Oracle® Fusion Middleware

Oracle API Gateway Concepts Guide

11g Release 2 (11.1.2.4.0)

July 2015

ORACLE

Oracle API Gateway Concepts Guide, 11g Release 2 (11.1.2.4.0)

27 July 2015

Contents

Preface

This document provides an overview of the API Gateway and describes its main concepts, features, and architecture.

Who should read this document

The intended audience for this document is API Gateway architects and evaluators, and all users who are new to API Gateway (for example, policy developers or system administrators). For details on installing API Gateway, see the *API Gateway Installation Guide*.

How to use this document

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

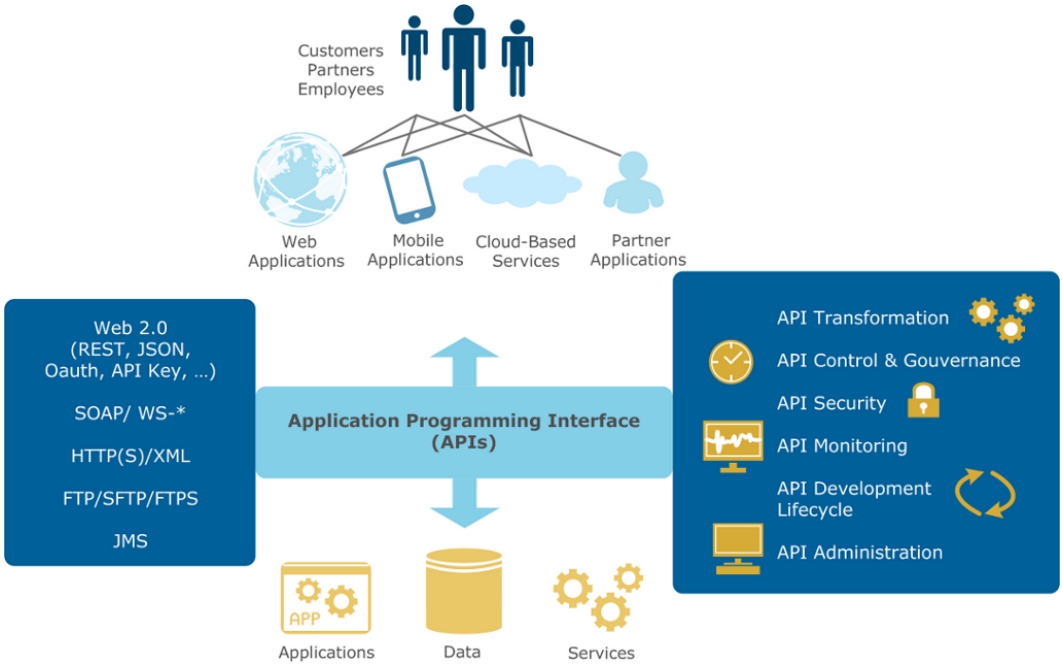
- *Introduction to API Gateway* on page 6 provides an overview of the services and infrastructure provided by API Gateway, and describes the main user roles and high-level functionality.
- *API Gateway tools* on page 14 introduces each of the main API Gateway component tools.
- *API Gateway architecture* on page 22 explains the main concepts and components in the API Gateway architecture, and shows example deployment scenarios.
- *API Gateway documentation* on page 29 shows where to look in the API Gateway documentation library for more details.

Introduction to API Gateway

1

Overview

Oracle API Gateway manages, delivers, and secures enterprise APIs, applications, and consumers. The following overview diagram shows the range of transports and protocols supported by API Gateway on the left, and the services that it provides on the right:



API Gateway services

The main services supported by Oracle API Gateway are described in this section.

API transformation

The API transformation features include the following:

- API virtualization and mediation
- Wide range of protocols, data formats, and standards

- Bi-directional transformation (for example, REST-to-SOAP, XML-to-JSON, and HTTP-to-JMS)

API control and governance

The API control and governance features include the following:

- Service Level Agreement (SLA) monitoring and enforcement
- Quota management, traffic throttling, and load balancing
- Content-based routing, blocking, and processing
- Auditing of transactions

API security

The API security features include the following:

- Protect APIs at all levels (interface, access, and data)
- Authentication and authorization
- Identity mediation and integration with IDM platforms
- Data monitoring, redaction, encryption, and signing
- Key and certificate management

API monitoring

The API monitoring features include the following:

- Real-time API monitoring, with alerting based on errors, exceptions, and thresholds
- Configurable logging of API transaction data
- Analyze API use for insight and trends
- Automated generation and delivery of reports

API development lifecycle

The API development features includes the following:

- Manage API lifecycle from creation to end-of-life
- Drag-n-drop policy creation with intuitive flow chart metaphor
- Extensive library of pre-built policy rules
- Interactive API testing tool
- Promotion between environments

API administration

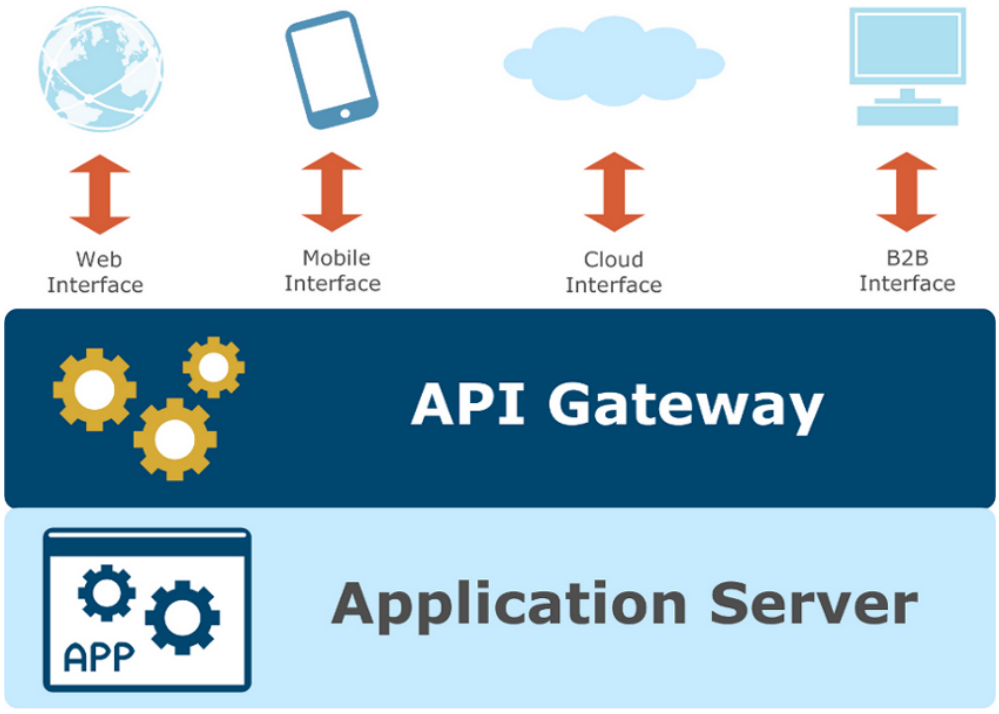
The API administration features include the following:

- Manage all aspects of the daily API operations
- Transaction management
- Tracing and debugging
- OAuth client management
- Managing JMS-based messaging

For more details, see *API Gateway features* on page 10.

API Gateway is core infrastructure

API Gateway does for APIs what the application server does for applications. This API Gateway role as core application infrastructure is shown as follows:



The API Gateway can be seen as the API runtime environment, which provides core services such as the following:

- Security (for example, authentication and authorization)
- Connectivity with a range of different protocols
- Virtualization

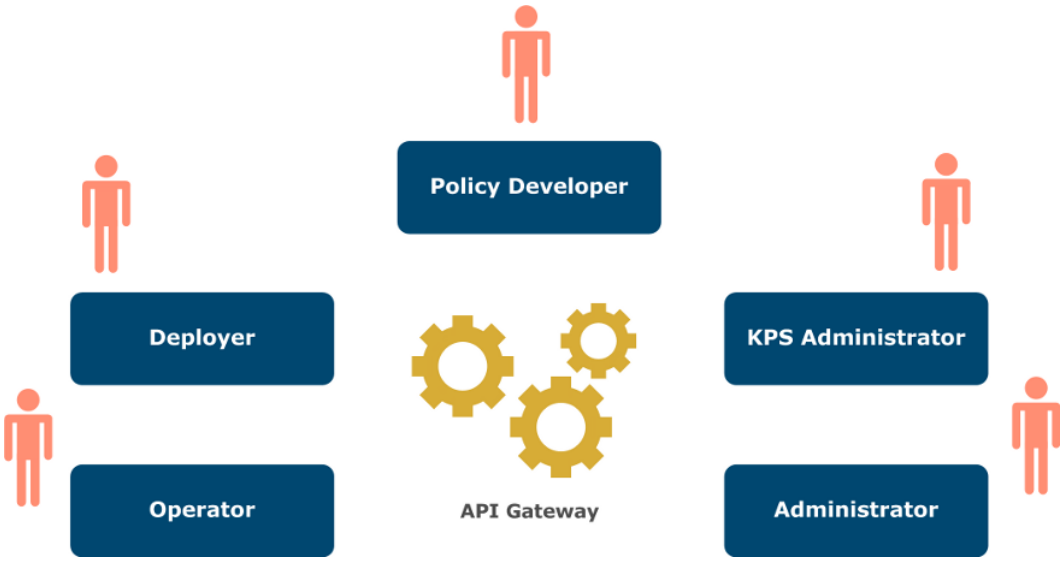
- Scalability and elasticity
- High availability
- Manageability (for example, using API Gateway Manager)
- Development simplicity

Because the API Gateway provides this core API infrastructure, developers can focus on providing the application logic. They no longer need to build these services into their application, and can leverage the core infrastructure provided by the API Gateway.

Previously, the API was not treated as a first class citizen, and in many cases was part of the application interface. However, the API Gateway sees the API as a first class artifact, with its own particular constructs, and its own runtime environment. The API Gateway provides all of the same benefits for the API that the application server provides for the application. In this way, it is important to distinguish between the API and the application as two distinct entities.

API Gateway user roles

API Gateway provides the following main user roles:



These user roles are described as follows:

- **Policy developer**
  This user role virtualizes APIs and develops policies for APIs. Policies are rules used to govern or manage an API (for example, for security, integration, SLA monitoring, or transformation). This is a technical developer role.
- **KPS administrator**
  This is a business or operational role managing dynamic policy configuration data in a Key Property Store (KPS). A KPS is used to store parameters that are passed into policies at runtime (for example, authorization levels, quotas, or customer details). This means that these details do not need to be configured by the policy developer.

- **API Gateway administrator**
  This role monitors, manages, and troubleshoots the API Gateway. It has full administrative privileges, including deployment of API Gateway configurations.

  This is the traditional system administration or operational role for the API Gateway. It involves keeping the API Gateway running, monitoring its operation, managing any settings, and performing any troubleshooting. This user typically works in an upstream staging or production environment instead of in a development environment.
- **API Gateway operator**
  This role monitors the API Gateway. It has read-only administrative capability. This is typically a production operations role.
- **Deployer**
  This role deploys API Gateway configurations using scripts. It has a restricted deployment role, and is typically used in production environments.

API Gateway features

API Gateway provides a comprehensive platform for managing, delivering, and securing APIs. It provides integration, acceleration, governance, and security for Web API and SOA-based systems. This section describes the high-level functionality available in API Gateway.

Integration

API Gateway provides the following integration features:

- **Identity management**

  API Gateway integrates with existing third-party Identity Management (IM) infrastructures to perform authentication and authorization of message traffic. For example, integration is provided with LDAP, Microsoft Active Directory, Oracle Access Manager, Computer Associates SiteMinder, Entrust GetAccess, IBM Tivoli Access Manager, RSA Access Manager, and other IM products. API Gateway also interoperates with leading integration products and platforms (for example, Microsoft .NET, Oracle WebLogic, IBM WebSphere, and SAP NetWeaver).
- **Scalability**

  API Gateway is designed to offer a highly flexible and scalable solution architecture. Administrators can deploy new API Gateway instances as needed, and deploy the same or different policies across a group of API Gateway instances as required. This enables administrators to apply polices at any point in their system. Policy enforcement points can be distributed around the network, anywhere traffic is being passed.
- **Pluggable pipeline**

  The API Gateway internal message-handling pipeline is extensible, enabling extra access control and content-filtering rules to be added with ease. Customers do not have to wait for a full product release before receiving updates of support for emerging standards and for additional adapters.

- **REST APIs**

  The API Gateway REST support enables you to make enterprise application data and operations available using Web APIs. For example, you can convert a legacy SOAP service, and deploy it as a REST API to be consumed by mobile apps. REST-to-SOAP conversion is easy to achieve using the API Gateway. It can expose REST APIs that map to SOAP services, dynamically creating a SOAP request based on the REST API call.

- **Internationalization (i18n)**

  API Gateway includes support for multi-byte message data and a wide range of international languages and character sets. For example, this includes requests in languages such as Chinese, German, French, Spanish, Danish, Serbian, Russian, Japanese, Korean, Greek, Arabic, Hebrew, and so on. The API Gateway supports character sets such as `UTF-8`, `KO-I8`, `UTF-16`, `UTF-32`, `ISO-8859-1`, `EUC-JP`, `US-ASCII`, `ISO-8859-7`, and so on.

Performance

API Gateway accelerates performance as follows:

- **Processing offload**

  You can use API Gateway to offload the heavy lifting of XML from application servers, and on to the network. This frees up resources on application servers and enables applications to run faster. Oracle's patented high-performance core XML acceleration engine, coupled with hardware acceleration ensures wire speed network performance.

- **Acceleration engine**

  The core acceleration engine is integrated into API Gateway to accelerate the essential XML security primitives. This engine provides XML processing at faster levels than those performed by common JAXP implementations in application servers and other applications that sit downstream from API Gateway. The acceleration engine performs Document Object Model (DOM) processing, XPath, JSON Path, XSLT conversion, and validation of XML and JSON.

- **Data enrichment**

  API Gateway can automatically populate content in XML and JSON documents from sources such as databases. By putting this functionality on to the network infrastructure, data is automatically populated in messages before they reach the consuming services. This simplifies and accelerates applications in ESBs and application servers.

Governance

API Gateway provides the following governance features:

- **Ease of deployment**

  API Gateway includes many features that speed up deployment. For example, certificates and private keys, necessary for XML security functions, are issued on board. API Gateway has a *deny-by-default* defense posture, to detect and block unauthorized deployments of services. Policies

can be re-applied across multiple endpoints using simple menus. Policies can also be imported and exported as XML files. This minimizes time needed to replicate policies across multiple API Gateways, or to move from a staging system to production environment.

- **Centralized management**

  A web-based system management dashboard provides centralized control of API Gateways in your domain. API Gateway Manager provides quick and easy access to enable you to manage your API Gateways and services. For example, you can use monitoring and a traffic log to monitor messages sent through API Gateways in your domain. All monitoring data can be aggregated across multiple API Gateway instances in a group or domain.

- The Policy Studio tool enables administrators to add security and management policies to the API Gateway, and to manage policy versions across multiple API Gateways. This enables enterprise policy management to be brought under centralized control, rather than be managed separately on each API Gateway.

- **Reporting**

  The API Gateway Analytics tool provides auditing and reporting on usage across all entry points and creates comprehensive reports to meet operational and compliance requirements. API Gateway Analytics also provides root cause analysis by identifying common failure points in multi-service transactions. If a service fails, and impacts the transaction as a whole, API Gateway Analytics can detect this and generate alerts.

- **Traffic throttling**

  API Gateway protects services from unanticipated traffic spikes by smoothing out traffic. It also limits clients to agreed service consumption levels in accordance with service usage agreements. This enables Oracle customers to charge their clients for different levels of service usage.

Security

API Gateway includes the following security features:

- **Identity mediation**

  Through its support for a wide range of security standards, API Gateway enables identity mediation between different identity schemes. For example, the API Gateway can authenticate external clients by user name and password, but then issue SAML tokens that are used for identity propagation to application servers.

- **API management**

  API Gateway enables you to secure Web APIs against attack and abuse. It also enables you to govern and meter access to and usage of Web APIs.API Gateway provides support for API management security standards such as OAuth. This enables you to share private resources with third-party websites without needing to provide credentials.

- **Application-level networking**

API Gateway routes data based on sender identity, content, and type. This enables messages to be sent to the appropriate application in a secure manner. It also enables *service virtualization*, where services are exposed to clients with virtual addresses to mask their actual addresses for security and application delivery. In this way, the API Gateway acts as an important control point for network traffic by shielding endpoint services from direct access.

- **Audit trail**

  API Gateway satisfies audit requirements by enabling service transactions to be archived in a tamper-proof store for subsequent audit. Oracle also facilitates privacy compliance support by allowing sensitive information, such as customer names, to be encrypted or stripped out of message traffic.

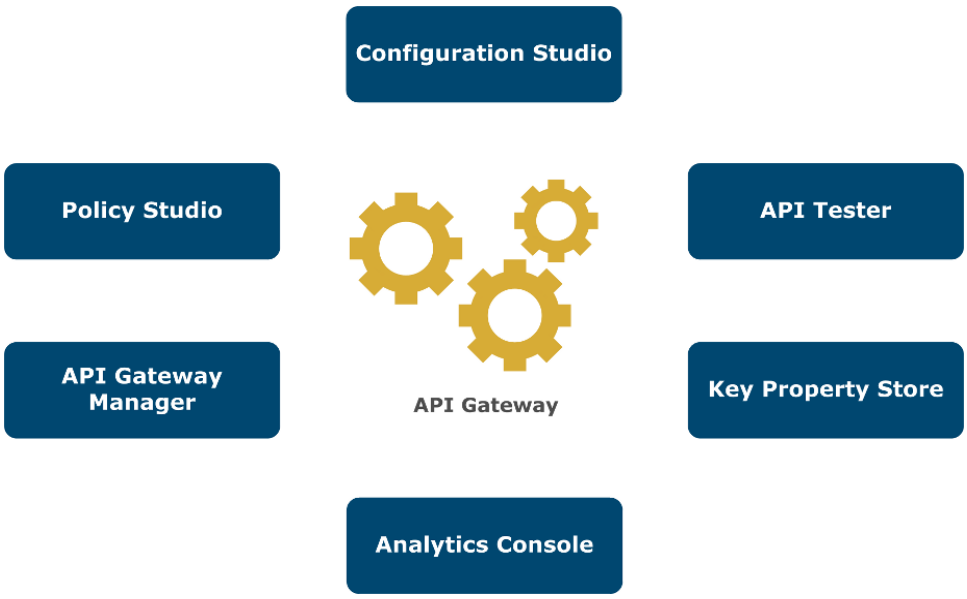
Form factors

API Gateway is available on Windows, Linux, and Solaris. For more details on supported platforms, see the *API Gateway Installation Guide*.

API Gateway tools 2

Overview

Oracle API Gateway provides powerful easy-to-use tools that enable you to develop, deploy, and manage API solutions. This topic introduces each of the API Gateway tools:



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

API Gateway

The central API Gateway core component is described as follows:

- Provides the runtime environment for exposing virtualized APIs and executing policies
- Implemented using combination of native code for performance and Java for extensibility
- Deployed and managed in a distributed environment of multiple servers providing scalability and availability
- Available in the following form factors:
  - Software—Windows, Linux, and Solaris

In enterprise organizations, the API Gateway is typically deployed in the DMZ between the public Internet and private intranet.

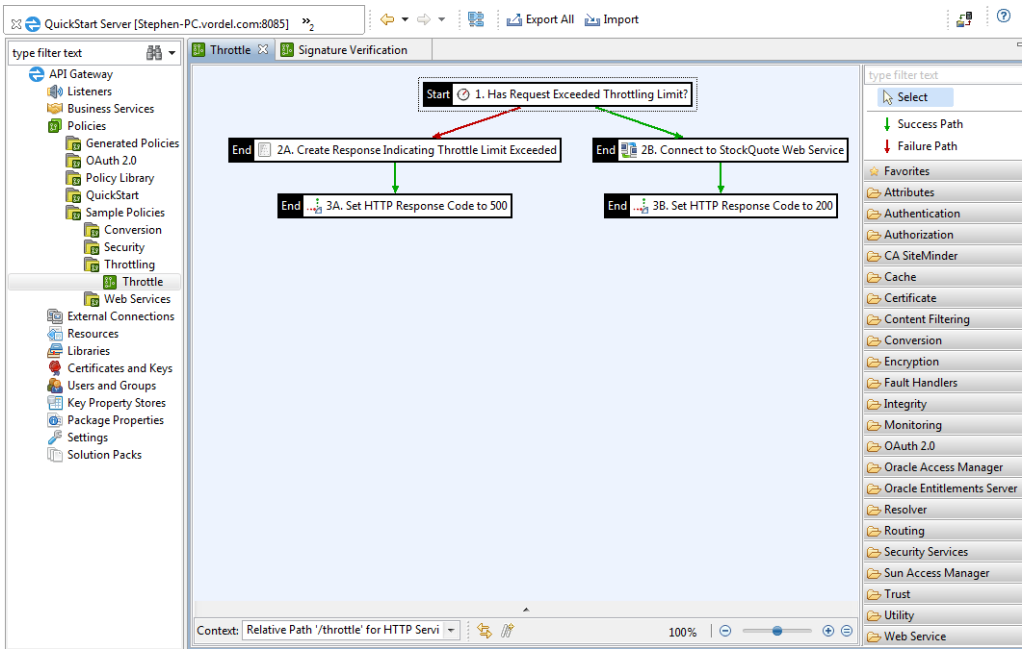
For more details, see *API Gateway architecture* on page 22.

Policy Studio

Policy Studio is graphical tool that enables you to virtualize APIs and develop policies (for example, to enforce security, compliance, and operational requirements). It includes the following features:

- Flow-chart style visualization for easy development and maintenance
- Graphical drag-n-drop user interface that enables you to drag filters (processing rules) on to the policy canvas and configure them
- Extensive library of filters to build powerful policies

The following example shows the policy canvas at the center and the filter library on the right:



A *filter* is an executable rule that performs a specific type of processing on a message. For example, the **Message Size** filter rejects messages that are greater or less than a specified size.

There are many categories of message filters available with the API Gateway (for example, Authentication, Authorization, Content Filtering, Conversion, Trust, and so on). In Policy Studio, a filter is displayed as a block of business logic that forms part of an execution flow known as a policy.

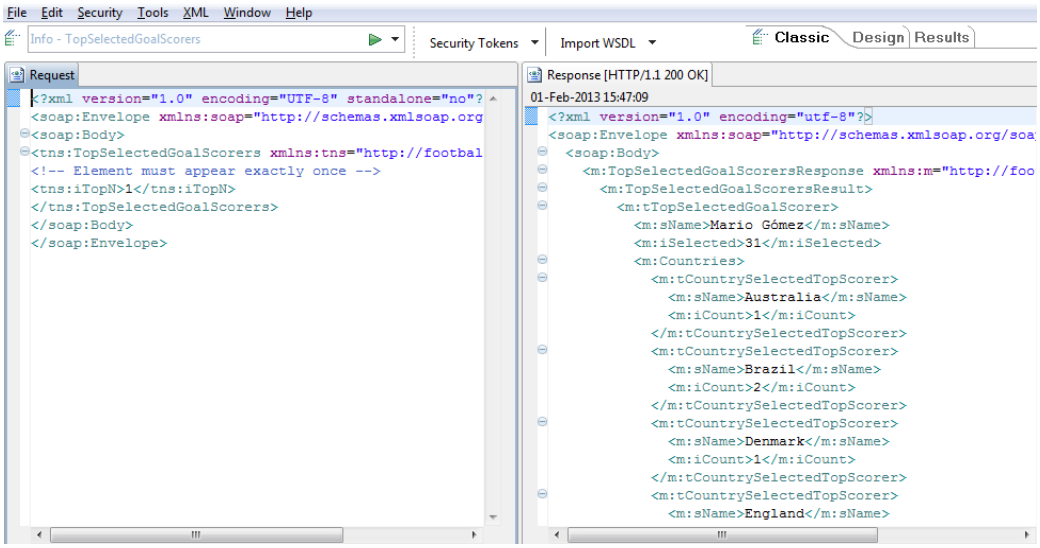
A *policy* is a network of filters in which each filter is a modular unit that processes a message. A message can traverse different paths through the policy, depending on which filters succeed or fail. For example, this enables you to configure policies that route messages that pass a **Schema Validation** filter to a back-end system, and route messages that pass a different **Schema Validation** filter to a different system.

A policy can also contain other policies, which enables you to build modular reusable policies. In Policy Studio, the policy is displayed as a path through a set of filters, as shown in the previous example.

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

API Tester

Oracle API Tester is a graphical tool that enables you to test API performance, scalability, and security. For example, you can use API Tester to send an example request message to a specific API service, and view the associated response.



API Tester includes the following features:

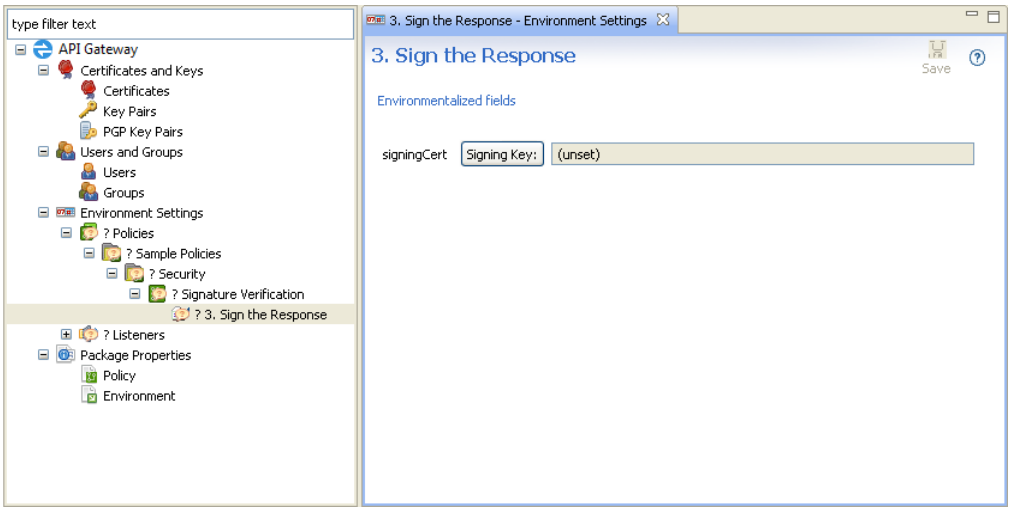
- REST API and SOAP Web services testing
- Security token insertion (for example, WS-Security and SAML)
- SOAP attachment management
- Simplified certificate and key management
- Test case creation and stress testing

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

Configuration Studio

Configuration Studio is a graphical tool used to promote API Gateway configuration from development environments to upstream environments (for example, testing or production).

Configuration Studio enables API Gateway administrators to take configuration prepared by policy developers, and to create environment-specific configuration for deployment. Configuration Studio is designed for the skills of upstream administrators, and does not assume expertise in policy development and policy configuration.



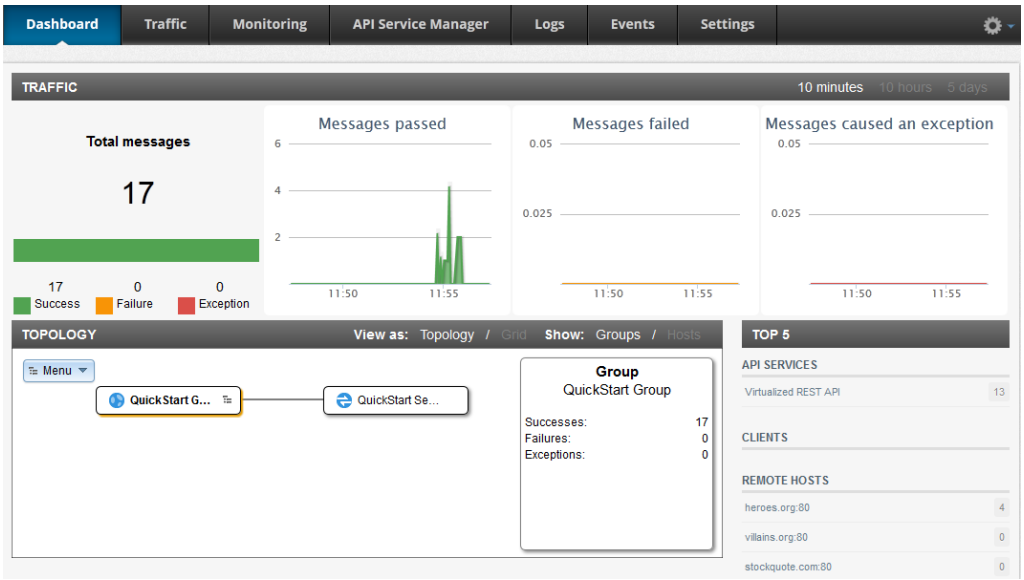
Configuration Studio enables administrators to perform tasks such as the following:

- Open a policy package (`.pol`) received from a development environment.
- Specify values for environment-specific settings selected in a development environment (for example, policy, listener, and external connections).
- Import or create environment-specific certificates and keys.
- Define environment-specific users and user groups.
- Export the environment package to a file on disk. The environment package is implemented as an `.env` file.

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

API Gateway Manager

API Gateway Manager is a Web-based administration console that enables you to perform operational monitoring, management, and troubleshooting.



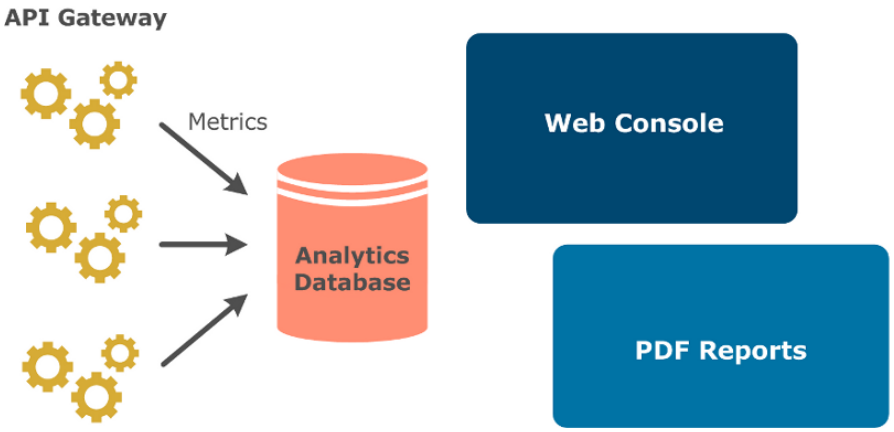
API Gateway Manager includes the following features:

- Dashboard displaying the distributed topology with a real-time overview of message traffic by domain, group, and API Gateway
- Real-time monitoring of message traffic and content, enabling easy identification of exceptions and drilling into message details
- Real-time monitoring of performance metrics by API service, system, and remote host
- Aggregated view of audit, alert, and SLA alert messages across the domain
- Centralized viewing of audit and debug logs of each API Gateway instance
- Managing dynamic system settings
- Managing user roles assigned in the domain

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

API Gateway Analytics

API Gateway Analytics is a Web-based monitoring and reporting console that enables you to generate scheduled reports and analyze API use in multiple API Gateways across the domain.



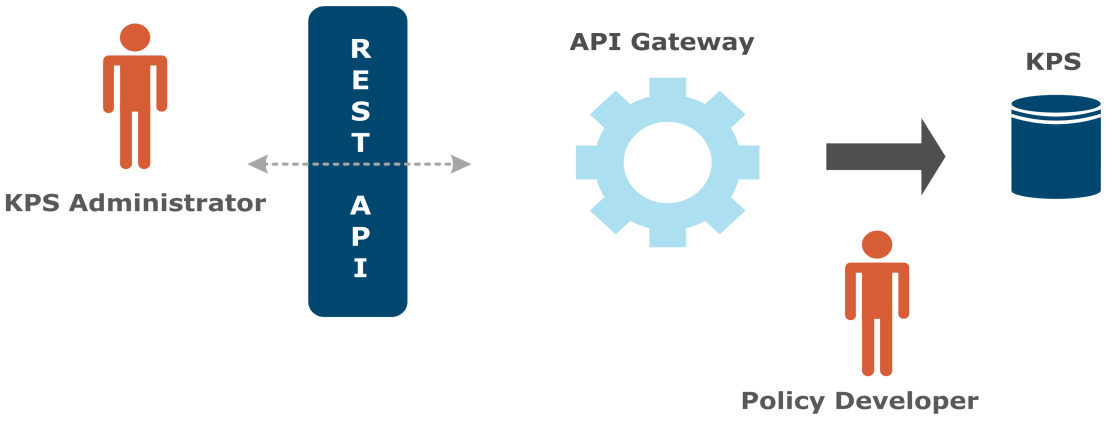
API Gateway Analytics includes the following features:

- Web-based console that monitors and reports on all API Gateways in the domain (multiple API Gateways are shown on the left in the diagram)
- Reporting over an extended time period rather than immediate operational monitoring
- Analysis of what APIs are used, how often APIs are used, when APIs are used, and who is using APIs
- Scheduled reports in PDF format can be emailed to specific users

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

Key Property Store

A Key Property Store (KPS) is used to store configuration parameters that are dynamically passed into policies at runtime. This enables policy configuration data to be managed directly by business or operational users at runtime, and allows dynamic change of policy behavior.



A KPS includes the following features:

- Policies look up configuration data in the KPS at runtime to dynamically determine behavior
- Policies developed in the Policy Studio use a selector syntax to specify context-sensitive lookup of policy configuration data at runtime from the KPS (for example, `${kps.CustomerProfiles[JoeBloggs].age}` obtains the age of the specified customer)
- Provides a cached read-frequently, write occasionally cache with backing stores
- Policy-specific UIs can be developed for business or operational users to manage the policy configuration data in the KPS

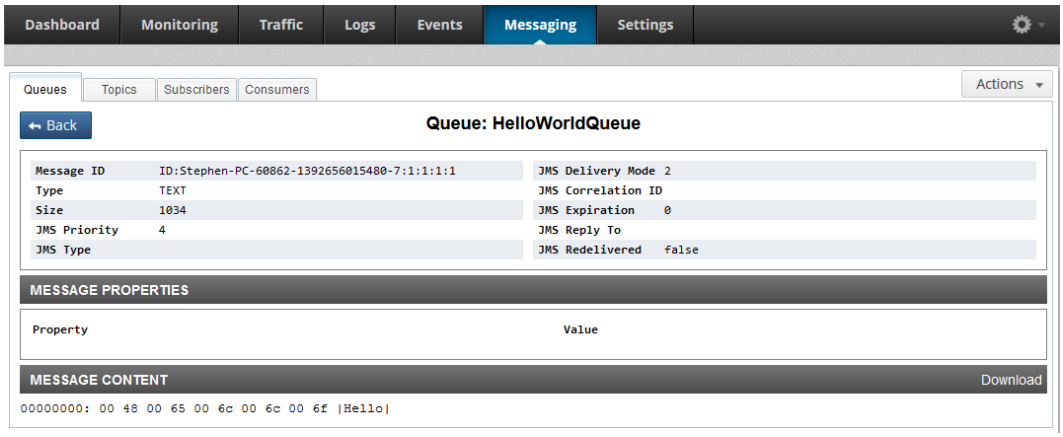
For more details, see the *API Gateway Key Property Store User Guide*.

Embedded Apache ActiveMQ

API Gateway can act as a native Java Message Service (JMS) provider by embedding Apache ActiveMQ. This enables the API Gateway to integrate external facing REST APIs and SOAP Web services with back-end systems and applications using reliable, asynchronous messaging.

For internal integration and ESB-style projects, API Gateway provides a messaging and mediation solution to route and transform messages flowing between applications and services. In addition, JMS queues hosted on the embedded ActiveMQ can be used by API Gateway policies to provide asynchronous policy behavior.

An ActiveMQ broker is embedded in each API Gateway instance, with brokers organized by API Gateway groups. An active/active deployment is supported to ensure high availability of the messaging infrastructure, with an external shared file system used for the persistent message store. Queue and topic management is integrated into the API Gateway Manager web console, which enables the API administrator to view queues and topics, messages on queues, and individual message contents. For example:



The API Gateway installation includes the ActiveMQ Java JMS 1.1 client library, which applications can use to send and receives message to and from the queues and topics hosted on the embedded ActiveMQ broker. In addition, ActiveMQ clients that use the OpenWire protocol (ActiveMQ default transport protocol) can interact with the embedded broker. For more details, see http://activemq.apache.org/openwire.html.

For details on how to manage ActiveMQ brokers embedded in the API Gateway, see the *API Gateway Administrator Guide*.

API Gateway architecture 3

Overview

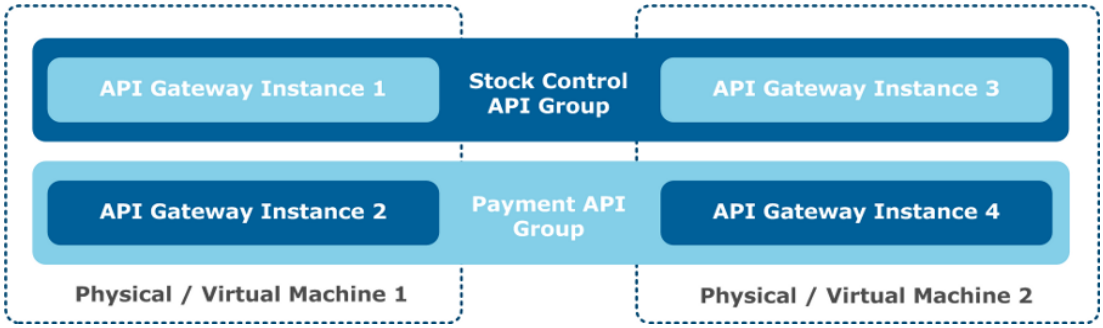
The API Gateway supports a distributed architecture based on groups of API Gateways in an administrative domain. The benefits of this architecture include the following:

- Managing a group of API Gateways as a single unit
- Solution partitioning by group
- Load balancing, scalability, and availability across the group
- Virtualization by separating logical and physical architectures—decoupling what is built from the physical architecture that runs it to enable infrastructure flexibility and scalability
- Running multiple isolated API applications on shared virtualized infrastructure
- Managing the domain based on administrative boundaries

API Gateway groups

An API Gateway group consists of one or more API Gateway instances that are managed as a unit and run the same configuration to virtualize the same APIs and execute the same policies. API Gateway groups enable you to organize API Gateway instances by solution type and manage them as a single entity.

The following diagram shows two API Gateway groups, each consisting of two API Gateway instances, distributed across two different host machines. Each API Gateway instance in the same group runs the same configuration to distribute the APIs and policies across both hosts for scalability and availability. Both groups run different configurations to virtualize different APIs, and run different policies that manage different solutions:



This group-based architecture is described as follows:

- API Gateways are deployed on the host machines.
- API Gateways are organized into groups of multiple API Gateways. A group must contain at least one API Gateway.
- All API Gateways in the group run the same configuration to virtualize the same APIs and execute the same policies. Partitioning of APIs and policies into different configurations should be performed by solution type.
- Groups span multiple host machines to provide availability, scalability, and load balancing.
- Management operations are performed on groups. For example:
  - Aggregating monitoring information from API Gateways in the group
  - Deploying API and policy configurations to all API Gateways in the group

Note Multiple API Gateways can run on the same host machine. However, each API Gateway would be in a different group and run a different configuration. There is no benefit to running multiple API Gateways in the same group on a single host machine.

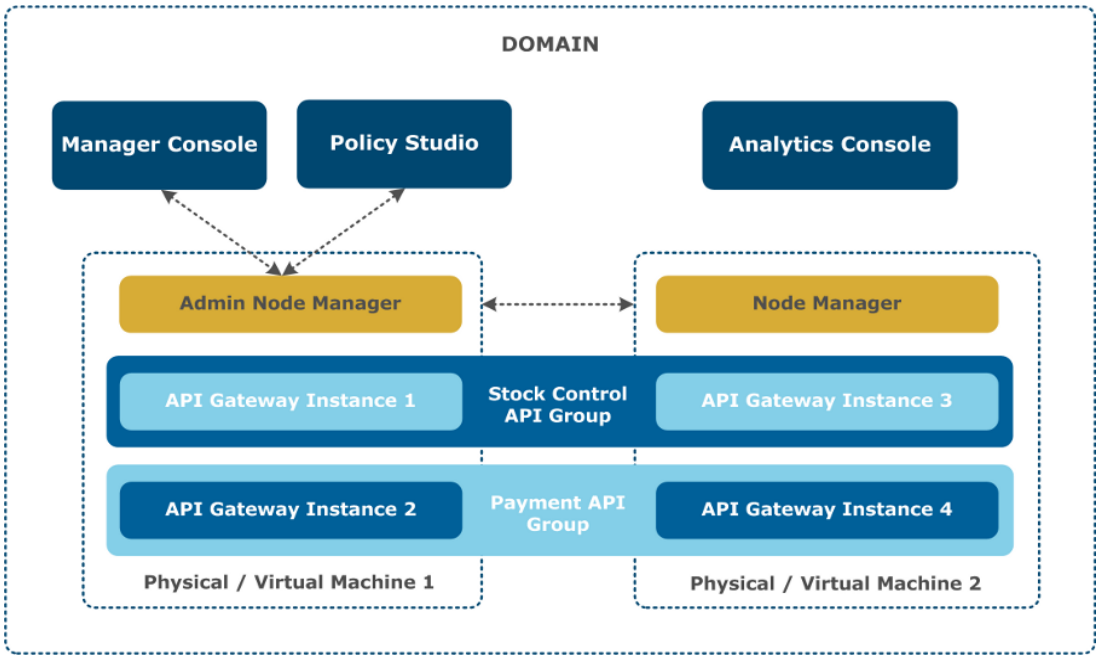
API Gateway domains

An API Gateway domain is a distinct administrative entity that consists of multiple groups spanning multiple host machines. Domains are scoped on the boundaries of administrative control, which may be organizational or geographical.

Multiple domains are possible based on different boundaries of administrative control. For example, you might have different domains for development and production environments, or different domains for each business unit.

Simple API Gateway domain

The following diagram shows the deployment of the two groups from the previous example in the context of a domain:



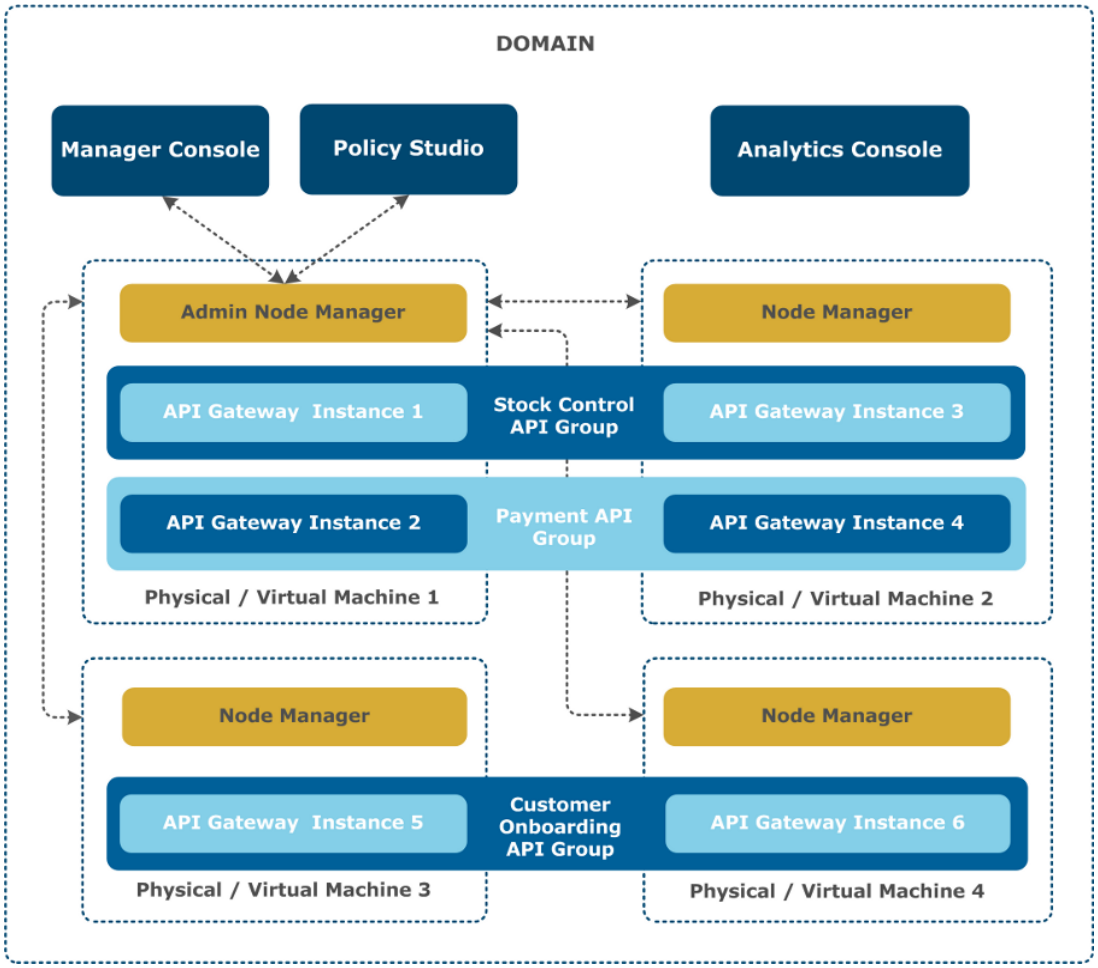
This domain-based architecture is described as follows:

- The Admin Node Manager in the domain is the central administration server for the entire domain, and is responsible for performing all management operations across the domain.
- The Node Manager (NM) on each machine manages all the local API Gateways on that machine, regardless of the group they are in. This includes the following:
  - Collecting monitoring information
  - Managing dynamic settings
  - Deploying API and policy configurations
- In addition to managing the local API Gateways on its host, the Admin Node Manager communicates with the NMs to perform management operations across the domain.
- Node Managers only communicate with the Admin Node Manager.
- The API Gateway Manager and Policy Studio tools connect to the Admin Node Manager.
- Role-Based Access Control (RBAC) for administrative users is across the domain. For example, an API Gateway administrator can log into API Gateway Manager and manage all API Gateways and groups in the domain.
- There is a single API Gateway Analytics database in a domain. All API Gateways record analytics information in this single database.

Note A single Admin Node Manager is deployed in the domain by default. However, you must configure at least two Admin Node Managers for high availability. For more details, see the *API Gateway Administrator Guide*.

Complex API Gateway domain

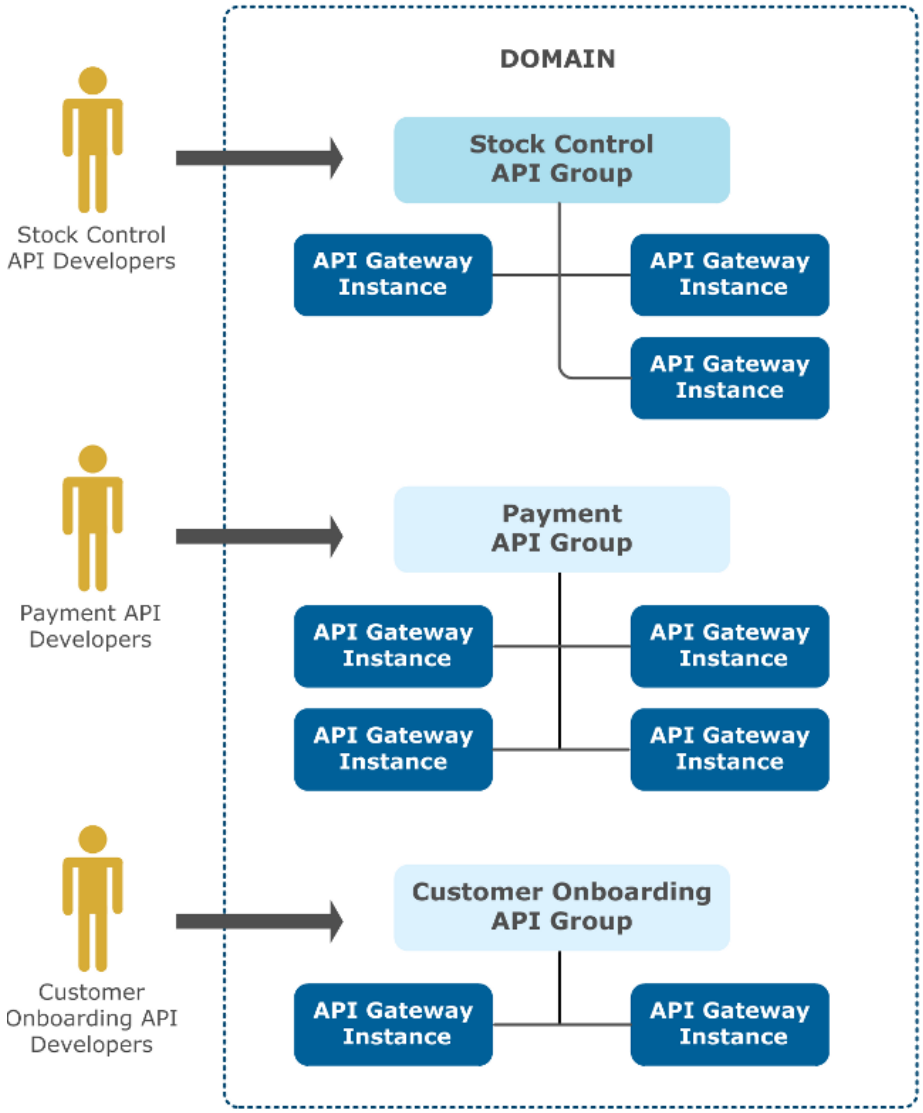
The following diagram shows a more complex domain with three groups distributed across four host machines:



Solution partitioning

API Gateway groups enable you to partition your APIs and policies by solution type. Partitioned APIs and policies associated with specific solutions are implemented in different API Gateway configurations, which are deployed to different groups and managed independently.

The following diagram shows an example API Gateway solution partitioned into groups:

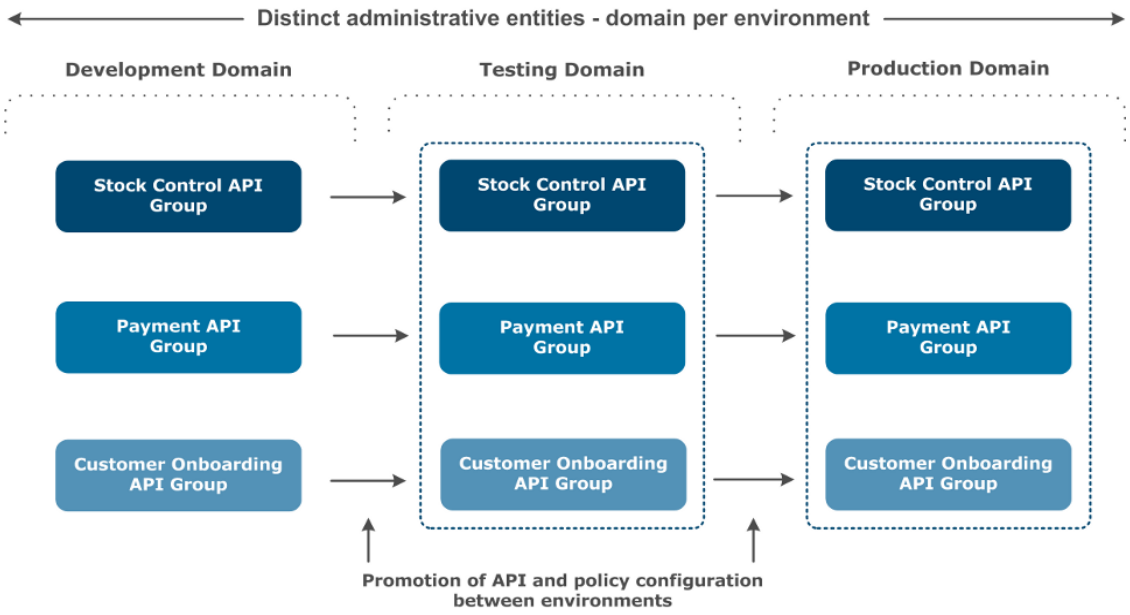


Virtualization

The API Gateway group and domain-based architecture enables virtualization by separating logical and physical architectures. The APIs and policies that are built and packaged into API Gateway configurations are decoupled from the physical architecture that they run, which provides flexibility and scalability of infrastructure.

Environment topology

The following diagram shows a typical environment topology that includes separate domains for each environment:



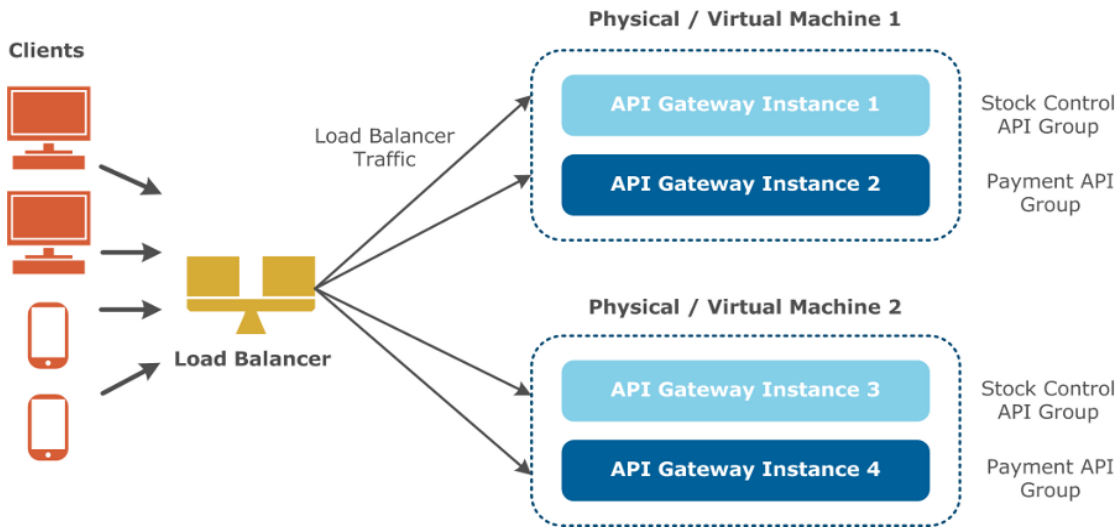
In this context, *promotion* refers to moving API Gateway configuration between environments and ensuring that environment-specific settings are properly configured. *Deployment* refers to the physical act of pushing configuration to an API Gateway instance (for example, using Policy Studio).

For details on how to promote between environments, see the *API Gateway Deployment and Promotion Guide*.

Availability, load balancing, and scalability

Availability and horizontal scalability is achieved by deploying multiple API Gateways on multiple hosts and load balancing across them using a standard load balancer. 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.

API Gateways being load balanced must run the same configuration to virtualize the same APIs and execute the same policies. If multiple 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 execution of policies is stateless, and the route that a message takes has no bearing on its processing. No session data is created, so there is no need to replicate session state across API Gateways. If the policies use caches and counters, these should be configured to use the distributed cache shared by all API Gateways. For more details on caching, see the *API Gateway Policy Developer Guide*.

API Gateway documentation 4

Overview

This topic shows where to look in the documentation library for more detailed information.

API Gateway library

The API Gateway documentation library includes the following user guides:

Document	Description
API Gateway Installation Guide	Describes how to install API Gateway components on all platforms, and how to upgrade API Gateway versions.
API Gateway Administrator Guide	Describes how to configure and manage the components in an API Gateway domain.
API Gateway Policy Developer Guide	Describes the main API Gateway features (for example, policies, filters, and configuration options), and how to configure them using the Policy Studio graphical tool.
API Gateway Deployment and Promotion Guide	Describes how to promote and deploy API Gateway configuration between different environments (for example, development, testing, and production).
API Gateway OAuth User Guide	Describes how to configure and manage the API Gateway for use with the OAuth open standard for authentication.
API Tester User Guide	Describes how to use the API Tester graphical tool to test REST-based APIs and SOAP-based web services.
API Gateway Developer Guide	Describes how to extend, leverage, and customize the API Gateway to suit the needs of your environment.
API Gateway Key Property Store User Guide	Describes how to configure and manage the API Gateway Key Property Store (KPS). This enables you to manage data referenced from policies running on the API Gateway.

Document	Description
API Gateway PassPort Interoperability Guide	Describes how to integrate API Gateway and Axway PassPort.
API Gateway Sentinel Interoperability Guide	Describes how to integrate API Gateway and Axway Sentinel.
API Gateway Validation Authority Interoperability Guide	Describes how to integrate API Gateway and Axway Validation Authority.

Glossary

Admin Node Manager
An API Gateway component responsible for managing API Gateway instances in a domain. For example, this includes collecting monitoring information, managing dynamic settings, and deploying API and policy configuration. The Admin Node Manager must be running to use the API Gateway management tools that connect to it (for example, Policy Studio and API Gateway Manager).

API
An Application Programming Interface (API) is a set of business services that an enterprise can expose to external customers, partners, or employees using a range of different technologies on a range of different devices. For example, APIs typically support HTTP requests and JSON or XML responses to enable mobile client applications.

API Gateway
A server-side application that manages, delivers, and secures APIs. API Gateway provides services such as API transformation, control and governance, security, monitoring, development lifecycle, and administration.

B2B
Business-to-Business

B2C
Business-to-Consumer

B2E
Business-to-Employee

Base64
A method of encoding 8-bit characters as ASCII printable characters. It is typically used to encode binary data so that it may be sent over text-based protocols such as HTTP and SMTP. Base64 is a scheme where 3 bytes are concatenated, and split to form 4 groups of 6-bits each. Each 6-bits is translated to an encoded printable ASCII character, using a table lookup. The specification is described in RFC 2045.

CA
A Certificate Authority (CA) issues digital certificates (especially X.509 certificates), and vouches for the binding between the data items in a certificate.

cacerts

A file used to keep the root certificates of signing authorities. This is typically stored in ..\jre\lib\security\cacerts. Each entry is identified by a unique alias, and is a key entry or a certificate entry. Key entries consist of a key pair, and certificate entries consist of just a certificate. Because you implicitly trust all CAs in the cacerts file for code signing and verification, you must manage the cacerts file carefully. The cacerts file should contain only certificates of the CAs you trust.

CMS

Content Management System

CRL

A Certificate Revocation List (CRL) is a signed list indicating a set of certificates that are no longer considered valid by the certificate issuer. CRLs may be used to identify revoked public-key certificates or attribute certificates, and may represent revocation of certificates issued to authorities or to users. The term CRL is also commonly used as a generic term applying to different types of revocation lists.

DName

A Distinguished Name (DName or DN) is an identifier that uniquely represents an object in the X.500 Directory Information Tree (DIT). A DName a set of attribute values that identify the path leading from the base of the DIT to the object that is named. An X.509 public-key certificate or CRL contains a DName that identifies its issuer, and an X.509 attribute certificate contains a DN or other form of name that identifies its subject.

Domain

An API Gateway domain consists of multiple groups of API Gateways spanning multiple host machines. A domain is a distinct administrative entity, which is managed separately by API Gateway tools such as API Gateway Manager and API Gateway Analytics.

ERP

Enterprise Resource Planning

Filter

An API Gateway filter is an executable rule that performs a specific type of processing on a message. For example, the Message Size filter rejects messages that are greater or less than a specified size. Many categories of message filters are available with the API Gateway (for example, Authentication, Authorization, Content filtering, Conversion, Trust, and so on). In Policy Studio, a filter is displayed as a block of business logic that forms part of an execution flow known as a policy.

Group

An API Gateway group consists of one or more API Gateway instances that are managed as a unit and run the same configuration to virtualize the same APIs and execute the same policies.

API Gateway groups enable you to organize API Gateway instances by solution type and manage them as a single entity.

HTTP

Hypertext Transfer Protocol (HTTP) is a protocol for distributed hypermedia systems. HTTP is the foundation of data communication for the World Wide Web. For more details, see http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol.

HTTPS

Hypertext Transfer Protocol Secure (HTTPS) is a protocol for secure communication over a computer network, and which is widely deployed on the Internet. It is the result of layering HTTP on top of the SSL/TLS protocol. For more details, see http://en.wikipedia.org/wiki/HTTP_Secure.

JMS

Java Message Service (JMS) is a messaging standard that enables application components based on Java 2 Enterprise Edition (J2EE) to create, send, receive, and read messages. It enables communication between different components of a distributed application to be loosely coupled, reliable, and asynchronous. For more details, see http://en.wikipedia.org/wiki/Java_Message_Service.

JSON

JavaScript Object Notation (JSON) is a lightweight data-interchange format, which is easy for humans to read and write, and easy for machines to parse and generate. JSON is based on a subset of the JavaScript programming language. Its text format is programming language independent, but uses conventions that are familiar to programmers of the C family of languages (for example, C, C++, C#, Java, JavaScript, Perl, and Python). For more details, see http://www.json.org.

JSON Path

JSON Path enables you to locate and process specific parts of a JSON document. It is available in programming languages such as JavaScript, Java, Python and PHP. For more details, see the JSON specification.

Keystore

The keystore file of the JDK contains your public and private keys. It has a file name of .keystore (leading dot makes the file read-only on Unix). It is stored in PKCS #12 format, contains both public and private keys, and is protected by a passphrase.

KPS

A Key Property Store (KPS) is a data management component in the API Gateway. Data in a KPS table is assumed to be read frequently and seldom written, and can be changed without incurring an API Gateway service outage. KPS tables are shared across an API Gateway group.

LDAP

LDAP is a lightweight version of Directory Access Protocol (DAP), which is part of X.500, a standard for directory services in a network. An LDAP directory stores information on resources in a hierarchical fashion, which makes data retrieval very efficient.

Node Manager

An API Gateway component that is responsible for managing API Gateway instances on a host machine. There must be one Node Manager on each managed host machine. A single Admin Node Manager communicates with all Node Managers in a domain to perform management operations.

OCSP

Online Certificate Status Protocol (OCSP) is an automated certificate checking network protocol. A client will query the OCSP responder for the status of a certificate. The responder returns whether the certificate is still trusted by the CA that issued it.

PEM

Privacy Enhanced Mail (PEM) was originally intended for securing email using various encryption techniques. Its scope widened for use in a broader range of applications, such as Web servers. Its format is essentially a base64-encoded certificate wrapped in BEGIN CERTIFCATE and END CERTIFICATE directives.

PKCS#12

A standard for storing private keys and X.509 certificates securely (for example, in a .p12 file).

Policy

An API Gateway policy is a network of filters in which each filter is a modular unit that processes a message. Messages can traverse different paths through the policy, depending on which filters succeed or fail. For example, you could configure policies routing messages that pass a Schema Validation filter to a back-end system, and routing messages that pass a different Schema Validation filter to another system. A policy can also contain other policies, which enables you to build modular reusable policies.

Private key

The secret component of a pair of cryptographic keys used for asymmetric cryptography.

Public key

The publicly-disclosed component of a pair of cryptographic keys used for asymmetric cryptography.

RBAC

Role-Based Access Control (RBAC) restricts system access to authorized users based on assigned roles. Permissions to perform specific system operations are assigned to specific roles, and system users are granted permission to perform specific operations only through their roles.

This simplifies system administration because users do not need to be assigned permissions directly, and instead acquire them through their assigned roles.

REST

Representational State Transfer (REST) is an architectural style for building large-scale distributed software that uses the technologies and protocols of the World Wide Web (for example, JSON/XML and HTTP). For more details, see http://en.wikipedia.org/wiki/Representational_state_transfer.

SAML

Security Assertion Markup Language (SAML) is an XML standard for establishing trust between entities. SAML assertions contain identity information about users (authentication assertions), and information about user access permissions of (authorization assertions). When a user is authenticated at a site, the site issues a SAML authentication assertion to the user. The user can use this assertion in requests at other affiliated sites. These sites need only check the details in the authentication assertion to authenticate the user. In this way, SAML allows authentication and authorization information to be shared between different sites.

SCM

Supply Chain Management

Selector

A special syntax that enables API Gateway configuration settings to be evaluated and expanded at runtime based on metadata values (for example, from a KPS, message attribute, or environment variable).

Signature

A value computed with a cryptographic algorithm and added to a data object in such a way that any recipient of the data can use the signature to verify the data's origin and integrity.

SOAP

Simple Object Access Protocol (SOAP) is an XML-based object invocation protocol. SOAP was originally developed for distributed applications to communicate over HTTP and corporate firewalls. SOAP defines the use of XML and HTTP to access services, objects, and servers in a platform-independent way. SOAP is a wire protocol that can be used to facilitate highly ultra-distributed architecture. For more details, see the SOAP specification.

SSL

Secure Sockets Layer (SSL) is an encrypted communication protocol for sending information securely across the Internet. It sits just above the transport layer, and below the application layer and transparently handles the encryption and decryption of data when a client establishes a secure connection to the server. It optionally provides peer entity authentication between client and server.

TLS

Transport Layer Security (TLS) is the successor to SSL 3.0. Like SSL, it allows applications to communicate over a secure channel.

UDDI

Universal Description, Discovery, and Integration (UDDI) is an XML-based lookup service for locating Web services on the Internet. For more details, see the UDDI standard.

URI

A Uniform Resource Identifier (URI) is a platform-independent way to specify a file or resource on the Web. Strictly speaking, every URL is also a URI, but not every URI is also a URL. For more details on URI formats, see RFC 2396 and RFC 2732.

WSDL

Web Services Description Language (WSDL) is an XML format for describing network services as a set of endpoints operating on messages containing document-oriented or procedure-oriented information. Operations and messages are described abstractly, and bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is extensible to allow description of endpoints and messages regardless of what message formats or network protocols are used. For more details, see the WSDL specification.

X.509

A standard that defines the contents and data format of a public key certificate.

XKMS

XML Key Management Specification (XKMS) uses XML to provide key management services so that a Web service can query the trustworthiness of a user's certificate over the Internet. XKMS aims to simplify application building by separating digital-signature handling and encryption from the applications themselves. For more details, see the XML Key Management specification.

XML

Extensible Markup Language (XML) is a subset of Structured General Markup Language (SGML). Its goal is to enable generic SGML to be served, received, and processed on the Web in a similar way to HTML. See the XML Specification for more details.

XPath

XML Path (XPath) is a language that describes how to locate and process specific parts of an XML document. For more details, see the XML Path Language specification.

XSL

XML Stylesheet Language (XSL) is used to convert XML documents into different formats, the most common of which is HTML. In a typical scenario, an XML document references an XSL stylesheet, which defines how the XML elements of the document should be displayed as HTML. This enables a clear separation of content and presentation.

XSLT

Extensible Stylesheet Language Transformation (XSLT) is used to convert XML documents into other XML documents or other formats (for example, HTML, plain text, or XSL Formatting objects).