricably bound:

WSSE Namespace	`http://schemas.xmlsoap.org/ws/2003/06/secext`
WSSU Namespace	`http://schemas.xmlsoap.org/ws/2003/06/utility`

First, enter the WSSE namespace in the **Name** field. Then enter the corresponding WSSU namespace in the **WSSU Namespace** field.

Further information

For details on XML Signature generation and verification filters, see the *API Gateway Policy Developer Guide*.

HTTP Session settings

The **HTTP Session** settings enable you to configure session management settings for the selected cache. For example, you can configure the period of timebefore expired sessions are cleared from the `HTTP Sessions` cache, which is selected by default.

To configure HTTP session settings, select the **Environment Configuration > Server Settings** node in the Policy Studio tree, and click **General > HTTP Session**.Alternatively, in the Policy Studio main menu, select **Tasks > ManageGateway Settings > General > HTTP Session**. To confirm updates to these settings, click **Apply changes** at the bottom right of the screen.

Configuration

Configure the following session settings:

Cache:
Specifies the cache that you wish to configure. Defaults to `HTTP Sessions`.To configure a different cache, click the button on the right, and select the cache touse. The list of currently configured caches is displayed in the tree.

To add a cache, right-click the **Caches** tree node, and select **Add Local Cache** or **Add Distributed Cache**. Alternatively, you can configure caches under the **Environment Configuration > Libraries** node in the Policy Studio tree. For more details, see the *API Gateway Policy Developer Guide*.

Clear Expired Sessions Period:
Enter the number of seconds before expired sessions are cleared from the selected cache. Defaults to `60`.

Zero downtime settings

The **Zero Downtime** settings enable you to configure zero downtime deployment and zero downtime shutdown. You can enable zero downtime deployment and set delays before and after deployment. You can also enable zero downtime shutdown and set the delay before shutdown.

To configure zero downtime settings, select the **Server Settings** node in the Policy Studio tree, and click **General > Zero Downtime**. To confirm updates to these settings, click **Save** at the bottom right of the window.

For more information of performing a zero downtime deployment, see *Perform zero downtime deployment* on page 145. For more information on performing a zero downtime shutdown, see *Perform zero downtime shutdown* on page 80.

Prerequisites

Zero downtime deployment and shutdown rely on the **Health Check LB** policy to alert the load balancer when a maintenance operation is about to begin. To use the zero downtime deployment or shutdown features, the Health Check LB policy must be present in your API Gateway configuration.

New projects

The Health Check LB policy is included in the default factory configuration.

When creating a new project in Policy Studio, choose **From a template configuration** and select the **Factory template with samples** template. This template includes the Health Check LB policy.

Existing projects

If you created a project in Policy Studio from any other template (for example, **Factory template**, **Team Development – Common Project**, or **Team Development – API Project**), you must manually import the Health Check LB policy into your configuration.

To import the Health Check LB policy, select **File > Import > Import Configuration Fragment** from the Policy Studio main menu, and select the following file:

```
$VDISTDIR/samples/SamplePolicies/HealthCheck/HealthCheckLB.xml
```

You must also add the Health Check LB policy to a listener so that the load balancer can ping it to determine the health of the API Gateway (for example, path `/healthchecklb` on HTTP port `8080`). For more information on listeners, see the *API Gateway Policy Developer Guide*.

Tip You can customize the Health Check LB policy for your own environment. For example, you can modify the response code and message that are returned to the load balancer.

Configuration

Configure the following zero downtime settings:

Zero-downtime deployment enabled:

Select the check box to enable zero downtime deployment. The default is disabled.

Delay before deployment:

Enter the delay before deployment in seconds. This is the delay in the API Gateway between receiving a deployment request and starting the deployment. During this delay, the Health Check LB policy returns an error response, giving the load balancer time to route traffic away from the API Gateway before the deployment begins. The default is 10 seconds. You can choose a value between 1 and 20 seconds.

Delay after deployment:

Enter the delay after deployment in seconds. This is the delay in the API Gateway between the end of deployment and a response being sent to the deployment request. During this delay, the Health Check LB policy returns a successful response, giving the load balancer time to start routing traffic to the API Gateway before deployment begins on the next API Gateway in the group. The default is 10 seconds. You can choose a value between 1 and 20 seconds.

Zero-downtime shutdown enabled:

Select the check box to enable zero downtime shutdown. The default is disabled.

Delay before shutdown:

Enter the delay before shutdown in seconds. This is the delay in the API Gateway between receiving a shutdown request and starting the shutdown procedure. During this delay, the Health Check LB policy returns an error response, giving the load balancer time to route traffic away from the API Gateway before shutdown begins. The default is 10 seconds. You can choose a value between 1 and 20 seconds.

Transaction audit log settings

One of the most important features of a server-based product is its ability to maintain highly detailed and configurable logging. It is crucial that a record of each and every transaction is kept, and that these records can easily be queried by an administrator to carry out detailed transaction analysis. In recognition of this requirement, the API Gateway provides detailed logging to a number of possible locations.

You can configure the API Gateway so that it logs information about all requests. Such information includes the request itself, the time of the request, where the request was routed to, and the response that was returned to the client. The logging information can be written to the console, log file, local/remote syslog, and/or a database, depending on what is configured in the logging settings.

The API Gateway can also digitally sign the logging information it sends to the log files and the database. This means that the logging information can not be altered after it has been signed, thus enabling an irreversible audit trail to be created.

Caution The transaction audit log includes the complete contents of HTTP requests, including HTTP headers, body, and attachments. This may include sensitive information. You must ensure that appropriate safeguards are in place to protect this information in the different audit log locations.

Configure log output

To edit the default API Gateway logging settings in the Policy Studio tree, select the **Server Settings** node, and click **Logging > Transaction Audit Log**.

You can configure the API Gateway to log to the following locations described in this topic.

Log to Text File

To configure the API Gateway to log in text format to a file, click the **Text File** tab, and select **Enable logging to file**. You can configure the following fields:

- **File Name**:
  Enter the name of the text-based file that the API Gateway logs to. The default is `transactionLog`.
- **File Extension**:
  Enter the file extension of the log file. Defaults to `.log`.
- **Directory**:
  Enter the directory of the log file in this field. By default, all log files are stored in the `/logs/transaction` directory of your API Gateway installation.
- **File Size (MB)**:
  Enter the maximum size that the log file grows to. When the file reaches the specified limit, a new log file is created. By default, the maximum file size is 1000 MB.
- **Roll Log Daily**:
  Specify whether to roll over the log file at the start of each day. This is enabled by default.
- **Number of Files**:
  Specify the number of log files that are stored. The default number is 20.
- **Format**:
  You can specify the format of the logging output using the values entered here. You can use selectors to output logging information that is specific to the request. The default logging format is as follows:

```
${level} ${timestamp} ${id} ${text} ${filterType} ${filterName}
```

  The available logging properties are described as follows:

  - **`level`**: The log level (`fatal`, `fail`, `success`).
  - **`timestamp`**: The time that the message was processed in user-readable form. For more details, see **Date format** in *General settings* on page 249.
  - **`id`**: The unique transaction ID assigned to the message.
  - **`text`**: The text of the log message that was configured in the filter itself. In the case of the **Log Message Payload** filter, the `${payload}` selector contains the message that was sent by the client.

- **`filterName`**: The name of the filter that generated the log message.
- **`filterType`**: The type of the filter that logged the message.
- **`ip`**: The IP address of the client that sent the request.

- **Signing Key**:
  To sign the log file, select a **Signing Key** from the Certificates Store that is used in the signing process. By signing the log files, you can verify their integrity at a later stage.

To confirm updates to these settings, click **Save** at the bottom right of the screen.

Log to XML File

To configure the API Gateway to log to an XML file, click the **XML File** tab, and select **Enable logging to XML file**. The log entries are written as the values of XML elements in this file. You can view historical XML log files (not the current file) as HTML for convenience by opening the XML file in your default browser. The `/logs/xsl/MessageLog.xsl` stylesheet is used to render the XML log entries in a more user-friendly HTML format.

You can configure the following fields on the **XML File** tab:

- **File Name**:
  Enter the name of the text-based file that the API Gateway logs to. By default, the log file is called `axway`.
- **File Extension**:
  Enter the file extension of the log file in this field. By default, the log file is given the `.log` extension.
- **Directory**:
  Enter the directory of the log file in this field. By default, all log files are stored in the `/logs/transaction` directory of your API Gateway installation.
- **File Size**:
  Enter the maximum size that the log file grows to. When the file reaches the specified limit, a new log file is created.By default, the maximum file size is 1000 kilobytes.
- **Roll Log Daily**:
  Specify whether to roll over the log file at the start of each day. This is enabled by default.
- **Number of Files**:
  Specify the number of log files that are persisted. The default number is 20.
- **Signing Key**:
  To sign the log file, select a **Signing Key** from the Certificates Store that will be used in the signing process. By signing the log files, you can verify their integrity at a later stage.

Log to Database

Using this option, you can configure the API Gateway to log messages to an Oracle, SQL Server, or MySQL relational database.

Note Before configuring the API Gateway to log to a database, you must first create the database tables that the API Gateway writes to. For details on setting up tables for supported databases, see the *API Gateway Installation Guide*.

When you have set up the logging database tables, you can configure the API Gateway to log to the database. Click the **Database** tab, and select **Enable logging to database** . You can configure the following fields on the **Database** tab:

- **Connection**:
  Select an existing database from the **Connection** drop-down list. To add a database connection, click the **External Connections** button on the left, right-click the **Database Connections** tree node, and select **Add a Database Connection**. For more details, see the *API Gateway Policy Developer Guide*.
- **Signing Key**:
  You can sign log messages stored in the database to ensure that they are not tampered with. Click **Signing Key** to open the list of certificates in the Certificate Store, and select the key to use to sign log messages.

Log to Local Syslog

To configure the API Gateway to send logging information to the local UNIX syslog, click the **Local Syslog** tab, and select the **Enable logging to local UNIX Syslog** checkbox. You can configure the following fields:

- **Select Syslog server**:
  Select the local syslog facility that the API Gateway should log to. The default is `LOCAL0`.
- **Format**:
  You can specify the format of the log message using the values (including selectors) entered in this field. For details on the properties that are available, see *Log to Text File* on page 259.

Log to Remote Syslog

To configure the API Gateway to send logging information to a remote syslog, click the **Remote Syslog** tab, and select the **Enable logging to Remote Syslog** checkbox. You can configure the following fields:

- **Syslog Server**:
  Select a previously configured **Syslog Server** from the list.For details on how to configure Syslog Server, see the topic on External Connections in the *API Gateway Policy Developer Guide*.
- **Format**:
  You can specify the format of the log message using the values (including properties) entered in this field. For details on the properties that are available, see *Log to Text File* on page 259.

Log to System Console

To configure the API Gateway to send logging information to the system console, click the **System Console** tab, and select the **Enable logging to system console**.

For details on how to use the **Format** field to configure the format of the log message, see *Log to Text File* on page 259.

Transaction access log settings

The access log records a summary of the request and response messages that pass through the API Gateway. By default, the API Gateway records this in the `access.log` file in the `log` directory. This file rolls over with a version number added for each new version of the file (for example, `access.log.0`, `access.log.1`, and so on).

The transaction access log file format is based on that used by Apache HTTP Server. This means that the log file can be consumed by third-party Web analytics tools such as Webtrends to generate charts and statistics.

Access log format

The syntax used to specify the access log file is based on the syntax of available patterns used by the access log files in Apache HTTP Server. For example, the default pattern used by API Gateway isas follows:

```
%h %l %u %t "%r" %s %b
```

The log format strings in this example are explained in *Supported log format strings* on page 262.

The following extract from the `access.log` file illustrates the log format resulting from the default access log patern:

```
s1.axway.com - lisa [09/05/2012:18:24:48 00] "POST / HTTP/1.0" 200 429
s2.axway.com - dave [09/05/2012:18:25:26 00] "POST / HTTP/1.0" 200 727
s3.axway.com - fred [09/05/2012:18:27:12 00] "POST / HTTP/1.0" 200 596
.................
.................
```

Supported log format strings

API Gateway supports the following subset of the Apache HTTP Server log format strings:

Log format string	Description
%a	Remote IP address.
%A	Local IP address.
%b	Bytes sent, excluding HTTP headers, in Common Log Format (for example, – instead of `0` if no bytes were sent).
%B	Bytes sent, excluding HTTP headers.
%D	Time taken to process the request, in milliseconds.
%h	Remote host name.
%H	Request protocol.
%I	Current request thread name (can compare later with stack traces).
%l	Remote logical user name (always –).
%m	Request method.
%p	Local port.
%q	Query string (prepended with `?` if it exists, otherwise an empty string).
%r	First line of the request that originated at the client.
%s	HTTP status code returned to the client in the response.
%t	Date and time of the request in Common Log Format.
%{format}t	Date and time, in any format supported by `SimpleDateFormat`.
%T	Time taken to process the request, in seconds.
%u	Remote user that was authenticated.
%U	Requested URL path.
%v	Local server name.

Log format string	Description
`%{xxx}i`	Incoming request header, where `xxx` is the header name.
`%{xxx}o`	Outgoing request header, where `xxx` is the header name.
`%{xxx}c`	Cookie value, where `xxx` is the cookie name.
`%{xxx}r`	API Gateway message attribute, where `xxx` is the attribute name.

Aliases for commonly used patterns

In addition, you can specify one of the following aliases for commonly used patterns:

Alias	Pattern
`common`	`%h %l %u %t "%r" %s %b`
`combined`	`%h %l %u %t "%r" %s %b "%{Referer}i" "%{User-Agent}i"`

For more details on Apache HTTP Server access log formats, see http://httpd.apache.org/docs/current/logs.html.

Configure the access log

To configure the access log in the Policy Studio tree, select the **Server Settings** node, and click **Logging** >**Transaction Access Log**. To confirm updates to these settings, click **Save** at the bottom right of the window.

You can configure the following fields to enable the server to write an access log to file:

Writing to Transaction Access Log:
Select whether to configure the API Gateway instance to start writing event data to the transaction access log. This setting is disabled by default.

File name:
Enter the name of the access log file. When the file rolls over (because the maximum file size has been reached, or because the date has changed), a suitable increment is appended to the file name. Defaults to `access`.

File extension:
Enter the file extension for the log file. Defaults to `.log`.

Directory:
Enter the directory for the access log file. Defaults to the `logs/access` directory of your product installation.

File size (MB):
Specify the maximum size that the log file is allowed reach before it rolls over to a new file. Defaults to `1000` MB.

Roll log daily:
Select whether to roll over the log file at the start of each day. This is enabled by default.

Number of log files:
Specify the number of log files that are stored. Defaults to `20`.

Format:
Enter the access log file format. This is based on the syntax used in Apache HTTP Server access log files, for example:

```
%h %l %u %t "%r" %s %b
```

For more details, see *Supported log format strings* on page 262.

Note These settings configure the access log at the API Gateway level. You must also configure the access log at the service level on a specific relative path.

For example, in the Policy Studio tree, select the relative path, right-click it in the **Resolvers** pane, and select **Edit**. Then click the **Logging Settings** tab, and select **Include in server access log records** . For more details, see the *API Gateway Policy Developer Guide*.

Redact sensitive details from the access log

The default syntax for the access log is as follows:

```
%h %l %u %t "%r" %s %b
```

The `%r` format string results in the entire HTTP request line being added to the access log file, including the query string. For example:

```
127.0.0.1 - - [02/07/2014:12:39:29 00] "POST /healthcheck?name=value HTTP/1.0" 200 19
```

The query string may contain sensitive information (for example, credit card number, or social security number). If you do not wish the query string to be included in the access log, it is recommended that you use the following format instead:

```
%h %l %u %t "%m %U% %H" %s %b
```

For example, this results in the following output instead:

```
127.0.0.1 - - [02/07/2014:12:39:29 00] "POST /healthcheck HTTP/1.0" 200 19
```

The "%m %U %H" options log the method, path, and HTTP version. This results in the same output as %r , but without the query string.

To confirm updates to these settings, click **Save** at the bottom right of the screen. Click **Deploy** in the toolbar to deploy the updated configuration to the API Gateway.

Transaction event log settings

The **Transaction Event Log** provides a summary of each API Gateway message transaction, which is written to a log file, and used to generate metrics for historic traffic (for example, in API Gateway Analytics or the **Monitoring** view in API Manager). In a distributed system with multiple API Gateway instances running, the events data is written to separate transaction event log files for each API Gateway instance.

The event log file data is processed by the local Node Manager every 5 minutes, aggregated into the appropriate metrics data, and then written to a database. API Manager can use the data from the database to display metrics in the system. Event log file data is written in JSON format, which also enables it to be integrated with third-party logging tools such as Splunk.

Note Node Manager processing of event log data is not enabled by default. You must enable the Node Manager to write metrics to the database. For more details, see *Configure API Gateway with the metrics database* on page 153.

For details on how metrics are displayed in API Gateway Analytics, see the *API Gateway Analytics User Guide*. For details on how metrics are displayed in API Manager, see the *API Manager User Guide*.

Transaction event log formats

Event log files are located in the events directory of your API Gateway installation by default. For example:

```
INSTALL-DIR/apigateway/events/group-2_instance-1.log
```

When each event log file has been processed (every 5 minutes), it can be moved to a processed directory. For example:

```
INSTALL-DIR/apigateway/events/processed
```

By default, files are deleted after being processed.

Entries in the transaction event log file are generated for different event types (for example, header, system, transaction, alert, and custom).

Event log header entries

Event log `header` entries contain details about the creation of the log file. For example, this includes when the log file is created, and on which host, domain, group, instance, and so on.

The fields in the header entries include the following:

Field	Description
type	`header` Entries of type header identify the API Gateway group and instance, one record per log file.
logCreationTime	Time the event log file was created.
hostname	Name of the host the API Gateway process is running on.
domainId	Topology ID of the API Gateway system.
groupId	API Gateway group ID.
groupName	API Gateway group name.
serviceId	API Gateway instance ID.
serviceName	API Gateway instance name.
version	API Gateway version.

The following example shows the JSON format used for `header` events:

```
{
   "type":"header",
   "logCreationTime":"2015-01-23 12:25:00.120",
   "hostname":"user1-PC",
   "domainId":"cfbe55d1-be45-4968-8b4b-f06a4db858b8",
   "groupId":"group-2",
   "groupName":"QuickStart Group",
   "serviceId":"instance-1",
   "serviceName":"QuickStart Server",
   "version":"7.6.2"
}
...
```

Event log system entries

Event log `system` entries contain details about the API Gateway system. For example, this includes details such as the amounts of disk space, memory, and CPU.

The fields in the system entries include the following:

Field	Description
type	`system` Entries of type system contain CPU and memory information, written once every minute.
time	Timestamp of the event (in milliseconds since the epoch, taken from `System.currentTimeMillis()`).
diskUsed	Percentage of disk used (disk on which the API Gateway instance is running: `$VINSTDIR`).
instCpu	Percentage of current process CPU usage (total usage divided by the number of cores).
sysCpu	Percentage of the global CPU usage on the system.
instMem	Current process resident memory size (in KB).
sysMem	System memory used (in KB).
sysMemTotal	System total memory size (in KB).

The following example shows the JSON format used for `system` events:

```
{
   "type":"system",
   "time":1422015900120,
   "diskUsed":30,
   "instCpu":1,
   "sysCpu":5,
   "instMem":533436,
   "sysMem":4641996,
   "sysMemTotal":16759240
}
...
```

Event log transaction entries

Event log `transaction` entries contain details about a specific message transaction. For example, this includes details such as the protocol, method, bytes sent and received, IP addresses, ports, service name, and so on.

The fields in the transaction entries include the following:

Field	Description
type	`transaction` Entries of type transaction describe a transaction and the transaction legs. A transaction entry is recorded for each API call.
time	Timestamp of the event (in milliseconds since the epoch, taken from `System.currentTimeMillis()`).
path	Resource path representing the transaction.
protocol	Inbound protocol.
protocolSrc	Local inbound protocol port (or path).
duration	Execution time of the transaction element (in ms).
status	Transaction result status.
serviceContexts	OAuth, web service, and service context elements. Each service context contains the following fields:
customMsgAtts	Custom message attributes that have been added to the event log (see *Configure the transaction event log* on page 274 for details of how to configure these attributes globally).
correlationId	Transaction correlation ID.

service	Service context name.
monitor	Indicates if metrics monitoring is enabled for this service.
client	Identity of the client.
org	Authentication organization name.
app	Authentication application name.
method	Protocol method used.
status	Execution status of this service context.
duration	Processing time of this service context.

Field	Description	
legs	Legs processed during the transaction. Each leg contains the following fields:	
	uri	URI path of request.
	status	HTTP status code returned.
	statustext	HTTP status message returned.
	method	HTTP method used.
	Vhost	Virtualized API's host.
	wafStatus	Threat protection profile status.
	bytesSent	Number of bytes sent.
	bytesReceived	Number of bytes received.
	remoteName	Name representing remote host of the transaction.
	remoteAddr	Remote host address of transaction.
	localAddr	Local host of transaction.
	remotePort	Remote port of transaction.
	localPort	Local port of transaction.
	Sslsubject	Subject name of peer certificate used to establish SSL connection.
	leg	Transaction element number.
	timestamp	Event timestamp.
	duration	Execution time of the transaction element (in ms).
	serviceName	API Gateway instance name.
	subject	Authenticated user (content of attribute `authentication.subject.id`).

Field	Description	
	operation	SOAP request method used.
	type	Protocol used.
	finalStatus	Status text of the transaction element execution.

The following example shows the JSON format used for an HTTP `transaction` event with a service context and inbound and outbound transaction legs:


```
{
   "type":"transaction",
   "time":1425291330502,
   "path":"/stockquote.asmx",
   "protocol":"http",
   "protocolSrc":"8080",
   "duration":1842,
   "status":"success",
   "serviceContexts":[
   {
      "service":"StockQuote",
      "monitor":true,
      "client":null,
      "org":null,
      "app":null,
      "method":"GetQuote",
      "status":"success",
      "duration":1824}],
   "customMsgAtts":{},
   "correlationId":"4038f4540400788ebe4f84ca",
   "legs":[
     {
        "uri":"/stockquote.asmx",
        "status":200,
        "statustext":"OK",
        "method":"POST",
        "vhost":null,
        "wafStatus":0,
        "bytesSent":1278,
        "bytesReceived":612,
        "remoteName":"127.0.0.1",
        "remoteAddr":"127.0.0.1",
        "localAddr":"127.0.0.1",
        "remotePort":"49104",
        "localPort":"8080",
        "sslsubject":null,
        "leg":0,
        "timestamp":1425291328660,
        "duration":1843,
        "serviceName":"StockQuote",
        "subject":null,
```

```
        "operation":"GetQuote",
        "type":"http",
        "finalStatus":"Pass"
      },
      {
       "uri":"/stockquote.asmx",
       "status":200,
       "statustext":"OK",
       "method":"POST",
       "vhost":null,
       "wafStatus":0,
       "bytesSent":736,
       "bytesReceived":1202,
       "remoteName":"www.webservicex.net",
       "remoteAddr":"173.201.44.188",
       "localAddr":"10.142.10.142",
       "remotePort":"80",
       "localPort":"49438",
       "sslsubject":null,
       "leg":1,
       "timestamp":1425291329916,
       "duration":566,
       "serviceName":"StockQuote",
       "subject":null,
       "operation":"GetQuote",
       "type":"http",
       "finalStatus":null
      }
    ]
}
...
```


Inbound and outbound transaction legs

In this example, the `legs` data is based on traffic monitoring, and its `duration` fields provide useful information. Leg `0` is always the inbound transaction, so its `duration` value is the overall transaction duration observed by API Gateway. Subsequent legs are outbound calls, so their `duration` value represents the back-end transaction duration observed by API Gateway.

Tip The `duration` value for leg `0` minus the sum of the duration of all subsequent legs should give you the time spent in the API Gateway for that transaction. In this example, this is 1843 ms – 566 ms = 1277 ms.

For more information about transactions and legs, see *Introduction to transactions and legs in API Gateway* on page 39.

The service context is an abstract concept, and the `duration` at this level measures time spent in that context only. The service context might be set in an arbitrary place in a policy, so this information is typically not as useful as the leg data—unless in composite services scenarios.

The top-level transaction `duration` is obtained separately, but should be similar to the leg `0` value.

Event log alert entries

Event log `alert` entries contain details about a specific system alert. For example, this includes details such as the protocol, method, bytes sent and received, IP addresses, ports, service name, and so on.

The fields in the alert entries include the following:

Field	Description
type	`alert` Entries of type alert describe API Gateway alerts.
time	Time the alert event was created.
alertType	Type of the alert: • AlertMessage • SlaBreachAlertMessage • SlaClearAlertMessage
level	Integer representing the level of the alert: • 1 (ERROR) • 2 (WARNING) • 3 (INFO)
id	Unique ID of the alert record.
srcId	API Gateway instance ID.
msgId	ID of the message processed during the alert.
defaultMsg	Custom message configured in the alert.
clientIP	Client IP address in use when alert was triggered.
policy	Policy name.
filter	Alert filter name.

The following example shows the JSON format used for an HTTP `transaction` event with a service context and inbound and outbound transaction legs:

```
{
   "type": "alert",
   "time": 1431948768350,
   "alertType": "AlertMessage",
```

```
    "level": 1,
    "id": "6985e4fe:14d66cc3493:-8000",
    "srcId": "user1-LinuxDev1:instance-1",
    "msgId": "Id-e0cd59550500ad6f16e3ce38",
    "defaultMsg": "This is an alert",
    "clientIP": "127.0.0.1",
    "policy": "My Policy Name",
    "filter": "Alert Filter Name"
}
```

Tip For details on `custom` event entries, see the API Gateway Javadoc available from Axway Support at https://support.axway.com. The `com.vordel.reporting.rtm.api.MetricGroup class` includes details on the Java API and the resulting metric event in the transaction event log.

Configure the transaction event log

To configure the transaction event log in the Policy Studio tree, select the **Server Settings** node, and click **Logging** > **Transaction Event Log**.

Configure the following fields to enable the API Gateway instance to write a transaction event log to a file:

Writing to Transaction Event Log:
Enables writing to an event log for all message transactions received by the API Gateway. This setting is enabled by default, and is required for API Gateway Analytics and API Manager metrics. For example, you could deselect this setting to optimize performance.

Write transaction event logs to directory:
Specifies the directory where transaction event logs are written. Defaults to `${environment.VDISTDIR}/events`.

Note If transaction event logs are being used to populate the metrics database, you must also update the `sourceEventLogDir` property in the Node Manager configuration if you change this directory.

System event frequency (secs):
Specifies how often in seconds that a system entry is written to each event log file. Defaults to `60` seconds. For more details, see *Event log system entries* on page 267.

Maximum disk space for event logs (MB):
Specifies the maximum amount of disk space used for event logs. When the directory reaches the specified limit, the oldest log files are deleted. Defaults to `1024` MB.

Check disk space interval (secs):
Specifies how often the amount of available disk space used for event logs is checked. Defaults to `600` seconds.

Select the message attributes to be stored in transaction events:
Enables you to specify custom message attributes to write to the transaction event logs (for example, the HTTP request URI). To specify an attribute, click **Add**, and enter the attribute name in the dialog.

To confirm updates to these settings, click **Save** at the bottom right of the window. Click **Deploy** in the toolbar to deploy the updated configuration to the API Gateway.

Open traffic event log settings

The **Open Traffic Event Log** settings enable you to configure the open traffic event logs written by the API Gateway instances. For example, you can enable open traffic event logging, configure where the logs are stored, and whether or not transaction payloads are stored.

For more information on open logging, see *Configure open logging* on page 172.

Configure the open traffic event log

To configure the open traffic event log in the Policy Studio tree, select the **Server Settings** node, and click **Logging** > **Open Traffic Event Log**.

Configure the following fields:

Enable Open Traffic Event Log:

Enables writing to an open event log. This setting is disabled by default.

Event Log Output:

Choose one of the following options:

- `Use filesystem` – Select this option to write the log data to a file. If you select this option you must enter a directory in the **Event logs directory** field. This can be a location on the local drive, NFSv4, or SAN file system. This is the default.
- `Use console / traces` – Select this option to stream the traffic logs to `stdout` (console/trace files).

Event logs directory:

Specifies the directory where open traffic event logs are written. Defaults to `${environment.VDISTDIR}/logs`.

Maximum disk space for logs (MB):

Specifies the maximum amount of disk space used for open traffic event logs. When the directory reaches the specified limit, the oldest log files are deleted. Defaults to `1024` MB.

Check disk space interval (secs):

Specifies how often the amount of available disk space used for event logs is checked. Defaults to `600` seconds. Enter `0` to disable disk space checks.

Payload Storage:

Choose one of the following options:

- `Do not store payloads` – Select this option to prevent received and sent payloads being stored. This is the default.
- `Use filesystem` – Select this option to store received and sent payloads on the file system. If you select this option you must enter a directory in the **Filesystem directory** field. This can be a location on the local drive, NFSv4, or SAN file system.

To confirm updates to these settings, click **Save** at the bottom right of the window. Click **Deploy** in the toolbar to deploy the updated configuration to the API Gateway.

Embedded ActiveMQ settings

The **Embedded ActiveMQ** settings enable you to configure settings for the Apache ActiveMQ messaging broker that is embedded in each API Gateway instance. You can also configure multiple embedded ActiveMQ brokers to work together as a network of brokers in a group of API Gateway instances. For more details, see Apache ActiveMQ website.

Configure Embedded ActiveMQ settings

Apache ActiveMQ 5.14.3 restricts serializing object message types. To allow the serialization, you must add the impacted packages to the system property in the `jvm.xml` file.

To allow serialization globally on all API Gateway instances, add the following to `INSTALL_DIR/system/conf/jvm.xml`:

```
<VMArg name="-Dorg.apache.activemq.SERIALIZABLE_PACKAGES=<list of
packages>"/>
```

Use comma to separate the packages. For example:

```
<VMArg name="-Dorg.apache.activemq.SERIALIZABLE_
PACKAGES=myorg.data,myorg2.data2">
```

To allow serialization on a particularAPI Gateway instance, add the above code to `INSTALL_DIR/groups/<group name>/<instance name>/conf/jvm.xml`.

To configure embedded ActiveMQ settings, select the **Server Settings** node in the Policy Studio tree, and click **Messaging > Embedded ActiveMQ**. To apply updates to these settings, click **Save** at the bottom right of the window.

General messaging settings

Configure the following ActiveMQ messaging settings:

Enable Embedded ActiveMQ Broker:

Specifies whether to enable starting up the ActiveMQ broker that is embedded in the API Gateway instance. This is not selected by default.

Address:

Specifies the IP address used to open a listening socket for incoming ActiveMQ connections. Defaults to `0.0.0.0`, which specifies that all interface addresses should be used.

Port:

Specifies the TCP port for incoming ActiveMQ connections. Defaults to `${env.BROKER.PORT}`, which enables the port number to be environmentalized. This means that the port number is specified in the `envSettings.props` file on a per-server basis. For more details on the `envSettings.props` file, see the *API Gateway DevOps Deployment Guide*. Alternatively, you can enter the port number directly in this field (for example, `61616`).

Shared Directory:

Specifies the location of the shared directory in your environment that is used by multiple embedded ActiveMQ brokers. This setting is required, and must be configured for high availability and failover. Defaults to `INSTALL_DIR/messaging-shared`.

When setting up a shared directory, do not use OCFS2, NFSv3, or other software where exclusive file locks do not work reliably. For details, see Apache ActiveMQ Shared File System Master Slave.

SSL settings

Configure the following settings to secure the communication with JMS clients, and between multiple embedded ActiveMQ brokers:

Enable SSL:

Specifies whether to use Secure Sockets Layer (SSL) to secure the communication with JMS clients, and between ActiveMQ brokers.

Server Cert:

When **Enable SSL** is selected, click to select the server certificate with a private key that is used for SSL communication between ActiveMQ brokers. For details on importing certificates into the certificate store, see *Manage X.509 certificates and keys* on page 106.

Accepted cipher suites:

When **Enable SSL** is selected, select which cipher suites should be accepted by the JMS server when the SSL communication is being established.

Note If no cipher suites are selected, the default cipher suites from the Java Security Socket Extension (JSSE) are used.

Require Client Certificates:

When **Enable SSL** is set, specifies whether to require client certificates for client SSL authentication. For example, for mutual (two-way) SSL communication, you must trust the issuer of the client certificate by importing the client certificate issuer into the certificate store. For details on importing certificates, see *Manage X.509 certificates and keys* on page 106.

When **Require client certificates** is selected, you can then select the root certificate authorities that are trusted for mutual (two-way) SSL communication between ActiveMQ brokers. For details on importing certificates into the API Gateway certificate store, see *Manage X.509 certificates and keys* on page 106.

Authentication settings

Configure the following to specify authentication settings between multiple embedded ActiveMQ brokers:

Note The authentication settings are also used by features on the **Messaging** tab in the API Gateway Manager web console (for example, sending messages and managing durable topic subscriptions). For more details, see *Manage embedded ActiveMQ messaging* on page 239.

Authenticate broker and client connections with the following policy:

When user name and password credentials are provided for inter-broker communication, they are delegated to the selected policy for authentication. By default, no policy is selected. To select a policy, click the button on the right, and select a preconfigured policy in the dialog.

Username:

Specifies the user name credential when connecting to other ActiveMQ brokers.

Password:

Specifies the password credential when connecting to other ActiveMQ brokers.

Communicate with brokers in the same group:

Every API Gateway instance belongs to a group. This setting specifies whether to communicate only with ActiveMQ brokers in the same API Gateway group. This is the default setting.

Or brokers outside the group registered with the following alias:

Specifies an alias name used to communicate with other ActiveMQ brokers registered with the same alias. This setting enables communication with ActiveMQ brokers that belong to different API Gateway groups.

Advanced settings

Configure the following advanced settings:

Maximum UI queue browsing size:

Enter the maximum number of messages loaded into memory and returned to the API Gateway Manager UI when listing messages contained in a queue. For more information on viewing the messages in a queue, see *Manage messages in a queue* on page 240.

Maximum memory usage:

Enter a value and select a unit (MiB or GiB) to specify the maximum amount of memory to use for non-persisted or temporary messages. The default value is 1 GiB.

Maximum disk store usage:

Enter a value and select a unit (MiB or GiB) to specify the maximum amount of disk space to use for persisted messages. The default value is 100 GiB.

Maximum temporary disk usage:

Enter a value and select a unit (MiB or GiB) to specify the maximum amount of disk space to use for temporary messages. The default value is 50 GiB.

Enable report of memory and disk usage (when exceeding 50%):

Select or deselect this option to enable or disable reporting of memory and disk usage when usage exceeds 50 percent of the specified maximums. When this option is selected, memory and disk usage information is written to the API Gateway trace log. For example:

```
INFO  [3d1b:00000000000000000000000] ActiveMQ memory usage reached 50%
INFO  [3d1b:00000000000000000000000] ActiveMQ memory usage reached 60%
ERROR [3d1b:00000000000000000000000] ActiveMQ memory usage reached 70%
ERROR [3d1b:00000000000000000000000] ActiveMQ memory usage reached 80%
ERROR [3d1b:00000000000000000000000] ActiveMQ memory usage reached 90%
FATAL [3d1b:00000000000000000000000] ActiveMQ memory usage reached 100%
ERROR [3d1b:00000000000000000000000] ActiveMQ memory usage went down to 90%
ERROR [3d1b:00000000000000000000000] ActiveMQ memory usage went down to 80%
ERROR [3d1b:00000000000000000000000] ActiveMQ memory usage went down to 70%
INFO  [3d1b:00000000000000000000000] ActiveMQ memory usage went down to 60%
INFO  [3d1b:00000000000000000000000] ActiveMQ memory usage went down to 50%
INFO  [3d1b:00000000000000000000000] ActiveMQ memory usage went down to 40%
```

Traffic monitoring settings

The **Traffic Monitor** settings enable you to configure the traffic monitoring available in the web-based API Gateway Manager tool. For example, you can configure where the data is stored and what message transaction details are recorded in the message traffic log.

To access the **Traffic Monitor** settings, click the **Server Settings** node in the Policy Studio tree, and click **Monitoring** > **Traffic Monitor**. To confirm updates to these settings, click **Save** at the bottom right of the screen.

To access traffic monitoring in API Gateway Manager, go to `http://localhost:8090`, and click the **Traffic** button in the toolbar.

Configuration

You can configure the following **Traffic Monitor** settings:

Enable Traffic Monitor:

Select whether to enable the web-based traffic monitoring in in API Gateway Manager. This is enabled by default.

Transaction Store Location Settings:

Enter the **Transaction Directory** that stores the traffic monitoring data files and database. You must enter either an absolute path, or a path relative to the API Gateway instance directory, which is the location where the server instance runs. For example:

```
INSTALL_DIR/apigateway/groups/GROUP/INSTANCE
```

The **Transaction Directory** defaults to `conf/opsdb.d`. If this directory or the database does not exist when the API Gateway starts up, they are recreated automatically.

Persistence Settings:

You can configure the following data persistence settings:

Record inbound transactions	Select whether to record inbound message transactions received by the API Gateway. This is enabled by default.
Record outbound transactions	Select whether to record outbound message transactions sent from the API Gateway to remote hosts. This is enabled by default.
Record policy path	Select whether to record the policy path for the message transaction, which shows the filters that the message passes through. This is enabled by default.
Record trace	Select whether to record the trace output for the message transaction. This is enabled by default.
Record sent data for transactions	Select whether to record the sent payload data for the message transaction. This is enabled by default.
Record received data for transactions	Select whether to record the received payload data for the message transaction. This is enabled by default.

Note These settings are global for all traffic passing through the API Gateway. You can override these persistence settings at the port level when configuring an HTTP or HTTPS interface. For more details, see "Configure HTTP services" in the *API Gateway Policy Developer Guide*. Details of inbound and outbound transactions are also written to the transaction event log.

If recording of inbound or outbound transactions is disabled in **Traffic Monitor** settings, transaction data will not be written to the event log. For more details, see *Transaction event log settings* on page 266.

Transaction File Management Settings:
You can use the following settings to configure transaction file management.

Maximum transaction file size	Enter the maximum size of the transaction file, and select the units from the list. The default value is 8 MiB. When this limit is reached, a new file is created.
Create new transaction file every day	Select the check box to force creation of a new transaction file every day at midnight, even if the maximum file size was not exceeded.
Maximum number of transaction files	Enter the maximum number of transaction files to store on disk. The default value is 128. When this limit is reached, old files that have no open transactions are purged. **Note** The number of transaction files might be exceeded if the oldest file still has open transactions.
Number of days after which transaction files will be purged	Enter the maximum number of days after which transaction files are purged. Files older than the specified number of days (based on their modification date) are purged. You can use a value of 0 to disable purging. The default value is 0.

Real-time monitoring metrics

You can configure real-time monitoring metrics for an API Gateway instance. For example, this enables you to specify monitoring of messages at the level of API services, methods,clients, and remote hosts. This is important when managing APIs because of requirements to bill clients for their API usage.

When real-time monitoring is enabled, monitoring data is stored in API Gateway memory and displayed in the API Gateway Manager web console. API Gateway Manager uses the configured real-time monitoring metrics to display graphical reports in its **Dashboard** and **Monitoring** views. For more details on viewing real-time metrics, see *Monitor services in API Gateway Manager* on page 147.

To configure real-time monitoring settings in the Policy Studio tree, select the **Server Settings** node, and click **Monitoring** > **Real Time Monitoring**.

Enable monitoring

Configure the following general settings:

Enable real-time monitoring:
This enables real-time monitoring globally for the API Gateway instance in the API Gateway Manager web console. This setting must be enabled to display monitoring data in the **Dashboard** and **Monitoring** views in API Gateway Manager, and is selected by default. To disable real-time monitoring, deselect this setting.

System metrics update frequency (secs):
Specifies how often in seconds that system metrics are measured (for example, CPU, disk space, and memory usage). Defaults to `3` seconds.

Configure real-time metrics

Configure the following settings in the **Real Time Monitoring Limits** section:

Note Real-time monitoring may have a negative impact on API Gateway performance. To optimize performance, disable monitoring for one or more metrics.

You should set the maximum services, methods, and remote hosts to values that will never be reached in normal operation. These settings protect the API Gateway by setting an upper limit on the amount of memory consumed by real-time monitoring.

Service:
Enables real-time monitoring of metrics data on the **API Services** tab. This is enabled by default.

Maximum Services:
Specifies the maximum number of API services that are monitored by the API Gateway. When the maximum is reached, the API Gateway stops collecting metrics for new services. Defaults to `10000`.

Method:
Enables real-time monitoring of metrics data on the **API Methods** tab. This is enabled by default.

Note To enable method monitoring, you must ensure that service monitoring is also enabled. Disabling service monitoring also disables method monitoring.

Maximum Methods:
Specifies the maximum number of API methods that are monitored by the API Gateway. When the maximum is reached, the API Gateway stops collecting metrics for new methods. Defaults to `100000`.

Remote Host:
Enables real-time monitoring of metrics data on the **Remote Hosts** tab. This is enabled by default. For more details on remote hosts, see the *API Gateway Policy Developer Guide*.

Maximum Remote Hosts:
Specifies the maximum number of remote hosts that are monitored by the API Gateway. When the maximum is reached, the API Gateway stops collecting metrics for new remote hosts. Defaults to `10000`.

Client:
Enables real-time monitoring of metrics data on the **Clients** tab. This is enabled by default.

Maximum Clients:
Specifies the maximum number of clients that are monitored by the API Gateway. When the maximum is reached, the API Gateway stops collecting metrics for new clients. Defaults to `10000`.

Note The number of unique clients that communicate with an API Gateway is potentially unbounded. The maximum number of clients is therefore a soft limit. When this is reached, monitoring stops for the oldest client and begins for the newest client instead.

For the other maximum values (services, methods, and remote hosts), exceptions are thrown and logged when the limits are reached.

To confirm updates to these settings, click **Save** at the bottom right of the window. Click **Deploy** in the toolbar to deploy the updated configuration to the API Gateway.

Note You must restart the API Gateway instance after changing any of the maximum values (for example, **Maximum Services**, **Maximum Methods**, **Maximum Clients**, or **Maximum Remote Hosts**).

Scheduled report storage settings

You can schedule API Gateway Analytics reports to run on a regular basis, and to email the results to users in PDF format. These reports include summary values at the top (for example, the number of requests, SLA breaches, alerts triggered, and unique clients in a specified week) followed by a table of services, and their aggregated usage data (for example, the number of requests on each service).

The report data is for the configured *current week of the report*, which is compared to the week before. You can set the configured *current week of the report* to be the actual current calendar week or any prior week (provided there is corresponding data in the database).

To configure scheduled report settings in Policy Studio, right-click the **Environment Configuration** > **Listeners** > **Axway Analytics** node in the Policy Studio tree, and select **Database Archive**.

Database configuration

Click the browse button the right, and select a pre-configured database connection in the dialog. This setting defaults to the `Default Database Connection`. To add a new database connection, right-click the **Database Connections** node, and select **Add DB connection**.

You can also edit or delete existing nodes by right-clicking and selecting the appropriate option. Alternatively, you can add database connections under the **Environment Configuration** > **External Connections** node in the Policy Studio tree view. For more details on creating database connections, see the *API Gateway Policy Developer Guide*.

Scheduled reports configuration

You can configure the following settings for scheduled reports:

Enable Report Generation:
Select whether to enable scheduled reports in PDF format. When selected, by default, this runs a scheduled weekly report on Monday morning at 0:01. For details on configuring a different time schedule, see the next setting. This setting is not selected by default.

When **Enable Report Generation** is enabled, you can configure the following settings on the **Report Generator Process** tab:

Connect to API Gateway Analytics as User:
Enter the user name and password used to connect to the report generator process. Defaults to the values entered using the `configureserver` script.

Output:
Enter the directory used for the generated report files in the **Output Directory** field, or click **Choose** to browse to the directory. Defaults to the directory entered using the `configureserver` script (for example, `c:\temp\reports`). You can also select to **Do not delete report files after emailing**. This setting is not selected by default.

SMTP configuration

When **Enable Report Generation** is enabled, you can configure the following settings on the **SMTP** tab. These settings default to those entered using the `configureserver` script. For more details, see the *API Gateway Installation Guide*.

Email generated reports:
Select whether to email generated PDF report files. This is not selected by default.

Do not delete report files after emailing:
Select whether to keep generated PDF report files after they are sent. Not selected by default.

Email Recipient (To):
Enter the recipient of the automatically generated email (for example, `user@mycorp.com`). Use a semicolon-separated list of email addresses to send reports to multiple recipients.

Email Sender (From):
The generated report emails appear *from* the sender email address specified here (for example, `no-reply@mycorp.com`).

Note Some mail servers do not allow relaying mail when the sender in the **From** field is not recognized by the server.

SMTP Server Settings:
Specify the following fields:

Outgoing Mail Server (SMTP)	Specify the SMTP server used to relay the report email (for example, `smtp.gmail.com`).
Port	Specify the SMTP server port to connect to. Defaults to port 25.
Connection Security	Select the connection security used to send the report email (`SSL`, `TLS`, or `NONE`). Defaults to `NONE`.

Log on Using:
If you are required to authenticate to the SMTP server, specify the following fields:

User Name	Enter the user name for authentication.
Password	Enter the password for the user name specified.

For more details on API Gateway Analytics, see the *API Gateway Analytics User Guide*.

Appendix A: Open logging schema

Each line in the Open logging JSON output consists of common properties, and one of four possible content records detailing an activity that occurred during runtime:

Common properties	Type	Description
`processInfo`	`object`	Contains details of the API Gateway process that logged the activity
`correlationId`	`string`	The transaction correlation ID allows all activities for a given transaction to be grouped together
`timestamp`	`integer`	The timestamp of when the activity was started; the meaning varies depending on the content record type. transactionElement - When the remote connection was first established with API Gateway transactionSummary - When the API Gateway started processing the request circuitPath - When the API Gateway started processing the request trace - When the trace message was created

#	Specific record properties	Type	Description
1	`circuitPath`	`array`	Contains details of the circuit paths executed during a transaction

```
Example of circuit path record:{
  "timestamp": 1530192405955,
  "correlationId": "15e2345b0400e43b9d9cbfaa",
  "processInfo": {
    "hostname": "redhat7",
    "domainId": "5fae6fa9-fa2c-495b-9400-7e65f55ee4a6",
    "groupId": "group-2",
    "groupName": "QuickStart Group",
    "serviceId": "instance-1",
    "serviceName": "QuickStart Server",
    "version": "7.6.2"
  },
  "circuitPath": [
    {
      "policy": "Health Check",
      "execTime": 0,
      "filters": [
        {
          "espk": "DEFAULT_PRIMARY_VordelGateway_7.6.2:638686972755456130",
          "name": "Set Message",
          "type": "ChangeMessageFilter",
          "class": "com.vordel.circuit.conversion.ChangeMessageFilter",
          "status": "Pass",
          "filterTime": 1530192405954,
          "execTime": 0
        },
        {
          "espk": "DEFAULT_PRIMARY_VordelGateway_7.6.2:5622982849029387930",
          "name": "Reflect",
          "type": "ReflectFilter",
          "class": "com.vordel.circuit.net.ReflectFilter",
          "status": "Pass",
          "filterTime": 1530192405954,
          "execTime": 0
        }
      ]
    }
  ]
}
```

#	Specific record properties	Type	Description
2	`transactionSummary`	`object`	Contains the high-level summary details of a transaction

```
Example of transaction summary record:{
  "timestamp": 1530192405955,
  "correlationId": "15e2345b0400e43b9d9cbfaa",
  "processInfo": {
    "hostname": "redhat7",
    "domainId": "5fae6fa9-fa2c-495b-9400-7e65f55ee4a6",
    "groupId": "group-2",
    "groupName": "QuickStart  Group",
    "serviceId": "instance-1",
    "serviceName": "QuickStart Server",
    "version": "7.6.2"
  },
  "transactionSummary": {
    "path": "/healthcheck",
    "protocol": "http",
    "protocolSrc": "8080",
    "status": "success",
    "serviceContexts": []
  }
}
```

#	Specific record properties	Type	Description
3	`transactionElement`	`object`	Contains detailed information for an indvidual element (leg) of a transaction

#	Specific record properties	Type	Description

```
Example of transaction element record:{
  "timestamp": 1530192405953,
  "correlationId": "15e2345b0400e43b9d9cbfaa",
  "processInfo": {
    "hostname": "redhat7",
    "domainId": "5fae6fa9-fa2c-495b-9400-7e65f55ee4a6",
    "groupId": "group-2",
    "groupName": "QuickStart Group",
    "serviceId": "instance-1",
    "serviceName": "QuickStart Server",
    "version": "7.6.2"
  },
  "transactionElement": {
    "leg": 0,
    "duration": 2,
    "serviceName": null,
    "operation": null,
    "finalStatus": "Pass",
    "protocolInfo": {
      "http": {
        "uri": "/healthcheck",
        "status": 200,
        "statusText": "OK",
        "method": "GET",
        "vhost": null,
        "wafStatus": 0,
        "bytesSent": 243,
        "bytesReceived": 89,
        "remoteName": "127.0.0.1",
        "remoteAddr": "127.0.0.1",
        "localAddr": "127.0.0.1",
        "remotePort": "38090",
        "localPort": "8080",
        "sslSubject": null,
        "authSubjectId": null
      },
      "recvHeader": "GET /healthcheck HTTP/1.1\r\nHost: 127.0.0.1:8080\r\nUser-Agent:
curl/7.47.0\r\nAccept: */*\r\n\r\n",
      "sentHeader": "HTTP/1.1 200 OK\r\nDate: Thu, 28 Jun 2018 13:26:45 GMT\r\nServer:
Gateway\r\nConnection: close\r\nX-CorrelationID: Id-15e2345b0400e43b9d9cbfaa 0\r\nAccept:
*/*\r\nHost: 127.0.0.1:8080\r\nUser-Agent: curl/7.47.0\r\nContent-Type:
text/xml\r\n\r\n",
      "recvPayload": null,
      "sentPayload": null
    }
  }
}
```

#	Specific record properties	Type	Description
4	`trace`	`object`	Contains a single trace message generated by an API Gateway process

```
Example of trace record: {
  "timestamp": 1530190069589,
  "correlationId": "00000000000000000000000000",
  "processInfo": {
    "hostname": "redhat7",
    "domainId": "5fae6fa9-fa2c-495b-9400-7e65f55ee4a6",
    "groupId": "group-2",
    "groupName": "QuickStart Group",
    "serviceId": "instance-1",
    "serviceName": "QuickStart Server",
    "version": "7.6.2"
  },
  "trace": {
    "level": "INFO",
    "thread": "617c",
    "data": "service started (version 7.6.2-2018-06-27 rev. 9af6443,  pid 24956)"
  }
}
```

Process information

Contains details of the API Gateway process that logged the activity:

Property	Type	Description
`domainId`	`string`	Topology ID of the API Gateway system
`groupId`	`string`	API Gateway group ID
`groupName`	`string`	API Gateway group name
`hostname`	`string`	Name of the host that the API Gateway process is running on
`serviceId`	`string`	API Gateway instance ID
`serviceName`	`string`	API Gateway instance name
`version`	`string`	API Gateway product version

Circuit path

Contains details of the circuit paths executed during a transaction:

Property	Type	Description
`execTime`	`integer`	Total execution time (ms)
`filters`	`array`	Filters executed in the circuit
`policy`	`string`	Policy name
`viaProperty`	`string`	Policy execution context description (optional)

Filter

Contains the details of a filter execution:

Property	Type	Description
`class`	`string`	Java class of the filter
`espk`	`string`	Entity store access key of the filter
`execTime`	`integer`	Total execution time (in milliseconds) of the filter
`filterMessage`	`string`	Filter completion message
`filterTime`	`integer`	Timestamp of when the filter was completed
`name`	`string`	Name of the filter
`status`	`string`	Filter execution status Possible values are: `["Pass","Fail","Error"]`
`subPaths`	`array`	Nested circuit paths executions
`type`	`string`	Type of the filter

Transaction summary

Contains the high-level summary details of a transaction:

Property	Type	Description
`path`	`string` or `null`	URI entry point of the transaction (HTTP only)
`protocol`	`string` or `null`	Inbound protocol used
`protocolSrc`	`string` or `null`	Local port or path of the inbound protocol
`serviceContexts`	`array`	OAuth, Web Service, and Service Context elements contain fields describing the service context of a transaction
`status`	`string`	Transaction result status Possible values are: `["success","failure","exception","unknown"]`

Service context

OAuth, Web Service, and Service Context elements contain fields describing the service context of a transaction:

Property	Type	Description
`app`	`string` or `null`	Authentication application name
`client`	`string` or `null`	Identity of the client
`duration`	`integer`	Total execution time of this service context in milliseconds
`method`	`string` or `null`	Protocol method used

Property	Type	Description
monitor	boolean	Is metrics monitoring enabled for this service
org	string or null	Authentication organization name
service	string	Service context name
status	string or null	Execution status of this service context Possible values are: ["success" ,"failure","exception","unknown",null]

Transaction element

Contains detailed information for an indvidual element (leg) of a transaction:

Property	Type	Description
duration	integer	Total execution time (in milliseconds) of this transaction element
finalStatus	string or null	Status text of the transaction element execution
leg	integer	Transaction element number
operation	string or null	SOAP request method used
protocolInfo	object	Protocol specific information of the transaction element
serviceName	string or null	SOAP service name used

Protocol information

Protocol specific information of the transaction element consists of common properties and one of five possible content records:

Common properties	Type	Description
`recvHeader`	`string` or `null`	Protocol header received
`recvPayload`	`string` or `null`	Storage path to the received payload
`sentHeader`	`string` or `null`	Protocol header sent
`sentPayload`	`string` or `null`	Storage path to the sent payload

#	Specific record properties	Type	Description
1	`http`	`object`	HTTP protocol properties
2	`websock`	`object`	Websocket protocol properties
3	`dirScan`	`object`	Directory scanning processing properties
4	`fileTransfer`	`object`	File transfer properties
5	`jms`	`object`	JMS protocol properties

HTTP

HTTP protocol properties:

Property	Type	Description
`authSubjectId`	`string` or `null`	Authentication subject ID (login)

Property	Type	Description
`bytesReceived`	`integer`	Number of bytes received
`bytesSent`	`integer`	Number of bytes sent
`localAddr`	`string`	Local host of transaction
`localPort`	`string`	Local port of transaction
`method`	`string`	HTTP method used
`remoteAddr`	`string`	Remote host address of transaction
`remoteName`	`string`	Name representing remote host of the transaction
`remotePort`	`string`	Remote port of transaction
`sslSubject`	`string` or `null`	Subject name of peer certificate used to establish SSL connection
`status`	`integer`	HTTP status code returned
`statusText`	`string`	HTTP status message returned
`uri`	`string`	URI path of request
`vhost`	`string` or `null`	Virtualized API's host
`wafStatus`	`integer`	Threat protection profile status Possible values are: `[0,1,2]`

Websock

Websocket protocol properties:

Property	Type	Description
`authSubjectId`	`string` or `null`	Authentication subject ID (login)

Property	Type	Description
bytesReceived	integer	Number of bytes received
bytesSent	integer	Number of bytes sent
length	integer	Length of payload in bytes
localAddr	string	Local host of transaction
localPort	string	Local port of transaction
mask	integer	Websocket header mask
opcode	string	Websocket opcode mode Possible values are: ["text","binary"]
origin	string	Origin of transaction Possible values are: ["client","server"]
remoteAddr	string	Remote host address of transaction
remoteName	string or null	Name representing remote host of the transaction
remotePort	string	Remote port of transaction
sslSubject	string or null	Subject name of peer certificate used to establish SSL connection
uri	string	URI path of request
wsPeerLocalAddr	string	Local host of Websocket connection
wsPeerLocalPort	string	Local port of Websocket connection
wsPeerRemoteAddr	string	Remote host of Websocket connection
wsPeerRemotePort	string	Remote port of Websocket connection

Directory scanning

Directory scanning processing properties:

Property	Type	Description
`authSubjectId`	`string` or `null`	Authentication subject ID (user login set by the policy, not by file processing)
`fileHidden`	`string`	True if file is a hidden file Possible values are: `["true","false"]`
`fileName`	`string`	File name
`filePath`	`string`	Path of directory containing the file
`fileSize`	`string`	Size of the file (if available)
`readOnly`	`string`	Read only status of the file (if available) Possible values are: `["true","false"]`
`status`	`string`	File operation status Possible values are: `["Success","Failed"]`
`statusText`	`string` or `null`	File operation error message (if available)

File transfer

File transfer properties:

Property	Type	Description
`authSubjectId`	`string`	Authentication subject ID (user login)
`direction`	`string`	File transfer direction Possible values are: `["up"]`

Property	Type	Description
`remoteAddr`	`string`	Remote host address of transaction
`serviceType`	`string`	Service protocol used Possible values are: `["ftp","ftps","sftp"]`
`size`	`integer`	Size of the file
`uploadFile`	`string`	Absolute path of payload file

JMS

JMS protocol properties:

Property	Type	Description
`authSubjectId`	`string` or `null`	Authentication subject ID (user login) if available
`jmsCorrelationID`	`string`	Message correlation ID
`jmsDeliveryMode`	`integer`	Message delivery mode (1 for non-persistent, 2 for persistent) Possible values are: `[1,2]`
`jmsDestination`	`string`	Message destination
`jmsExpiration`	`integer`	Message expiration timestamp
`jmsMessageID`	`string`	Message ID
`jmsPriority`	`integer`	Messaage priority rank
`jmsProviderURL`	`string`	Provider URL configured in JMS service
`jmsRedelivered`	`integer`	Indicates if message is marked for potential redelivery (0 for no) Possible values are: `[0,1]`
`jmsReplyTo`	`string`	Reply destination configured in message

Property	Type	Description
`jmsStatus`	`string`	JMS operation status Possible values are: `["Success","Failed"]`
`jmsStatusText`	`string` or `null`	JMS operation error message (if available)
`jmsTimestamp`	`integer`	Time at which the message was originally handed off to a provider to be sent
`jmsType`	`string`	Message type identifier

Trace message

Contains a single trace message generated by an API Gateway process:

Property	Type	Description
`level`	`string`	Trace level Possible values are: ["FATAL" ,"REPORT","ERROR","INFO","MIN","DEBUG","DATA"]
`thread`	`string`	The ID of the thread that logged the trace message
`data`	`string`	The contents of the trace message