ADMINISTRATOR GUIDE

Axway API Gateway

Version 7.3.0

08 May 2014

Contents

Introduction to API Gateway administration

Overview

Axway API Gateway is a comprehensive platform for managing, delivering, and securing APIs. This guide explains how to administer and manage the API Gateway platform. This chapter 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 (for details on supported platform versions, see the *API Gateway Installation and Configuration Guide*)
- Hardware Appliance—a hardware and software combination, which includes a crypto acceleration card and an optional Hardware Security Module (HSM)
- Virtual Appliance—a virtual image of the appliance for VMware, Oracle VM, and Amazon Machine Image

For example, API Gateway customers might use a mix of different software for development and initial testing of policies, and then use appliances for preproduction and live production systems.

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

The API Gateway platform is administered by the following groups:

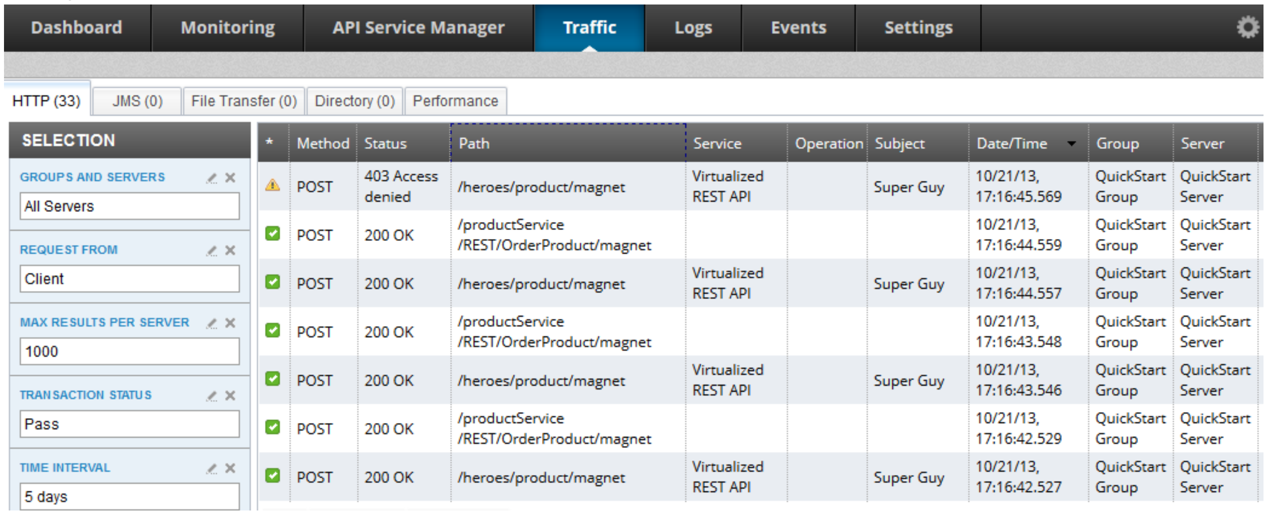
- Operations—Runtime management of message traffic, logs and alerts, and high availability is performed by Operations staff.
- Architecture—Design-time policy definition, which determines the behavior of the API Gateway platform, is performed by Security Architects and Systems Architects.

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:



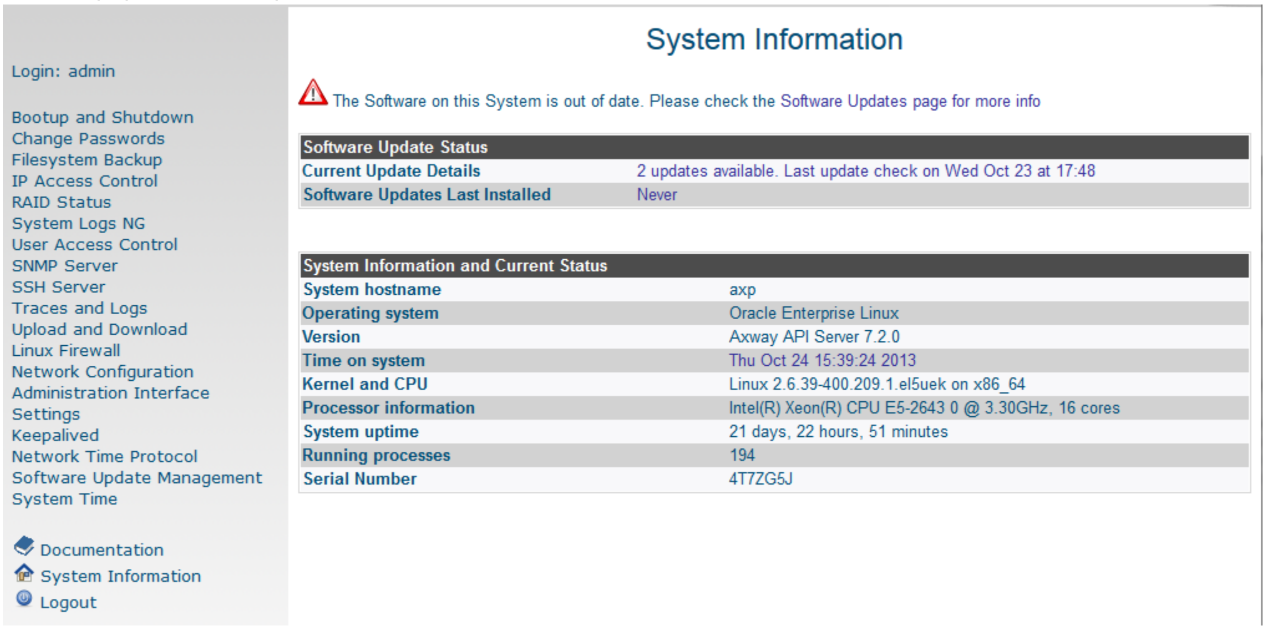
GROUPS AND SERVERS	All Servers
REQUEST FROM	Client
MAX RESULTS PER SERVER	1000
TRANSACTION STATUS	Pass
TIME INTERVAL	5 days

*	Method	Status	Path	Service	Operation	Subject	Date/Time	Group	Server
⚠	POST	403 Access denied	/heroes/product/magnet	Virtualized REST API		Super Guy	10/21/13, 17:16:45.569	QuickStart Group	QuickStart Server
✓	POST	200 OK	/productService /REST/OrderProduct/magnet				10/21/13, 17:16:44.559	QuickStart Group	QuickStart Server
✓	POST	200 OK	/heroes/product/magnet	Virtualized REST API		Super Guy	10/21/13, 17:16:44.557	QuickStart Group	QuickStart Server
✓	POST	200 OK	/productService /REST/OrderProduct/magnet				10/21/13, 17:16:43.548	QuickStart Group	QuickStart Server
✓	POST	200 OK	/heroes/product/magnet	Virtualized REST API		Super Guy	10/21/13, 17:16:43.546	QuickStart Group	QuickStart Server
✓	POST	200 OK	/productService /REST/OrderProduct/magnet				10/21/13, 17:16:42.529	QuickStart Group	QuickStart Server
✓	POST	200 OK	/heroes/product/magnet	Virtualized REST API		Super Guy	10/21/13, 17:16:42.527	QuickStart Group	QuickStart Server

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.

The API Gateway appliance and virtual appliance platforms also provide the powerful API Gateway Web Administration Interface to control common system-specific tasks performed by the Operations team. The Web Administration Interface runs by default and makes the configuration of all system related tasks relatively quick and easy.



System Information

The Software on this System is out of date. Please check the Software Updates page for more info

Software Update Status	
Current Update Details	2 updates available. Last update check on Wed Oct 23 at 17:48
Software Updates Last Installed	Never

System Information and Current Status	
System hostname	axp
Operating system	Oracle Enterprise Linux
Version	Axway API Server 7.2.0
Time on system	Thu Oct 24 15:39:24 2013
Kernel and CPU	Linux 2.6.39-400.209.1.el5uek on x86_64
Processor information	Intel(R) Xeon(R) CPU E5-2643 0 @ 3.30GHz, 16 cores
System uptime	21 days, 22 hours, 51 minutes
Running processes	194
Serial Number	4T7ZG5J

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

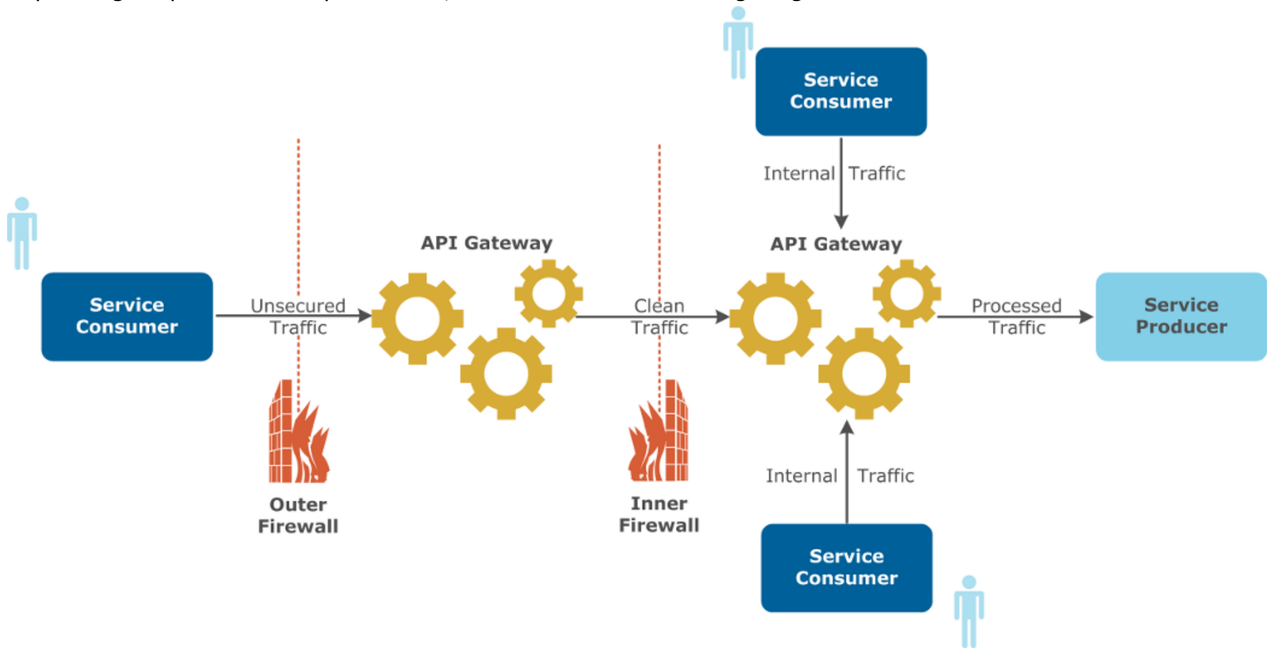
System architects and security architects have an overarching view of enterprise IT infrastructures, and so are more concerned with using the API Gateway to help integrate and secure existing enterprise systems. Architects wish to create API policies and integrate with third-party systems.

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 User 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 the API 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 and/or Node Manager.

You can deploy API Gateway Analytics on any supported host platform (for example, Windows, Linux, or Solaris) depending on its availability in your architecture. For more details on supported platform versions, see the *API Gateway Installation and Configuration Guide*.

Important

API Gateway Analytics supports a range of databases for storing historic reports and metrics (for example, Oracle, DB2, MySQL, and Microsoft SQL Server). It is not advised to 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 Installation and Configuration 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 User 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 the next topic.
2. Installing API Gateway components. See the *API Gateway Installation and Configuration Guide*.
3. Configuring a domain. This is described in Part 4, “Deploy API Gateway configuration”.
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 Installation and Configuration Guide*.

Plan an API Gateway system

Overview

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. The most important factors to look at when architecting an API Gateway deployment are as follows:

- What type of API Gateway policies will run on the system?
- What type of message traffic will the API Gateway be required to process?
- What is the expected distribution of the traffic?
- How important is the system, and does it need to be always available?
- How to make changes to the system without incurring system outage
- Planning for outage and disaster recovery

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 username/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 (Axway API Tester) to help you with this process.
- Use the Axway API Gateway Manager console 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.

- View information on historic network usage in Axway API Gateway Analytics.
- Test the performance capability of the backend 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 User 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

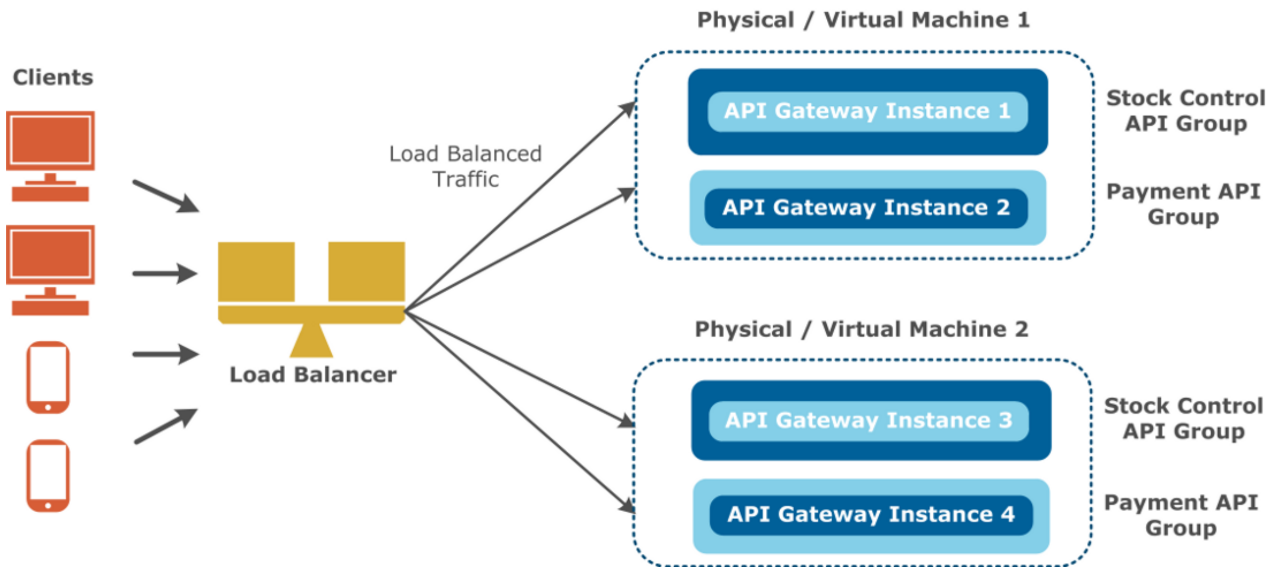
You should adhere to the following guidelines when analyzing message traffic:

- Take traffic distribution into account when calculating performance requirements.
- If traffic bursts cause problems for service producers, consider using the API Gateway to smooth the traffic. There are a number of techniques for doing this.
- Take message size distribution into account when running performance tests.

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 User 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 *Deployment and Promotion 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 backend 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 is 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 the API Gateway, which uses it for validation and authentication.
- **Load balancer is configured for SSL pass-through, and all traffic passed to the 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.

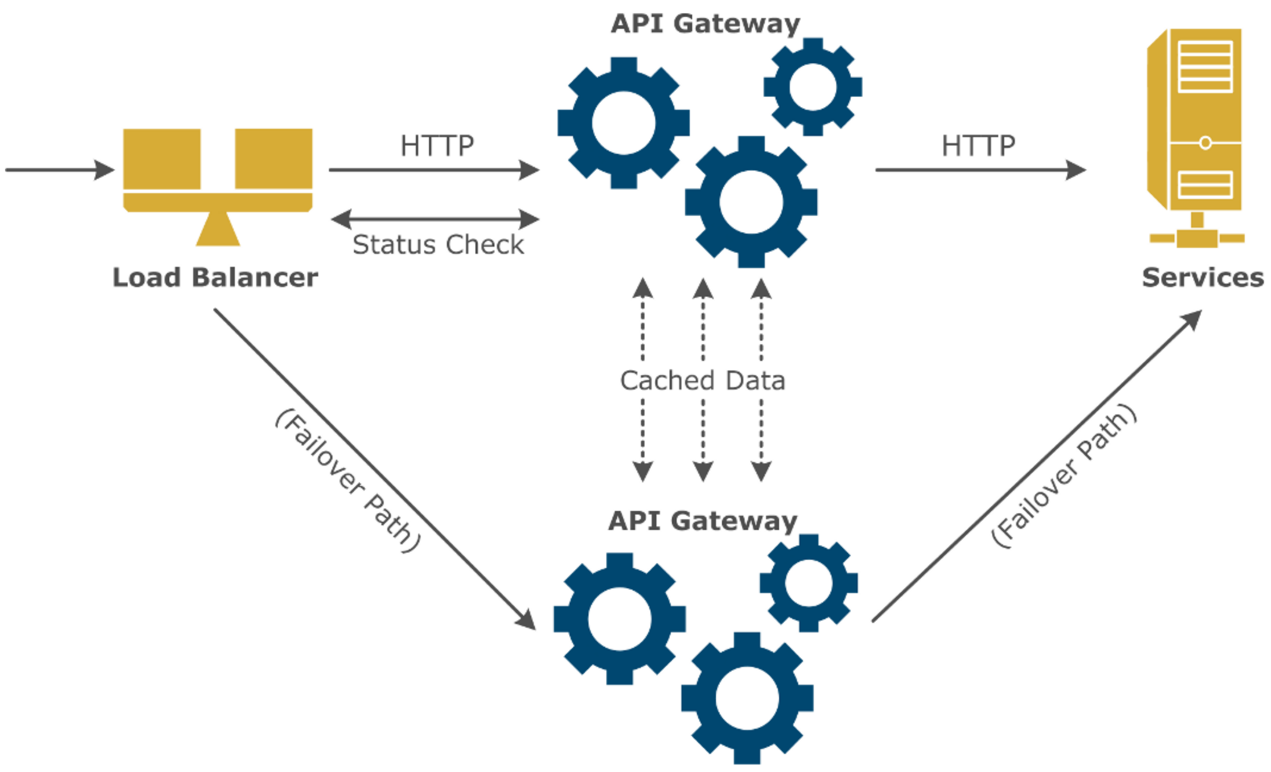
Note

The API Gateway can also optionally use a cryptographic accelerator for SSL termination. For more details, see the *API Gateway User Guide*. This is configured by default on the API Gateway Appliance.

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 more details on the Admin Node Manager 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 User 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 or API Gateway Analytics, 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 backend 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 and API Gateway Analytics. Interfaces are also provided to standard monitoring tools such as syslog and Simple Network Management Protocol (SNMP).

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.

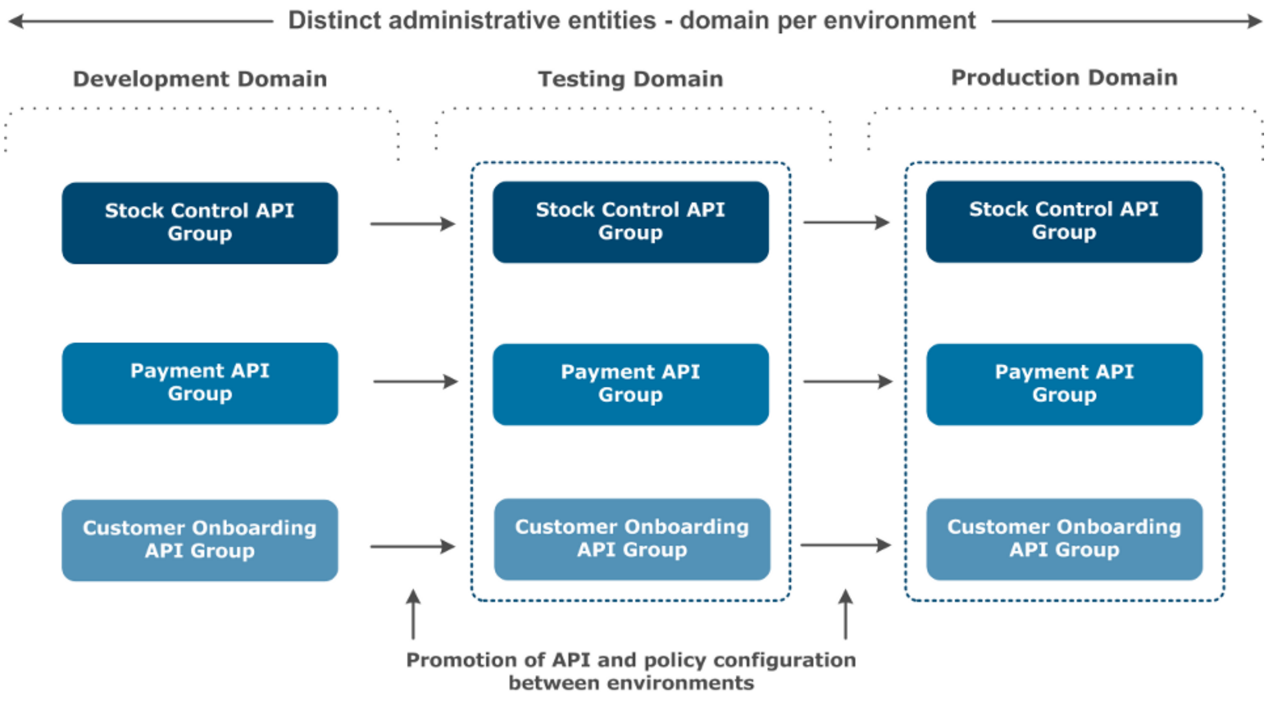
Development staging and testing

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:


Distinct administrative entities - domain per environment
Development Domain
Testing Domain
Production Domain
Stock Control API Group
Payment API Group
Customer Onboarding API Group
Promotion of API and policy configuration between environments

Staging and testing guidelines

The following guidelines apply to development staging and testing:

- 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.

For more details, see the *API Gateway Deployment and Promotion Guide*.

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 from `admin/changeme`). For more details, see *Manage Admin users*.
- 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 *Manage certificates and keys*.
- 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 chapter on "Configuring HTTP Services" in the *API Gateway User 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*.
- You can configure user access at the following levels:
  - Policy Studio, API Gateway Manager, and Configuration Studio users—see *Manage Admin users*.
  - API Gateway users—see *Manage API Gateway users*.
  - Role-based access—see *Configure Role-Based Access Control (RBAC)*.

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 9am in the morning and 5pm 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 backend 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 appliances for stage testing
   - A factor of safety on the load of 2
5. The number of API Gateways should be as follows:

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

   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

Overview

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, and logging and reporting. API Gateway uses Java Database Connectivity (JDBC) to manage database connections. The following databases are supported:

- MySQL
- Oracle
- DB2
- Microsoft SQL Server

For details on supported versions, see the *API Gateway Installation and Configuration 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 antivirus vendor.
- Licensing for the AV engine is performed through the AV vendor distribution channels.

For more details, see the AntiVirus Integration Guides in the *API Gateway Interoperability 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/TCP), 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 a database for later analysis using API Gateway Analytics. For more details on logging, see Part 5, "Troubleshoot your API Gateway installation".

- 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 a Admin Node Manager. For more details, see Part 8, “Monitoring and reporting”.
- 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 User 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 User 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:

- 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.

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.

Firewall modes

You can configure the API Gateway in the following modes:

- Block unidentified traffic, which 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.

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

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:

- 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).

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.

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, usernames and passwords, certificates, organization structures, and so on. API Gateways must be able to interact with user stores without compromising them.

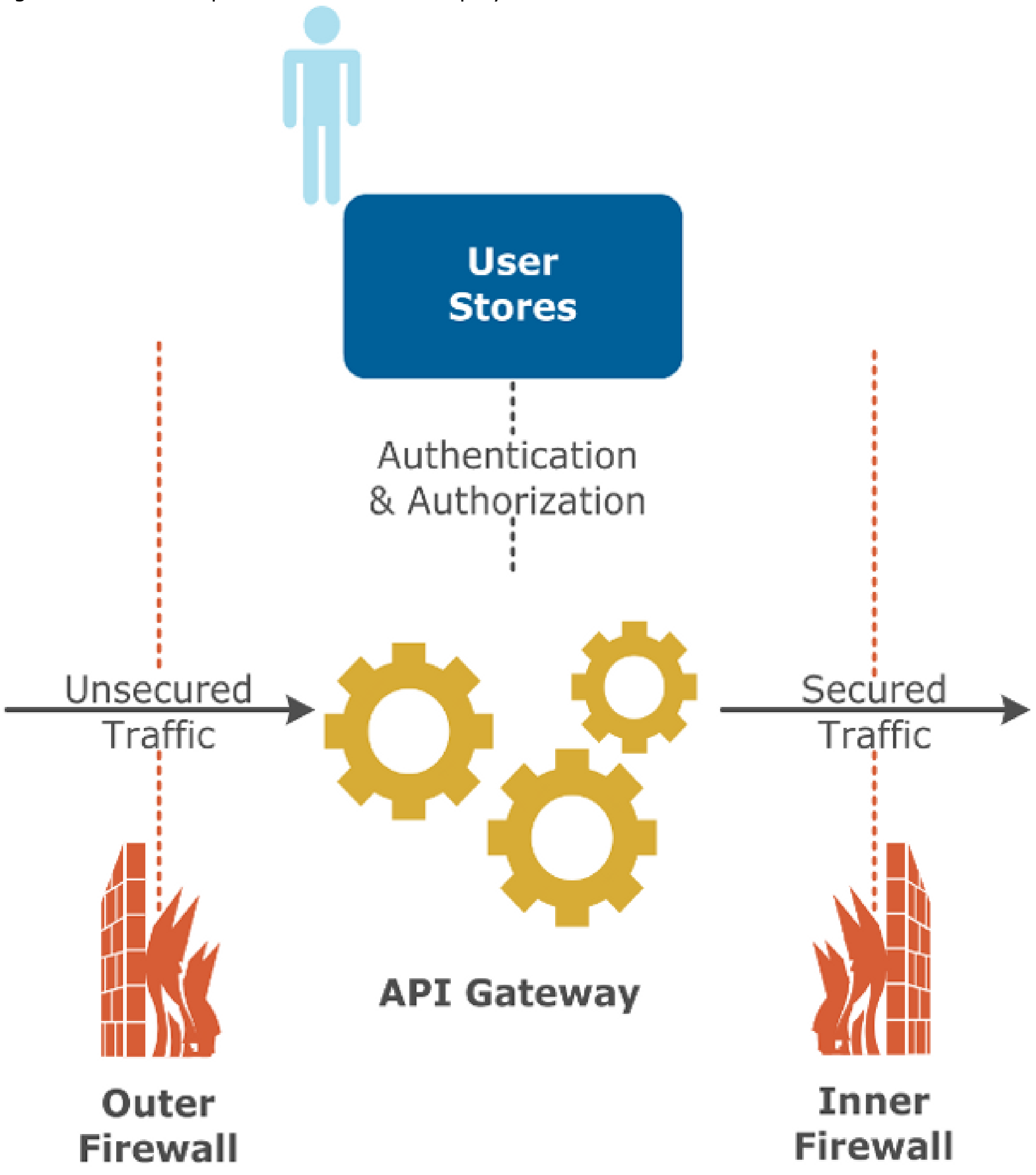
The API Gateway can use LDAP directories to retrieve user information such as the following:

- 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 Integration Guide*.

Simple inline user store deployment

The following diagram shows a simple inline user store deployment:



Advantages
This architecture has the following advantages:

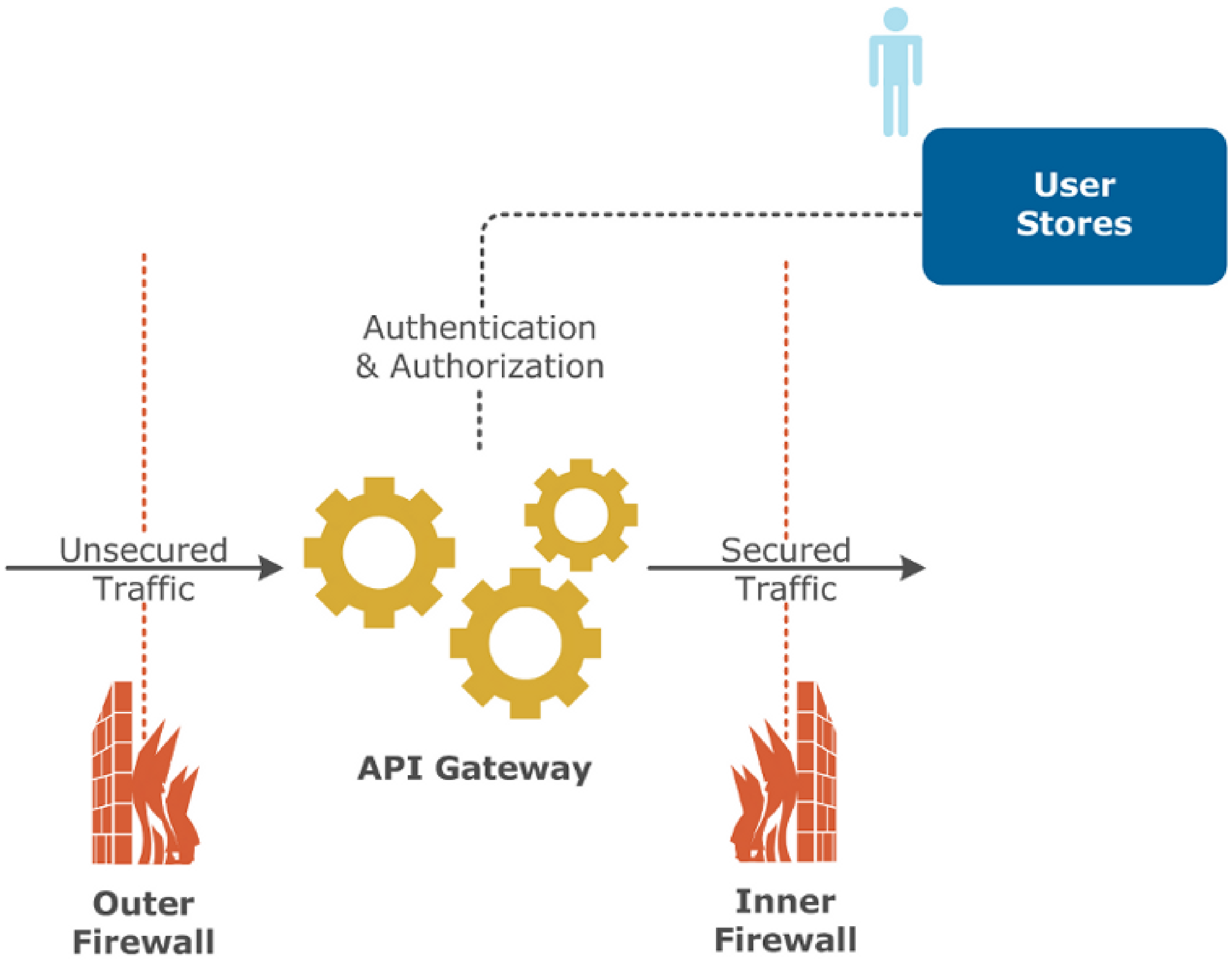
- This is simple and easy to set up.
- There is a single entry point through the DMZ for all message traffic.
- This is the least expensive option.

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.

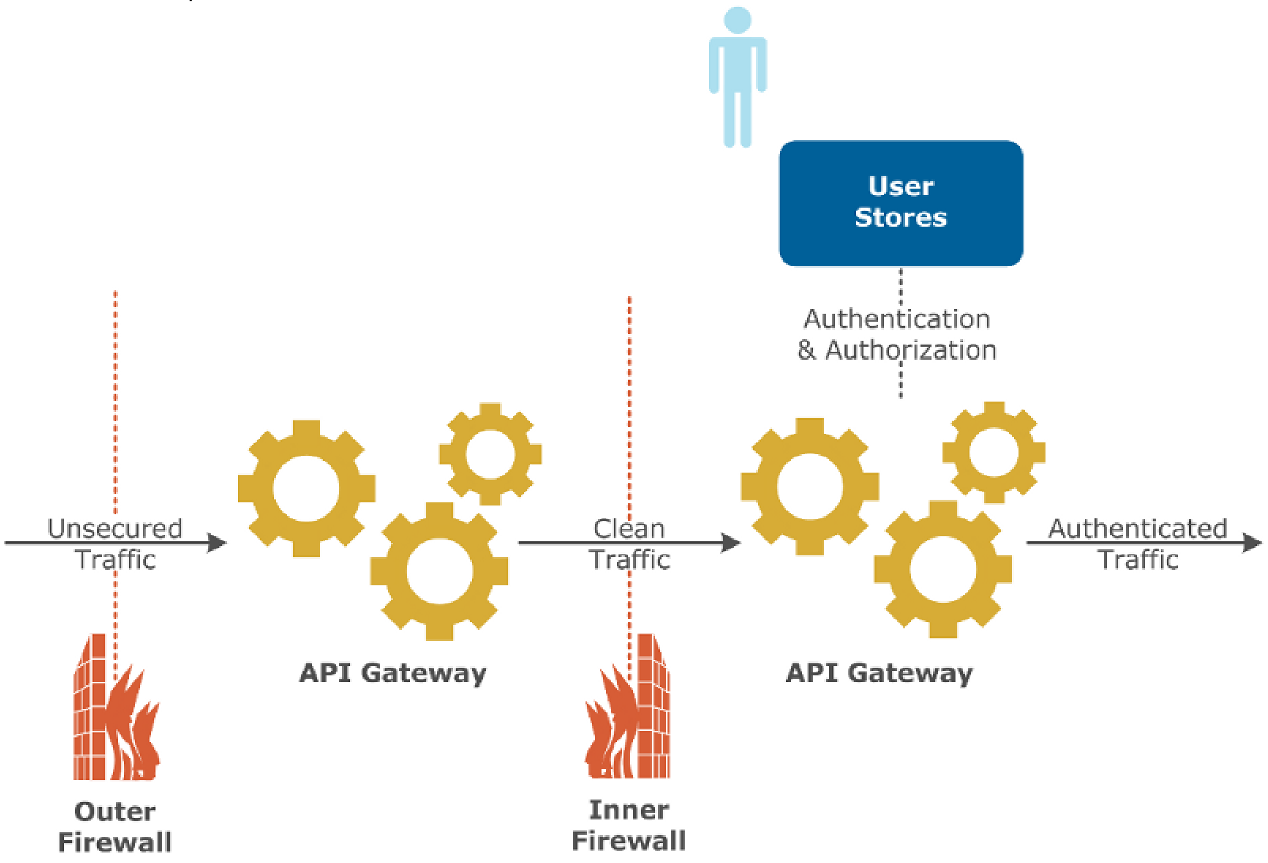
Disadvantages

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:

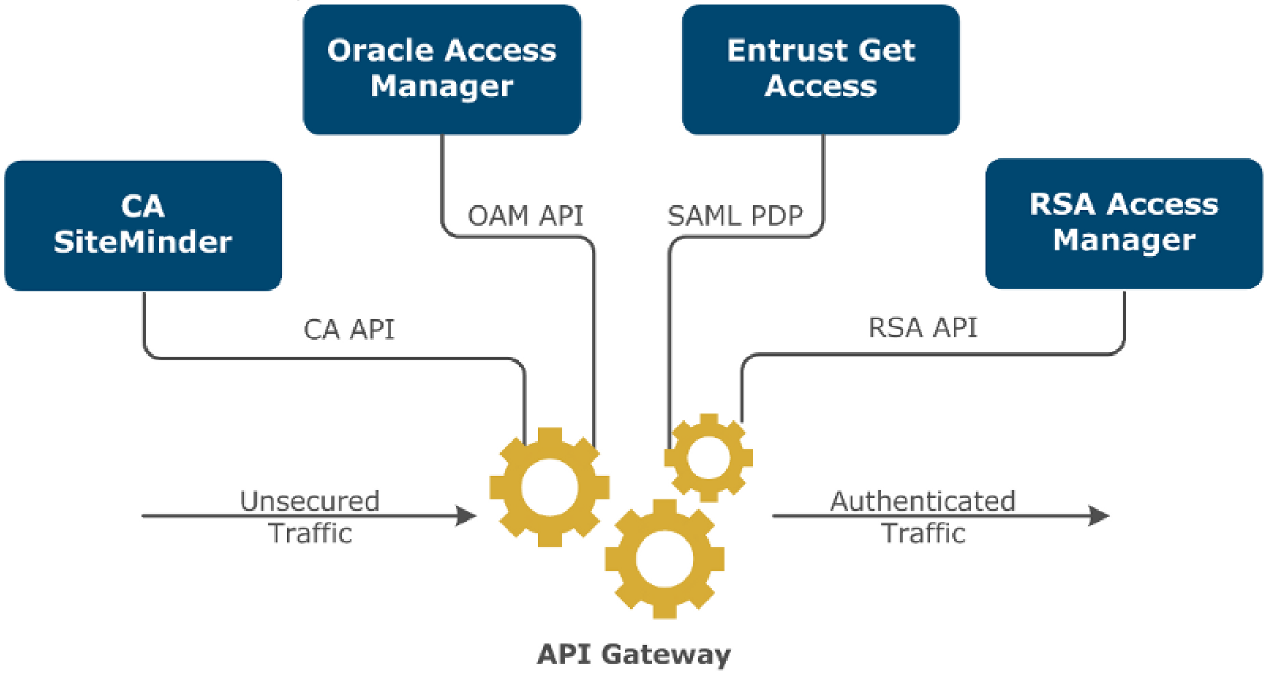
- This is the most secure deployment available.
- By separating threats from access control, it is easy to run separate security policies for internal and external traffic.

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:

- 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.

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 certificates and keys*.

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 User 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 Management Guide*.

Software Configuration Managment

The API Gateway does not include its own built-in Software Configuration Managment 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 Concurent 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 enviroments. 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 Deployment and Promotion Guide*. For details on exporting API Gateway policies and general configuration, and on Compare/Merge, see the *API Gateway User Guide*.

Configure a managed domain

Overview

This topic describes how to use the `managedomain` script to configure a managed API Gateway domain on the command line. 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.

Important

This chapter assumes that you have already installed the API Gateway (see the *API Gateway Installation and Configuration 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*

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 Windows and Linux/Solaris 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.

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

Windows	`INSTALL_DIR\apigateway\Win32\bin`
UNIX/Linux	`INSTALL_DIR/apigateway/posix/bin`

Note

To register an API Gateway instance as a service on Windows or Linux/UNIX, you must run the `managedomain` command as `Administrator` on Windows or `root` on Linux/UNIX.

For more details on `managedomain` options, see *Managedomain Command Reference*.

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:

Windows	`INSTALL_DIR\apigateway\Win32\bin`
UNIX/Linux	`INSTALL_DIR/apigateway/posix/bin`

2. Enter the following command:

```
managedomain
```

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
```

Important

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:

Windows	`INSTALL_DIR\apigateway\Win32\bin`
UNIX/Linux	`INSTALL_DIR/apigateway/posix/bin`

3. Enter the following command:

```
managedomain
```

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 the 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 Windows or UNIX services for the API Gateway, you must run `managedomain` on same host.

You must run `startinstance` on the host on which you intend to start the instance. On UNIX/Linux, 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 the API Gateway tools*.

Manage domain topology in API Gateway Manager

Overview

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:

- Create and delete API Gateway groups
- Lock and unlock API Gateway groups
- Create and delete API Gateway instances
- Start and stop API Gateway instances
- Create and edit API Gateway configuration tags
- Deploy API Gateway configuration

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 a managed domain*.

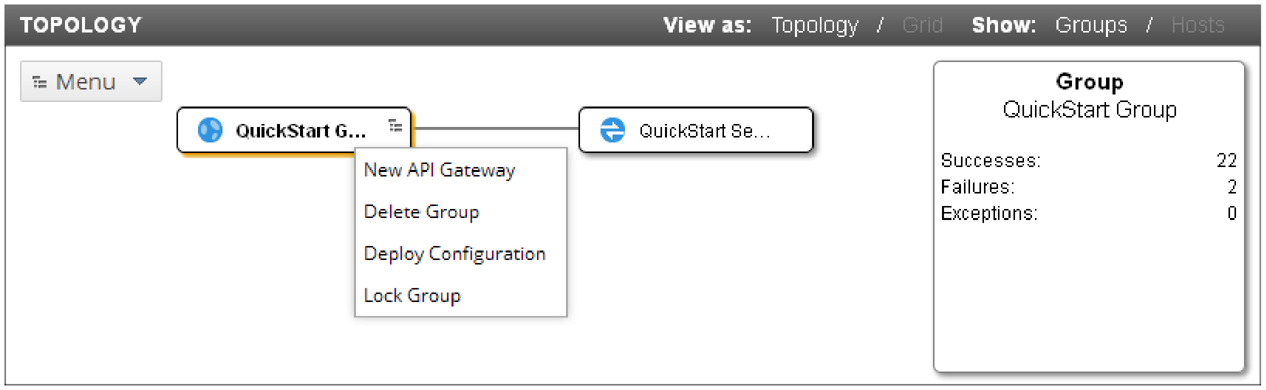
The API Gateway Manager web console is available from the following URL:

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

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

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 admin 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 API Gateway instances

To use the API Gateway Manager to create an API Gateway instance, 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 **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`).
4. Click **OK**.

Delete API Gateway instances

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

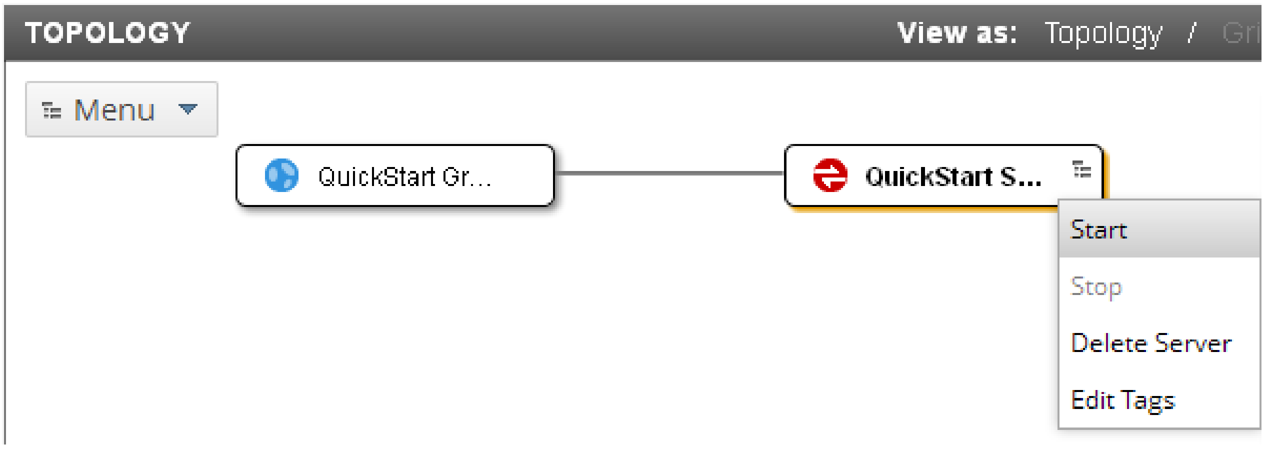
1. Ensure that the API Gateway instance has been stopped.
2. Hover over the API Gateway instance in the topology view, and click the edit button on the right.
3. Select **Delete Server**.
4. Click **OK**.

Start API Gateway instances

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 API Gateway instances

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 API Gateway tags

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. In the dialog, click the green plus icon to add a tag.
3. Enter a **Tag** name (for example, `Department`), and a **Value** (for example, `Engineering`).
4. 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:

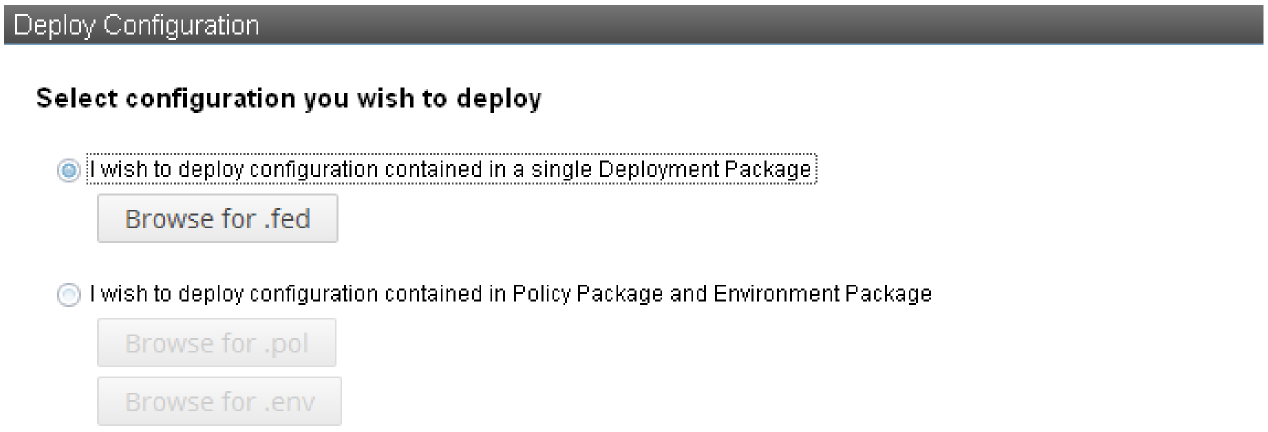
- deployment package (`.fed`)
- policy package (`.pol`)
- environment package (`.env`)

For more details on configuration packages, see the *API Gateway Promotion and 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 Configuration

Select configuration you wish to deploy

[✓ I wish to deploy configuration contained in a single Deployment Package]

I wish to deploy configuration contained in Policy Package and Environment Package

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**.

Managedomain Command Reference

Overview

For an overview of the managedomain command, see *Configure a managed domain*. To run the command, enter `managedomain` in the following directory, and follow the instructions at the command prompt:

Windows	`INSTALL_DIR\apigateway\Win32\bin`
UNIX/Linux	`INSTALL_DIR/apigateway/posix/bin`

Host Management

The `managedomain` command 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. You must ensure that the host is registered in order to create and run API Gateways. For example, you can specify the following: • If host is an Admin Node Manager • Use SSL • Hostname • Node Manager name • Node Manager port • Node Manager passphrase • Windows/UNIX service for Node Manager • Trust store details If the host you are registering is not the Admin Node Manager, you must specify the Admin Node Manager host details. The Admin Node Manager must also be running.
2	`Edit a host`	Edit the details for a host registered in a domain topology. Used occasionally. You can update the following: • Hostname • Node Manager name • Node Manager port • Node Manager passphrase • Windows/UNIX service for Node Manager • Use SSL When you get a license for an evaluation mode API Gateway, you must use this option to change the

Option	Description	Why Use this Option
		host from `127.0.0.1` to a network reachable address or hostname. 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. This option is only for use on a remote non-Admin Node Manager node that is being removed from the topology. You must also stop the remote Node Manager process. This option will not work on the Admin Node Manager host, or if run locally on the host to be removed.
4	`Change credentials for Admin Node Manager, currently connecting as: user admin with truststore None`	By default, you connect to the Node Manager using `managedomain` with the credentials `admin/changeme`. 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.

API Gateway Management

The `managedomain` command 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 an API Gateway (rename, change management port)`	Rename the API Gateway instance, enable/disable SSL, 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 an 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.
9	`Delete a tag from an 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 a Windows/UNIX service for existing local API Gateway Group Management`	Must be run by a user with permission to create a service on the host operating system (`root` on Linux, or `Administrator` on Windows). When run on Linux,

Option	Description	Why Use this Option
		adds an `init.d` script.

Group Management

The `managedomain` command 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` command 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 Manager • Groups • API Gateway instances (tags)
`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 sync of all topologies.
`17`	`Reset the local topology`	For advanced users. Delete the contents of the `apigateway/groups` directory. This means that you would need to 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` command 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.
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` command 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 first stop the Node Manager on the localhost before running this option. You must run this option on all hosts in the domain.

Start and stop the API Gateway

Overview

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 *Configure a managed domain*.

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*.

Start the Node Manager

The following instructions describe how to start the Node Manager on the command line on Windows and UNIX:

Windows
Complete the following steps to start the Node Manager on Windows:

1. Open a DOS prompt at the `/Win32/bin` directory of your API Gateway installation.
2. Run the `nodemanager.bat` file.
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*.

Linux/Solaris
To start the Node Manager on Linux/Solaris, complete the following instructions:

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*.

Start the API Gateway instance

The following instructions describe how to start the API Gateway instance on the command line on Windows and UNIX:

Windows
Complete the following steps to start the API Gateway on Windows:

1. Open a DOS prompt at the `/Win32/bin` directory of your API Gateway installation.
2. Use the `startinstance` command to start the API Gateway, for example:

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

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 Setting Passphrases.
4. When the API Gateway has successfully started up, you can run the `policystudio.exe` file from your Policy Studio installation directory.
5. When the Policy Studio is starting up, you are prompted for connection details for the API Gateway. For more details, see Connecting to the API Gateway.

Linux/Solaris
To start the API Gateway instance and the Policy Studio on Linux/Solaris, 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 the 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 Setting Passphrases.
4. When the API Gateway has successfully started up, run the `policystudio.sh` file in your Policy Studio installation directory. For example:

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

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

Tip

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

```
INSTALL_DIR\Win32\bin>startinstance

usage: "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 (UNIX only).
`-h <directory>`	Specifies the service instance directory.
`-k`	Kills the instance (UNIX only).
`-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 (UNIX only).
`-e`	Specifies the end of arguments for `vshell`.
`-D prop[=value]`	Sets a configuration file property.

Linux/Solaris
The `-d`, `-s` and `-k` options on UNIX 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
```

Windows
The Service Control Manager (SCM) on Windows requires more direct interaction with the process than on UNIX. The product will run as a service, but needs to be started by the SCM.

The `-d` and `-k` options can be used internally as part of the SCM interaction, but are not for interactive use on Windows. These are used to differentiate between a command-line startup and a startup from the SCM. `-s` is not implemented on Windows. When installed as a service, you can control and query the API Gateway using the SCM (for example, using the applet, or the `net` or `sc` commands).

Connect to the 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
```

where `HOST` points to the IP address or hostname of the machine on which the API Gateway runs. For more details, see the *API Gateway User Guide*.

Stop the API Gateway instance

Linux/Solaris
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
```

Windows
To stop the API Gateway instance, you can enter **Ctrl-C** on the command prompt. If the API Gateway instance is installed as a Windows service, you should use the Windows Services tool.

Stop the Node Manager

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

```
./nodemanager -k
```

Windows
To stop the Node Manager, you can enter **Ctrl-C** on the command prompt. If the Node Manager is installed as a Windows service, you should use the Windows Services tool.

Start the API Gateway tools

Overview

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 and Configuration Guide*.

Configure a managed domain
If you have not already created a domain, you can use the `managedomain` script to configure a domain. You should also 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 hostname or IP address of the machine on which the API Gateway is running (for example, `https://localhost:8090/`).
2. Enter your user name and password. The default values are as follows:

User Name	`admin`
Password	`changeme`

3. Click the appropriate button in the API Gateway Manager screen 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 the API Gateway. For example, all HTTP, HTTPS, JMS, File Transfer, and Directory messages processed by the API Gateway.
- **Monitoring**: Real-time monitoring of all the traffic processed by the 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 the **Policy Studio**, click a link to connect to the Admin Node Manager session.
3. In the **Open Connection** dialog, click **OK** to accept the defaults.
4. The **API Gateway** instance is displayed in the **Topology** view.
5. In the **Topology** view, double-click the **API Gateway** instance to load the configuration for the active API Gateway.
6. If a passphrase has been set, enter it in the **Enter Passphrase** dialog, and click **OK**. Alternatively, if no passphrase has been set, click **OK**. For more details, see *Configure an API Gateway encryption passphrase*.

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*.

Configure an API Gateway encryption passphrase

Overview

By default, the API Gateway configuration data is stored unencrypted. However, you can encrypt sensitive information such as passwords and private keys using a 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 the Policy Studio, or when the API Gateway is starting up, so that the encrypted data can be decrypted. You can enter the encryption passphrase at the level of an API Gateway group or an API Gateway 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).

Note

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 the encryption passphrase when connecting to the API Gateway with the Policy Studio, in your API Gateway configuration file, or when the API Gateway is starting up. It also describes how to change the API Gateway group passphrase when it has been set initially.

Configure the passphrase in Policy Studio

You can use the Policy Studio topology view to set the passphrase used to encrypt the API Gateway group data. This is the table displayed when you connect to the Admin Node Manager. To change the passphrase, right-click the API Gateway group name in the table (for example, **QuickStart Group**), and select **Change Passphrase**.

Complete the following fields on the **Change Group Passphrase** dialog:

Old Passphrase:
Enter the old passphrase that you wish to change in this field. Alternatively, you can leave this field blank if you are setting the passphrase for the first time.

New Passphrase:
Enter the new passphrase.

Confirm New Passphrase:
Re-enter the new passphrase to confirm it.

Warning

It is *crucial* that you remember the passphrase when you change it. Failure to remember the passphrase results in the loss of private key data.

Enter the passphrase when you connect in Policy Studio

When you have set the encryption passphrase for the API Gateway group configuration data, you must enter this passphrase every time you connect to the API Gateway in the Policy Studio. You can enter it in the **Passphrase** field of the **Open File** dialog, which is displayed when connecting to a configuration file. Alternatively, you can enter it in the **Enter Passphrase** dialog, which is displayed before editing an active server configuration.

Note

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.

Enter the passphrase in a file or on startup

For the API Gateway to read (decrypt) encrypted data from its configuration, it must be primed with the passphrase key. You can do this using the Policy Studio, as explained in the previous section. You can also enter the passphrase directly in a configuration file, or prompt for it at startup.

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

```
INSTALL_DIR/system/conf/nodemanager.xml
```

This file contains values for general system settings, such as the server name and trace level, and also (if required) the passphrase key to use to decrypt encrypted API Gateway configuration data.

Typically, the passphrase is only entered directly in the file if the server must be started as a Windows service or UNIX 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. You should specify the passphrase as the value of the `secret` attribute as follows, where `"myPassphrase"` is the encryption passphrase:

```
secret="myPassphrase"
```

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

```
INSTALL_DIR/groups/group-id/instance-id/conf/service.xml
```

Prompt for the passphrase on 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 Windows service or UNIX daemon, 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)"
```

Important

If you use this option, you must take care to remember the encryption passphrase. Failure to use the correct passphrase results in loss of private key data, and may prevent the API Gateway from functioning correctly.

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

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) must be the same as the passphrase in the source configuration (testing). If you are using a different passphrase in each environment, before the deployment takes place, you must make a copy of the configuration (for example, `.fed` file), and set it with the passphrase of the target configuration.

For more details, see the *API Gateway Deployment and Promotion Guide*.

Run API Gateway as non-root on UNIX/Linux

Overview

In a typical deployment on Linux or Solaris, the process for the API Gateway instance runs as `root`. This is typically used to enable the process to listen on Internet domain privileged ports (port numbers less than `1024`). However, this is a potential security issue because the root user has read/write access to all files on the system. Therefore, if the API Gateway process ever becomes compromised, it could be used to return the contents of, or overwrite sensitive files, and the operating system would not prevent this. A solution to this problem is to run the API Gateway as a non-root user, but still allow the API Gateway process to bind to privileged ports.

Important

The steps in this topic apply to the following API Gateway processes:

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

Linux capabilities

Traditional UNIX implementations distinguish between the following categories of processes:

- *privileged* processes whose effective user ID is `0` (`superuser` or `root`)
- *unprivileged* processes whose effective user ID is non-zero

Privileged processes bypass all kernel permission checks where unprivileged processes are subject to full permissions checking based on the processes credentials, usually from the effective user ID and effective group ID. More recent versions of the kernel divide up the privileges traditionally associated with the `superuser` into a set of capabilities that can be independently enabled or disabled. This allows a more fine-grained control of permissions for a process.

The capability to use with the API Gateway is `CAP_NET_BIND`, which specifically allows binding to privileged ports. The method by which this capability is set on the API Gateway is supported from kernel 2.6.24 onwards. If the kernel version is before 2.6.33, you must enable `CONFIG_SECURITY_FILE_CAPABILITIES`.

Before you begin

The sections that follow describe the steps that you must perform to run the API Gateway as an unprivileged user. This topic assumes that the new unprivileged user is named `admin`, and that the location of your API Gateway installation is `/home/admin/apigateway`.

Modify API Gateway file ownership

The unprivileged user must have ownership of the API Gateway files. If there is a pre-existing API Gateway installation, you must change the ownership of the API Gateway files to the new user that you intend to run the API Gateway with.

You can use the following command to change the user and group ownership of all files under the installation directory:

```
# chown -R user:usergroup /path/to/apigateway/installation
```

For example:

```
# chown -R admin:admin /home/admin/apigateway
```

SSL accelerators for HSM

When using a Hardware Security Module (HSM), the unprivileged user must have access to the device corresponding to the crypto 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`
- **Utimaco**: `/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
```

If the unprivileged user is different from `admin`, run the following command:

```
# sed -i /admin/d /lib/udev/load_{cavium,utimaco}
```

Set the CAP_NET_BIND capability on vshell

You must add the Linux capability to allow the API Gateway to listen on ports less than `1024` to the `vshell` file.

You can use the following command:

```
# setcap 'cap_net_bind_service=+ep' /path/to/apigateway/installation//platform/bin/vshell
```

For example:

```
# setcap 'cap_net_bind_service=+ep' /home/admin/apigateway/platform/bin/vshell
```

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

```
# getcap /home/admin/apigateway/platform/bin/vshell
```

For example, the output of this command should be as follows:

```
/home/admin/apigateway/platform/bin/vshell = cap_net_bind_service+ep
```

Note

For API Gateway versions prior to 6.3.0 (for example, 6.1.2), this path should be `platform/libexec/vshell` instead of `platform/bin/vshell`.

Install the libcap2 package if required

Depending on your Linux distribution, this may involve installing an additional software package. If the `setcap` command cannot be found, try installing the `libcap2` package.

For yum-based systems (for example, Redhat Enterprise Linux, CentOS, Oracle Enterprise Linux), you can use the following command:

```
# yum install libcap2
```

For Debian or Ubuntu, you can use the following command:

```
# apt-get install libcap2-bin
```

API Gateway appliance version 7.1.0 or later

You must set an additional privilege for API Gateway appliance version 7.1.0 or later. This step applies if you have an appliance and wish to run the `vshell` processes as a user other than the default `admin` user.

Run the following command (all on one line):

```
# setcap 'cap_net_bind_service=+ep cap_sys_rawio=+ep'
   /path/to/apigateway/installation/platform/bin/vshell
```

For example:

```
# setcap 'cap_net_bind_service=+ep cap_sys_rawio=+ep'
   /home/admin/apigateway/platform/bin/vshell
```

To verify that the permissions have been set, run the following command:

```
# getcap /home/admin/apigateway/platform/bin/vshell
```

For example, the output of this command should be as follows:

```
/home/admin/apigateway/platform/bin/vshell = cap_net_bind_service,cap_sys_rawio+ep
```

Add API Gateway library locations

When executing a file with elevated permissions, 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. You can do this using the steps described in this section:

1. Create an `ld.so.conf` file with the API Gateway shared object locations
2. Run `ldconfig` to reload the `ld.so.cache` file

Create the ld.so.conf file

For a 64-bit API Gateway installation, create the following file:

```
/etc/ld.so.conf.d/gateway-libs.conf
```

And add the following lines:

```
/home/admin//apigateway/platform/jre/lib/amd64/server
/home/admin/apigateway/platform/jre/lib/amd64
/home/admin/apigateway/platform/lib/engines
/home/admin/apigateway/platform/lib
/home/admin/apigateway/ext/lib
```

For a 32-bit API Gateway installation, create the following file:

```
/etc/ld.so.conf.d/gateway-libs.conf
```

And add the following lines:

```
/home/admin/gateway/platform/jre/lib/i386/server
/home/admin/gateway/platform/jre/lib/i386
/home/admin/gateway/platform/lib/engines
/home/admin/gateway/platform/lib
/home/admin/gateway/ext/lib
```

Note

You should replace `/home/admin/gateway` with the root of your API Gateway installation.

Run ldconfig

After creating the `/etc/ld.so.conf.d/gateway-libs.conf` file, run the following command to reload the library cache file:

```
# ldconfig
```

Modify the init.d script to use sudo

If there is an `init.d` script that runs the API Gateway as a service, you must change this to execute as the unprivileged user. The easiest way is by using `sudo -u admin` in the start command of the script.

For API Gateway version 6.1, this means changing the following line in the `/etc/init.d/vordelgateway` script:

```
daemon --pidfile $PIDFILE $DAEMON $DAEMON_ARGS >> $LOGFILE 2>&1
```

to the following:

```
daemon --pidfile $PIDFILE --user=admin $DAEMON $DAEMON_ARGS >> $LOGFILE 2>&1
```

Modify the jvm.xml file

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, add an extra line after the following line:

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

3. For a 64-bit API Gateway installation, add 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"/>
   ```

4. For a 32-bit API Gateway installation, add the following:

   ```
   <VMArg name="-Djava.library.path=$VDISTDIR/$DISTRIBUTION/jre/lib/i386/server:
      $VDISTDIR/$DISTRIBUTION/jre/lib/i386:$VDISTDIR/$DISTRIBUTION/lib/engines:
      $VDISTDIR/ext/$DISTRIBUTION/lib:$VDISTDIR/ext/lib:$VDISTDIR/$DISTRIBUTION/jre/lib:
      system/lib:$VDISTDIR/$DISTRIBUTION/lib"/>
   ```

Restart the API Gateway

When you have completed the steps described in this topic, start the API Gateway with the unprivileged user.

Run API Gateway as non-root on Solaris

On Solaris 10, a similar system is provided by adding the `net_privaddr` privilege to a user.

Note

On Solaris, you must modify the user instead of the API Gateway process (as already described for Linux in this topic).

In this case, you can modify an existing user (for example, `gwadmin`) as follows:

```
# usermod -K defaultpriv=basic,net_privaddr gwadmin
```

Configure High Availability

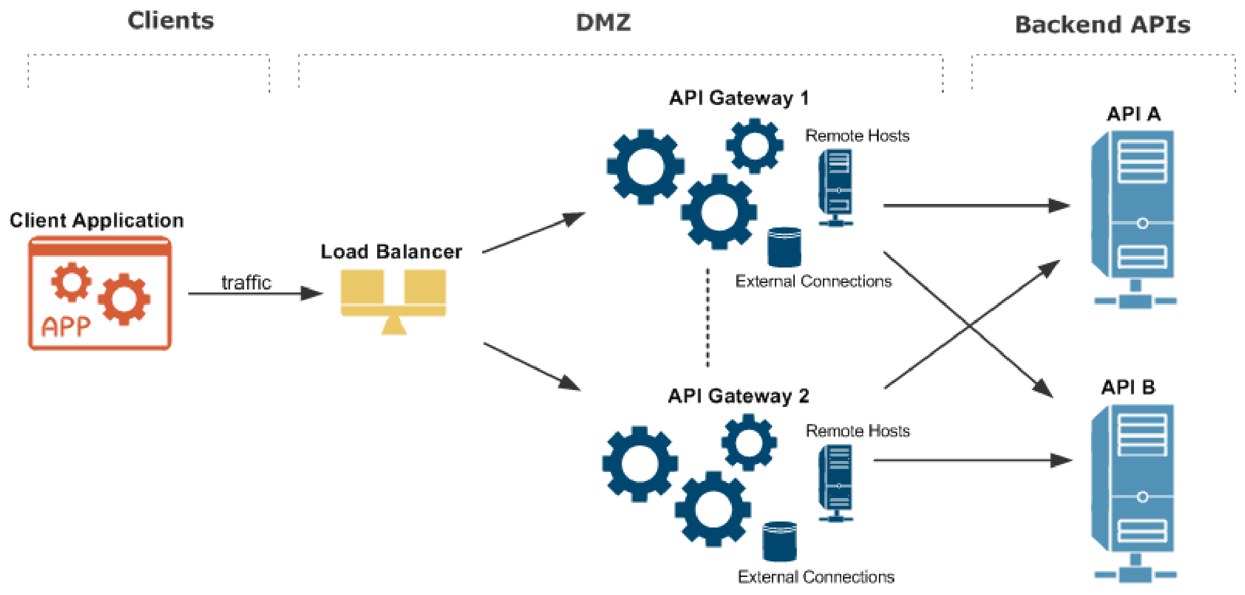
Overview

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 backend 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, embedded 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, Local 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 backend API systems, and which can balance the message load based on specified priorities for Remote Hosts.
- Each API Gateway instance contains an embedded 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 filesystem.
- Each backend 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`).

Important

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, the API Gateway can see that incoming messages are sent from specific client and load balancer IP addresses. The 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, the API Gateway sees only the virtual service address of the load balancer.

In addition, the API Gateway can also act as load balancer on the outbound connection to the backend APIs. For more details, see the section called “Remote Hosts”.

Java Message System

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

- SonicMQ
- Tibco Rendezvous
- Fiorano
- OpenJMS
- JBoss Messaging

The 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 User Guide*. For details on configuring HA in supported third-party JMS systems, see the user documentation available from your JMS provider.

The API Gateway also provides an embedded Apached ActiveMQ server in each API Gateway instance. For more details, see the section called "Embedded Apache ActiveMQ".

File Transfer Protocol

The API Gateway supports the following protocols:

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

When using FTP protocols, the 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 User Guide*.

Remote Hosts

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

- Mapping a hostname 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 *API Gateway User 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 User Guide*. The 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 the 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 *API Gateway User Guide*. For details on how to configure HA for any external third-party systems, see the product documentation provided by your third-party vendor.

Important

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

Embedded Apache ActiveMQ

The 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).

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*
- *Manage embedded ActiveMQ messaging*

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

Embedded Apache Cassandra database

Each API Gateway instance contains an embedded Apache Cassandra database for default persistent data storage. This Cassandra database is used 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 the embedded Apache Cassandra database for HA. The embedded Cassandra database is API Gateway group-aware, which means that all API Gateways in a group can share the same Cassandra data source. In a production environment, you must configure the Cassandra data source to provide HA for the API Gateway group.

For more details on Apache Cassandra, see http://cassandra.apache.org/. For details on the KPS, see the *Key Property Store User Guide*, available from Axway Support.

Note

All nodes in a Cassandra cluster must run on the same operating system (for example, Windows, Linux, or Solaris).

Configure a seed node

HA Cassandra configuration requires multiple API Gateway instances running on separate host nodes. This section describes how to configure a seed node and add additional nodes with a replication factor of two. For production environments, you should design your API Gateway topology and replication factors before you begin.

To configure a Cassandra seed node, perform the following steps:

1. Decide which API Gateway instance in the group is the seed node. Seed nodes must be running when adding new nodes to the system.
2. All Cassandra configuration is stored at the instance level in `cassandra.yaml`, for example:

   ```
   <install-dir>/apigateway/groups/group-2/instance-1/conf/kps/cassandra/cassandra.yaml
   ```

   In `cassandra.yaml`, set `listen_address` and `seed_provider` to a network address on the host. This address must be accessible by other API Gateway instances in the group.

   Note

   `rpc_address` should remain as `localhost` for this configuration.
3. Start the API Gateway instance with Java Management Extensions (JMX) enabled using the `-DenableJMX` flag. For example:

   ```
   startinstance -g "Group A" -n "API Gateway 1" -DenableJMX
   ```
4. Wait for the API Gateway instance to start.

Note

This is the same as the default single node configuration except for the following:

- JMX is enabled for Cassandra administration
- Cassandra is listening for other Cassandra nodes on a network accessible address (`listen_address`)
- Cassandra is still listening for client connections on `localhost` only (`rpc_address`)

Configure subsequent nodes

On second and subsequent API Gateway nodes, perform the following steps:

1. In `cassandra.yaml`, set the following:
   - `listen_address` to a network address on the host, which must be accessed by other API Gateway nodes in the group (change from `localhost`)
   - `seed_provider` to the address of the seed node (change from `localhost`)
2. Start the API Gateway instance with JMX enabled, for example:

   ```
   startinstance -g "Group A" -n "API Gateway 1" -DenableJMX
   ```

3. Wait for the API Gateway instance to start.

View embedded Cassandra configuration

If you run the `kpsdamin` tool, and select **Embedded Service Administration** > **Show Configuration**, you can view output for the configured nodes in the cluster. For example, this includes details such as the following for each node:

- `seeds`: the seed node(s), which are common for each node in the cluster
- `listen address`: the node address, which is different for each node
- Cassandra clients connect to a `localhost` address on port `9160` (server: `rpc_address`, `rpc_port`, client: `hosts: localhost:9160`)

Note

You must ensure the Admin Node Manager is running on each node to retrieve this information.

View hidden KPS configuration in Policy Studio

To display hidden KPS collection configuration in Policy Studio, perform the following steps:

1. Edit the following file:

   ```
   <install-dir>\policystudio\policystudio.ini
   ```

2. Add the following line, and save the update:

   ```
   -Dshow.internal.kps.collection=true
   ```

3. Start Policy Studio using the `policystudio` command.

Set the replication factor

You can use the Cassandra command-line interface (`cassandra-cli`) to set the replication factor. Perform the following steps:

1. Enter `cassandra-cli [-h hostname]` to connect the Cassandra command-line interface to the specified node.
2. Enter `use kps;` to use the `kps` keyspace.
3. Set `strategy_options = {replication_factor:2};`.

4. Enter `quit;`.
5. Run the following command against each API Gateway instance to synchronize the update:

   ```
   nodetool -h server-address repair kps
   ```

   Wait until the command completes before moving to the next API Gateway instance. The time depends on how much data you have, but on a new cluster this should happen quickly.
6. Run the following command against each API Gateway instance to display cluster information:

   ```
   nodetool -h server-address ring kps
   ```

   You should see an effective ownership of 100% on each node.
7. Start Policy Studio enabled to show the hidden Cassandra Data Sources. For more details, see the section called "View hidden KPS configuration in Policy Studio").
8. In the Policy Studio tree, under **Key Property Stores**, select **Data Sources** > **Embedded Storage** at the API Gateway KPS collection and/or table level. You can edit the collection if all tables use the default data source in that collection.

   For OAuth applications, you can select OAuth Token Stores in Cassandra under **Libraries** > **OAuth2 Stores**.
9. Set the read and write consistency level to at least ONE.
10. Deploy the configuration.

Configure API Gateway group-wide HA

This section describes how to share a KPS collection across API Gateway groups.

Note

Before performing the steps in this section, second and subsequent nodes must be registered as hosts of the seed node using the `managedomain` command. For more details, see *Configure a managed domain*.

Perform the following steps:

1. Set up a KPS collection for an initial group as described in the section called "Set the replication factor". Ensure this is working before setting up HA for the group.
2. In the Policy Studio tree, right-click the KPS collection, and select **Export** to export the configuration in an `.xml` file.
3. On the new node, click **Import** in the Policy Studio toolbar, and browse to the previously exported `.xml` file.
4. Select the `.xml` file, and click **Open** to import the selected configuration.
5. Deploy the updated configuration.

The HA configuration can then be applied to the nodes in the second and subsequent groups in this way. The seed nodes are used across the groups. You can use the seed node from the initial group if this has been configured.

Configure Java Management Extensions

Cassandra uses Java Management Extensions (JMX) for administration tasks such as repairing and compacting data. JMX must be enabled for Cassandra HA configuration. Cassandra JMX configuration is stored in:

```
<install-dir>/groups/<group-id>/<instance-id>/conf/kps/cassandra/jvm.xml
```

To enable this configuration, you must specify `enableJMX` when starting the API Gateway, for example:

```
startinstance -g "Group A" -n "API Gateway 2" -DenableJMX
```

The default JMX configuration is an insecure JMX configuration using port `7199`. This is the Cassandra JMX monitoring port. After the initial handshake, the JMX protocol requires that the client reconnects on a randomly chosen port (`1024+`). The following shows some example settings:

```
<?xml version="1.0" encoding="UTF-8"?>
<ConfigurationFragment>
    <if property="enableJMX">
        <VMArg name="-Dcom.sun.management.jmxremote"/>
        <VMArg name="-Dcom.sun.management.jmxremote.port=7199"/>
        <VMArg name="-Dcom.sun.management.jmxremote.authenticate=false"/>
        <VMArg name="-Dcom.sun.management.jmxremote.ssl=false"/>
    </if>
</ConfigurationFragment>
```

Cassandra housekeeping tasks

You should run the Cassandra `nodetool repair` and `compact` commands at regular intervals against all Cassandra nodes in the cluster. Ideally, these commands should be performed at different times on different nodes. For more details, see http://wiki.apache.org/cassandra/Operations.

You can use the `kpsadmin` tool to issue these commands to Cassandra nodes on a user-defined schedule. For example:

```
kpsadmin -housekeeping <nodelist> <commands> <cron schedule>
```

This syntax is described as follows:

- *`nodelist`* is a comma-separated list of JMX `hostname:port` pairs
- *`commands`* is a comma-separated list of of Cassandra commands (`ring`, `repair`, or `compact`)
- *`cron schedule`* is a cron expression that specifies when the command runs

For example, you can use the following command to run repair, compact, and ring every day against `localhost` at 23:15:

```
kpsadmin -housekeeping "localhost:7199" "repair,compact,ring"  "0 15 23 ? * *"
```

For more details on configuring Cassandra, see:
http://www.datastax.com/documentation/cassandra/1.2/webhelp/

Manage certificates and keys

Overview

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's certificate, or the issuing CA's certificate into the API Gateway's certificate store.

In addition to importing CA certificates, you can also import and create server certificates and private keys in the certificate store. 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 private keys

To view the lists of certificates and private keys stored in the certificate store, select **Certificates and Keys** > **Certificates** in the tree on the left of the Policy Studio. The certificates and keys are listed on the following tabs in the **Certificates** window on the right:

- **Certificates with Keys**: Server certificates with associated private keys.
- **Certificates**: Server certificates without any associated private keys.
- **CA**: Certification 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.

Configure an X.509 certificate

To create a certificate and private key, click the **Create/Import** button. 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 the **Edit** button to configure the *Distinguished Name* (DName) of the subject.
- **Alias Name**:
  This mandatory field enables you specify a friendly name (or alias) for the certificate. Alternatively, you can click **Use Subject** button to add the DName of the certificate in the text box instead of a certificate alias.
- **Public Key**:
  Click the **Import** button 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 this setting if you wish 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 button on the right to select an issuer certificate. This setting is not selected by default.

- **Validity Period**:
  The dates specified here define 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 this button to import a certificate (for example, from a `.pem` or `.der` file).
- **Export Certificate**:
  Use this option to export the certificate (for example, to a `.pem` or `.der` file).

Bind to a certificate at runtime

Alternatively, when configuring an HTTPS interface, you can also specify a certificate to bind to at runtime. In the **Configure HTTPS Interface** dialog, click **X.509 Certficate**, and select **Bind the Certificate at runtime**. For example, you can enter `${env.serverCertificate}`, and enter the certificate as an environment variable in the `envSettings.props` file.

For more details on configuring HTTPS interfaces, see the *API Gateway User Guide*. For more details on specifying environment variables, see the *Deployment and Promotion Guide*.

Configure a private key

Use the **Private Key** tab to configure details of the private key. By default, private keys are stored locally in the certificate store. They can also be stored on a Hardware Security Module (HSM), if required.

Private Key Stored Locally:
Select the **Private key stored locally** radio button. The following configuration options are available for keys that are stored locally in the certificate store:

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

Private key stored on HSM:
If the private key that corresponds to the public key stored in the certificate resides on a HSM, select the **Private key stored on HSM** radio button. Configure the following fields to associate a key stored on a HSM with the current certificate:

- **Engine Name**:
  Enter the name of the OpenSSL Engine to use to interface to the HSM. All vendor implementations of the

OpenSSL Engine API are identified by a unique name. Please refer to your vendor's HSM or OpenSSL Engine implementation documentation to find out the name of the engine.

- **Key Id**:
  The value entered is used to uniquely identify a specific private key from all others that may be stored on the HSM. On completion of the dialog, this private key is associated with the certificate that you are currently editing. Private keys are identified by their key Id by default.
- **Use Public Key**:
  Select this option if the HSM allows identifying a specific private key based on its associated public key, instead of using the private key Id. This option is not selected by default.
- **Conversation**:
  If the HSM requires the server to provide a specific response to a specific request from the HSM, you can enter the response in this field. This enables the server to conduct an automated dialog with a HSM when it requires access to a private key. For example, in a simple case, the server response might be a specific passphrase. For more details, the *API Gateway User Guide*.

Global options

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/or private key.

Manage certificates and keystores

On the main **Certificates** window, you can click the **Edit** button to edit an existing certificate. You can also click the **View** button to view the more detailed information on an existing certificate. Similarly, you can click the **Remove** button to remove a certificate from the certificate store.

Export certificates to a keystore

You can also export a certificate to a Java keystore. You can do this by clicking the **Keystore** button 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 the **Export to Keystore** button 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 the **Keystore** button on the main **Certificates** window. On the **Keystore** window, browse to the location of the keystore by clicking the 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 that you want to import, and click the **Import to Trusted certificate store** button. 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.

Configure key pairs

To configure public-private key pairs in the certificate store, select **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 the **Add** button 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 the **Load** buttons to select the public key and/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.

Manage OpenSSH keys

You can use the `ssh-keygen` command provided on UNIX 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`

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

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

Configure PGP key pairs

To configure Pretty Good Privacy (PGP) key pairs in the certificate store, select **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** button on the right, and configure the following settings in the dialog:

- **Alias**:
  Enter a unique name for the PGP key pair.
- **Load**:
  Click the **Load** buttons to select the public key and/or private key files to use.

Note

The PGP keys added must not be passphrase protected.

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 [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`

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

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

Manage API Gateway settings

Overview

You can configure the underlying settings for the 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** screen. Click or expand a tree node on this screen to configure the appropriate settings. You can confirm changes to these settings by clicking the **Apply Changes** button at the bottom right of each screen.

API Management settings

The **API Management** settings enable you to configure the API Management features that are available when the Axway API Manager product is installed with the 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 Management 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*.

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 User Guide*.

MIME/DIME

The 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. This also applies to Direct Internet Message Encapsulation (DIME), which is a streamlined version of MIME. The MIME/DIME settings list the default MIME/DIME types that the API Gateway can filter on. These types are then used by the **Content Types** filter to determine which MIME/DIME types to block or allow through to the back end Web service. For more details, see *MIME/DIME settings*.

Namespaces

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

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*.

Logging settings

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

Transaction Log

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

Access Log

The **Access Log** records a summary of all request and response messages that pass through the 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 *Access Log settings*.

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*.

Monitoring settings

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

Metrics

The **Metrics** settings enable you to configure statistics about the messages that the API Gateway instances in a database. The API Gateway Analytics monitoring tool can then poll this database, and produce charts and graphs showing how the API Gateway is performing. For more details, see *Real-time monitoring metrics*.

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*.

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 the 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 User Guide*.

Security Service Module

You can configure the API Gateway to act as an Oracle Security Service Module (SSM) to enable integration with Oracle Entitlements Server 10g. The 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 User Guide*.

Important

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 the 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 User Guide*.

Tivoli

You can configure how the 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 User Guide*.

Manage API Gateway deployments

Overview

When connected to the Admin Node Manager server in Policy Studio, you can deploy configurations to API Gateway instances running in groups in a domain. In Policy Studio, the **Group / API Gateway** topology view enables you to edit the configuration of currently running API Gateway instances. You can update the downloaded configuration, and deploy it to the server, where it can be reloaded later. You can deploy modified configuration to multiple API Gateway instances managed by Policy Studio. You can also create groups and API Gateway instances.

The web-based API Gateway Manager enables you to deploy configurations to API Gateway instances running in groups in a domain, to create groups and API Gateway instances, and to manage Admin 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.

Connect to a server in Policy Studio

Before starting Policy Studio, you should first ensure that the Admin Node Manager and the server instance that you wish to connect to have been started.

When Policy Studio starts up, click a link to a server to display the **Open Connection** dialog. You can use this dialog to specify **Connection Details** (for example, host, port, user name, and password) or to specify **Saved Sessions**. If you wish to connect to the server using a non-default URL, click **Advanced**, and enter the **URL**. The default Admin Node Manager URL is:

```
https://localhost:8090/api
```

Alternatively, you can connect to a server configuration file by clicking the **Open File** button. For more details on connecting using a server URL, configuration file, or deployment archive, see the *API Gateway User Guide*.

Note

You must connect to the Admin Node Manager server to deploy API Gateway configuration or manage multiple API Gateway instances in your network.

When the connection to the server has been made, the **Group / API Gateway** topology view is displayed. This displays the list of server instances currently managed by the Admin Node Manager in Policy Studio, and enables you to manage the configuration for server instances.

Edit a server configuration in Policy Studio

The **Group / API Gateway** topology view lists all available instances in each group. Double-click an instance name in the list to load its active configuration. Alternatively, right-click an instance name, and select **Edit Configuration**. The active server configuration is loaded and displayed in the following format: **InstanceName [HostName:Port]** (for example, **test_server [roadrunner.acme.com:8085]**).

When an active server configuration is loaded, its services are displayed under the **Listeners** node in the Policy Studio tree on the left. Expand one of the top-level nodes in the tree to display additional details (for example, **Business Services**, **External Connections**, **Resources**, **Libraries**, or **Settings**).

When editing an active server configuration, you can deploy updates using the **Deploy** button in the toolbar (alternatively, press **F6**). You can also deploy configuration packages in the **Group / API Gateway** topology view. For more details, see *Deploy API Gateway configuration*.

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 groups and API Gateway instances, and to deploy configuration. For details on how to access the API Gateway Manager, see *Start the API Gateway tools*.

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 User Guide*.

Manage Admin users in API Gateway Manager

You can add new Admin Users to enable role-based access to the API Gateway configuration managed by Policy Studio and API Gateway Manager. The default `admin` 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 Admin Users, click the **Settings** > **Admin Users** tab in the API Gateway Manager. For more details, see *Manage Admin users*.

For more details on role-based access, see *Configure Role-Based Access Control (RBAC)*.

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 User Guide*.

Deploy API Gateway configuration

Overview

You can edit API Gateway configuration offline, and then deploy later to a specified API Gateway instance using API Gateway configuration packages. 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. While 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 Deployment and Promotion Guide*.

You can use Policy Studio to create packages (`.fed`, `.pol`, or `.env`). You can also use Policy Studio to deploy an existing package or factory configuration on selected API Gateway instances, or to deploy a currently loaded configuration. You can use the API Gateway Manager console to deploy a package in a browser. Alternatively, you can use the `managedomain` script to create and deploy deployment packages (`.fed` files) on the command line.

Create a package in Policy Studio

You can create an API Gateway configuration package for a currently loaded configuration, or for a selected server instance in the **Group / API Gateway** topology view.

Currently loaded configuration
To create a package (`.fed`, `.pol`, or `.env`) for a currently loaded API Gateway configuration, perform the following steps:

1. In the main menu, select **File** > **Save** followed by the appropriate option:
   - **Deployment Package**
   - **Policy Package**
   - **Environment Package**
2. Enter a filename, and click **Save**.

Group / API Gateway view
To create a deployment package (`.fed`) in the API Gateway **Group / API Gateway** topology view, perform the following steps:

1. Right-click a server instance in the tree, and select **Save Deployment Package**.
2. Browse to a directory in the dialog.
3. Click **OK**. The file is saved to the specified directory (for example, `c:\temp\5c3b2a3c-23a5-4261-87cb-eca150f0a037.fed`.
4. Click **OK**.

Configure package properties in Policy Studio

You can view or modify API Gateway configuration package properties for a currently loaded configuration, or for a selected server instance in the **Group / API Gateway** topology view.

Currently loaded configuration
To view and modify configuration properties for a currently loaded API Gateway configuration, perform the following steps:

1. In the Policy Studio tree, and select **Package Properties** > **Policy** or **Environment**.
2. Enter values for the appropriate configuration properties (for example, **Name**, **Description**, or **Version**).
3. 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`).
4. If you wish to remove a property, click the red (x) button on the right of the property.
5. Click **Save** on the top right of the screen.

Group / API Gateway view
To view and modify configuration properties for a selected server instance in the API Gateway **Group / API Gateway** topology view, perform the following steps:

1. Right-click a server in the tree, and select **View/Modify Properties**.
2. Select the **Policy Properties** or **Environment Properties** tab.
3. Enter values for the appropriate configuration properties (**Name**, **Description**, and **Version**).
4. 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`).
5. If you wish to remove a property, click the red (x) button on the right of the property.
6. Click **Update Configuration Properties**.
7. Click **OK**.

For details on customizing the default package properties displayed, see the **Topology Screen** settings in the *API Gateway User Guide*.

Deploy packages in Policy Studio

You can use the Policy Studio to deploy configuration packages to selected API Gateway instances in the **Group / API Gateway** topology view.

Deploy a deployment package
To deploy an existing deployment package (`.fed` file) in the **Group / API Gateway** view, perform the following steps:

1. Click the **Deploy** button in the toolbar.
2. In the **Select the servers(s) you wish to deploy to** section, select a server group from the **Group** list, and select the server instance(s) in the box below.
3. In the **Select the configuration you wish to deploy** section, select **I wish to deploy configuration contained in a single Deployment Package**.
4. In the **Deployment Package** field, click **Browse for .fed**, and select the `.fed` file.
5. Click **Deploy** to upload the package to the Admin Node Manager and deploy to the selected server(s).
6. When the package has deployed, click **Finish**.

Deploy policy and environment packages

To deploy an existing policy package (`.pol` file) and environment package (`.env` file) in the **Group / API Gateway** view, perform the following steps:

1. Click the **Deploy** button in the toolbar.
2. In the **Select the servers(s) you wish to deploy to** section, select a server group from the **Group** list, and select the server instance(s) in the box below.
3. In the **Select the configuration you wish to deploy** section, select **I wish to deploy configuration contained in Policy Package and Environment Package**.
4. In the **Policy Package** field, click **Browse for .pol**, and select the `.pol` file.
5. In the **Environment Package** field, click **Browse for .env**, and select the `.env` file.
6. Click **Deploy** to upload these packages to the Admin Node Manager and deploy to the selected server(s).
7. When the package has deployed, click **Finish**.

Deploy a factory configuration in Policy Studio

To deploy a default factory configuration in the **Group / API Gateway** view, perform the following steps:

1. Click the **Deploy** button in the toolbar.
2. In the **Select the servers(s) you wish to deploy to** section, select a server group from the **Group** list, and select the server instance(s) in the box below.
3. In the **Select the configuration you wish to deploy** section, select **I wish to deploy a factory configuration**.
4. Click **Deploy** to deploy the configuration to the selected server(s).

Deploy currently loaded configuration in Policy Studio

You can also deploy updates to a currently loaded configuration in Policy Studio when editing the configuration. To deploy a currently loaded configuration, perform the following steps:

1. Click the **Deploy** button on the right in the toolbar.
2. In the **Select the servers(s) you wish to deploy to** section, select a server group from the **Group** list, and select the server instance(s) in the box below.
3. Click **Deploy**, and wait for the deployment to complete.
4. Click **Finish**.

Push configuration to a group in Policy Studio

When there is more than one API Gateway instance in a group, and configuration becomes out of sync between instances, you can select which configuration to push to the group. Perform the following steps in Policy Studio:

1. In the **Group / API Gateway** topology view, right-click the API Gateway instance configuration that you want to deploy to other instances in the group.
2. Select **Push this API Gateway's configuration to the group**.
3. In the wizard, select the API Gateway instances in the group that you wish to deploy to.
4. Click **Deploy** to deploy to the selected instances in the group.
5. Click **Finish**.

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 cancelled.

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.

Deploy on the command line

You can create and deploy a deployment package (`.fed`) using the `managedomain` script in the following directory:

Windows	`INSTALL_DIR\Win32\bin`
UNIX/Linux	`INSTALL_DIR/posix/bin`

The deployment options in the `managedomain` script 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
```

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*.

Configure API Gateway tracing

Overview

By default, the API Gateway outputs tracing and debugging information to record information about its execution. For example, this includes details such as services starting or stopping, and messages sent through the API Gateway. This information then can 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 location:
  - Admin Node Manager: `<install-dir>/trace`
  - API Gateway instance: `<install-dir>/groups/<group-id>/<instance-id>/trace`
  - API Gateway Analytics: `<install-dir>/trace`
- Windows or UNIX console window for the running server

The **Logs** view in API Gateway Manager enables you to view and search the contents of the API Gateway trace log, transaction log, access log, and domain audit log. This topic explains how to configure the trace log only. For more details, see *Configure API Gateway logging and events*.

View API Gateway trace files

Each time the API Gateway starts up, by default, it outputs a trace file to the API Gateway `trace` directory (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`

where `FATAL` is the least verbose and `DATA` is the most verbose. The default trace level is `INFO`.

Set trace levels

You can set the trace level using the following different approaches:

Startup trace	When the Admin Node Manager is starting up, it gets its trace level from the `tracelevel` attribute of the `SystemSettings` element in `/system/conf/nodemanager.xml`. You can set the trace level in this file if you need to diagnose boot up issues.
Default Settings trace	When the 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, select a **Trace level** from the drop-down list.
Interface level trace	You can configure an HTTP/HTTPS interface with a different trace level to that specified in the Default Settings. For example, the default API Gateway management port (`8090`) has a trace level set to `ERROR` to ensure it is not too verbose. To configure the trace level for an interface in the Policy Studio, right-click the interface under the **Listeners** node, select **Edit**, and select a **Trace 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 the section called "Configure trace and log settings".

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.xml`. By default, `trace.xml` contains the following setting:

```
<FileRolloverTrace maxfiles="500" filename="%s_%Y%m%d%H%M%S.trc"/>
```

This setting means that the API Gateway writes Node Manager trace output to `nodemanageron`*`hostname_timestamp`*`.trc` (for example, `nodemanageron127.0.0.1_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`.

FileRolloverTrace Attributes

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.
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 the maximum integer value (2147483647).
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 the 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.

Writing Trace Output to Syslog
On UNIX and 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, the API Gateway outputs the status of every policy and filter that it processes into the trace file.

Debugging a Filter
The trace output for a specific filter takes the following format:

```
Filter name {
    Trace for the filter is indented
    to the following point to make it clear
    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 is 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`, the API Gateway outputs 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 the section called “Set API Gateway trace levels”.

Run trace at DATA level

When the trace level is set to `DATA`, the API Gateway writes the contents of the messages that it receives and sends to the trace file. This enables you to see what messages the API Gateway has received and sent (for example, to reassemble a received or sent message).

Note

On Windows, you can not rely on the console output because it truncates large messages.

Searching by thread

Every HTTP request handled by the 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 the API Gateway by following the trace by thread ID. Because multiple messages can be processed by the 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** textbox, 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 the 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 the 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 the 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 the API Gateway classpath. This is because some third-party products that the API Gateway interoperates with require log4j. The configuration for log4j is found in the API Gateway `INSTALL_DIR/system/lib` directory in the `log4j.properties` file.

For example, to specify that the log4j appender sends output to the API Gateway trace file, add the following setting to your `log4j.properties` file:

```
log4j.rootLogger=DEBUG, A1, Vordel
log4j.appender.Vordel=com.vordel.trace.VordelTraceAppender
```

Get help with API Gateway

Context-sensitive help is available from the Policy Studio screens. Click the **Help** button on any screen to display the relevant help page for that screen. If you require further information or assistance, please contact the Axway Support team.

Details for 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, Axway API Gateway 7.3.0).
- Details of patches that were applied to the product, if any.
- Platform 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`.

Configure API Gateway logging and events

Overview

You can use the **Logs** and **Events** view in the API Gateway Manager web console to view and search the API Gateway log and event files. The **Logs** view enables you to configure and manage following API Gateway logging options:

- **Domain Audit**: Displays management changes at the API Gateway domain level (for example, updates to API Gateway configuration, topology, login, or deployment). The domain audit log is configured by default.
- **Trace Log**: Records detailed diagnostic and debugging information on API Gateway instance execution (for example, services starting or stopping, and messages sent through the API Gateway). The trace log is configured by default. You should submit trace log files when raising issues with Axway Support.
- **Transaction Log**: Records the message transaction log entries generated by each API Gateway filter as the message passes through the filter (for example, success, failure, or abort). You can configure different log output destinations.
- **Access Log**: Provides a summary of the HTTP request and response messages in Apache HTTP Server format. You can configure the access log per path.

This topic explains how to view and configure the domain audit log, transaction log and events, and access log. For details on configuring the trace log, see *Configure API Gateway tracing*.

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, logins, 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. However, you can configure filtering options such as the number of events displayed, time interval, and event type.

For example, the displayed event types include the following:

- Configuration
- Service
- Application
- Topology
- User store
- Key Property Store (KPS)
- Organization
- Portal admin

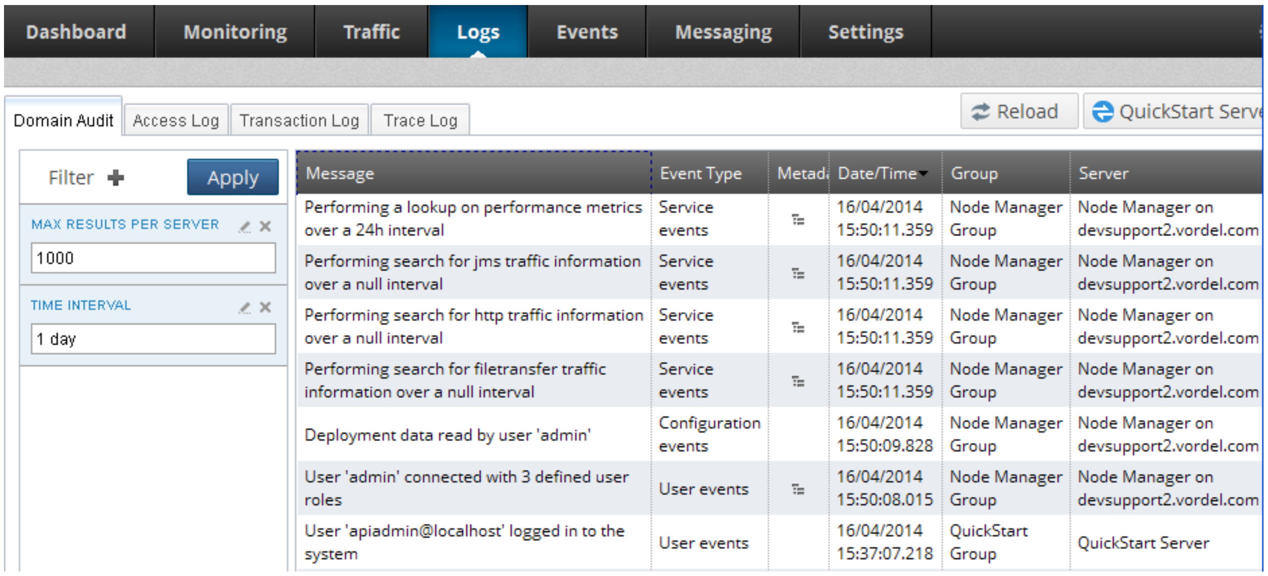
View in API Gateway Manager

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 the **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.



MAX RESULTS PER SERVER: 1000

TIME INTERVAL: 1 day

Message	Event Type	Metad	Date/Time	Group	Server
Performing a lookup on performance metrics over a 24h interval	Service events		16/04/2014 15:50:11.359	Node Manager Group	Node Manager on devsupport2.vordel.com
Performing search for jms traffic information over a null interval	Service events		16/04/2014 15:50:11.359	Node Manager Group	Node Manager on devsupport2.vordel.com
Performing search for http traffic information over a null interval	Service events		16/04/2014 15:50:11.359	Node Manager Group	Node Manager on devsupport2.vordel.com
Performing search for filetransfer traffic information over a null interval	Service events		16/04/2014 15:50:11.359	Node Manager Group	Node Manager on devsupport2.vordel.com
Deployment data read by user 'admin'	Configuration events		16/04/2014 15:50:09.828	Node Manager Group	Node Manager on devsupport2.vordel.com
User 'admin' connected with 3 defined user roles	User events		16/04/2014 15:50:08.015	Node Manager Group	Node Manager on devsupport2.vordel.com
User 'apiadmin@localhost' logged in to the system	User events		16/04/2014 15:37:07.218	QuickStart Group	QuickStart Server

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 1 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.

Customize domain audit log output

You can also customize and filter 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.
`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 logs per API Gateway

The API Gateway provides detailed transaction logging for specific message filters (for example, the request, the time of the request, where the request was routed to, and the response returned to the client). You can configure transaction logging to a number of different locations:

- Text file
- XML file
- Database
- Local syslog
- Remote syslog
- System console

Transaction logging is not configured by default. To configure where transaction logging information is sent, perform the following steps:

1. In the Policy Studio tree, select the **Server Settings** > **Logging** > **Transaction Log**.
2. Specify the required settings on the appropriate tab (for example, **Text File**, **Database**, or **XML File**).
3. When finished, click **Apply Changes** at the bottom right.
4. Click the **Deploy** button in the toolbar to deploy your settings to the API Gateway.

For details on configuring all the available options, see the topic on *Transaction Log settings*.

Configure transaction logs per filter

You can also configure the transaction 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 Log Level and Message** screen.
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.

For more details on transaction logging for specific message filters, see the *API Gateway User Guide*.

Configure 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 the **Server Settings** > **Logging** > **Access Log**.
2. Specify the required settings (for example, remote hostname, user login name, and authenticated user name).
3. When finished, click **Apply Changes** at the bottom right.
4. Click the **Deploy** button in the toolbar to deploy your settings to the API Gateway.

For details on configuring all the available options, see *Access Log settings*.

Manage API Gateway events

The **Events** view in API Gateway Manager enables you to view and search the contents of the following API Gateway events:

- **Transaction**: When transaction logging is configured per filter, this displays an in-memory list of transaction log events. A transaction log destination does not need to be enabled for the in-memory list of events to be updated. For more details, see the section called “Configure transaction logs per filter”
- **Alerts**: When **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.
- **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 details on how to configure **Alert** filters and **SLA Alert** filters in your policies, see the *API Gateway User Guide*.

API Gateway performance tuning

Overview

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, client threads, and Java memory.

Note

This topic applies to performance tuning for both API Gateway and API Gateway Analytics.

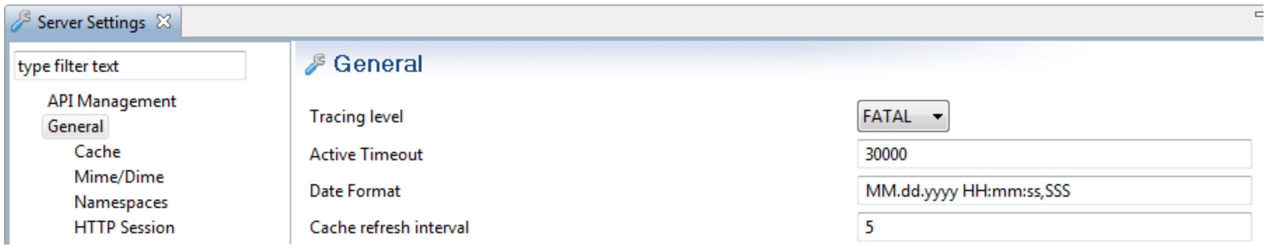
General performance tuning

You can optimize API Gateway performance by using Policy Studio to configure the general settings described in this section.

Minimize tracing

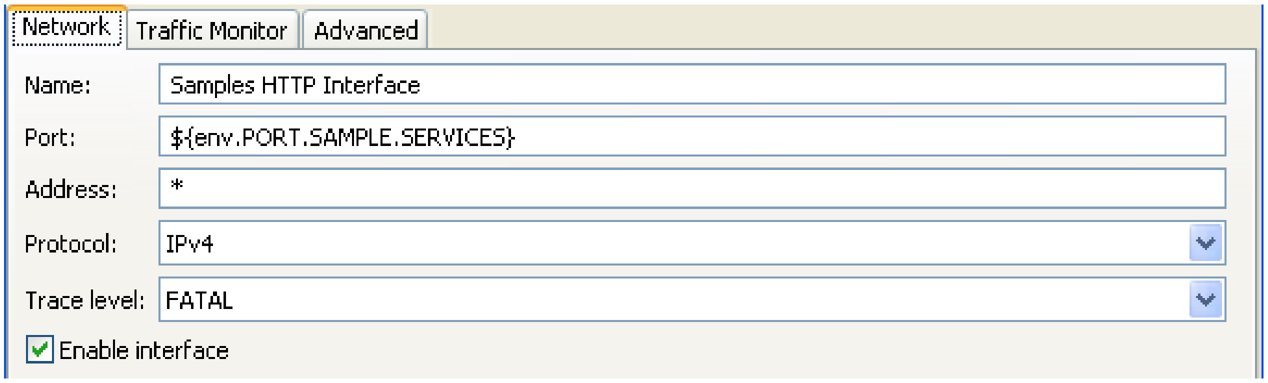
The **Trace Log** is displayed in the **Logs** view in the API Gateway Manager web console. When tracing is running at a verbose level (for example, DEBUG), this means that the API Gateway is doing more work and is very dependent on disk input/output. You can set a less verbose trace level for an API Gateway instance or API Gateway port interface (for example, ERROR or FATAL).

To set the tracing for an API Gateway instance, select **Server Settings** > **General** in the Policy Studio tree, and select the **Trace Level** (for example, FATAL):



Tracing level	FATAL
Active Timeout	30000
Date Format	MM.dd.yyyy HH:mm:ss,SSS
Cache refresh interval	5

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 **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):



Name: Samples HTTP Interface

Port: ${env.PORT.SAMPLE.SERVICES}

Address: *

Protocol: IPv4

Trace level: FATAL

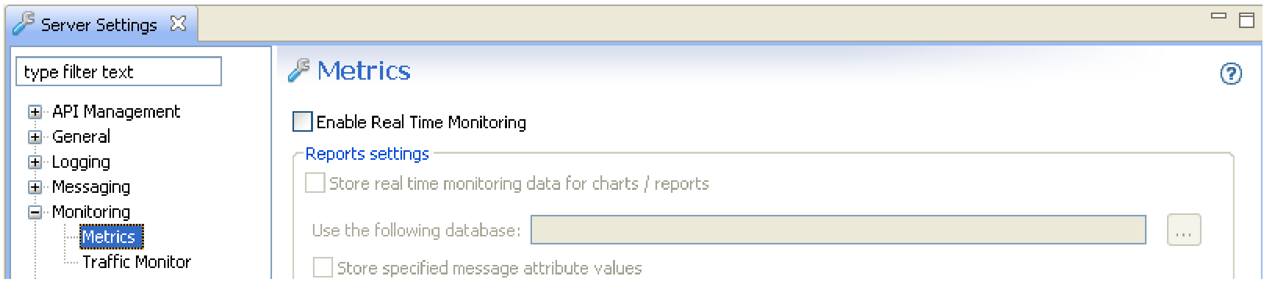
[✓] Enable interface

For more details, see *Configure API Gateway tracing*

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 the API Gateway. You can remove this overhead by disabling real-time monitoring.

To disable in Policy Studio, select **Server Settings** > **Monitoring** > **Metrics**, and deselect **Enable Real-Time Monitoring**:



[ ] Enable Real Time Monitoring

Reports settings

[ ] Store real time monitoring data for charts / reports

Use the following database:

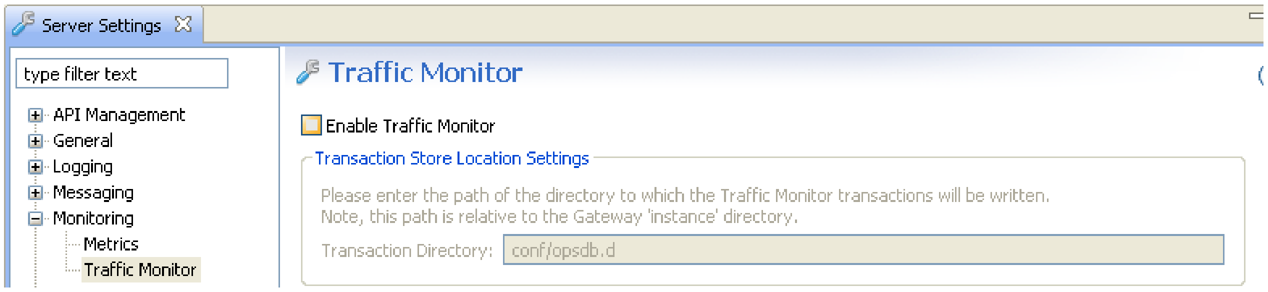
[ ] Store specified message attribute values

For more details, see *Real-time monitoring metrics*

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 **Server Settings** > **Monitoring** > **Traffic**, and deselect **Enable Traffic Monitor**:



[ ] Enable Traffic Monitor

Transaction Store Location Settings

Please enter the path of the directory to which the Traffic Monitor transactions will be written.
Note, this path is relative to the Gateway 'instance' directory.

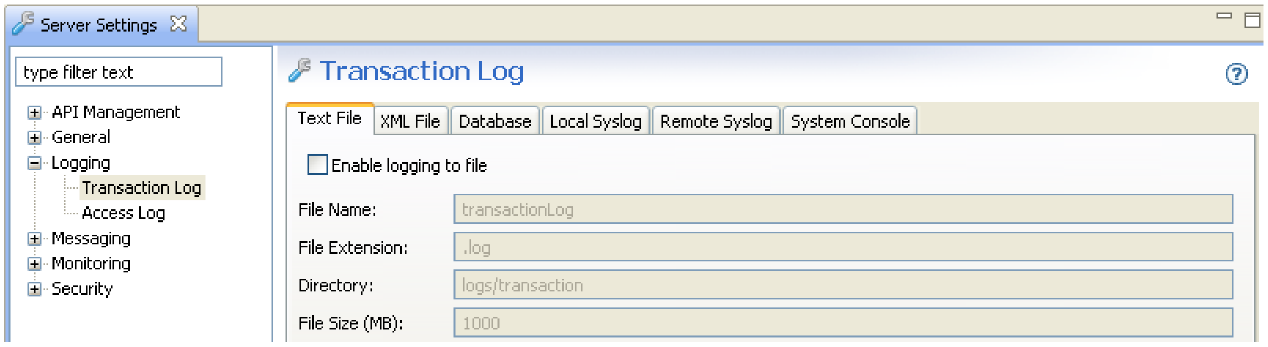
Transaction Directory: conf/opsdb.d

For more details, see *Traffic monitoring settings*

Disable transaction logging

The **Transaction Log** is displayed in the **Logs** view in the API Gateway Manager web console. You should ensure that the API Gateway is not sending transaction log messages or events to transaction log destinations. This is because the performance of the 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 **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:



Enable logging to file [ ]

File Name: transactionLog

File Extension: .log

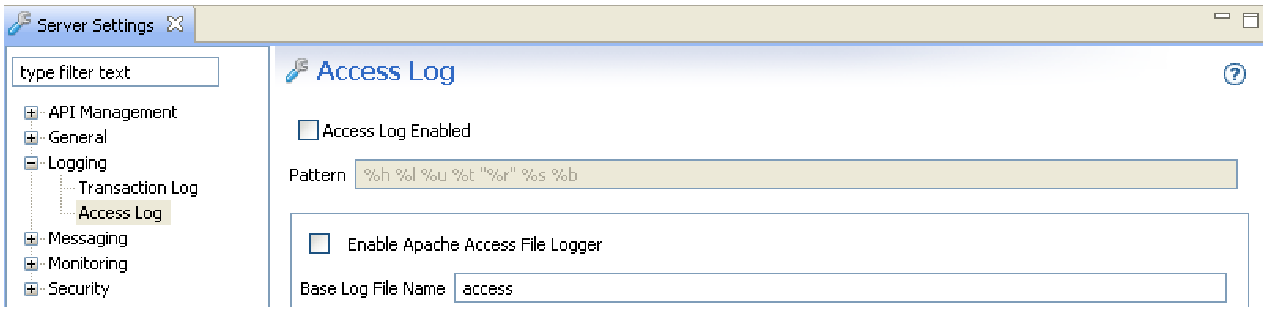
Directory: logs/transaction

File Size (MB): 1000

For more details, see *Transaction Log settings*

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 **Server Settings** > **Logging** > **Access Log**, and deselect **Access Log Enabled**:



Access Log Enabled [ ]

Pattern %h %l %u %t "%r" %s %b

Enable Apache Access File Logger [ ]

Base Log File Name access

For more details, see *Access Log settings*

Advanced performance tuning

You can also use the advanced configuration settings described in this section to optimize API Gateway performance.

Configure spill 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 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.

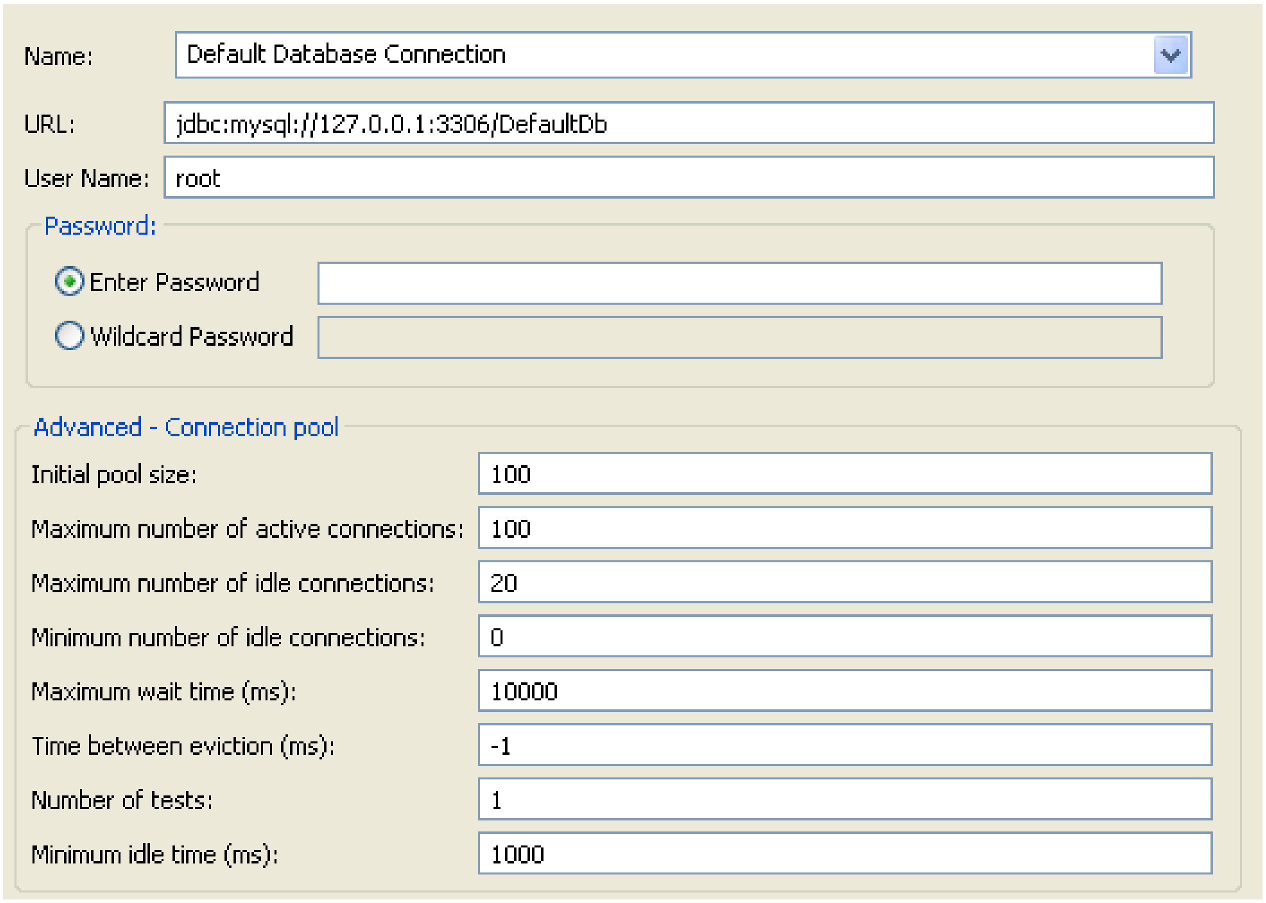
Configure database pooling

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 **External Connections** > **Database Connections** > **Add Database Connection**. For example:



Name:	Default Database Connection
URL:	jdbc:mysql://127.0.0.1:3306/DefaultDb
User Name:	root

Password:

- [✓ Enter Password]
- Wildcard Password

Advanced - Connection pool

Initial pool size:	100
Maximum number of active connections:	100
Maximum number of idle connections:	20
Minimum number of idle connections:	0
Maximum wait time (ms):	10000
Time between eviction (ms):	-1
Number of tests:	1
Minimum idle time (ms):	1000

Configure HTTP keep alive

In HTTP/1.1, by default, the connection between a client and a server is maintained 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 keepalive mechanism.

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 User Guide*.

Note

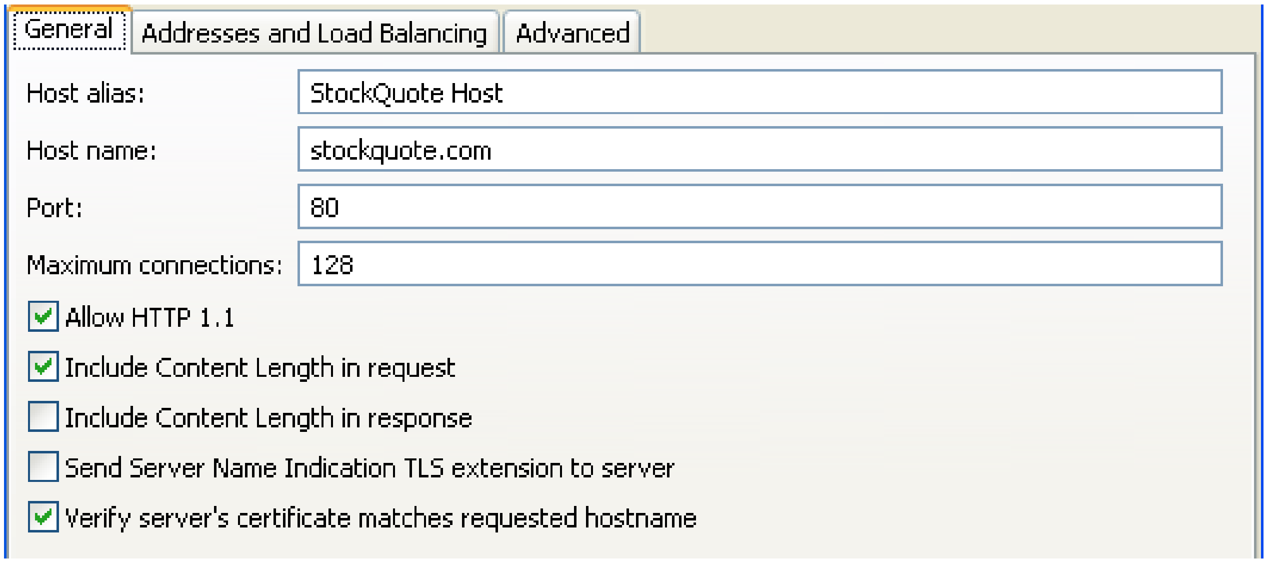
For conformance with the HTTP/1.1 specification, the client must send a `Host` header in this configuration, so you must pass a further `-aHost:localhost` argument to `sr`. If the connection keep alive is working correctly, `sr` reports a larger number of transactions to connections in its periodic output.

Configure chunked encoding

For interoperability reasons, the API Gateway normally does not use chunked encoding when talking to a remote server. Because of the HTTP protocol, the 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 the 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 **Listeners** > **API Gateway** > **Sample Services**. Right-click the remote host, select **Edit**, and select **Allow HTTP 1.1**:



Host alias: StockQuote Host
Host name: stockquote.com
Port: 80
Maximum connections: 128
[✓] Allow HTTP 1.1
[✓] Include Content Length in request
[ ] Include Content Length in response
[ ] Send Server Name Indication TLS extension to server
[✓] Verify server's certificate matches requested hostname

Single test client and sever

If a performance test uses one client and one target service, it is recommended that you use the following settings:

```
sysctl net.ipv4.tcp_tw_recycle=1
sysctl net.ipv4.tcp_tw_reuse=1
```

A TCP/IP connection is identified by the following properties:

- Remote IP address
- Local IP address
- Remote port
- Local port

When you have a fixed client IP address, a fixed server IP address, and a fixed server port, this leaves 65536 client ports, which means you can only have 65536 live connections at a time.

In a TCP/IP connection, one side always ends up in the `TIME_WAIT` state, which is part of the protocol. This period can last for two minutes by default. In a naive implementation, this means you can establish at most 65536 connections every 4 minutes. There are ways to mitigate this in the protocol, and most TCP stacks can reuse connections in the `TIME_WAIT` state more rapidly than this. But under certain circumstances, this can still be a problem. The `sysctl` settings shown above make the Linux kernel more aggressive in the way it recycles `TIME_WAIT` connections.

JVM memory

If processing many large XML documents, you should reduce the Java Virtual Machine (JVM) memory so that garbage collection is performed more often. Create a file called `jvm.xml` in the `system/conf` directory of the API Gateway installation with the following contents:

```
<ConfigurationFragment>
    <VMArg name="-Xmx100m"/>
    <VMArg name="-Xms100m"/>
    <VMArg name="-Xincgc"/>
    <Print message="JVM heap limited"/>
</ConfigurationFragment>
```

The amount of memory used by the server is limited in several ways, and one of these limits is the normal Java *heap space*. The API Gateway can use memory that is not visible to the JVM, but is *pinned* by it, so it is not released until the JVM decides to collect its garbage. If the amount of memory pinned is much larger than the size of the objects that the JVM sees (the bits pinning it down), the Java heap setting does not represent the amount of heap memory used. So the problem is worked around by reducing the Java heap space used in this scenario.

For example, by setting the Java heap size to 100 MB instead of 1024 MB, the Java garbage collector runs when the Java heap use is close to 100 MB, while the amount of memory physically pinned by the JVM may be closer to 1000 MB. Without this adjustment, the memory usage would grow beyond what is possible for a 32-bit process.

In addition, you may wish to consider reducing the number of threads that the API Gateway uses for processing incoming messages. You can do this by adding a `maxThreads="64"` attribute in the `SystemSettings` element in `system/conf/service.xml`. This setting also helps in configuring the API Gateway to back off from the target service (if the API Gateway can process more load than the target service).

Number of client threads on Linux

If there are more than 200 client threads connecting, you must change the following line in the `venv` script in the `posix/bin` directory of your API Gateway installation:

```
ulimit -n unlimited ...
```

to the following:

```
ulimit -n 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.

Manage API Gateway users

Overview

The **Users and Groups** node in the Policy Studio tree enables you to manage API Gateway users and groups, which are stored in the API Gateway user store.

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 *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. For more details, see *Manage Admin users*.

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 **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.

Add user details
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.
- **X.509 Cert**:
  Click the **X.509 Cert** button to load the user's certificate from the **Certificate Store**.

Add user attributes
You can 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).

Add attributes
The **Attributes** tab enables you to configure user attributes as simple name-value pairs. The following are examples of user attributes:

- `role=admin`
- `email=niall@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 **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.

Add group details
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.

Add group attributes
You can specify optional attributes at the group level on the **Attributes** tab. For more details, see the section called “API Gateway user attributes”.

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

Overview

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*.

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.

Important

An Admin user *cannot* delete itself.

Remove the default Admin user
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`

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`	KPS Web UI	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 the 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/or `API Server Administrator`).
7. Click **Create**.

Remove an Admin user

To remove an Admin user, select it in the **Username** list, and click the **Delete** button. 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 roles
To add or delete specific roles, you must edit the available roles in the `adminUsers.json` and `acl.json` files in the `conf` directory of your API Gateway installation.

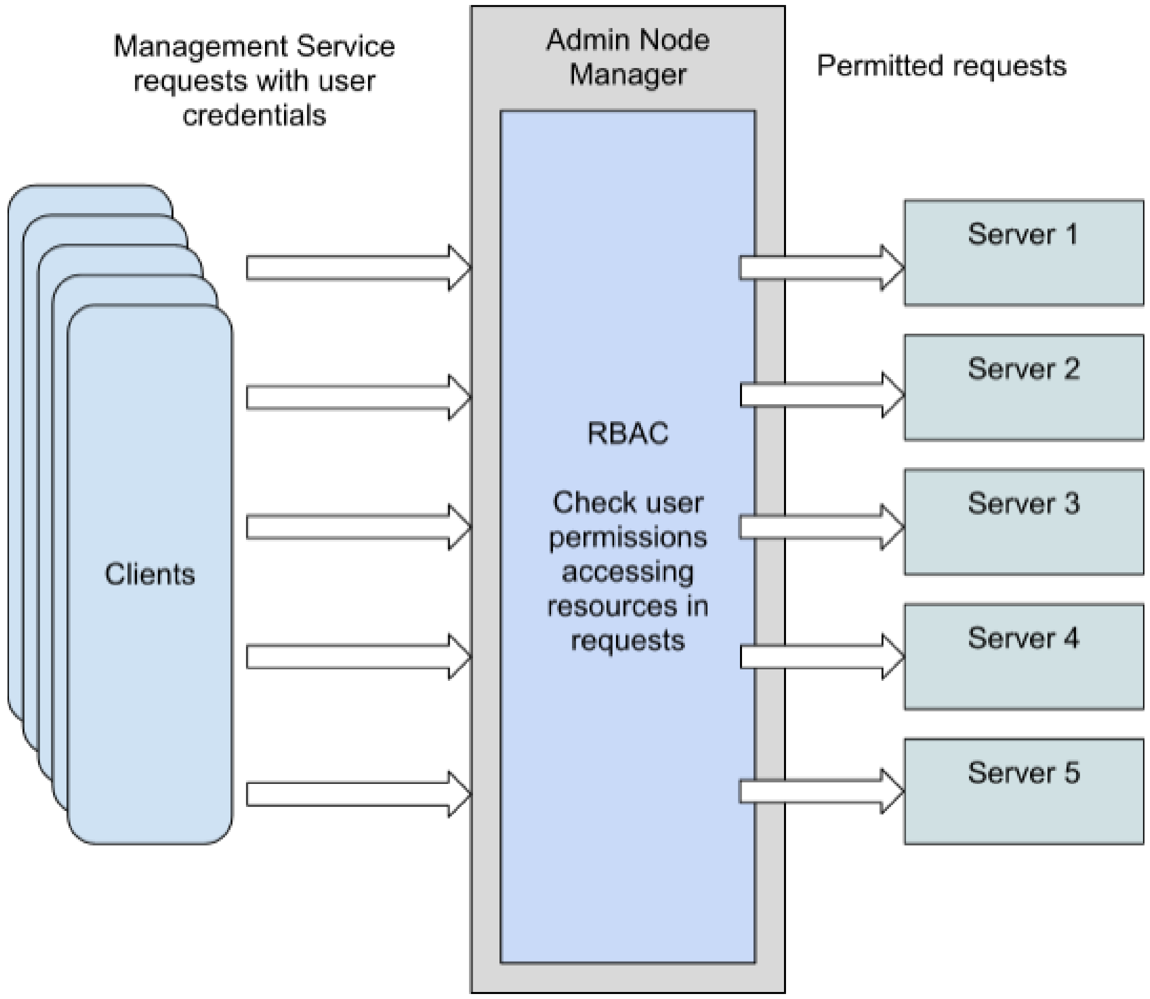
For more details on role-based access, see *Configure Role-Based Access Control (RBAC)*. For more details on role-based access, see the *API Gateway Administrator Guide*.

Configure Role-Based Access Control (RBAC)

Overview

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:


Management Service requests with user credentials
Admin Node Manager
Permitted requests
Clients
RBAC
Check user permissions accessing resources in requests
Server 1
Server 2
Server 3
Server 4
Server 5

Note

This topic applies to both API Gateway and API Gateway Analytics.

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 the topic on *Manage Admin users*.

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 the section called “Management service roles and permissions”.

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" : "7.3.0",
  "version" : 1,
  "timestamp" : 0,
  "adminUsers" : [ {
    "id" : "user-1",
    "name" : "admin",
    "roles" : [ "role-1", "role-6", "role-7" ]
  } ],
  "credentials" : {
    "user-1" : "$ygvfNZsiLEcQJGMe8vqmqw==$2jcZcvFctadAISBuvhJqt9NfUtk7zeoOHo8cFMySjvw="
  },
  "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).

For details on configuring an LDAP repository, see the following topics:

- *Active Directory for authentication and RBAC of management services*
- *OpenLDAP for authentication and RBAC of management services*

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 the table in the section called "Management service roles and permissions".

Important

The roles defined in the `acl.json` file should exist in the user store used to authenticate the users and load their roles and/or 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 the table in the section called "Management service roles and permissions".
- 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" : {
```

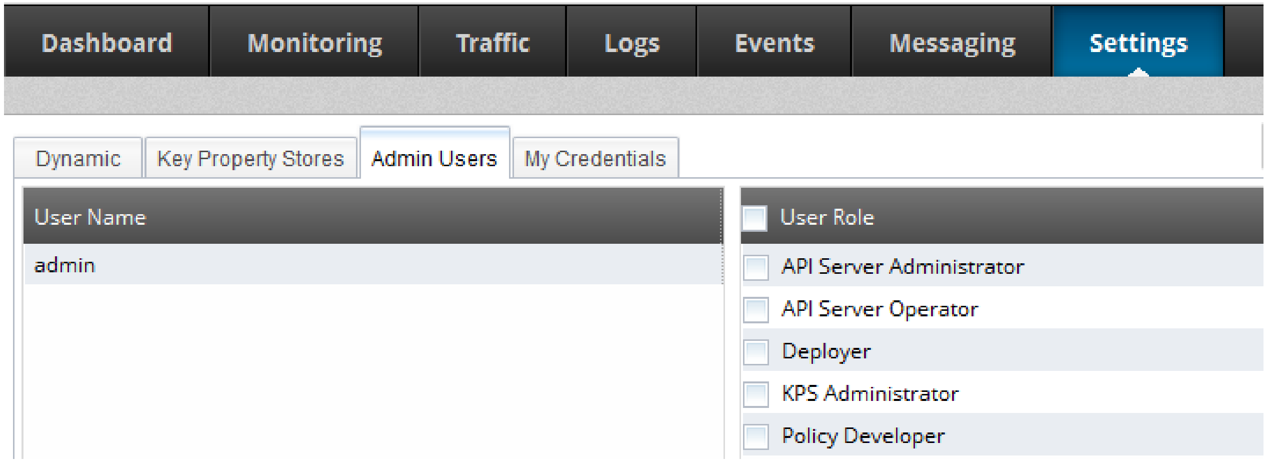
```
        "methods" : [ "GET" ],
        "paths" : [ "root", "site images" ]
        },
        ....
},

"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:



User Name
admin

	User Role
☐	API Server Administrator
☐	API Server Operator
☐	Deployer
☐	KPS Administrator
☐	Policy Developer

Manage RBAC user roles
When you click **Create** to create a new user, you can select the roles to assign to 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.

Important

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 the topic on *Manage Admin users*.

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`
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	• `emc`

Management Service	**Default Roles**	**Permissions**
	• API Server Operator	• `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`

Active Directory for authentication and RBAC of management services

Overview

This topic explains how to use Local Directory Access Protocol (LDAP) to authenticate and perform Role-Based Access Control (RBAC) of management services. You can use the sample **Protect Management Interfaces (LDAP)** policy instead of the **Protect Management Interfaces** policy. This means that an LDAP repository is used instead of the local Admin User store for authentication and RBAC of users attempting to access management services. This topic describes how to configure the server to use an example Microsoft Active Directory LDAP repository.

Note

This topic applies to both API Gateway and API Gateway Analytics.

Step 1: create an Active Directory group

To create a new user group in Active Directory, perform the following example steps:

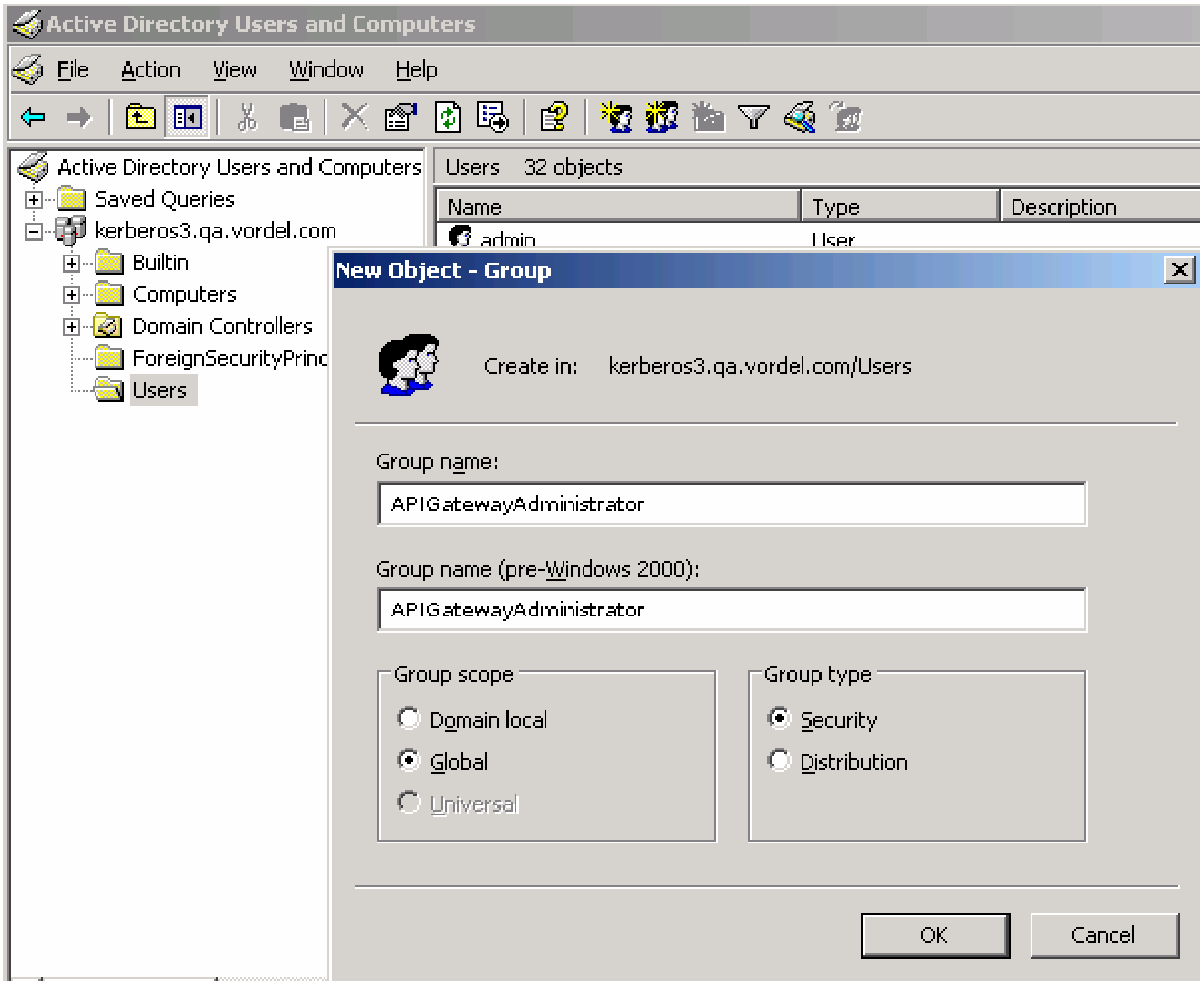
1. Click **Start** > **Administrative Tools** > **Active Directory Users and Computers**.
2. On the **Users** directory, right-click, and select **New** > **Group**.
3. Enter the Group name (for example, `APIGatewayAdministrator`).

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
```



Create in: kerberos3.qa.vordel.com/Users

Group name: APIGatewayAdministrator

Group name (pre-Windows 2000): APIGatewayAdministrator

Group scope: ( ) Domain local [✓ Global] ( ) Universal

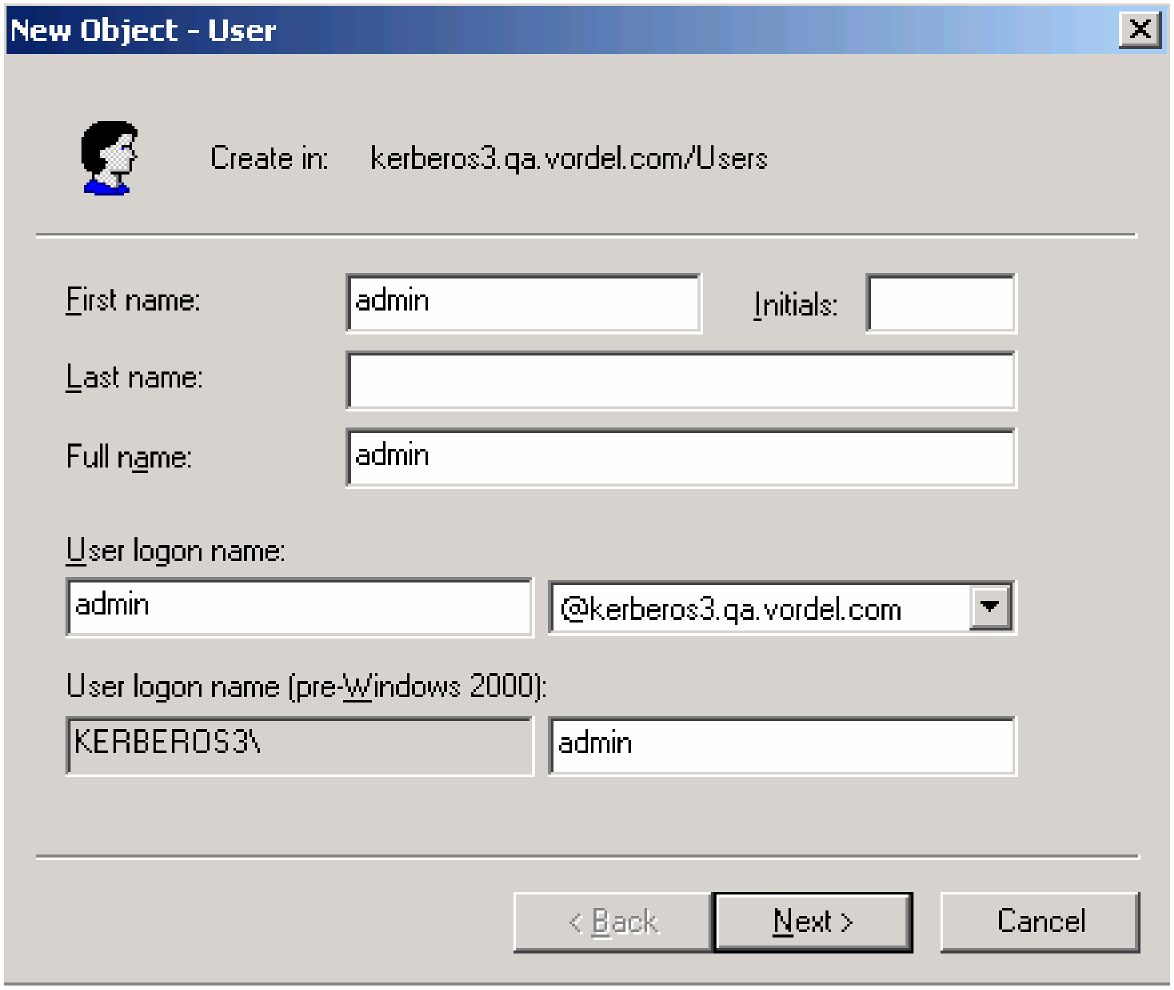
Group type: [✓ Security] ( ) Distribution

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 password of `changeme` because this 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`).



Create in: kerberos3.qa.vordel.com/Users

First name: admin

Initials:

Last name:

Full name: admin

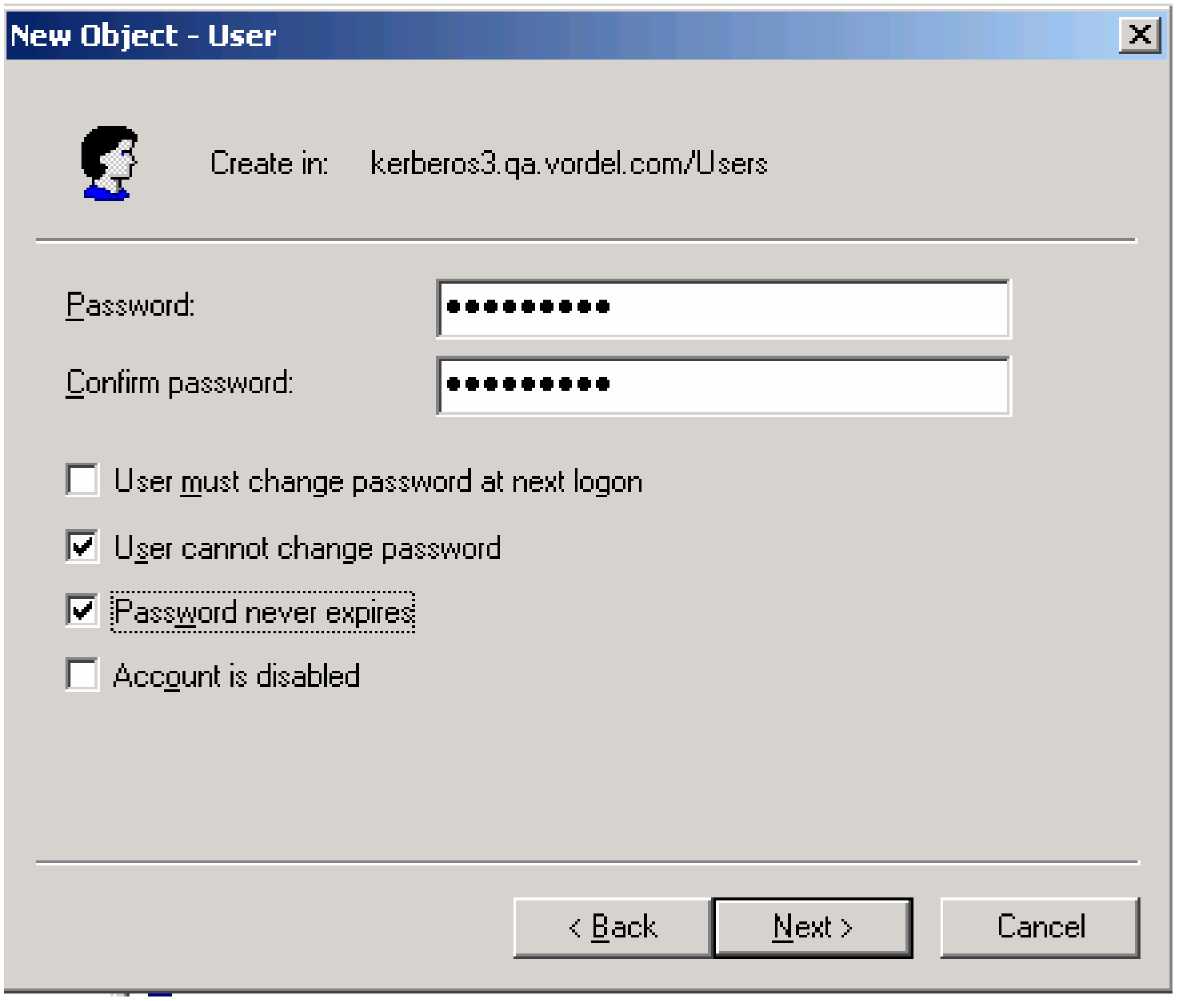
User logon name:

admin @kerberos3.qa.vordel.com

User logon name (pre-Windows 2000):

KERBEROS3\ 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**.


New Object - User

Create in: kerberos3.qa.vordel.com/Users

Password: ●●●●●●●●●●

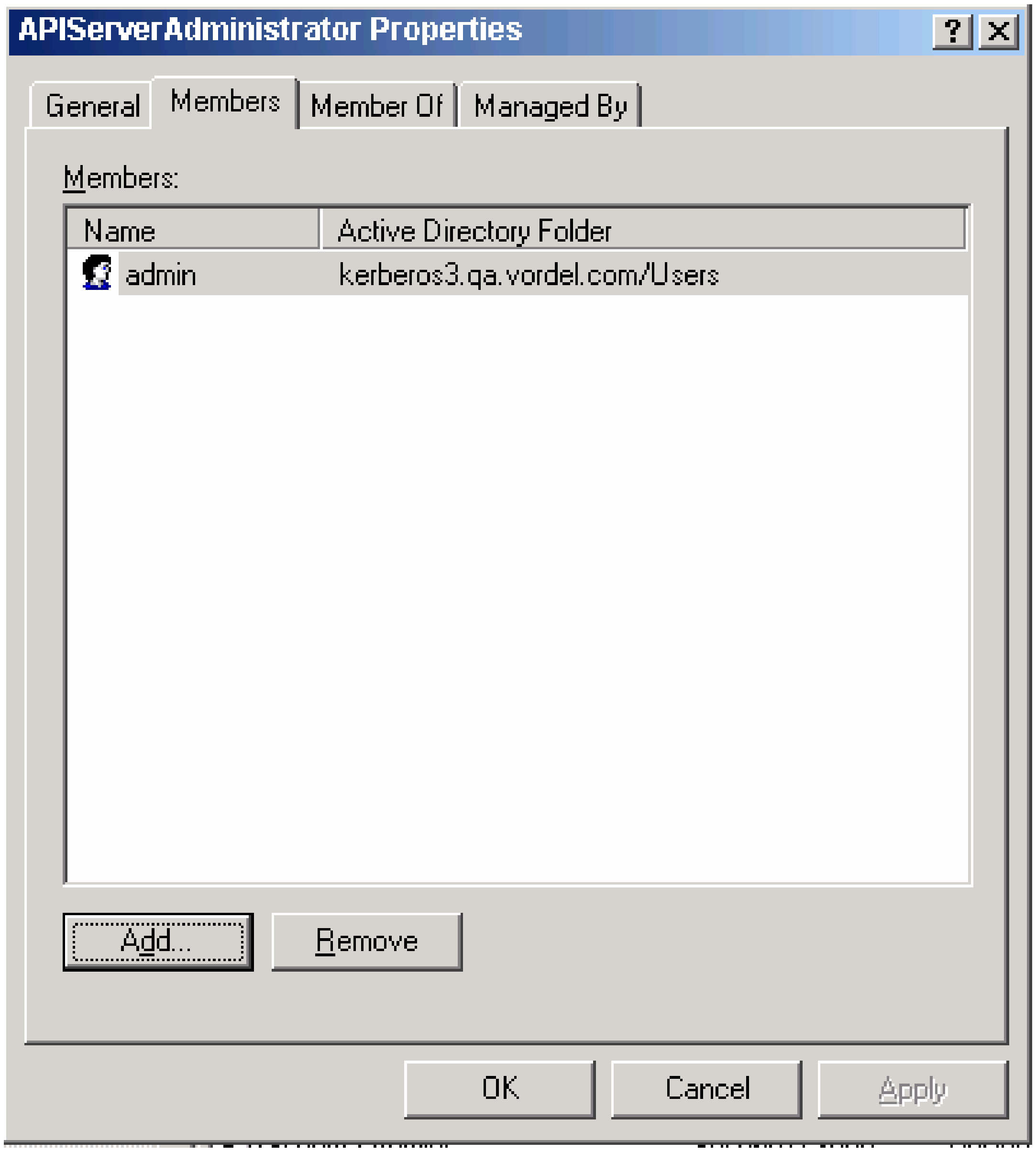
Confirm password: ●●●●●●●●●●

- [ ] User must change password at next logon
- [✓] User cannot change password
- [✓] Password never expires
- [ ] Account is disabled

Adding 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**.



Name	Active Directory Folder
admin	kerberos3.qa.vordel.com/Users

You can view the newly created user using an LDAP Browser.

Note

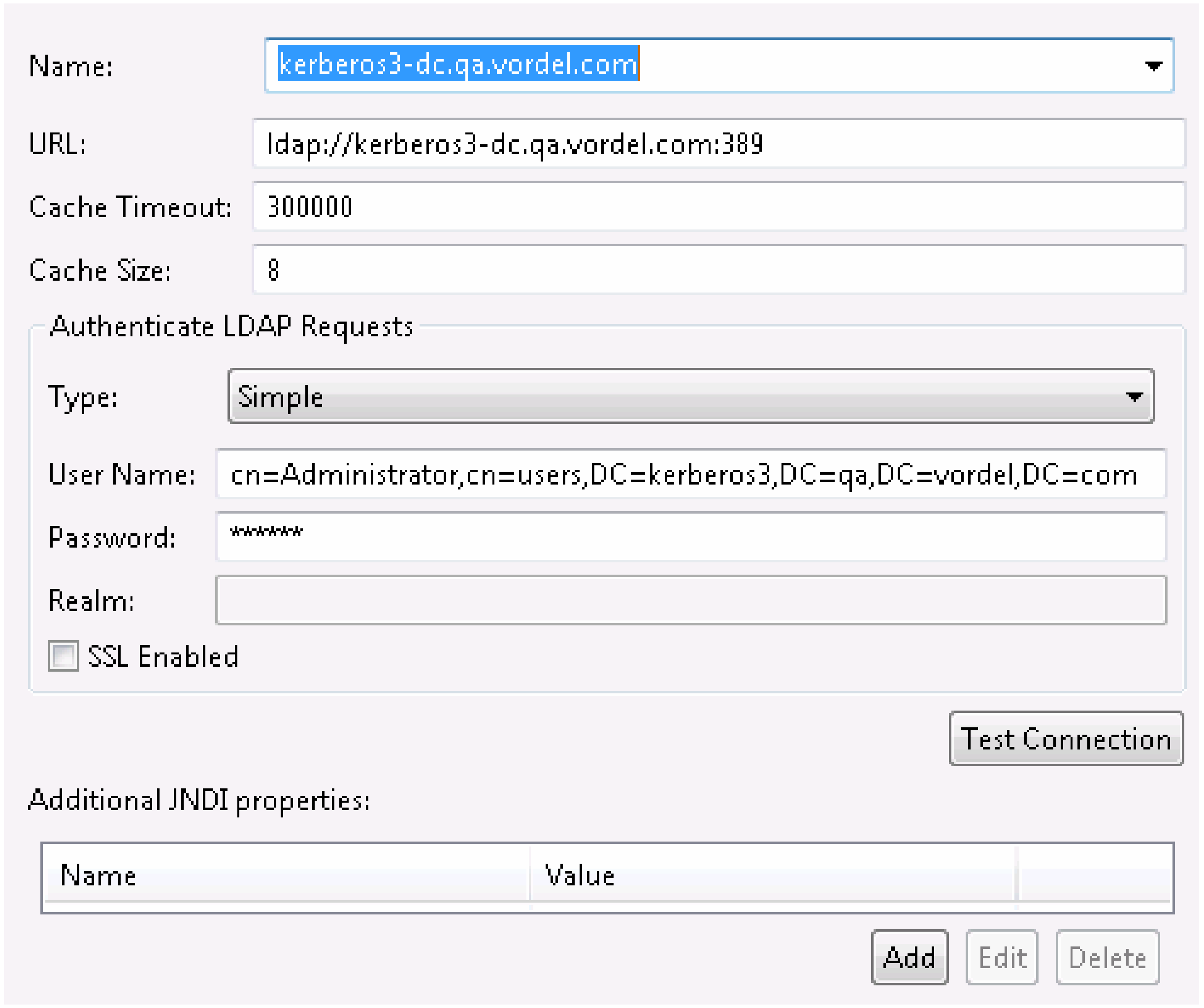
The `memberOf` attribute points to the Active Directory group. The user has an instance of this attribute for each group they are a member of.

Step 3: create an LDAP connection

To create an new LDAP Connection, perform the following steps:

1. In the Policy Studio, select **Open File** , and select the Admin Node Manager configuration file (for example, `INSTALL_DIR\apigateway\conf\fed\configs.xml`.
2. In the Policy Studio tree, select **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:



Name: kerberos3-dc.qa.vordel.com
URL: ldap://kerberos3-dc.qa.vordel.com:389
Cache Timeout: 300000
Cache Size: 8
Authenticate LDAP Requests
Type: Simple
User Name: cn=Administrator,cn=users,DC=kerberos3,DC=qa,DC=vordel,DC=com
Password: *******
Realm:
SSL Enabled

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:

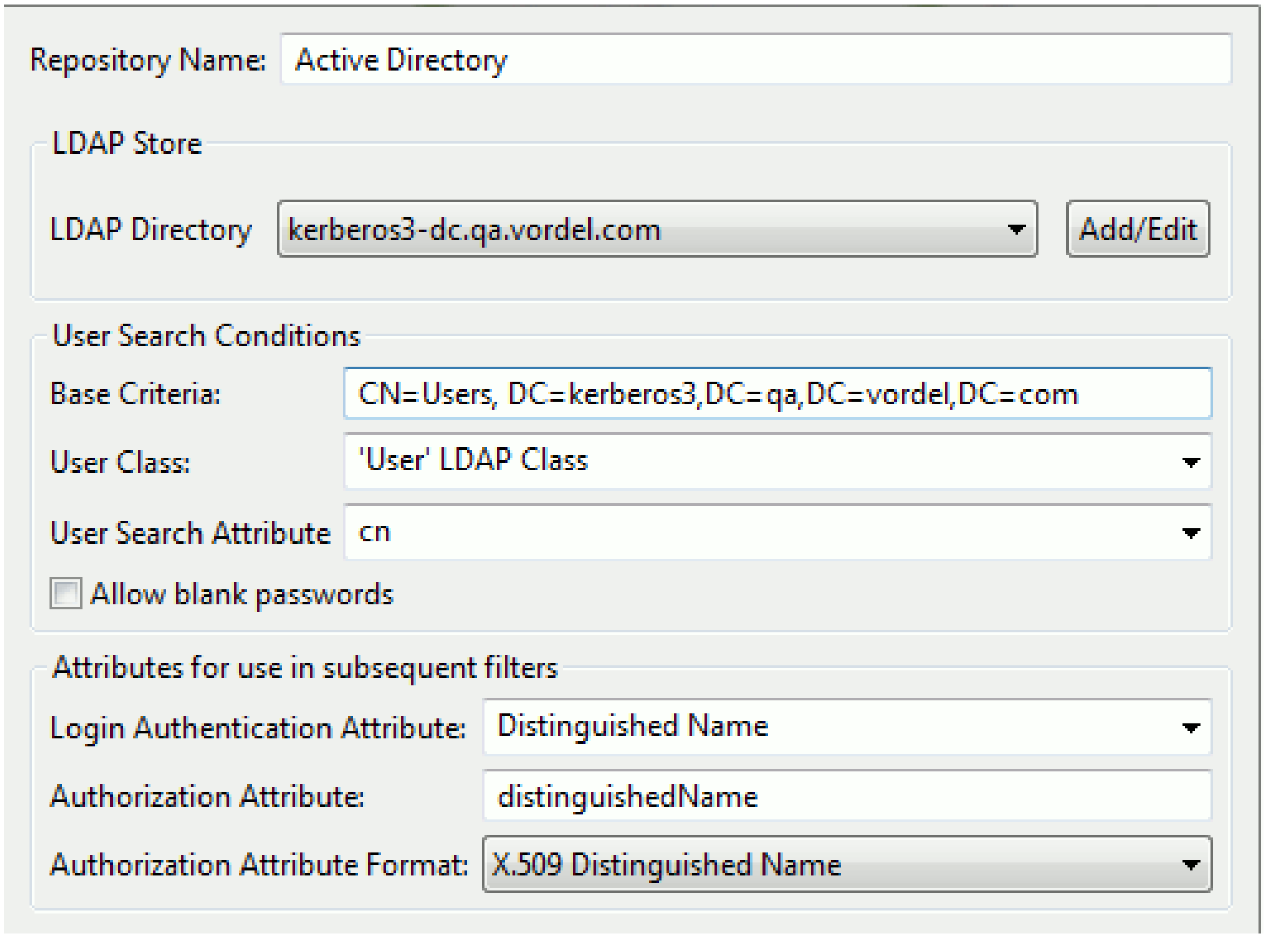
1. In the Policy Studio tree, select **External Connections** > **Authentication Repository Profiles** > **LDAP Repositories**.
2. Right-click, and select **Add a new Repository**.
3. Complete the following fields in the dialog:

Repository Name	Enter an appropriate name for the repository.
LDAP Directory	Use the LDAP directory created in the section called "Step 3: create an LDAP connection".
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 the section called "Step 5: create a test policy for LDAP authentication and RBAC".

An example value of the `authentication.subject.id` message attribute is as follows:

```
CN=admin, CN=Users,DC=kerberos3,DC=qa,DC=vordel,DC=com
```



Repository Name: Active Directory

LDAP Store

LDAP Directory: kerberos3-dc.qa.vordel.com

User Search Conditions

Base Criteria: CN=Users, DC=kerberos3,DC=qa,DC=vordel,DC=com

User Class: 'User' LDAP Class

User Search Attribute: cn

☐ Allow blank passwords

Attributes for use in subsequent filters

Login Authentication Attribute: Distinguished Name

Authorization Attribute: distinguishedName

Authorization Attribute Format: X.509 Distinguished Name

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 the section called "Step 5: create a test policy for LDAP authentication and RBAC". For details on using OpenLDAP, see *OpenLDAP for authentication and RBAC of management services*.

Step 5: create a test policy for LDAP authentication and RBAC

To avoid locking yourself out of the Policy Studio, you can create a 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). For an example policy, select **Policies** > **Management Services** > **Sample LDAP Policies** > **Protect Management Interfaces (LDAP)** when the Admin Node Manager configuration is loaded.

Create the test policy
Perform the following steps:

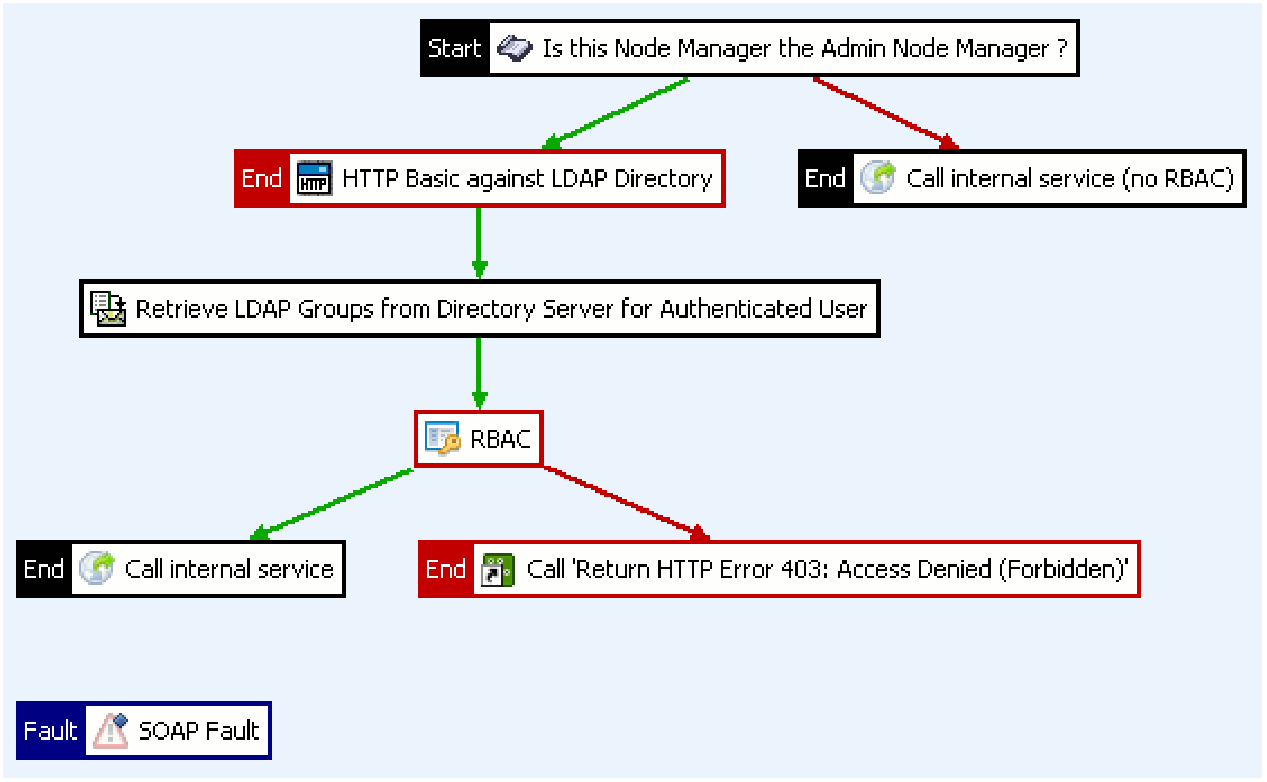
1. Select **Open File** and load the Admin Node Manager configuration file in the Policy Studio. For example:

```
INSTALL_DIR/apigateway/conf/fed/configs.xml
```

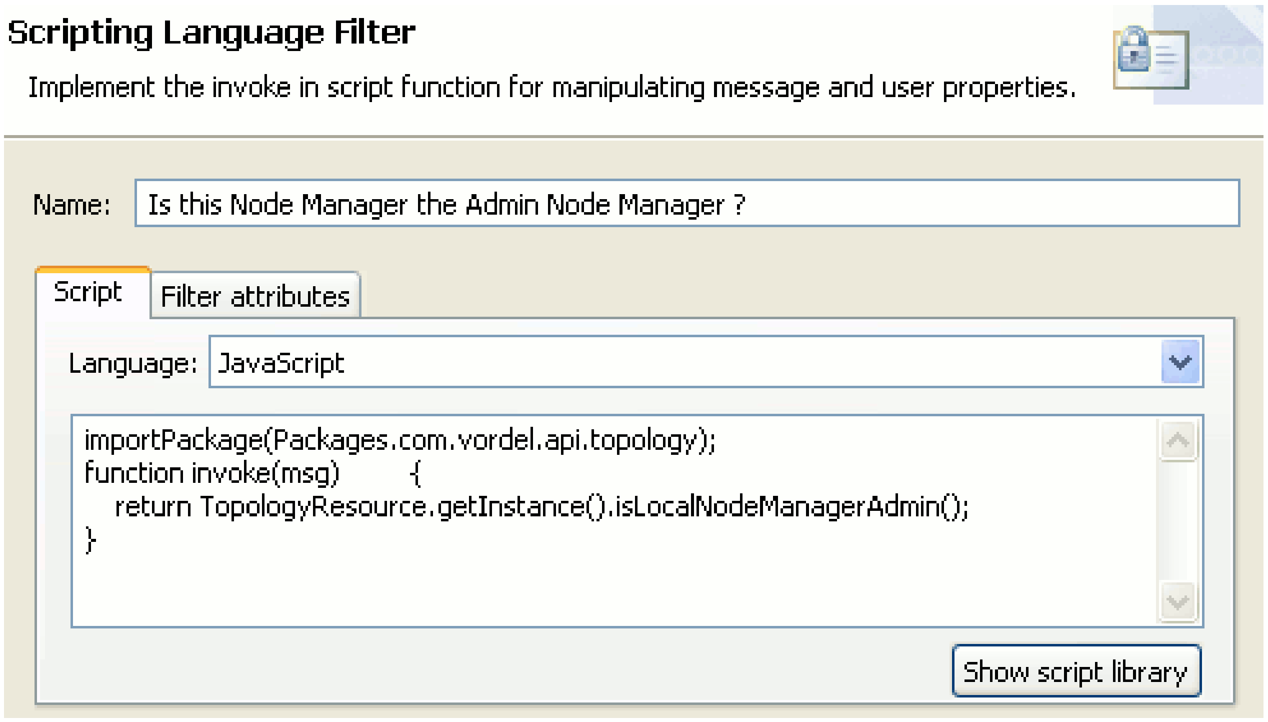
2. Right-click the **Policies** node in the tree view of the Policy Studio, and select **Add Policy**.
3. Enter a suitable name (for example `Test Policy`) for the new policy in the **Name** field, and click **OK**.
4. Click the new policy in the tree to start configuring the policy filters. You can configure the policy by dragging the required filters from the filter palette on the right, and dropping them on to the policy canvas.

For more details on creating policies, see the *API Gateway User Guide*.

Configure the test policy
Configure the test policy with the following filters:


Start
Is this Node Manager the Admin Node Manager ?
End
HTTP Basic against LDAP Directory
End
Call internal service (no RBAC)
Retrieve LDAP Groups from Directory Server for Authenticated User
RBAC
End
Call internal service
End
Call 'Return HTTP Error 403: Access Denied (Forbidden)'
Fault
SOAP Fault

Scripting Language filter
This includes the following settings:



Scripting Language Filter

Implement the invoke in script function for manipulating message and user properties.

Name: Is this Node Manager the Admin Node Manager ?

Language: JavaScript

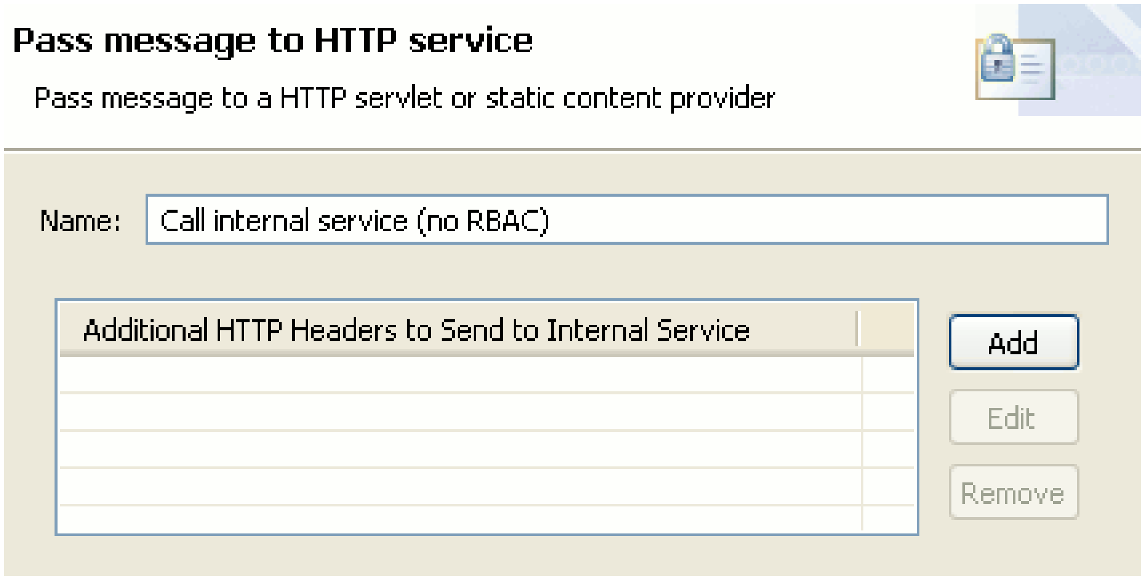
```
importPackage(Packages.com.vordel.api.topology);
function invoke(msg)         {
   return TopologyResource.getInstance().isLocalNodeManagerAdmin();
}
```

The **Scripting Language** filter performs the following tasks:

1. Returns true if the Node Manager is the Admin Node Manager and passes control to the **HTTP Basic Authentication** filter.
2. Otherwise, calls the **Call Internal Service (no RBAC)** filter without adding the `authentication.subject.role` and `authentication.subject.id` HTTP headers.

Call Internal Service (no RBAC) filter

This filter is called without adding any HTTP headers as follows:



Pass message to HTTP service

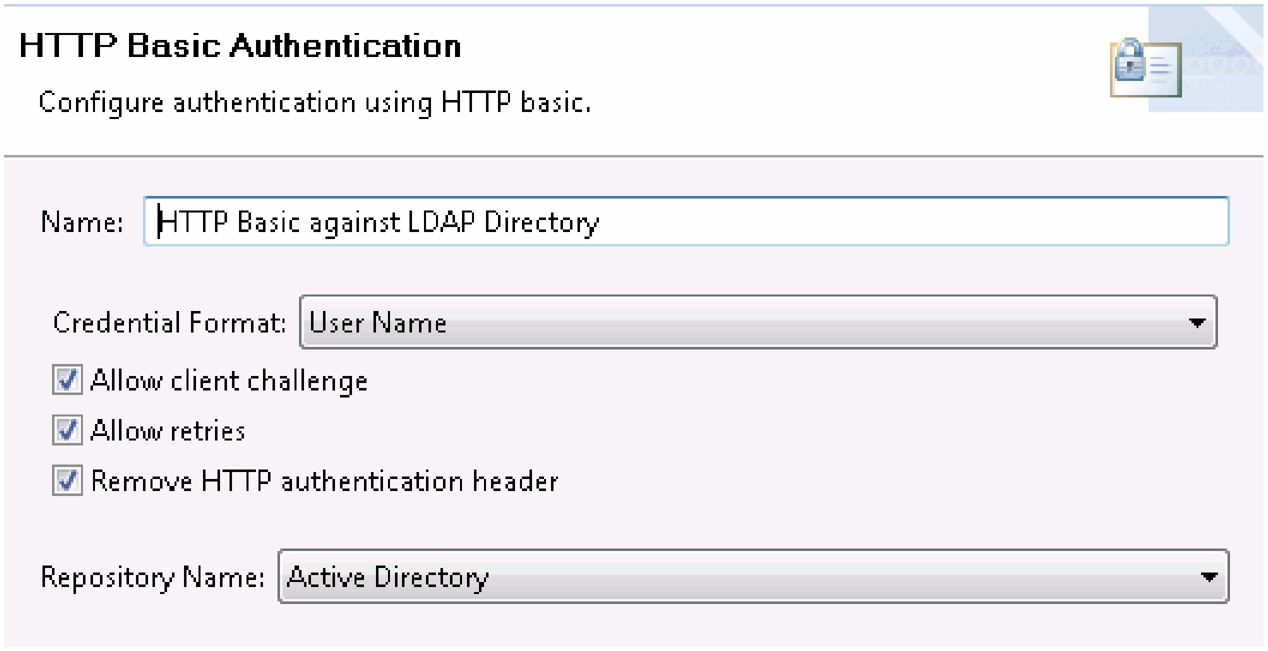
Pass message to a HTTP servlet or static content provider

Name: Call internal service (no RBAC)

Additional HTTP Headers to Send to Internal Service

HTTP Basic Authentication filter

This filter uses the LDAP repository configured in the section called "Step 4: create an LDAP repository", and includes the following settings:


HTTP Basic Authentication

Configure authentication using HTTP basic.

Name: HTTP Basic against LDAP Directory

Credential Format: User Name

☑ Allow client challenge

☑ Allow retries

☑ Remove HTTP authentication header

Repository Name: Active Directory

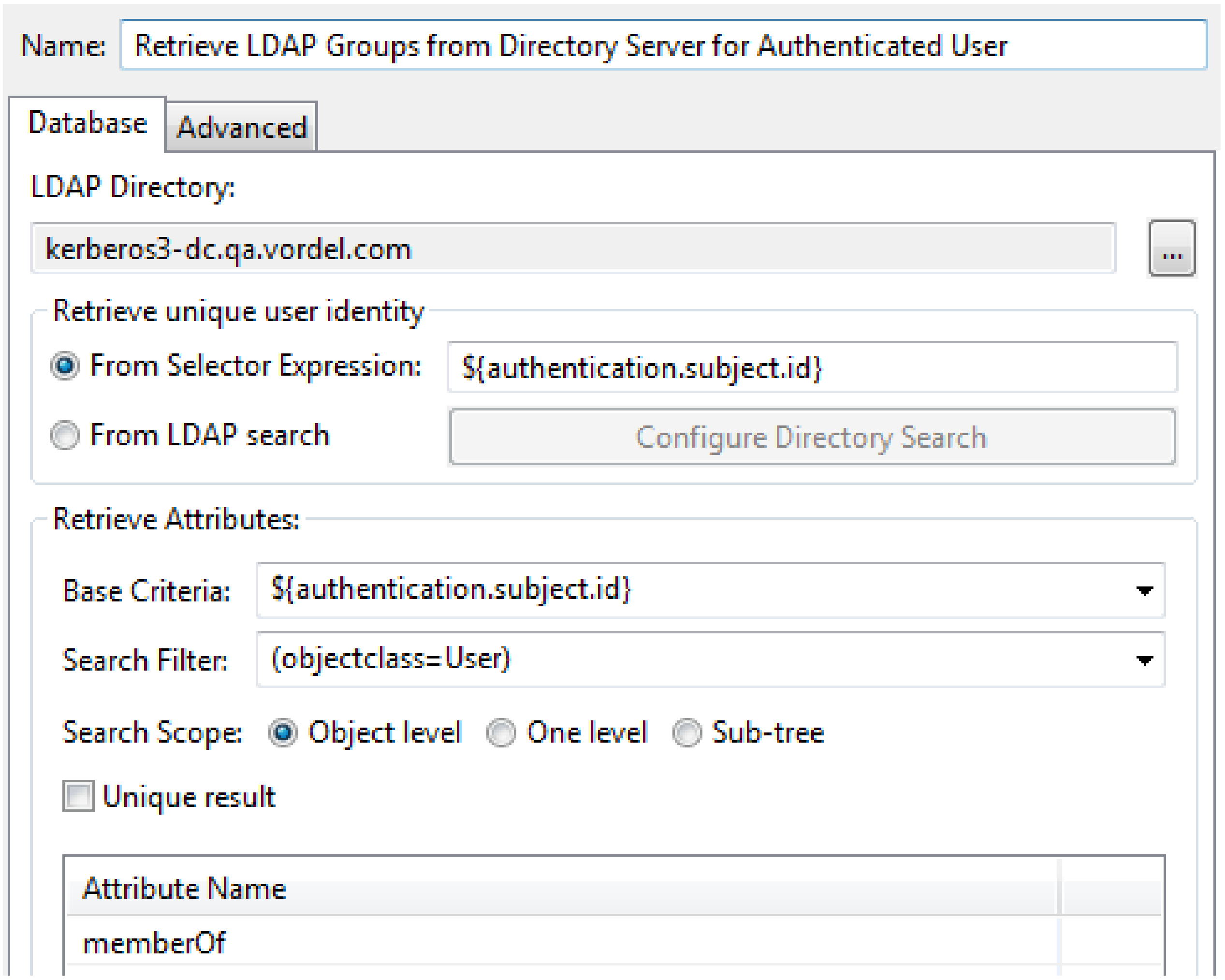
The **HTTP Basic Authentication** filter performs the following tasks:

1. Connects to the LDAP directory using the connection details specified in the LDAP directory.
2. Finds the user using the specified base criteria and search filter.
3. If the user is found, verifies the user's name and password against the LDAP repository by performing a bind.
4. If authentication fails, always throws a 401. This allows retry for browser users.
5. The `distinguishedName` (DName) is held in the `authentication.subject.id` message attribute. This is specified by the **Authorization Attribute** field in the LDAP repository configuration.
6. The user’s roles (LDAP groups) are not available yet.

Retrieve Attributes from Directory Server filter

On the **Database** tab, this filter uses the LDAP directory configured in the section called “Step 3: create an LDAP connection”, and includes the following settings:

- **Base Criteria**:
  `${authentication.subject.id}`
- **Search Filter**:
  `(objectclass=User)`
- **Attribute Name**:
  `memberOf`



Name: Retrieve LDAP Groups from Directory Server for Authenticated User

LDAP Directory: kerberos3-dc.qa.vordel.com

Retrieve unique user identity

[✓ From Selector Expression:] ${authentication.subject.id}

From LDAP search

Retrieve Attributes:

Base Criteria: ${authentication.subject.id}

Search Filter: (objectclass=User)

Search Scope: [✓ Object level] One level Sub-tree

Unique result: ☐

Attribute Name
memberOf

Note

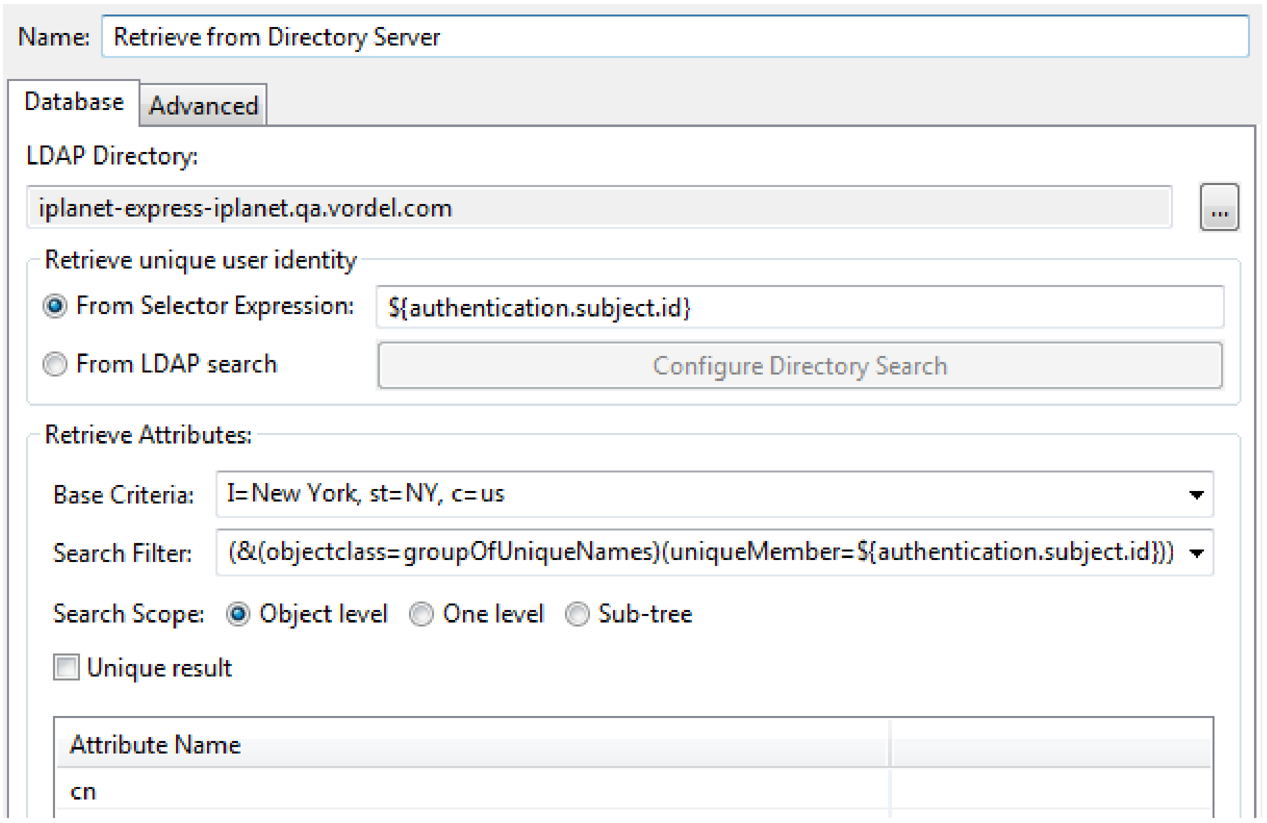
On the **Advanced** tab, you must ensure that the **Enable legacy attribute naming for retrieved attributes** setting is selected.

The **Retrieve Attributes from Directory Server** filter performs the following tasks:

1. Using the user’s DName as the search start point, find the user’s `memberOf` attribute, and load the LDAP groups for the user.
2. If the user is in one group, the group name is contained in the `user.memberOf` message attribute.
3. If the user is in multiple (n) LDAP groups, the group names are held in `user.memberOf.1` … `user.memberOf.n` message attributes.

Alternatively, the following screen shows an example of querying an Oracle Directory Server repository. The following **Search Filter** setting returns the authenticated user’s groups instead of the user object:

```
(&(objectclass=groupOfUniqueNames)(uniqueMember=${authentication.subject.id}))
```


Name: Retrieve from Directory Server

LDAP Directory: iplanet-express-iplanet.qa.vordel.com

Retrieve unique user identity

(●) From Selector Expression: ${authentication.subject.id}

( ) From LDAP search

Retrieve Attributes:

Base Criteria: l=New York, st=NY, c=us

Search Filter: (&(objectclass=groupOfUniqueNames)(uniqueMember=${authentication.subject.id}))

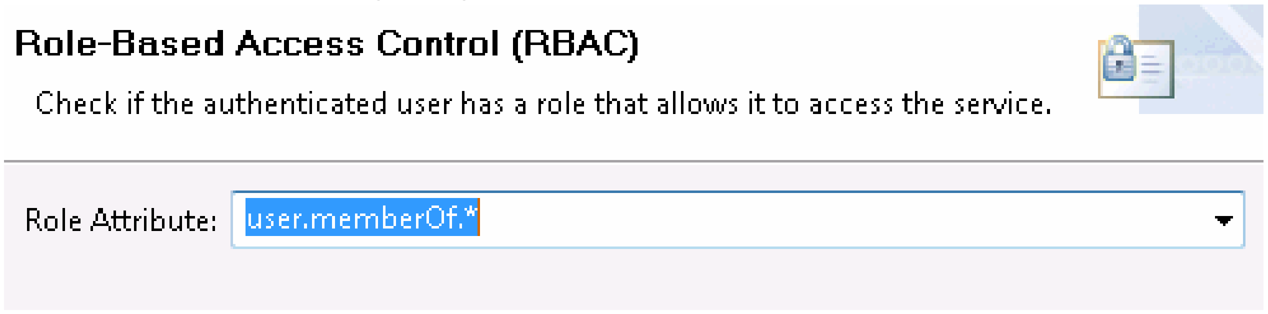
Search Scope: (●) Object level ( ) One level ( ) Sub-tree

[ ] Unique result

Attribute Name	
cn	

You should be able to query any LDAP directory in this way. Assuming that the user’s groups or roles can be retrieved as attributes of an object, the query does not need to be for the user object.

Management Services RBAC filter
This filter includes a the following setting:


Role-Based Access Control (RBAC)

Check if the authenticated user has a role that allows it to access the service.

Role Attribute: user.memberOf.*

The **Management Services RBAC** filter performs the following tasks:

1. Reads the roles from the `user.memberOf.*` message attribute. It understands the meaning of the wildcard, and loads the roles as required. It creates a string version of the roles, and places it in the `authentication.subject.role` message attribute for consumption by the **Call Internal Service** filter, which receives the roles as an HTTP header value.
2. Determines which Management Service URI is currently being invoked.
3. Returns true if one of the roles has access to the Management Service currently being invoked, as defined in the `acl.json file`.
4. Otherwise, returns false and the **Return HTTP Error 403: Access Denied (Forbidden)** policy is called. This message content of this filter is shown when a valid user has logged into the browser, but their role(s) does not give them access to the URI they have invoked. For example, this occurs if a new user is created and they have not yet been assigned any roles.

Test the policy configuration

To test this policy configuration, perform the following steps:

1. Update the `acl.json` file with the new LDAP group as follows:

```
"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 as follows:

```
{
  "name" : "CN=APIGatewayAdministrator,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. In the Policy Studio tree, select **Listeners** > **Node Manager** > **Add HTTP Services**, and enter a service name (for example, `LDAP Test`).
4. Right-click the HTTP service, and select **Add Interface** > **HTTP**.
5. Enter an available port to test the created policy (for example, `8888`), and click **OK**.
6. Right-click the HTTP service, and select **Add Relative Path**.
7. Enter a relative path (for example, `/test`).
8. Set the **Path Specify Policy** to the **Protect Management Interfaces (LDAP)** policy, and click **OK**.
9. Close the connection to the Admin Node Manager file, and restart the Admin Node Manager so it loads the updated configuration.
10. Use API Tester to call `http://localhost:8080/test`.
11. 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).

Important

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** > **Open**, and browse to `configs.xml` in the first copy of the `fed` directory.
4. Under the **Listeners** > **Management Services** node, select the `/` and the `/configuration/deployments` relative paths, and set the **Path Specify Policy** to the **Protect Management Interfaces (LDAP)** policy.
5. Remove the previously created `LDAP Test` HTTP Services.

6. Close the connection to the file.
7. Copy the `fed` directory back to the Admin Node Manager’s `conf` directory.
8. Reboot the Admin Node Manager.
9. 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`.

Adding 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.

OpenLDAP for authentication and RBAC of management services

Overview

This topic explains how to use Local Directory Access Protocol (LDAP) to authenticate and perform Role-Based Access Control (RBAC) of management services. You can use the sample **Protect Management and Interfaces (LDAP)** policy instead of the **Protect Management Interfaces** policy. This means that an LDAP repository is used instead of the local Admin User store for authentication and RBAC of users attempting to access the management services. This topic describes how to reconfigure the server to use OpenLDAP as the LDAP repository, and to use the Apache Directory Studio as an LDAP browser.

Note

This topic applies to both API Gateway and API Gateway Analytics.

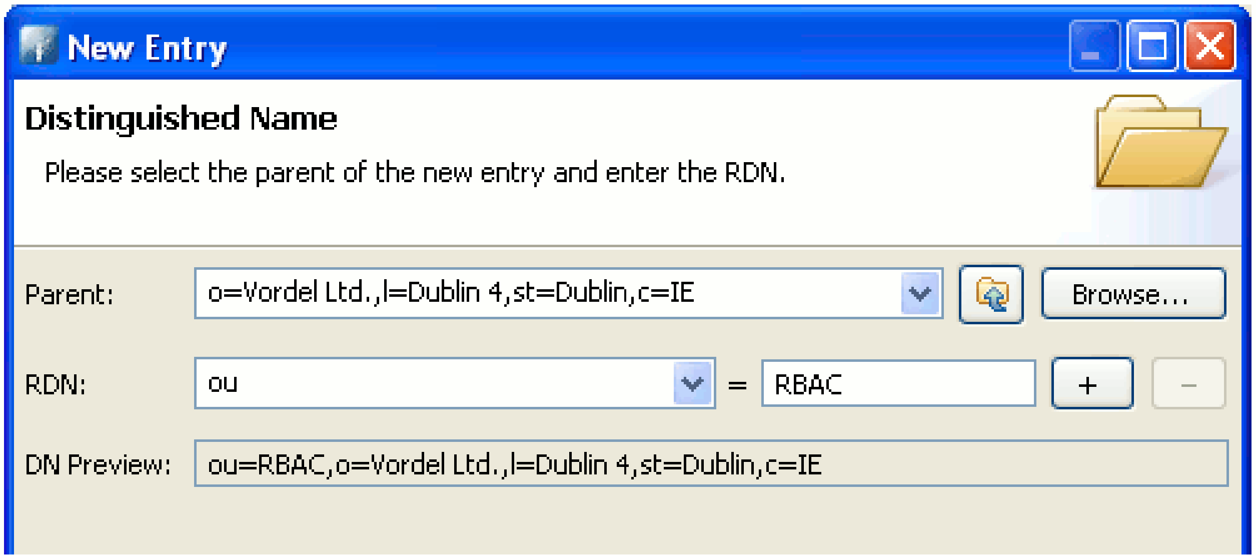
Prerequisites
This example assumes that you already have configured connection to the OpenLDAP server and setup your organization groups and users that you wish to use to perform RBAC. For example:

- **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**.


New Entry

Distinguished Name

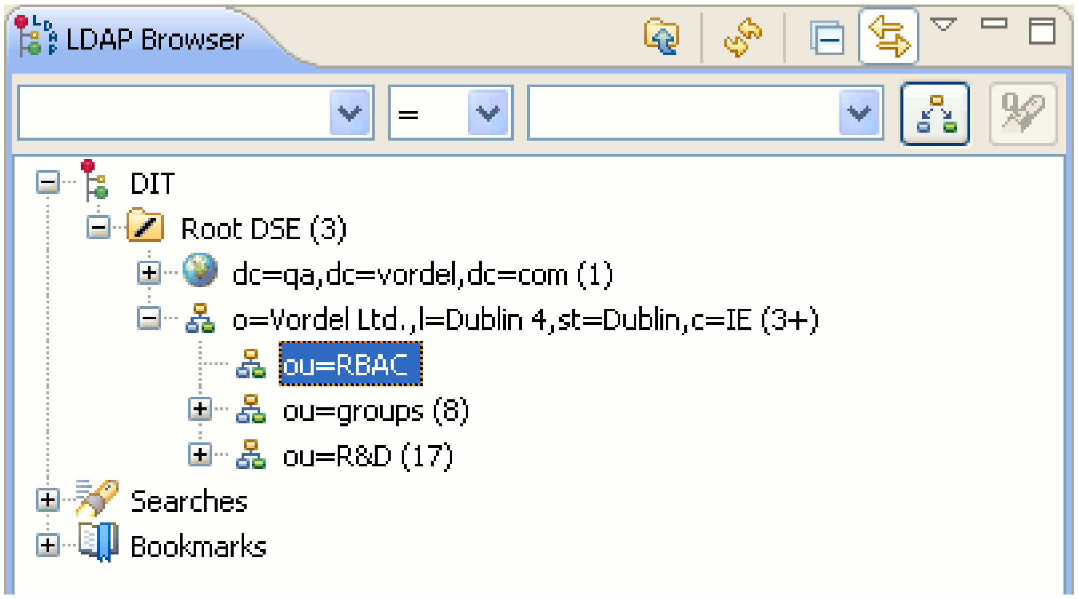
Please select the parent of the new entry and enter the RDN.

Parent: o=Vordel Ltd.,l=Dublin 4,st=Dublin,c=IE

RDN: ou = RBAC

DN Preview: ou=RBAC,o=Vordel Ltd.,l=Dublin 4,st=Dublin,c=IE

You can view the new group using an LDAP Browser. For example:



```
DIT
  Root DSE (3)
    dc=qa,dc=vordel,dc=com (1)
    o=Vordel Ltd.,l=Dublin 4,st=Dublin,c=IE (3+)
      ou=RBAC
      ou=groups (8)
      ou=R&D (17)
Searches
Bookmarks
```

Step 2: add RBAC roles to the OpenLDAP RBAC group

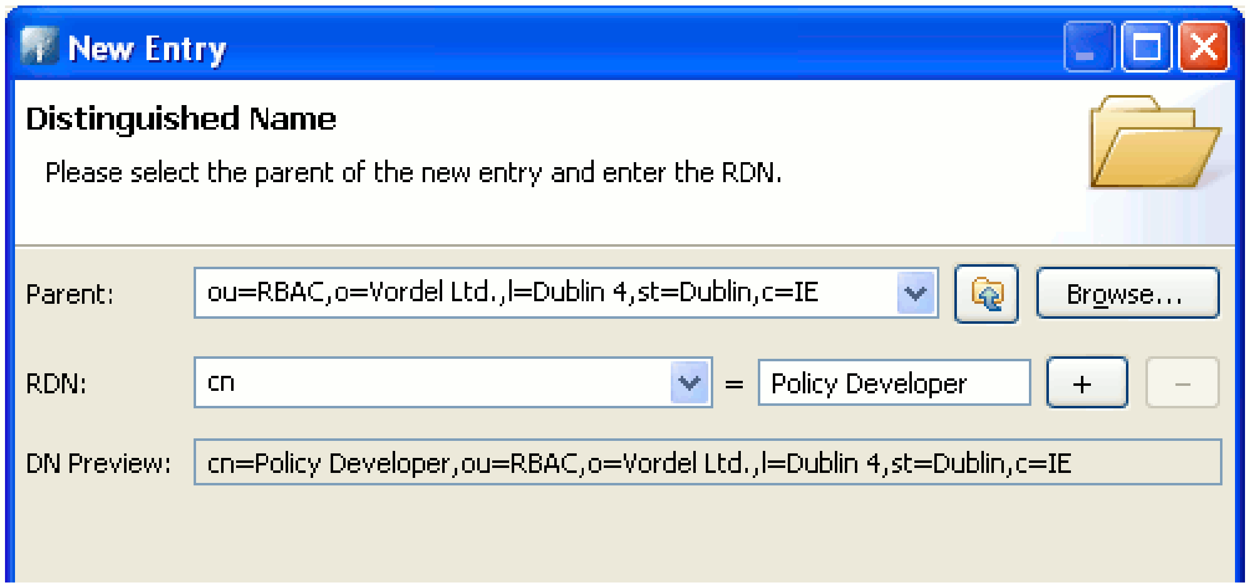
You must add the following default RBAC roles to the `ou=RBAC,o=Vordel Ltd.,l=Dublin 4,st=Dublin,C=IE` group to give the LDAP users appropriate access to the API Gateway management services:

- `API Gateway Administrator`
- `API Gateway Operator`
- `KPS Administrator`
- `Policy Developer`
- `Deployer`

These RBAC roles are located in the `roles` section of the `acl.json` file.

Adding 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**.



Distinguished Name

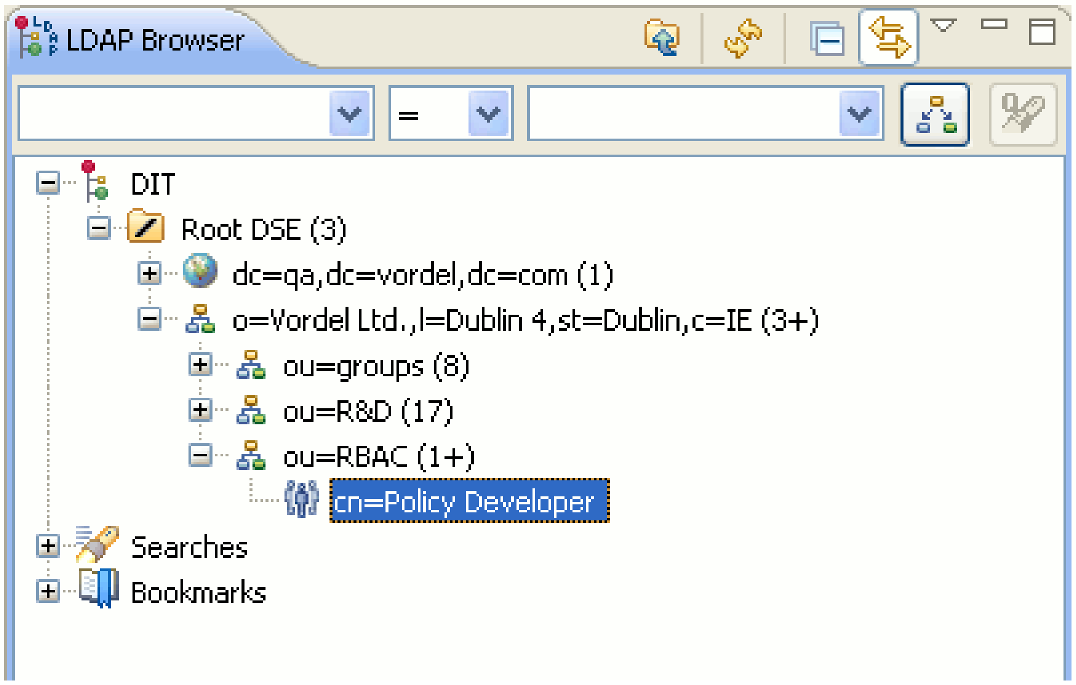
Please select the parent of the new entry and enter the RDN.

Parent: ou=RBAC,o=Vordel Ltd.,l=Dublin 4,st=Dublin,c=IE

RDN: cn = Policy Developer

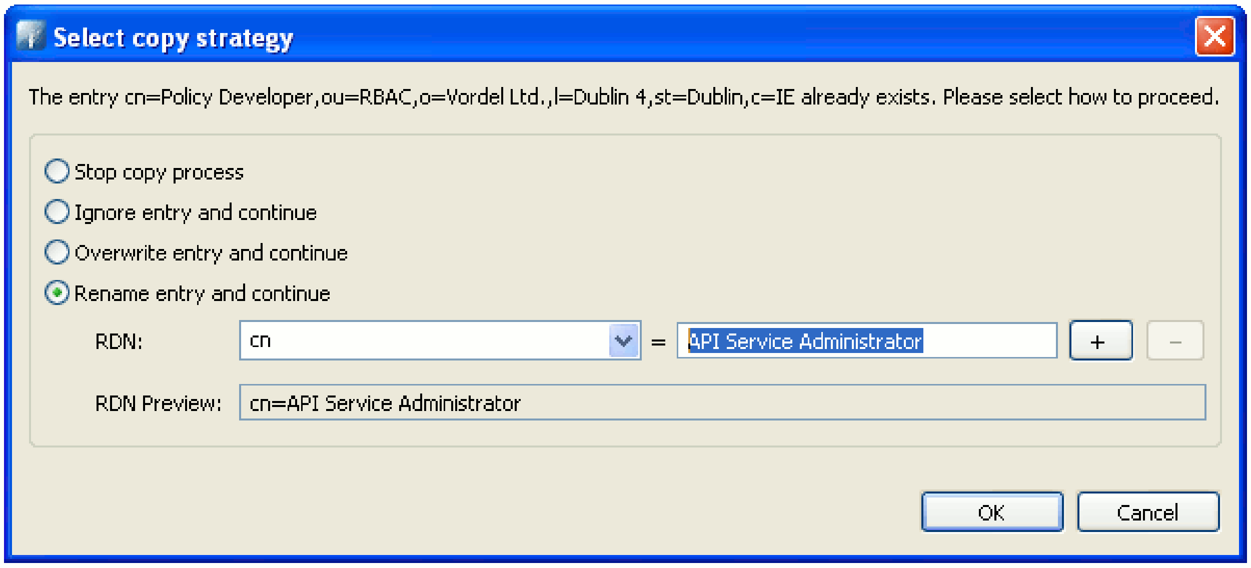
DN Preview: cn=Policy Developer,ou=RBAC,o=Vordel Ltd.,l=Dublin 4,st=Dublin,c=IE

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:



Select copy strategy

The entry cn=Policy Developer,ou=RBAC,o=Vordel Ltd.,l=Dublin 4,st=Dublin,c=IE already exists. Please select how to proceed.

Stop copy process [ ]
Ignore entry and continue [ ]
Overwrite entry and continue [ ]
[✓ Rename entry and continue]

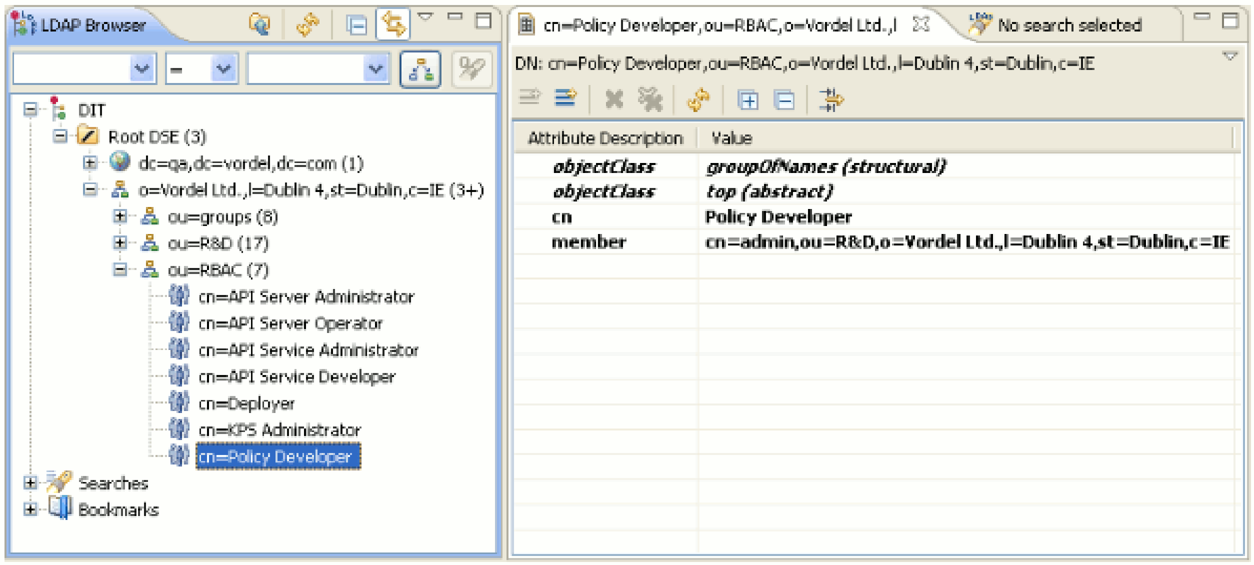
RDN: cn = API Service Administrator

RDN Preview: cn=API Service Administrator

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:



DN: cn=Policy Developer,ou=RBAC,o=Vordel Ltd.,l=Dublin 4,st=Dublin,c=IE

- DIT
  - Root DSE (3)
    - dc=qa,dc=vordel,dc=com (1)
    - o=Vordel Ltd.,l=Dublin 4,st=Dublin,c=IE (3+)
      - ou=groups (8)
      - ou=R&D (17)
      - ou=RBAC (7)
        - cn=API Server Administrator
        - cn=API Server Operator
        - cn=API Service Administrator
        - cn=API Service Developer
        - cn=Deployer
        - cn=KPS Administrator
        - cn=Policy Developer

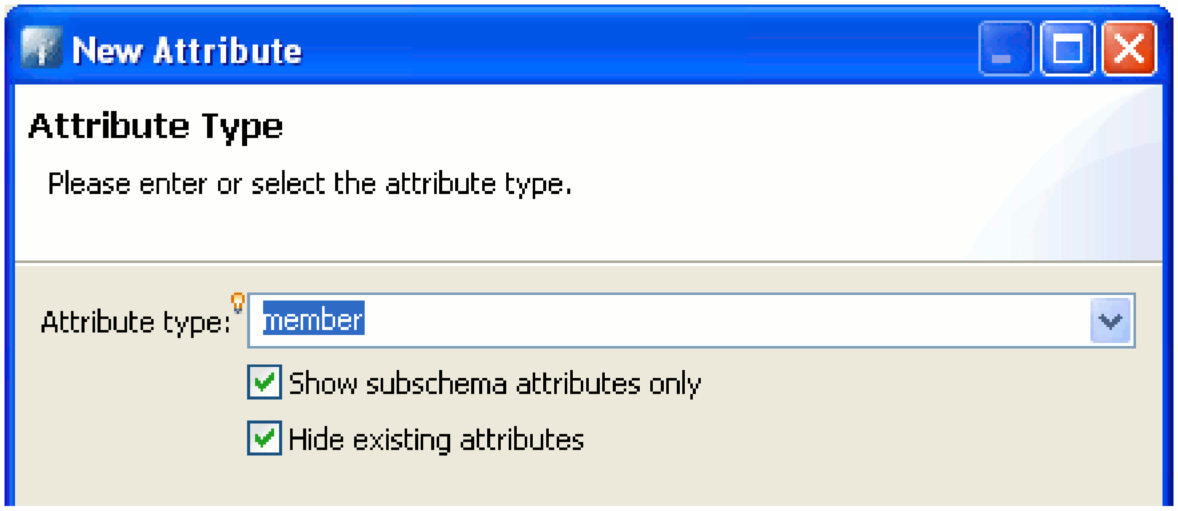
Attribute Description	Value
objectClass	*groupOfNames (structural)*
objectClass	*top (abstract)*
cn	Policy Developer
member	cn=admin,ou=R&D,o=Vordel Ltd.,l=Dublin 4,st=Dublin,c=IE

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.



Attribute Type

Please enter or select the attribute type.

Attribute type: member

[✓] Show subschema attributes only

[✓] Hide existing attributes

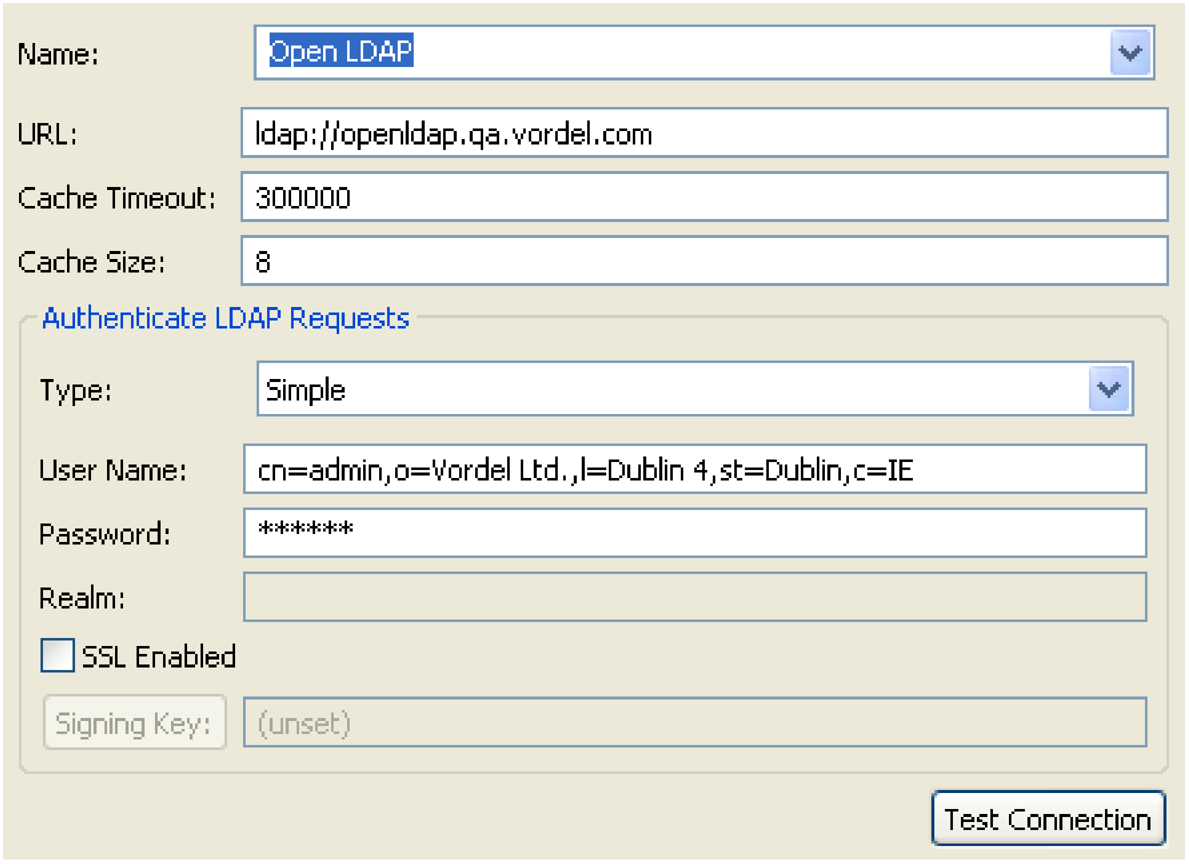
4. Click **Finish**.
5. In the **DN Editor** dialog, enter the user Distinguished Name (for example, `cn=joe.bloggs,o=Vordel Ltd.,l=Dublin 4,st=Dublin,c=IE`).
6. Click **OK**.

The `cn=joe.bloggs,o=Vordel Ltd.,l=Dublin 4,st=Dublin,c=IE` new user has been added to the RBAC `API Gateway Administrator` role.

Step 4: create an LDAP connection

To create an new LDAP Connection, perform the following steps:

1. In the Policy Studio, select **Open File** , and select the Admin Node Manager configuration file (for example, `INSTALL_DIR\apigateway\conf\fed\configs.xml`).
2. In the Policy Studio tree, select **External Connections** > **LDAP Connections**.
3. Right-click, and select **Create an LDAP Connection**.
4. Complete the fields in the dialog as appropriate. For example:



Name:	Open LDAP
URL:	ldap://openldap.qa.vordel.com
Cache Timeout:	300000
Cache Size:	8

Authenticate LDAP Requests

Type:	Simple
User Name:	cn=admin,o=Vordel Ltd.,l=Dublin 4,st=Dublin,c=IE
Password:	*******
Realm:	

☐ SSL Enabled

Signing Key:	(unset)

Note

The specified **User Name** should be an LDAP administrator that has access to search the full directory for users.

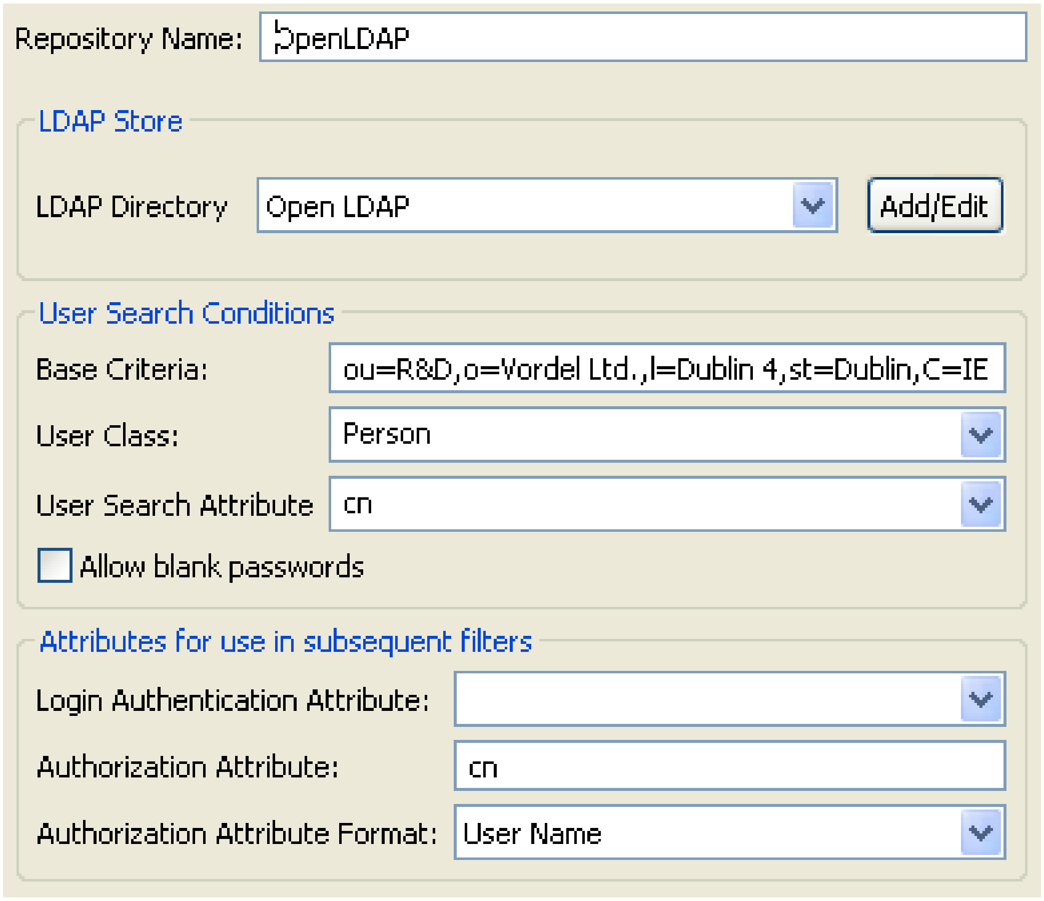
5. Click **Test Connection** to ensure the connection details are correct.

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 the section called "Step 4: create an LDAP connection".
Base Criteria	Enter the LDAP node that contains the users (for example, see the LDAP Browser screen in the section called "Step 3: add users to the OpenLDAP RBAC group").
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`. The `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 the section called "Step 6: create a test policy for LDAP authentication and RBAC".


Repository Name: OpenLDAP

LDAP Store

LDAP Directory: Open LDAP

User Search Conditions

Base Criteria: ou=R&D,o=Vordel Ltd.,l=Dublin 4,st=Dublin,C=IE

User Class: Person

User Search Attribute: cn

☐ Allow blank passwords

Attributes for use in subsequent filters

Login Authentication Attribute:

Authorization Attribute: cn

Authorization Attribute Format: User Name

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 *Active Directory for authentication and RBAC of management services*. For an example of querying an Oracle Directory Server repository, see the **Retrieve Attributes from Directory Server** filter in the section called “Step 5: create a test policy for LDAP authentication and RBAC”.

Step 6: create a test policy for LDAP authentication and RBAC

To avoid locking yourself out of the Policy Studio, you can create a 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). For an example policy, select **Policies** > **Management Services** > **Sample LDAP Policies** > **Protect Management Interfaces (LDAP)** when the Admin Node Manager configuration is loaded.

Create the test policy
Perform the following steps:

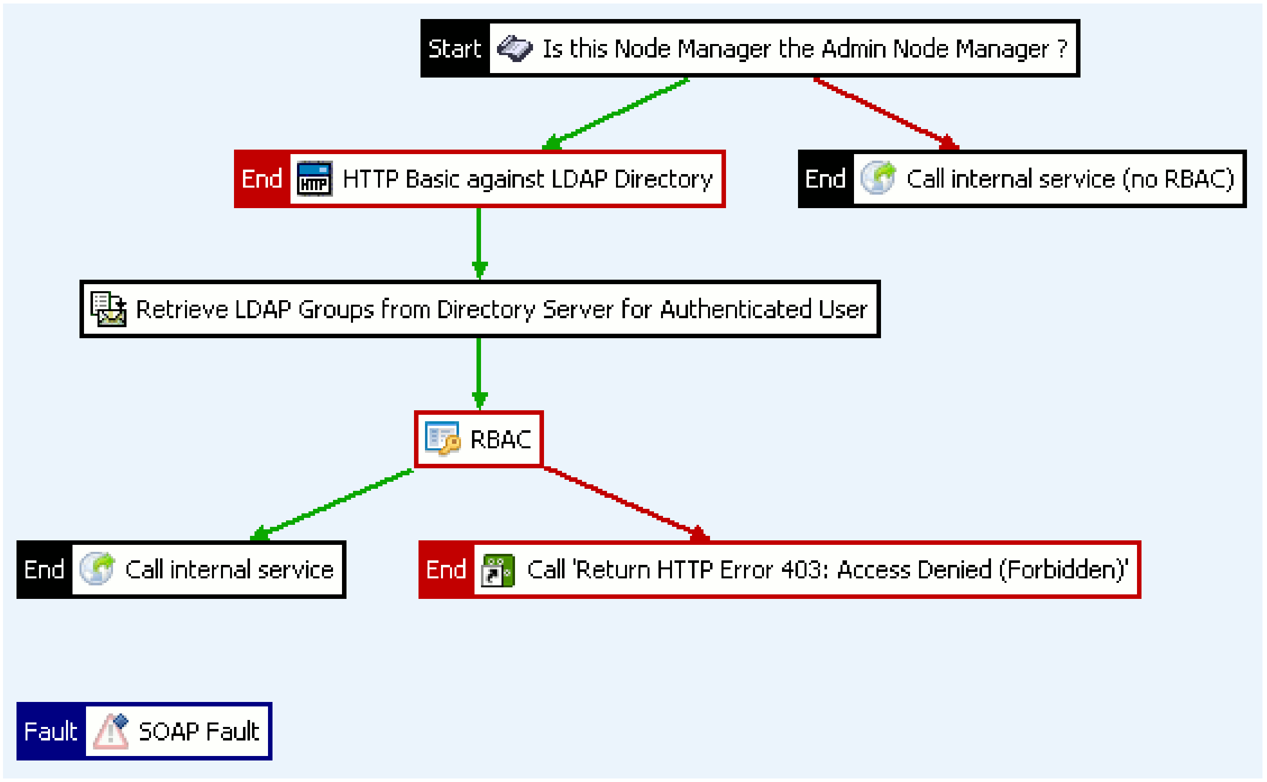
1. Select **Open File** and load the Admin Node Manager configuration file in the Policy Studio. For example:

   ```
   INSTALL_DIR/apigateway/conf/fed/configs.xml
   ```

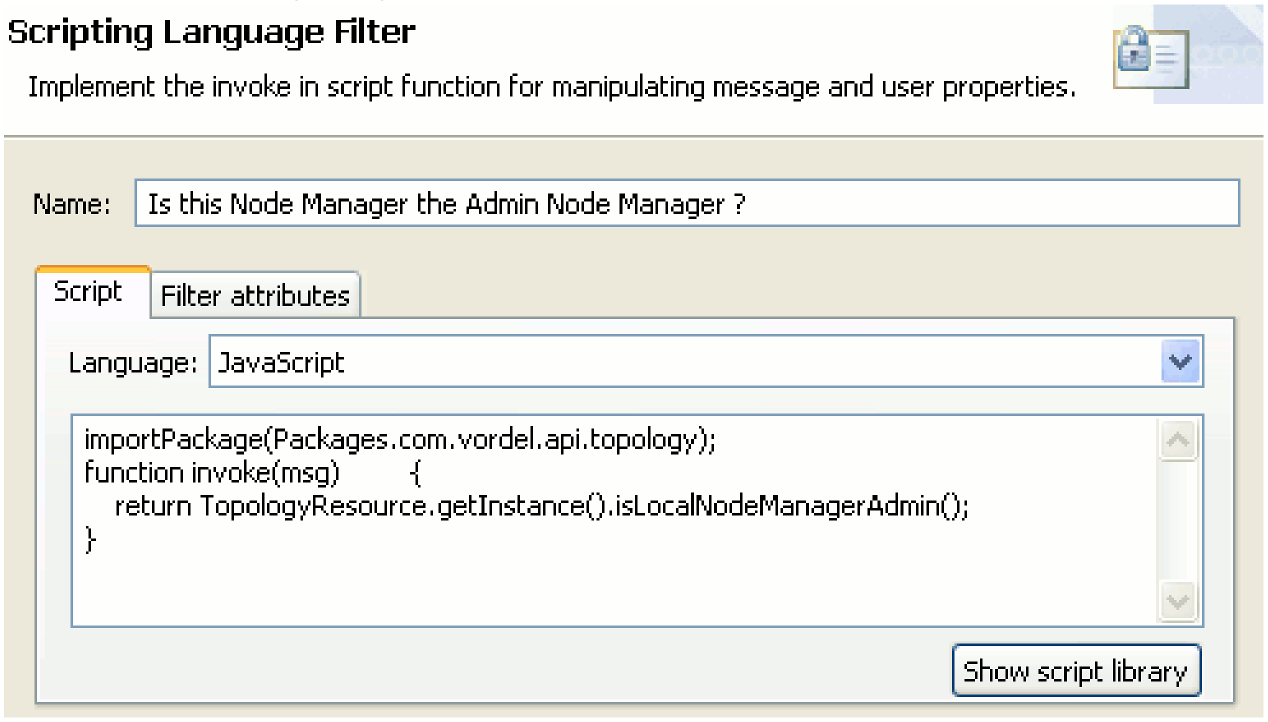
2. Right-click the **Policies** node in the tree view of the Policy Studio, and select **Add Policy** .
3. Enter a suitable name (for example `Test Policy`) for the new policy in the **Name** field, and click the **OK** button. The new policy is now visible in the tree view.
4. Click the new policy in the tree view to start configuring the filters for the policy. You can easily configure the policy by dragging the required filters from the filter palette on the right of the Policy Studio, and dropping them on to the policy canvas.

For more details on creating policies, see the *API Gateway User Guide*.

Configure the test policy
Configure the test policy with the following filters:


Start
Is this Node Manager the Admin Node Manager ?
End
HTTP Basic against LDAP Directory
End
Call internal service (no RBAC)
Retrieve LDAP Groups from Directory Server for Authenticated User
RBAC
End
Call internal service
End
Call 'Return HTTP Error 403: Access Denied (Forbidden)'
Fault
SOAP Fault

Scripting Language filter
This includes the following settings:


Scripting Language Filter

Implement the invoke in script function for manipulating message and user properties.

Name: Is this Node Manager the Admin Node Manager ?

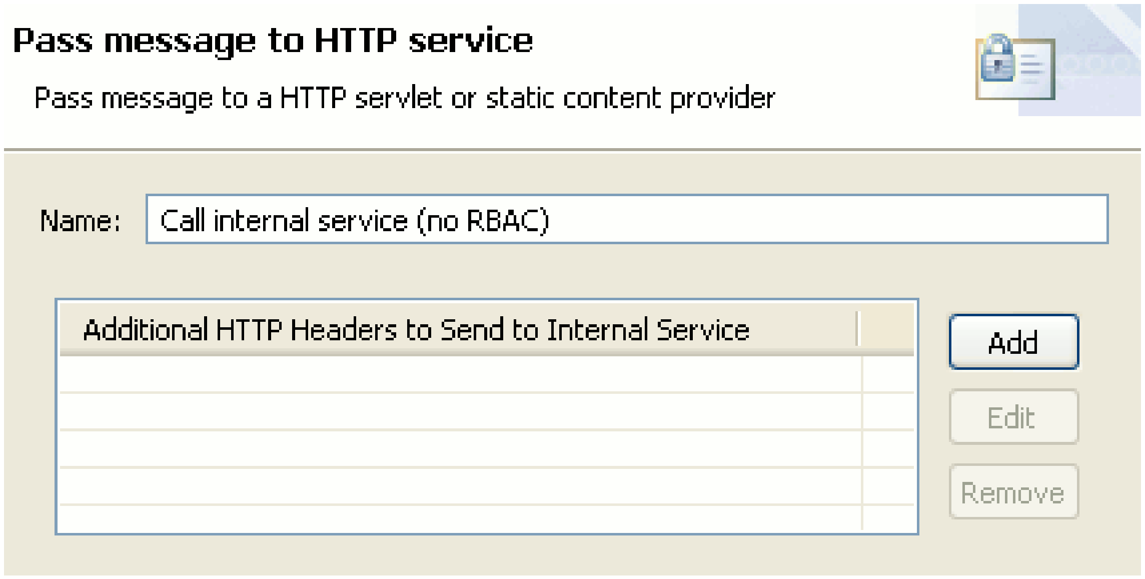
Language: JavaScript

```
importPackage(Packages.com.vordel.api.topology);
function invoke(msg)        {
   return TopologyResource.getInstance().isLocalNodeManagerAdmin();
}
```

The **Scripting Language** filter performs the following tasks:

1. Returns true if the Node Manager is the Admin Node Manager, and passes control to the **HTTP Basic Authentication** filter.
2. Otherwise, calls the **Call Internal Service (no RBAC)** filter without adding the `authentication.subject.role` and `authentication.subject.id` HTTP headers.

Call Internal Service (no RBAC) filter
This filter is called without adding any HTTP headers as follows:


Pass message to HTTP service

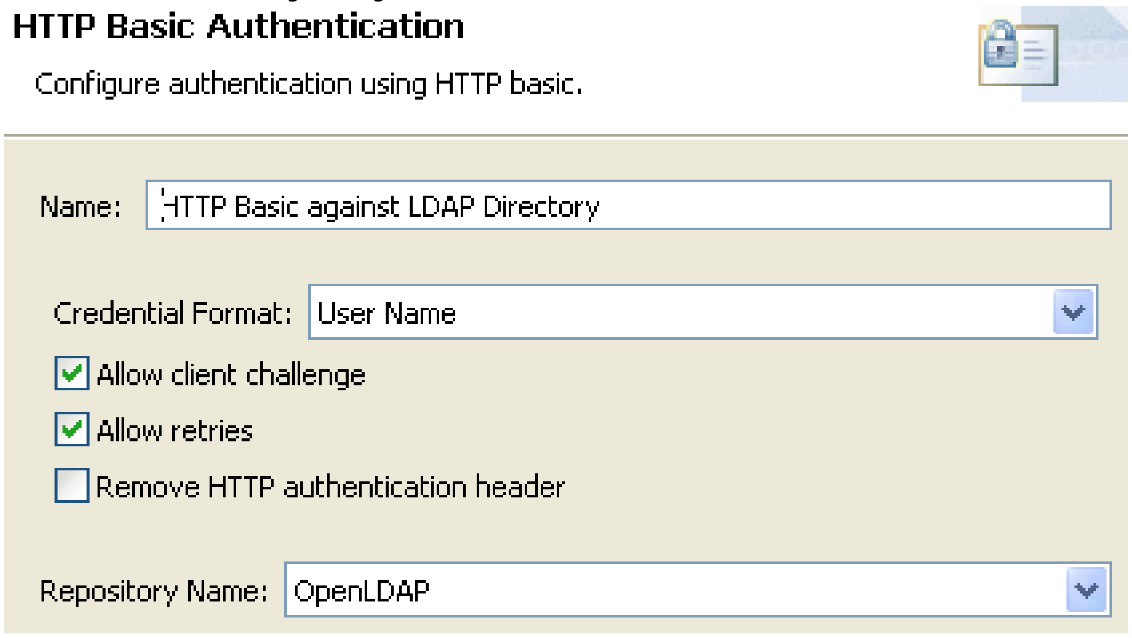
Pass message to a HTTP servlet or static content provider

Name: Call internal service (no RBAC)

Additional HTTP Headers to Send to Internal Service

HTTP Basic Authentication filter

This filter uses the LDAP repository configured in the section called "Step 5: create an OpenLDAP repository", and includes the following settings:



HTTP Basic Authentication

Configure authentication using HTTP basic.

Name: HTTP Basic against LDAP Directory

Credential Format: User Name

[✓] Allow client challenge

[✓] Allow retries

[ ] Remove HTTP authentication header

Repository Name: OpenLDAP

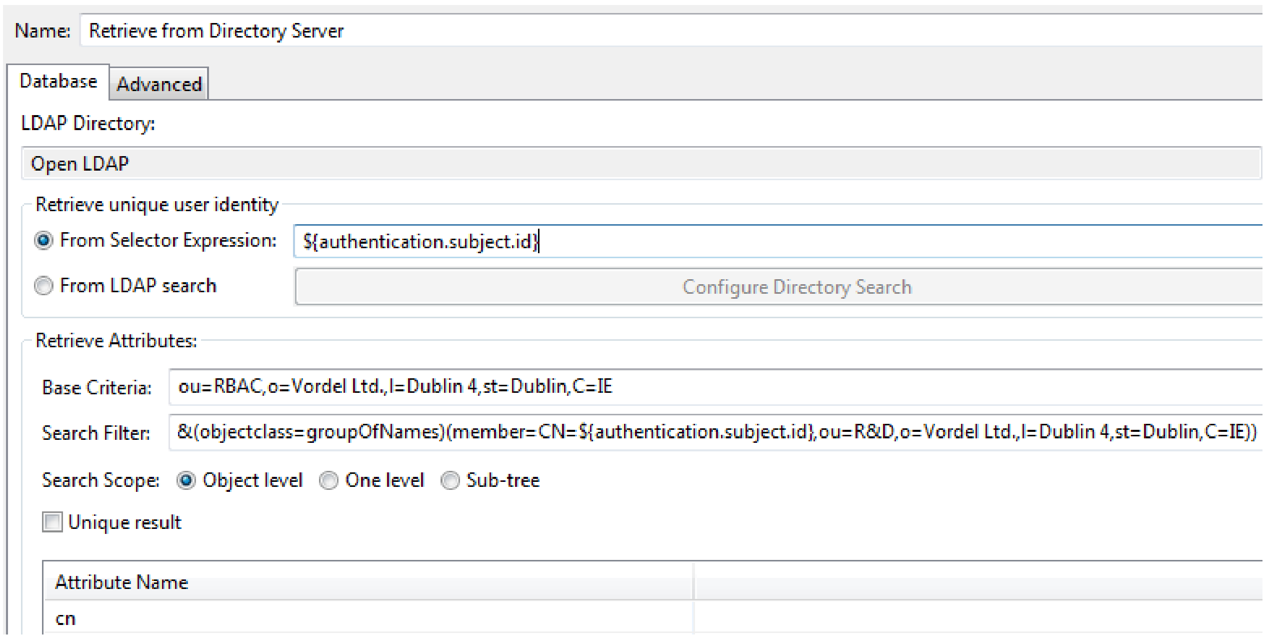
The **HTTP Basic Authentication** filter performs the following tasks:

1. Connects to the LDAP directory using the connection details specified in the LDAP directory.
2. Finds the user using the specified base criteria and search filter.
3. If the user is found, verifies the user's name and password against the LDAP repository by performing a bind.
4. If authentication fails, always throws a 401. This allows retry for browser users.
5. The `distinguishedName` (DName) is held in the `authentication.subject.id` message attribute. This is specified by the **Authorization Attribute** field in the LDAP repository configuration.
6. The user's roles (LDAP groups) are not available yet.

Retrieve Attributes from Directory Server filter

On the **Database** tab, this filter uses the LDAP directory configured in the section called "Step 4: create an LDAP connection", and includes the following settings:

- **Retrieve Unique User Identity From Selector Expression**:
  `${authentication.subject.id}`
- **Base Criteria**:
  `ou=RBAC,o=Vordel Ltd.,l=Dublin 4,st=Dublin,C=IE`
- **Search Filter**:
  `(&(objectclass=groupOfNames)(member=CN=${authentication.subject.id},ou=R&D,o=Vordel Ltd.,l=Dublin 4,st=Dublin,C=IE))`
- **Attribute Name**:
  `cn`


Name: Retrieve from Directory Server

LDAP Directory: Open LDAP

Retrieve unique user identity

● From Selector Expression: ${authentication.subject.id}

○ From LDAP search

Retrieve Attributes:

Base Criteria: ou=RBAC,o=Vordel Ltd.,l=Dublin 4,st=Dublin,C=IE

Search Filter: &(objectclass=groupOfNames)(member=CN=${authentication.subject.id},ou=R&D,o=Vordel Ltd.,l=Dublin 4,st=Dublin,C=IE))

Search Scope: ● Object level ○ One level ○ Sub-tree

☐ Unique result

Attribute Name
cn

Note

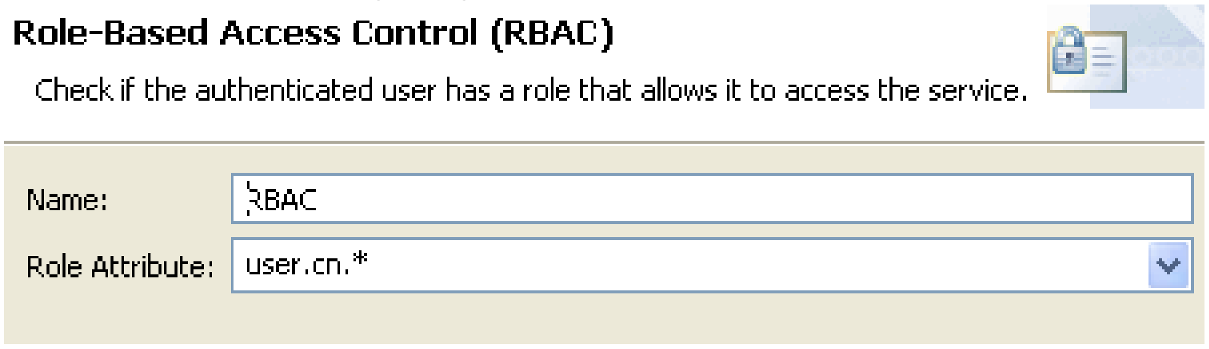
On the **Advanced** tab, you must ensure that the **Enable legacy attribute naming for retrieved attributes** setting is selected.

The **Retrieve Attributes from Directory Server** filter performs the following tasks:

1. Using the user's DName as the search start point, finds the user's `cn` attribute, and loads the LDAP groups for the user.
2. If the user is in one group, the group name is contained in the `user.cn` message attribute.
3. If the user is in multiple (`n`) LDAP groups, the group names are contained in `user.cn.1` ... `user.cn.n` message attributes.

Management Services RBAC filter

This filter includes a the following setting:


Role-Based Access Control (RBAC)

Check if the authenticated user has a role that allows it to access the service.

Name: RBAC

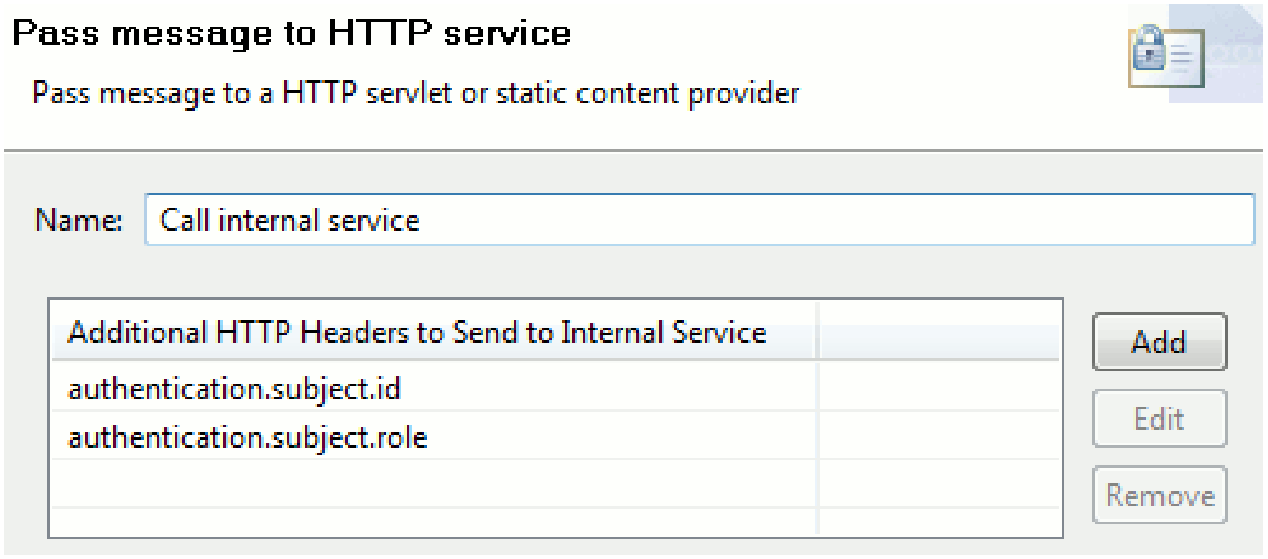
Role Attribute: user.cn.*

The **Management Services RBAC** filter performs the following tasks:

1. Reads the roles from the `user.cn.*` message attribute. It understands the meaning of the wildcard, and loads the roles as required. It creates a string version of the roles, and places it in the `authentication.subject.role` message attribute for consumption by the **Call Internal Service** filter, which receives the roles as an HTTP header value.
2. Determines which Management Service URI is currently being invoked.

3. Returns true if one of the roles has access to the Management Service currently being invoked, as defined in the `acl.json` file.
4. Otherwise, returns false and calls the **Return HTTP Error 403: Access Denied (Forbidden)** policy. The message content of this filter is shown when a valid user has logged into the browser, but their role(s) does not give them access to the URI they have invoked. For example, this occurs if a new user is created and they have not yet been assigned any roles.

Call Internal Service filter
This filter includes a the following settings:



Pass message to HTTP service

Pass message to a HTTP servlet or static content provider

Name: Call internal service

Additional HTTP Headers to Send to Internal Service	
authentication.subject.id	
authentication.subject.role	

The **Call Internal Service** filter sends the message to the internal service with the following additional HTTP headers:

- `authentication.subject.role`
- `authentication.subject.role`

Note

The `authentication.subject.id` message attribute is specified using `${authentication.subject.orig.id}` because `authentication.subject.id` holds the full DName, and the `cn` only needs to be passed to the services.

Test the policy configuration
To test this policy configuration, perform the following steps:

1. In the Policy Studio tree, select **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:8080/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** > **Open**, and browse to `configs.xml` in the first copy of the `fed` directory.
4. Under the **Listeners** > **Management Services** node, select the `/` and the `/configuration/deployments` relative paths, and set the **Path Specify Policy** to the **Protect Management Interfaces (LDAP)** policy.
5. Remove the previously created `LDAP Test` HTTP Services.
6. Close the connection to the file.
7. Copy the `fed` directory back to the Admin Node Manager's `conf` directory.
8. Reboot the Admin Node Manager.
9. 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 embedded ActiveMQ messaging

Overview

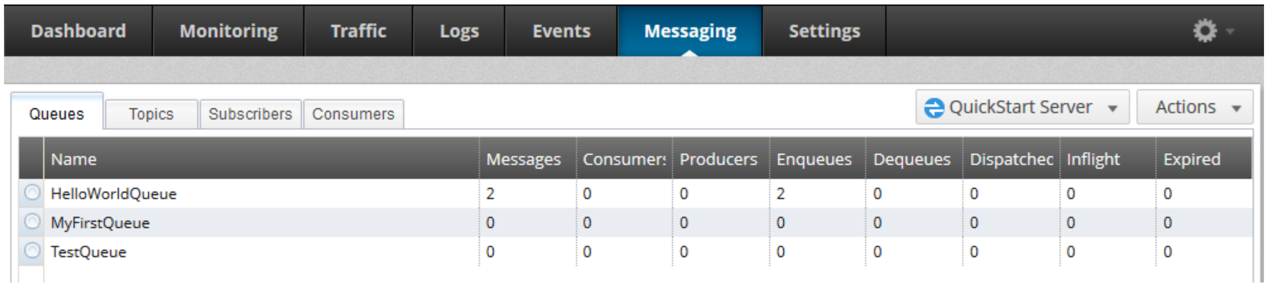
The **Messaging** view in API Gateway Manager enables you to manage the Apache ActiveMQ messaging broker that is embedded in the API Gateway instance. For example, this includes managing JMS message queues, topics, subscribers, and consumers, monitoring server connections, and so on. For more details on Apache ActiveMQ, see http://activemq.apache.org/. For details on the embedded ActiveMQ architecture, see the *API Gateway Concepts Guide*.

Important

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*.

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:



Name	Messages	Consumer:	Producers	Enqueues	Dequeues	Dispatchec	Inflight	Expired
HelloWorldQueue	2	0	0	2	0	0	0	0
MyFirstQueue	0	0	0	0	0	0	0	0
TestQueue	0	0	0	0	0	0	0	0

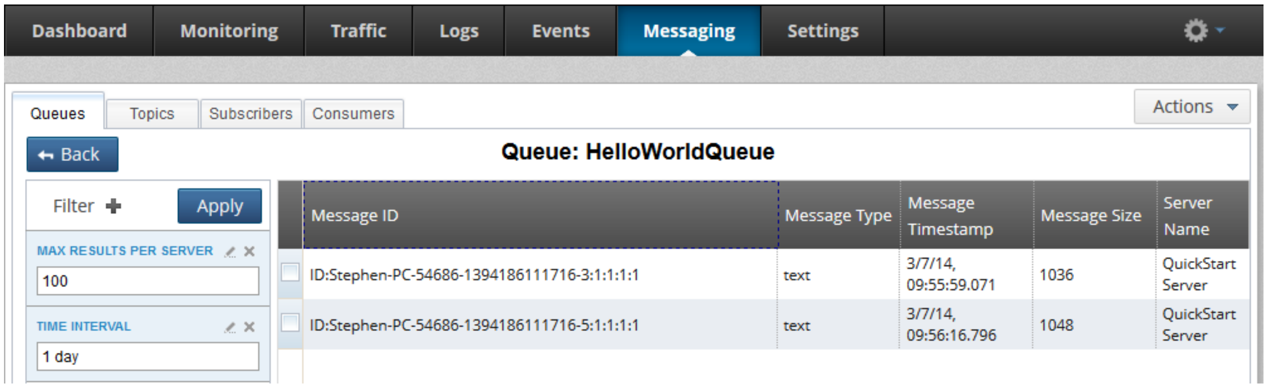
You can use the **Actions** drop-down menu on the right of the screen to perform the following tasks:

- 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:



Queue: HelloWorldQueue

MAX RESULTS PER SERVER: 100

TIME INTERVAL: 1 day

Message ID	Message Type	Message Timestamp	Message Size	Server Name
ID:Stephen-PC-54686-1394186111716-3:1:1:1:1	text	3/7/14, 09:55:59.071	1036	QuickStart Server
ID:Stephen-PC-54686-1394186111716-5:1:1:1:1	text	3/7/14, 09:56:16.796	1048	QuickStart Server

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**:

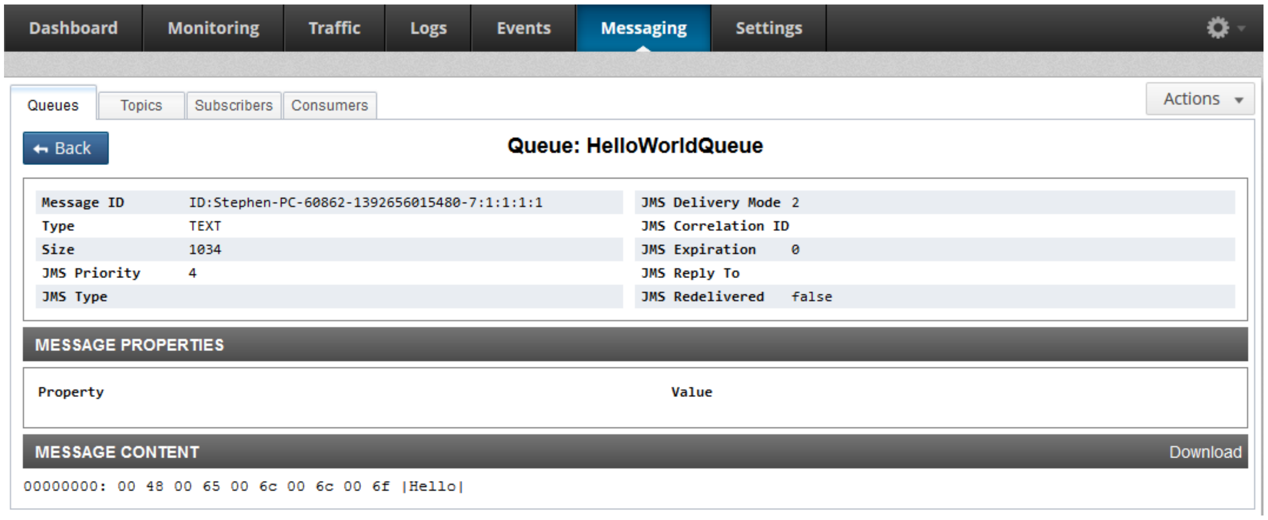
Setting	**Description**
Destination API Gateway	Specifies the required destination API Gateway for the message (for example `APIGateway1`).
Message Type	Specifies the required type of the message (`Text` or `Binary`). Defaults to `Text`.
JMS Reply To	Specifies the optional destination to send a reply 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

Setting	Description
	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.



Queue: HelloWorldQueue

Message ID	ID:Stephen-PC-60862-1392656015480-7:1:1:1:1	JMS Delivery Mode	2
Type	TEXT	JMS Correlation ID	
Size	1034	JMS Expiration	0
JMS Priority	4	JMS Reply To	
JMS Type		JMS Redelivered	false

MESSAGE PROPERTIES

Property	Value

MESSAGE CONTENT Download

```
00000000: 00 48 00 65 00 6c 00 6c 00 6f |Hello|
```

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

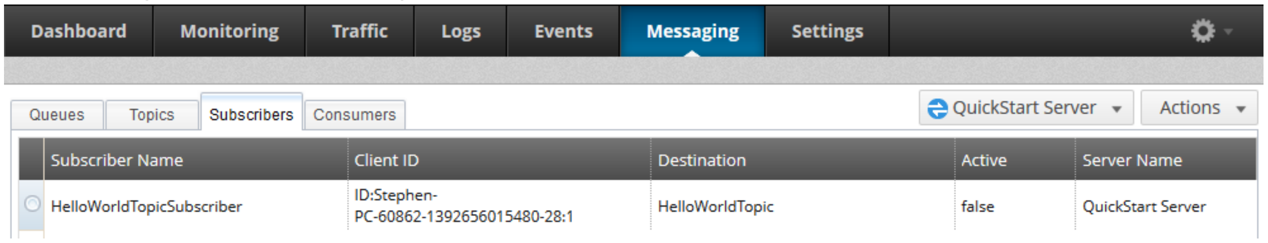
To create a new subscriber, select **Actions** > **Create Subscriber**, and complete the following details:

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 User Guide*.

The following screen shows a newly created subscriber:

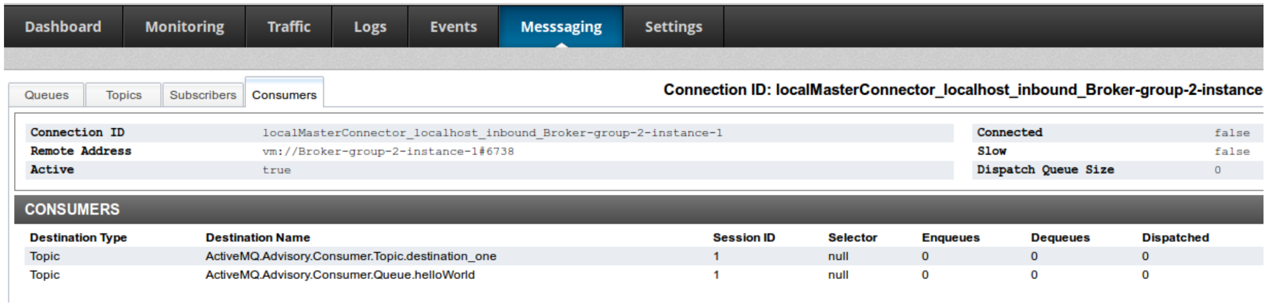


Subscriber Name	Client ID	Destination	Active	Server Name
HelloWorldTopicSubscriber	ID:Stephen-PC-60862-1392656015480-28:1	HelloWorldTopic	false	QuickStart Server

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:



Connection ID: localMasterConnector_localhost_inbound_Broker-group-2-instance

Connection ID	localMasterConnector_localhost_inbound_Broker-group-2-instance-1	Connected	false
Remote Address	vm://Broker-group-2-instance-1#6738	Slow	false
Active	true	Dispatch Queue Size	0

CONSUMERS

Destination Type	Destination Name	Session ID	Selector	Enqueues	Dequeues	Dispatched
Topic	ActiveMQ.Advisory.Consumer.Topic.destination_one	1	null	0	0	0
Topic	ActiveMQ.Advisory.Consumer.Queue.helloWorld	1	null	0	0	0

Monitor services in API Gateway Manager

Overview

This topic explains how to monitor example services using the API Gateway Manager monitoring tools.

Enable monitoring

You must first ensure that traffic monitoring and real-time monitoring are enabled:

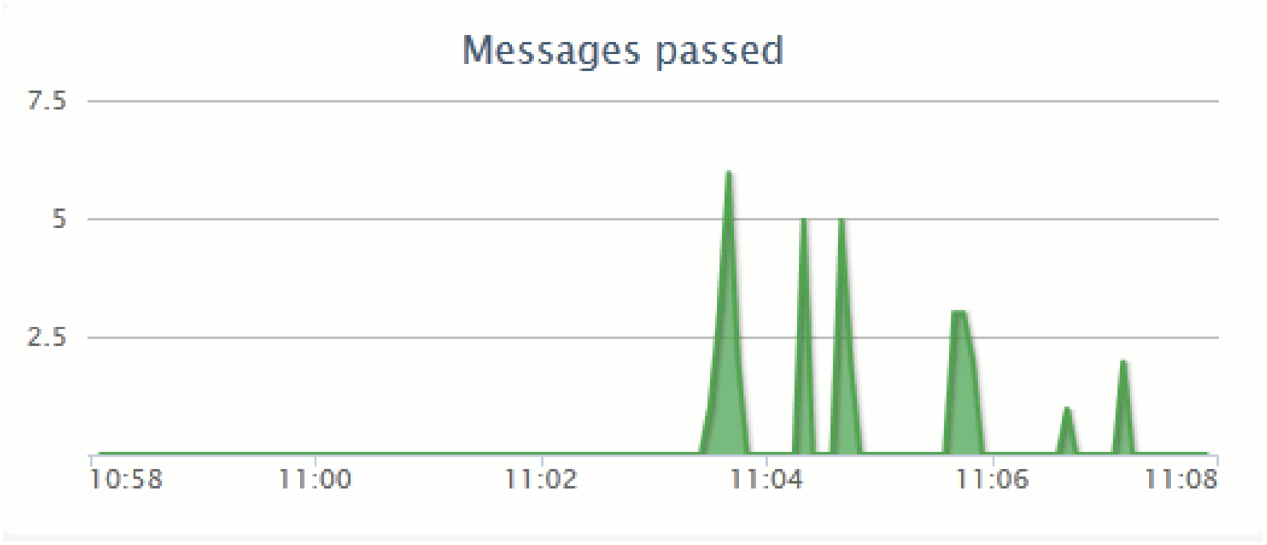
1. In the Policy Studio tree, select the **Settings** node, and select the **Traffic Monitor** tab at the bottom.
2. On the **Traffic Monitor** tab, ensure **Enable Traffic Monitor** is selected.
3. Select the **Metrics** tab at the bottom, and ensure **Enable real time monitoring** is selected.
4. Click the **Deploy** icon in the Policy Studio toolbar to deploy these settings to the API Gateway. Alternatively, press F6.

Important

Traffic monitoring is required for the purposes of this demo. However, 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*.

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

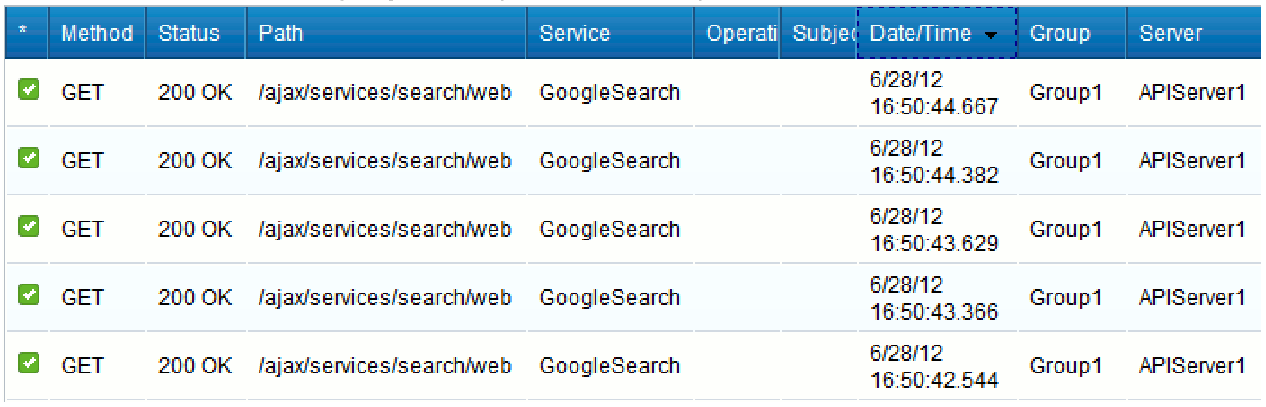
Tester or the Send Request (SR) tool), the message status is displayed in the **TRAFFIC** section.

View message traffic

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	Operati	Subjec	Date/Time	Group	Server
✓	GET	200 OK	/ajax/services/search/web	GoogleSearch			6/28/12 16:50:44.667	Group1	APIServer1
✓	GET	200 OK	/ajax/services/search/web	GoogleSearch			6/28/12 16:50:44.382	Group1	APIServer1
✓	GET	200 OK	/ajax/services/search/web	GoogleSearch			6/28/12 16:50:43.629	Group1	APIServer1
✓	GET	200 OK	/ajax/services/search/web	GoogleSearch			6/28/12 16:50:43.366	Group1	APIServer1
✓	GET	200 OK	/ajax/services/search/web	GoogleSearch			6/28/12 16:50:42.544	Group1	APIServer1

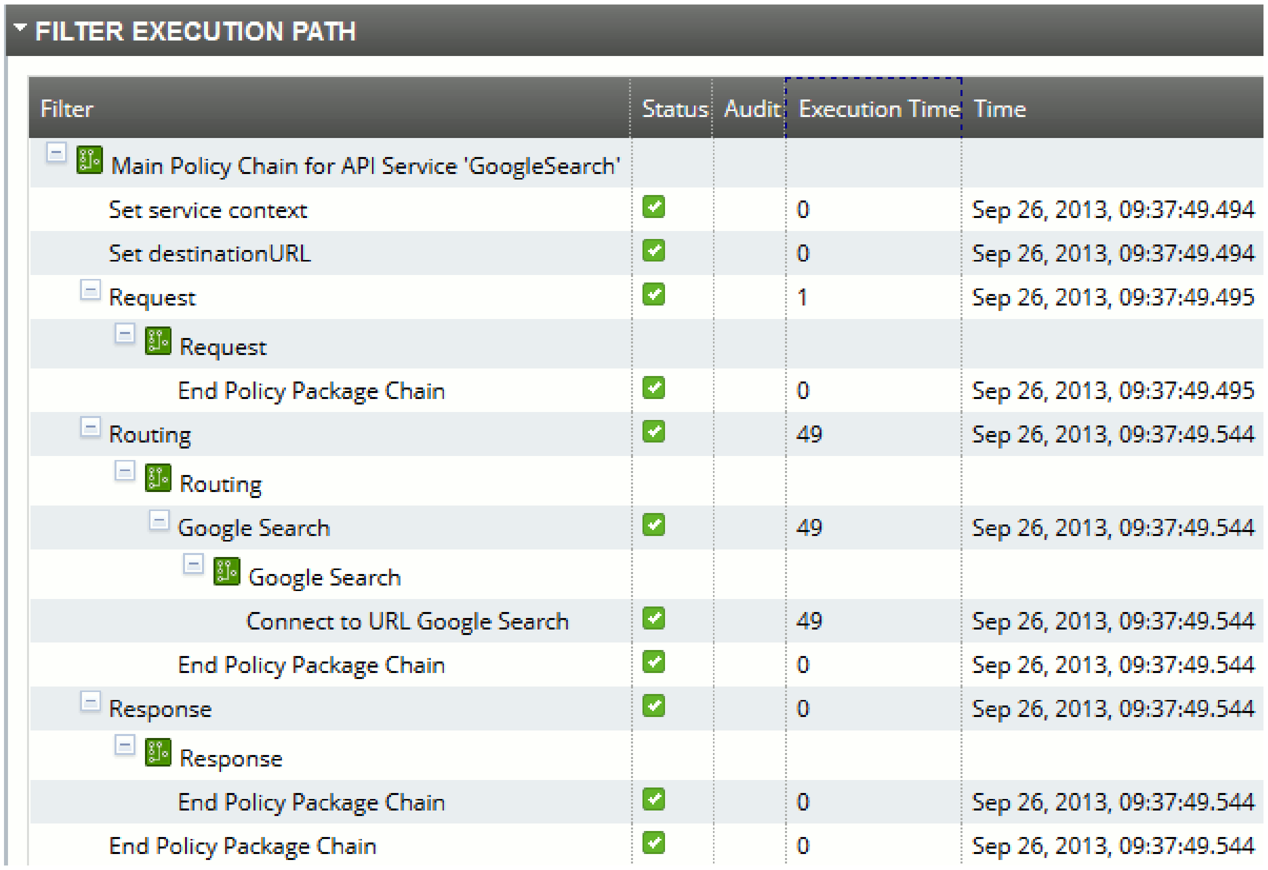
Filter message traffic

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

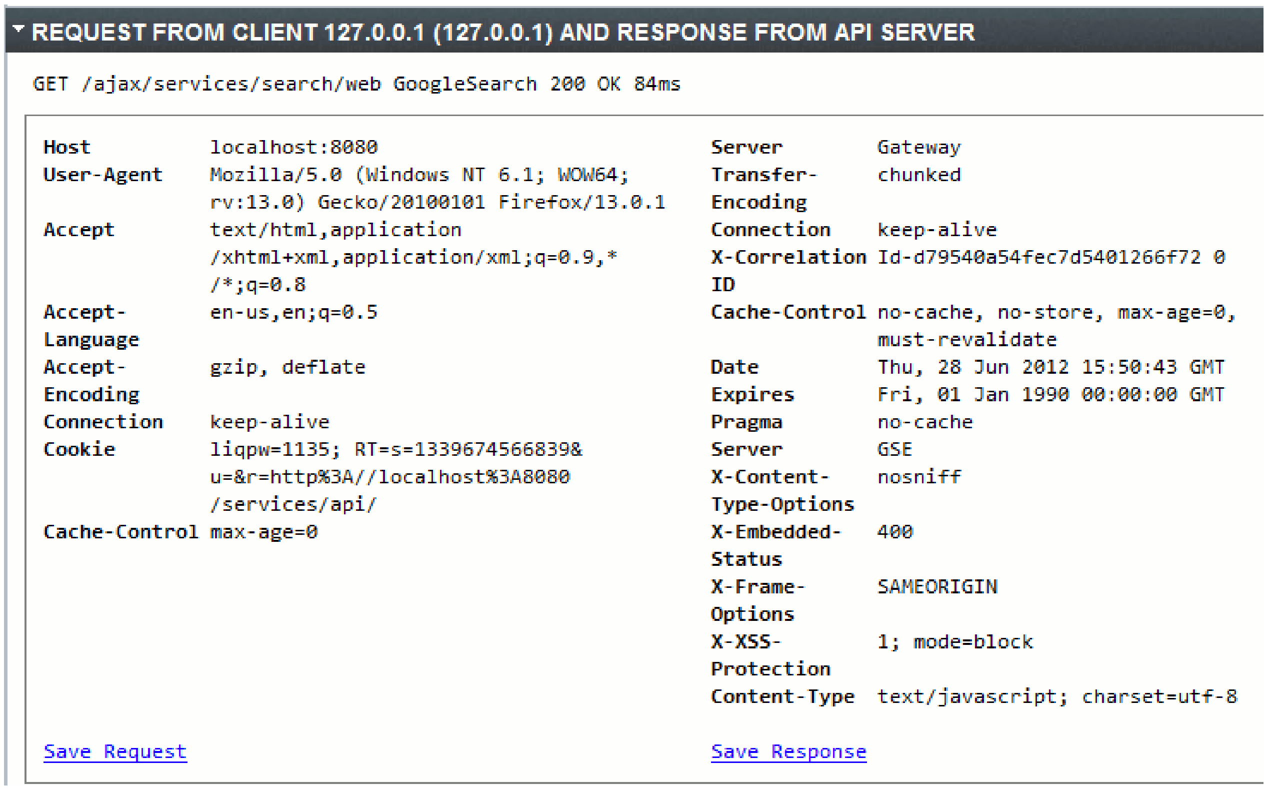
When you click a selected message listed on the **Traffic** tab, this displays the message filter execution path and the contents of each request message and response message. The following example displays the message path for a simple Google Search message:



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
```

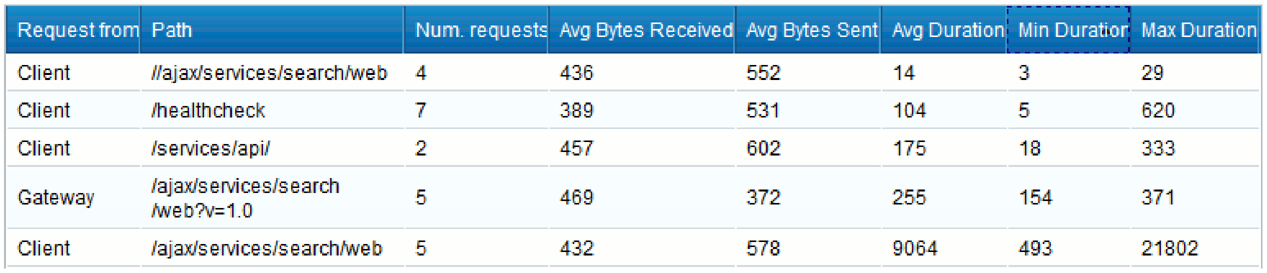
Host	localhost:8080
User-Agent	Mozilla/5.0 (Windows NT 6.1; WOW64; rv:13.0) Gecko/20100101 Firefox/13.0.1
Accept	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language	en-us,en;q=0.5
Accept-Encoding	gzip, deflate
Connection	keep-alive
Cookie	liqpw=1135; RT=s=1339674566839&u=&r=http%3A//localhost%3A8080/services/api/
Cache-Control	max-age=0

Server	Gateway
Transfer-Encoding	chunked
Connection	keep-alive
X-Correlation ID	Id-d79540a54fec7d5401266f72 0
Cache-Control	no-cache, no-store, max-age=0, must-revalidate
Date	Thu, 28 Jun 2012 15:50:43 GMT
Expires	Fri, 01 Jan 1990 00:00:00 GMT
Pragma	no-cache
Server	GSE
X-Content-Type-Options	nosniff
X-Embedded-Status	400
X-Frame-Options	SAMEORIGIN
X-XSS-Protection	1; mode=block
Content-Type	text/javascript; charset=utf-8

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:

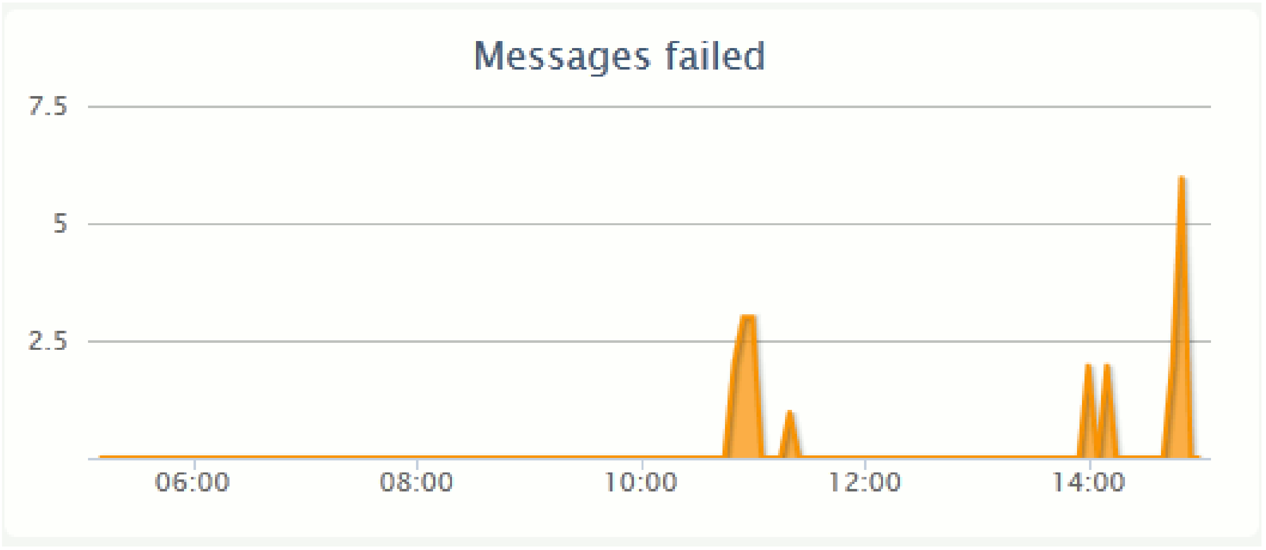


Request from	Path	Num. requests	Avg Bytes Received	Avg Bytes Sent	Avg Duration	Min Duratior	Max Duration
Client	//ajax/services/search/web	4	436	552	14	3	29
Client	/healthcheck	7	389	531	104	5	620
Client	/services/api/	2	457	602	175	18	333
Gateway	/ajax/services/search /web?v=1.0	5	469	372	255	154	371
Client	/ajax/services/search/web	5	432	578	9064	493	21802

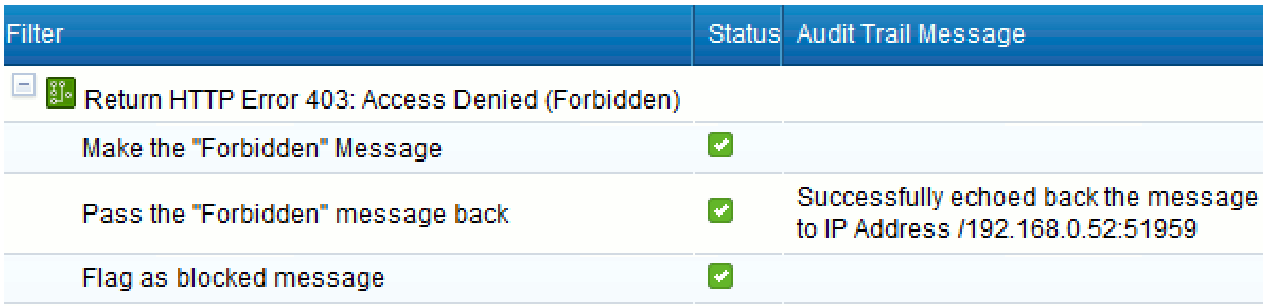
Filter performance statistics
You can click the **Apply** in the left pane to filter the performance statistics displayed based on different criteria. By default, the statistics are grouped by path name, with a time interval of 1 day. You can select different criteria from the **GROUP BY** and **TIME 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:



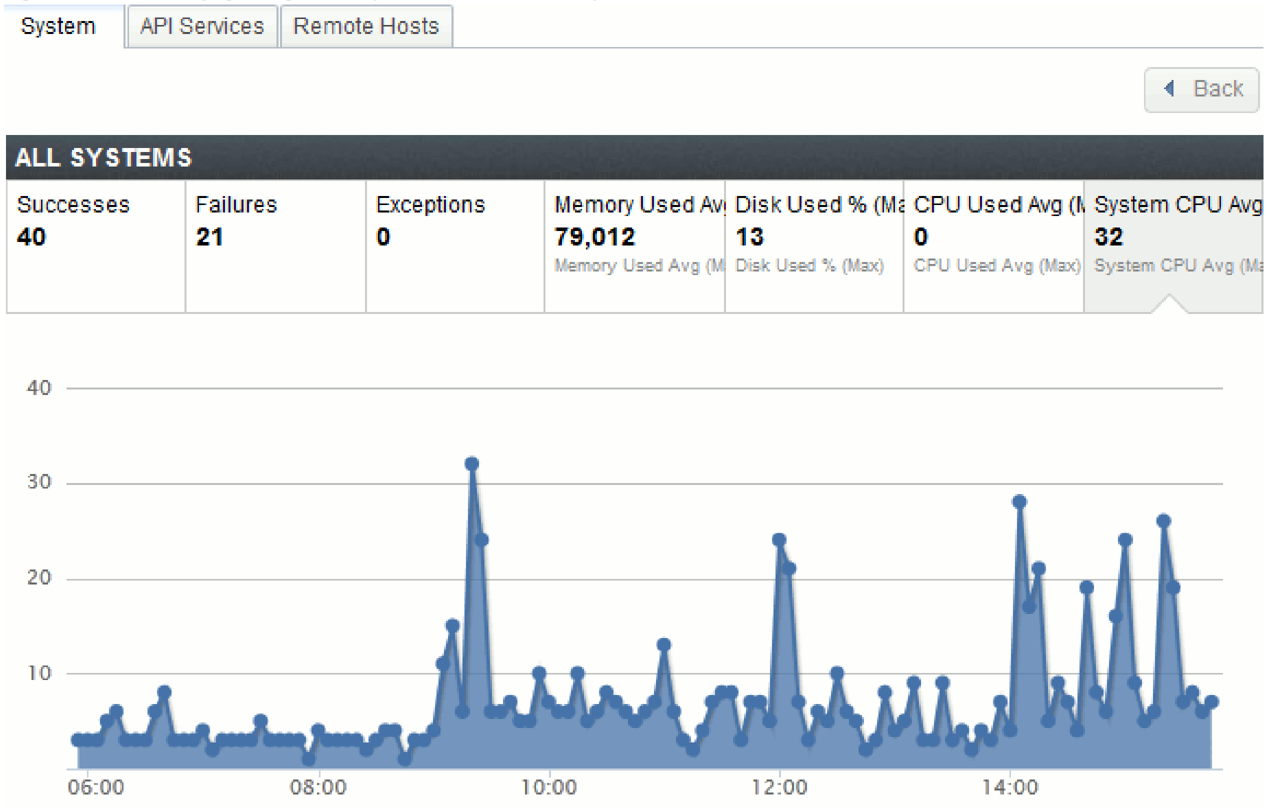
You can click the chart to display the list of failed messages automatically filtered on the **Traffic** tab. Click a message in the list to display the filter execution path and message content. The following example shows the execution path of a malformed message that has been blocked by the API Gateway:



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	✓	

Monitor system data

The **Monitoring** view enables you to monitor **System**, **API Services**, and **Remote Host** statistics. For example, on the **System** tab, when you click a panel in the **ALL SYSTEMS** section at the top of the screen, 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:



Successes	Failures	Exceptions	Memory Used Av	Disk Used % (Ma	CPU Used Avg (M	System CPU Avg
40	21	0	79,012	13	0	32
			Memory Used Avg (M	Disk Used % (Max)	CPU Used Avg (Max)	System CPU Avg (Ma

Configure trace and log settings

You can click the **Settings** button in the toolbar to configure trace, logging, and monitoring settings on-the-fly. These are dynamic settings, so you do not need to refresh or deploy to the API Gateway. For example, the top-level **SYSTEM SETTINGS** enable you to specify whether inbound and outbound transactions, the policy path, and message trace are recorded. You can also select an HTTP interface in the tree on the left to configure the **INTERFACE SETTINGS**, **TRAFFIC MONITORING SETTINGS**, and interface trace level.

Select the API Gateway instance on the left to configure the trace level. Finally, you can also select a Relative Path or Service on the left, and select the **SERVICE SETTINGS**, **TRANSACTION LOGGING LEVEL**, or

TRANSACTION PAYLOAD LOGGING on the right.

For more details on how to configure tracing for the API Gateway, see *Configure API Gateway tracing*. For more details on logging for specific message filters, see the *API Gateway User Guide*.

Monitor and report on services with API Gateway Analytics

This tutorial shows how to monitor an example service using the monitoring tools provided with the API Gateway. API Gateway Analytics is a separately installed component that enables you to monitor services and to generate reports on the stored message traffic history in your domain. For more details, see *Reporting with API Gateway Analytics*. For more details, see the *API Gateway Administrator Guide*.

Configure the API Gateway for API Gateway Analytics

Overview

This topic explains how to configure the API Gateway to store message metrics in the same reports database used by API Gateway Analytics. It also includes how to configure database logging and monitoring settings.

Important

To view API Gateway metrics in API Gateway Analytics, you must configure the API Gateway for use with API Gateway Analytics.

Prerequisites
This topic assumes that you have already installed and configured API Gateway Analytics using the steps described in the *Axway API Gateway Installation and Configuration Guide*.

Connect to the API Gateway

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*.
2. On the Policy Studio welcome page, click the Admin Node Manager server session to make a connection.
3. Specify your connection details (host, port, user name, and password). The default connection URL is:

   ```
   https://localhost:8090/api
   ```

   where `HOST` points to the IP address or hostname of the machine on which the API Gateway is running. For more details, see the *API Gateway User Guide*.
4. In the API Gateway topology view, double-click the **API Gateway** instance to load its configuration.

Configure the database connection

To configure the database connection, perform the following steps:

1. Expand the **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 the same reports database created when API Gateway Analytics was installed. For more details, see the *API Gateway User Guide*.

You can verify that your database connection is configured correctly by clicking the **Test Connection** button on the **Configure Database Connection** dialog. You can also view the contents of your server `.trc` file in the `INSTALL_DIR/trace` directory. For more details on tracing, see *Configure API Gateway tracing*.

Configure the database logging

To configure the database logging settings, perform the following steps:

1. In the Policy Studio tree view, select the **Settings** node.
2. Select the **Transaction Log** tab at the bottom, select the **Database** tab, 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 the same reports database configured when API Gateway Analytics was installed.

Note

If you wish to write the contents of message requests and responses to the database, you must configure the **Log Message Payload** filter for the relevant policies (for example, at the start and end of the policy). For more details, see the *API Gateway User Guide*.

Configure monitoring settings

To configure the API Gateway to monitor traffic and store message metrics in the reports database, perform the following steps:

1. In the Policy Studio tree, select the **Settings** node, and select the **Metrics** tab at the bottom of the screen.
2. Ensure **Enable real time monitoring** is selected.
3. In the **Reports settings** panel, select **Store real time monitoring data for charts/reports**, and specify the database connection that points to the reports database.
4. If you wish to enable monitoring of traffic per API Gateway, select the **Traffic Monitor** tab at the bottom of the screen, and ensure **Enable Traffic Monitor** is selected.

Important

Enabling traffic monitoring has a negative impact on performance. If you wish to maximize performance, do not enable this setting. For more details, see *Traffic monitoring settings*.

Deploy 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 audit trail and message metrics data to the reports database. This database is then queried by API Gateway Analytics to produce reports showing system health, service usage, clients, message size and volume, and so on.

Note

These monitoring settings are API Gateway instance-wide. You can also send metrics data to the database at the interface level. Right-click the HTTP Services Group (for example, **Default Services**), and select **Edit**. To monitor traffic for this Services Group, ensure that **Include in real-time monitoring** is selected. Similarly, you can disable monitoring for this Services Group by unselecting this setting.

Reporting with API Gateway Analytics

Overview

API Gateway Analytics monitors, records, and reports on the history of message traffic between API Gateway instances and various services, remote hosts, and clients running in a managed domain. You can use API Gateway Analytics to monitor traffic and perform root cause analysis at the level of the domain, API Gateway instance, service, remote host, and client. You can also filter the display based on any selected time period. For example, this defaults to the last 7 days, but you can specify any date range.

This topic describes how to use the views available in the API Gateway Analytics web-based interface to monitor your domain. It assumes that you have already performed the steps described in *Configure the API Gateway for API Gateway Analytics*.

Launch API Gateway Analytics

To launch API Gateway Analytics, perform the following steps:

1. Start the API Gateway Analytics server using the `analytics` script in the `/bin` directory of your API Gateway Analytics installation.
2. Using the default port, connect to the API Gateway Analytics interface in a browser at the following URL:

   ```
   http://HOST:8040/
   ```

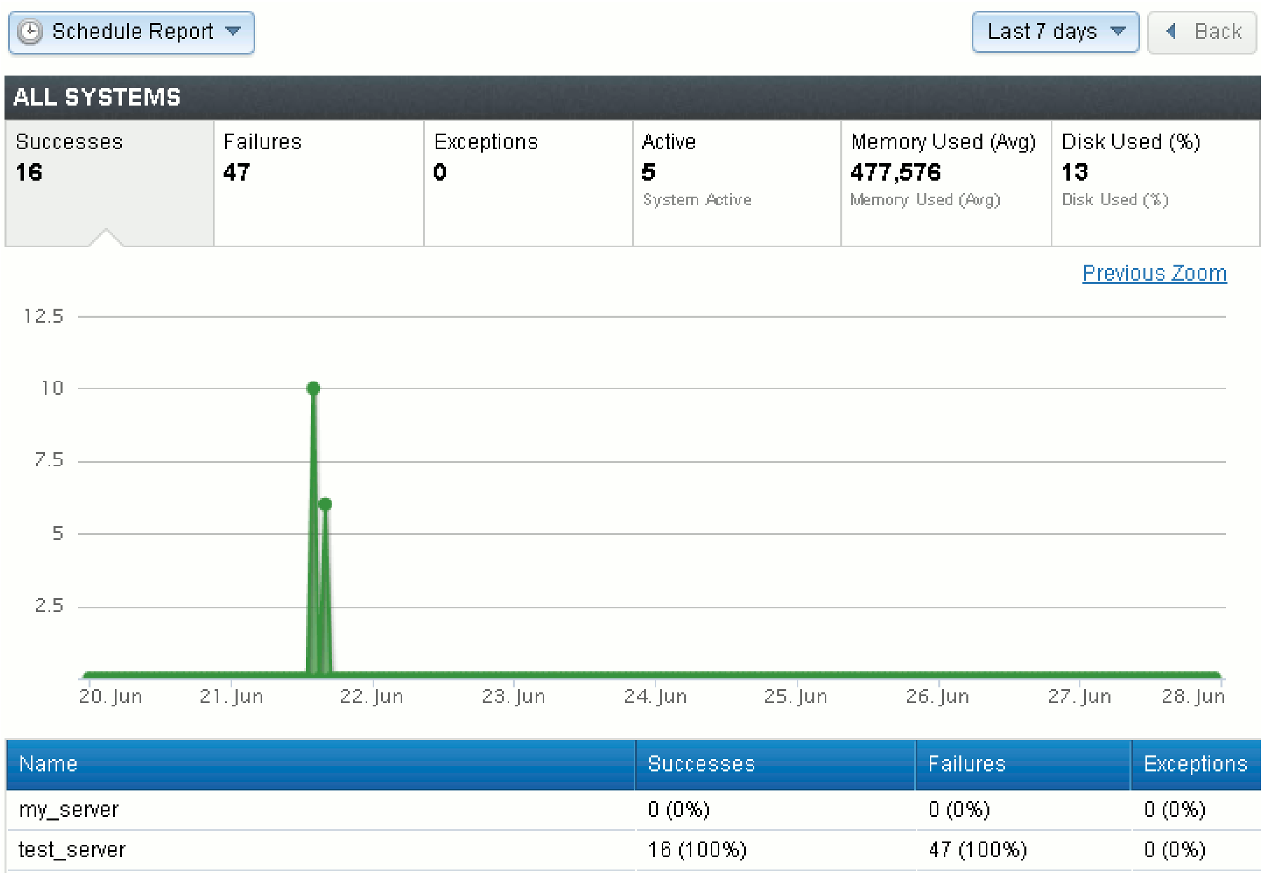
   where `HOST` points to the IP address or hostname of the machine on which API Gateway Analytics is installed.
3. Log in using the default `admin` user with password `changeme`. You can edit this user in Policy Studio using the **Users** interface from the Policy Studio tree view.

System

In the API Gateway Analytics **System** view, click a panel in the **ALL SYSTEMS** section at the top to display graph for the selected system-level metrics below. For example, the available metrics include the following:

- **Successes**
- **Failures**
- **Exceptions**
- **Active**
- **Memory Used (Avg)**
- **Disk Used (%)**

The following example shows messages successfully sent displayed in a simple domain with two API Gateway instances:



Successes	Failures	Exceptions	Active	Memory Used (Avg)	Disk Used (%)
16	47	0	5	477,576	13
			System Active	Memory Used (Avg)	Disk Used (%)

Name	Successes	Failures	Exceptions
my_server	0 (0%)	0 (0%)	0 (0%)
test_server	16 (100%)	47 (100%)	0 (0%)

The table at the bottom shows all the API Gateway instances that are sending monitored traffic to protected services, clients, and remote hosts in your domain. You can click an API Gateway instance in the table to drill down and view graphs for the selected instance (for example, **SYSTEM - Test Server**). You can click the **Back** button at the top right to return to the **ALL SYSTEMS** view.

Note

Each of the API Gateway instances must already be configured to store message metrics in the same database that API Gateway Analytics is configured to read from. This enables API Gateway Analytics to obtain the API Gateway metrics and logs from the database for display. For more details, see the chapter on configuring the database in the *Axway API Gateway Installation and Configuration Guide*.

API Services

The **API Services** view shows metrics for services that are virtualized by API Gateway instances in your domain. You can virtualize services using the web-based API Gateway Manager tool and the Axway Policy Studio. For details, see the *API Gateway User Guide*.

In the **API Services** view, click a panel in the **ALL API SERVICES** section at the top to display graph for the selected service-level metric below. For example, the available metrics include the following:

- **Messages**
- **Successes**
- **Failures**
- **Exceptions**

- **Processing Time (Min)**
- **Processing Time (Max)**
- **Processing Time (Avg)**

The table at the bottom shows all the services protected by API Gateway instances in your domain. You can click a service in the table to drill down and view graphs for the selected service (for example, **API SERVICE - Google Search**). You can click the **Back** button at the top right to return to the **ALL API SERVICES** view.

Note

A service must have been sent a message before it is displayed in the **API Services** view.

Remote Hosts

The **Remote Hosts** view displays metrics for all the remote hosts that have been configured in your domain. It shows details such as the number of message transactions that have been sent to this remote host, together with the total number of bytes sent to and received from this host.

In the **Remote Hosts** view, click a panel in the **ALL REMOTE HOSTS** section at the top to display graph for the selected remote host metric below. For example, the available metrics include the following:

- **Transactions**
- **Volume Bytes (In)**
- **Volume Bytes (Out)**
- **Response Time (Min)**
- **Response Time (Max)**
- **Response Time (Avg)**

The table at the bottom shows all the remote hosts connected to by API Gateway instances in your domain. You can click a remote host in the table to drill down and view graphs for the selected remote host (for example, **REMOTE HOST - stockquote.com:80**). This also displays which services have been connecting to the remote host. You can click the **Back** button at the top right to return to the **ALL REMOTE HOSTS** view.

Clients

This **Clients** view displays metrics for all clients that have been authenticated for services in your domain. In the **Clients** view, click a panel in the **ALL CLIENTS** section at the top to display graph for the selected clients metric below. For example, the available metrics include the following:

- **Messages**
- **Successes**
- **Failures**
- **Exceptions**

The table at the bottom shows all clients that have been authenticated for services in your domain. You can click a client in the table to drill down and view graphs for the selected client (for example, **CLIENT - test.client**). This also displays which services have been connected to by the client. You can click the **Back** button at the top right to return to the **ALL CLIENTS** view.

Audit Trail

The **Audit Trail** view enables you to filter the transaction log messages generated by API Gateway instances in your domain. You can filter log messages by clicking the **Search** button on the right in the toolbar. The **Query Editor** enables you to create a query to filter log messages by details such as time period, severity level, filter type, filter name, and log message text. When you have added your search criteria, click **Search** at the bottom to run the query. You can also save the query for later use.

When you click **Search**, the log messages that match the search criteria specified in the query are displayed in the table. For example, the details displayed in the table include the log message text, API Gateway name, alerts, and time. You can also double-click an item in the list for more details (for example, transaction ID, filter category, and filter name).

Reports

API Gateway Analytics uses message metrics stored in the centralized database by the API Gateway instances running in your domain. The API Gateway stores metrics for the virtualized services that it exposes, and for the services, and client and remote host connections that it protects. API Gateway Analytics can generate usage reports and charts based on this data, and display them in the **Reports** view.

You can create reports by selecting the appropriate button in the API Gateway Analytics toolbar, and clicking the **Schedule Report** button. For example, to generate reports on API Services, perform the following steps:

1. Click the **API Services** button in the API Gateway Analytics toolbar.
2. Click the **Schedule Report** button at the top left.
3. Complete the following fields:

Schedule	Select when to schedule the report. Defaults to `Now`.
Time	If you do not select to run the report now, select the time of day to run the report.
On	If you do not select to run the report now or daily, select the day of week on which to run the report.
From	Select the period of time for the report. Defaults to `Last 7 days`.
Email	Enter the email address to which to send the report.

4. Click **Create**.

Similarly, you can follow the same sequence of steps to generate reports in the **System**, **Remote Hosts**, and **Clients** views.

Custom reporting

API Gateway Analytics enables you to configure custom reports to provide flexible reporting to suit various requirements. This includes viewing any available metric for each target report type, grouping metrics, and filtering metrics.

Custom Report Types
Custom reporting enables you to produce the following types of reports:

- API service usage
- API service usage for selected API service(s)
- Client usage

- Client usage for selected API service(s)
- Client(s) that used an API service
- API service(s) that a client used

Grouping and Filtering Metrics
Custom reporting enables you to group by or filter by any or all of the following:

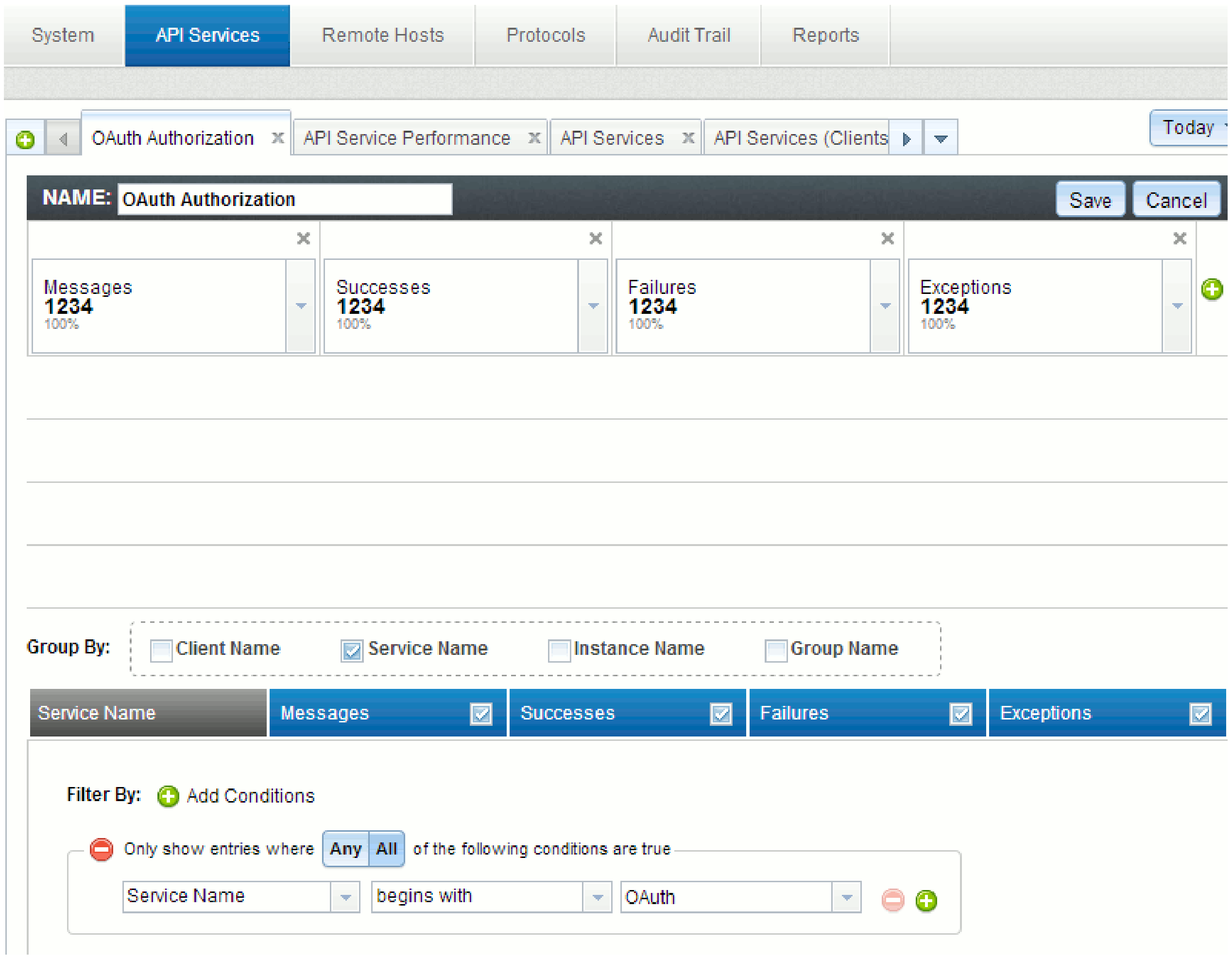
- Client Name
- Service Name
- Instance Name
- Group Name

Note

The group-by mechanism only applies to the data table below the report chart. The chart remains the same.

These filtering and grouping mechanisms enable you to answer questions such as **Client(s) that used an API Service**, or **API service(s) that a client used**. For example, to show clients that used `WebService1`, you can create a custom report that groups by **Client Name** and filters where **Service Name** is `WebService1`.

Defining custom reports
API Gateway Analytics enables you to define custom report names, the metrics to be displayed and charted, and what to display on drill through. For example, the following screen shows a custom report named **OAuth Authorization**, which is grouped by **Service Name**, and filtered by service name beginning with `OAuth`.



NAME: OAuth Authorization

Messages	Successes	Failures	Exceptions
1234	1234	1234	1234
100%	100%	100%	100%

Group By: [ ] Client Name [✓ Service Name] [ ] Instance Name [ ] Group Name

Service Name	Messages ✓	Successes ✓	Failures ✓	Exceptions ✓

Filter By: Add Conditions

Only show entries where [✓ Any] All of the following conditions are true

Service Name | begins with | OAuth

Configure scheduled reports

Overview

You can schedule reports to run on a regular basis, and have the results emailed to the user 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 reports, right-click the **Listeners** > **API Gateway 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 **External Connections** node in the Policy Studio tree view. For more details on creating database connections, see the *API Gateway User 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 username and password used to connect to the report generator process. Defaults to the values entered using the `configureserver` script (for example, `admin/changeme`).

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 `configureanalytics` script.

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.

Purge the reports database

Overview

You can use the `dbpurger` command to connect to your reports database and to purge old data. This command also enables you to retain a specified amount of data, and to archive all data.

For details on configuring the connection to your reports database, see the *API Gateway Installation and Configuration Guide*.

Run the dbpurger command

You can run the `dbpurger` command from the following directory:

Windows	`INSTALL_DIR\analytics\Win32\bin`
Linux/UNIX	`INSTALL_DIR/analytics/posix/bin`

Command 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 commands

This section shows examples of running `dbpurger` in default interactive mode and of specifying command-line options.

Running 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 command-line options to dbpurger

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.bat --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.bat --dbname="Default Database Connection" --archive --out=archive.dat
```

Configure a DNS service with wildcards for virtual hosting

Overview

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 User 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 and UNIX 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/.

UNIX 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` to 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.
`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 the section called "Configure domain zone files").
`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.

General settings

Overview

The top-level **General** settings screen in Policy Studio enables you to set global configuration settings to optimize the behavior of the API Gateway for your environment.

To configure the **General** settings, in the Policy Studio main menu, select **Tasks** > **Manage Gateway Settings** > **General**. Alternatively, 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 the 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 **Default Settings** screen:

Setting	**Description**
Trace level	Enables you to set the trace level for the API Gateway at runtime. Select the appropriate option from the **Trace Level** drop-down list. Defaults to `INFO`.
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, the API Gateway closes the connection. This guards against a host closing the connection in the middle of sending data. For example, if the host's network connection is pulled out of the machine while in the middle of sending data to the API Gateway. When the 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. You can configure this setting on a per-host basis using the **Remote Hosts** interface. For more details, see the *API Gateway User Guide*.
Date format	Configures the format of the date for the purposes of tracing, logging, and reporting. Defaults to `MM.dd.yyyy HH:mm:ss,SSS`. For more information, see `http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html`
Cache refresh interval	Configures the number of seconds that the server caches data loaded from an external source (external database, LDAP directory, and so on) before refreshing the data from that source. The default value is `5` seconds. If you do not wish any caching to occur, set this value to `0`.
Idle timeout	The API Gateway supports HTTP 1.1 persistent connections. The **Idle Timeout** specified in milliseconds is the time that the API Gateway waits after sending a message over a persistent connection before it closes the connection.

Setting	Description
	Typically, the host tells the API Gateway that it wants to use a persistent connection. The 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, the 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 User 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 the API Gateway can allocate to each request. This setting helps protect against denial of service caused by undue pressure on memory. **Tip** 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. You also can configure this setting at the HTTP/S interface level. For more details, see the *API Gateway User Guide*.
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 the API Gateway.
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 the 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 the 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 the API Gateway instance can accept from peers. The available content

Setting	Description
	encodings include `gzip` and `deflate`. Defaults to no context encodings. For more details, see the *API Gateway User Guide*.
Output encodings	Click the browse button to specify the HTTP content encodings that the API Gateway instance can apply to outgoing messages. The available content encodings include `gzip` and `deflate`. Defaults to no context encodings. For more details, see the *API Gateway User 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.
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 the 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 a API Gateway policy. An example correlation ID value is `Id-54bbc74f515d52d71a4c0000`. This setting is selected by default.

MIME/DIME settings

Overview

The MIME/DIME settings list a number of default common content types that are used when transmitting MIME messages. You can configure the 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 the 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 this filter, see the *API Gateway User Guide*.

Configuration

To configure the MIME/DIME settings, in the Policy Studio main menu, select **Tasks** > **Manage Gateway Settings** > **General** > **MIME/DIME**. Alternatively, in the Policy Studio tree, select the **Server Settings** node, and click **General** > **MIME/DIME**. To confirm updates to these settings, click **Apply changes** at the bottom right of the screen.

The MIME/DIME 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/DIME Type** dialog, enter the new content type in the **MIME or DIME 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

Overview

The 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 **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 the API Gateway. In a similar manner to the way in which the API Gateway handles WSSE namespaces, the 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. The 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 the 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, the API Gateway allows the user to specify other attributes that should be supported in this manner. By default, the 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 the 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 the API Gateway.

The API Gateway attempts to identify WS Security blocks belonging to 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 the 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.

HTTP Session settings

Overview

The **HTTP Session** settings enable you to configure session management settings for the selected cache. For example, you can configure the period of time before expired sessions are cleared from the `HTTP Sessions` cache, which is selected by default.

To configure HTTP session settings, select the **Server Settings** node in the Policy Studio tree, and click **General** > **HTTP Session**. Alternatively, in the Policy Studio main menu, select **Tasks** > **Manage Gateway 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 to use. 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 **Libraries** node in the Policy Studio tree. For more details, see the *API Gateway User Guide*.

Clear Expired Sessions Period:
Enter the number of seconds before expired sessions are cleared from the selected cache. Defaults to `60`.

Transaction Log settings

Overview

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.

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 Log**. Alternatively, in the Policy Studio main menu, select **Tasks** > **Manage Gateway Settings** > **Logging** > **Transaction Log**. To confirm updates to these settings, click **Apply changes** at the bottom right of the screen. 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 the **Enable logging to file** checkbox. You can configure the following fields:

- **File Name**:
  Enter the name of the text-based file that the API Gateway logs to. By default, the log file is called `transactionLog`.
- **File Extension**:
  Enter the file extension of the log file in this field. By default, this has a `.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 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.
- **`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.

Log to XML File

To configure the API Gateway to log to an XML file, click the **XML File** tab, and select the **Enable logging to XML file** checkbox. 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 and Configuration 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 User Guide*.
- **Signing Key**:
  You can sign log messages stored in the database to ensure that they are not tampered with. Click the **Signing Key** button to open the list of certificates in the Certificate Store. You can then 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 the section called “Log to Text File”.

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 drop-down list.
- **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 the section called “Log to Text File”.

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** checkbox. For details on how to use the **Format** field to configure the format of the log message, see the section called “Log to Text File”.

Access Log settings

Overview

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 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.

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:

```
%h %l %u %t "%r" %s %b
```

The log format strings in this example are explained as follows:

%h	Remote hostname or IP address.
%l	Remote logical username.
%u	Remote user that was authenticated (for example, Distinguished Name of a certificate).
%t	Date and time of the request in Common Log Format.
%r	First line of the request that originated at the client.
%s	HTTP status code returned to the client in the response.
%b	Bytes sent, excluding HTTP headers.

The following extract from the `access.log` file illustrates the resulting log format:

```
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
................
................
................
```

For more details on Apache HTTP Server Access Log formats, see the following:

- http://httpd.apache.org/docs/current/logs.html
- http://httpd.apache.org/docs/current/mod/mod_log_config.html#formats

Configure the Access Log

To configure the Access Log in the Policy Studio tree, select the **Server Settings** node, and click **Logging** > **Access Log**. Alternatively, in the Policy Studio main menu, select **Tasks** > **Manage Gateway Settings** >

Logging > **Access Log**. To confirm updates to these settings, click **Apply changes** at the bottom right of the screen.

You can configure the following fields to enable the server to write an Access Log to file:

Enable Apache Access File Logger:
Select whether to configure the API Gateway instance to start writing event data to the Access Log. This setting is disabled by default.

Pattern:
Enter the Access Log file pattern. 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 the section called “Log Format”.

Base Log File Name:
Enter the name of the Access Log file in this field. 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`.

Directory Name:
Enter the directory for the Access Log file. Defaults to the `logs/access` directory of your product installation.

Log File Extension:
Enter the file extension for the log file. Defaults to `.log`.

Max Files:
Specify the number of log files that are stored. Defaults to `20`.

Max Log File Size:
Specify the maximum size that the log file is allowed reach before it rolls over to a new file. Defaults to `1000` kilobytes.

Roll Over Daily:
Select whether to roll over the log file at the start of each day. This is enabled by default.

Note

These settings configure the Access Log at the API Gateway level. You can configure the Access Log at the service level on a relative path. For more details, see the *API Gateway User Guide*.

Embedded ActiveMQ settings

Overview

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

To configure Embedded ActiveMQ settings, select the **Server Settings** node in the Policy Studio tree, and click **Messaging** > **Embedded ActiveMQ**. Alternatively, in the Policy Studio main menu, select **Tasks** > **Manage Gateway Settings** > **Messaging** > **Embedded ActiveMQ**. To apply updates to these settings, click **Apply changes** at the bottom right of the screen.

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, see the *Deployment and Promotion 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`.

SSL settings

Configure the following settings to secure the communication between multiple embedded ActiveMQ brokers:

Enable SSL:
Specifies whether to use Secure Sockets Layer (SSL) to secure the communication between ActiveMQ brokers.

Server Cert:
When **Enable SSL** is set, 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 certificates and keys*.

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 certificates and keys*.

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 certificates and keys*.

Authentication settings

Configure the following to specify authentication settings between multiple embedded ActiveMQ brokers:

Authenticate broker and client connections with the following policy:
When username/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 pre-configured policy in the dialog.

Username:
Specifies the username 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.

Traffic monitoring settings

Overview

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 **Sever Settings** node in the Policy Studio tree, and click **Monitoring** > **Traffic Monitor**. Alternatively, in the main menu select **Tasks** > **Manage Gateway Settings**, and click **Monitoring** > **Traffic Monitor**. To confirm updates to these settings, click **Apply changes** 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.

Important

Traffic monitoring may have a negative impact on performance. If you wish to maximize performance, you can disable traffic monitoring.

Configuration

You can configure the following **Traffic Monitor** settings:

Enable Traffic Monitor:
Select whether to enable the web-based Traffic Monitor tool. 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 the location relative to the API Gateway instance directory, which is the location where the server instance runs (for example, `<install-dir>/apigateway/groups/<group-id>/<instance-id>`). 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	Select whether to record the received payload data for the message

transactions	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 the *API Gateway User Guide*.

Transaction Expiry Settings:
The following settings configure when a new transaction file is created. This occurs when the maximum transaction file size is exceeded or when the maximum number of transactions is reached (whichever occurs first).

Target transaction file size	Enter the target size of the transaction file, and select the units from the list. Defaults to `8` MB. When this limit is exceeded, no new transactions are added to the file, and a new file is created. **Note** Transaction files may exceed this limit due to continuing traffic on existing transactions. The current file is closed only when all current transactions using the file are complete. This means that the specified file size may be exceeded, but no more transactions are stored in the file when this limit is reached. For example, with the default of 8 MB, assume the current file size is 7 MB, and a 100 MB transaction is being logged, making a total of 107 MB. No more transactions are logged in this file, and a new file is created.
Maximum number of transactions per file	Enter the maximum number of transactions stored in the file. Defaults to `100000` transactions. When this limit is reached, no new transactions are written. The current file is closed when all current transactions using the file are completed.
Maximum number of transaction files	Enter the maximum number of transaction files to store on disk at any one time. Defaults to `128`. When this limit is reached, old files that have no open transactions are purged. **Note** The number of transaction files may be exceeded if the oldest file still has open transactions.

Real-time monitoring metrics

Overview

You can configure real-time monitoring metrics at the API Gateway instance level. For example, these enable you to specify monitoring of the message content. API Gateway Manager and API Gateway Analytics use this data to display graphical reports in their web-based interfaces. In addition, for API Gateway Analytics, you can also specify the database to which the instance writes metrics data.

To configure real-time monitoring metrics, select the **Server Settings** node in the Policy Studio tree, and click **Monitoring** > **Metrics**. Alternatively, in the main menu select **Tasks** > **Manage Gateway Settings**, and click **Monitoring** > **Metrics**. To confirm updates to these settings, click **Apply changes** at the bottom right of the screen.

Configure metrics settings

The following settings are available:

Enable real time monitoring:
This enables real-time monitoring globally for the API Gateway. This setting must be selected to monitor traffic processed by the API Gateway, and is enabled by default. To disable real-time monitoring, disable this setting.

Important

Enabling real-time monitoring has a negative impact on performance. If you wish to maximize performance, disable this setting.

Configure reports settings

Store real time monitoring data for charts/reports:
If you wish real-time monitoring data to be written to a database, select this setting. This enables the following fields.

Use the following database:
Click the button on the right to select a database in which you wish to store the metrics data. If you have already configured the database connection, you can select it in the tree. To add a database connection, right-click the **Database Connections** tree node, and select **Add DB Connection**. Alternatively, you can add database connections under the **External Connections** node in the Policy Studio tree view. For more information on configuring database connections, see the *API Gateway User Guide*.

Note

API Gateway Analytics must be configured to read metrics data from the database that the API Gateway is configured to write the metrics data to. For more details, see the chapter on configuring the database in the *Axway API Gateway Installation and Configuration Guide*.

Store specified message attribute values:
Select this setting to store the values of specified messages attributes in the database. Click the **Add** button, enter the appropriate message attribute name in the **Value** field, and click **OK**. Repeat to add multiple

message attributes.

When specified message attributes are present in a message running through the API Gateway, their values are stored the database. For example, you could specify `my.example.attribute` to be stored in the database. For any message running through the API Gateway that has the `my.example.attribute` attribute set, the message ID, attribute name, and the value (for example, `My example message subject`) are stored in database. You could then use this information to view reports for messages with specific message subjects or IDs.

The specified attributes add metadata to the audit trail for a transaction, which can be queried in the API Gateway Analytics console. A common use case is when the group or organization of an authenticated user is stored and queried for a transaction. In this case, the message attribute on the white board that contains the user's organization should be stored with the audit trail on completion of the transaction. In this way, the additional attributes to be stored with a transaction are then available to searched on in the audit trail in API Gateway Analytics

CPU, Memory, Disk polling interval:
Specifies the interval in seconds when polling CPU use, free memory, and disk settings used in system monitoring. Defaults to `3` seconds.