Oracle® Fusion Middleware

API Gateway Concepts Guide
11g Release 2 (11.1.2.2.0)

August 2013

Oracle API Gateway Concepts Guide, 11g Release 2 (11.1.2.2.0)

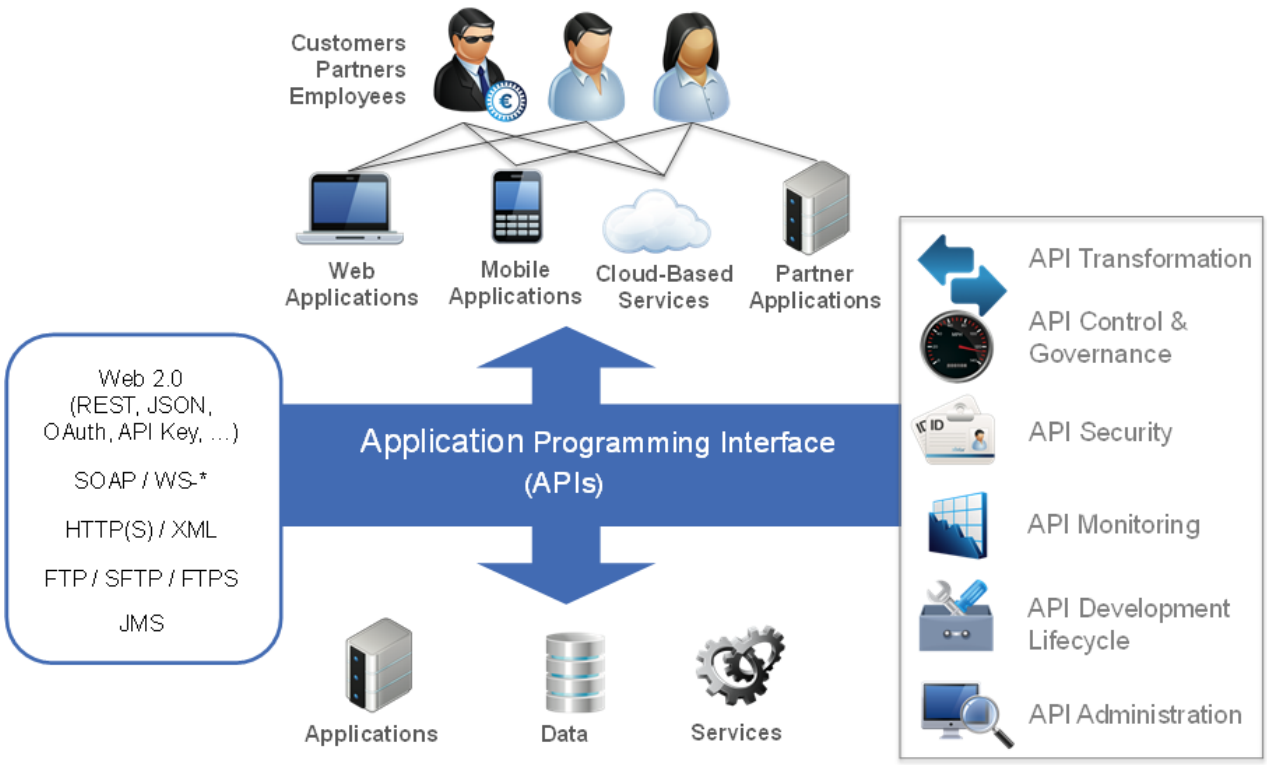
28 August 2013

Contents

Introducing Oracle API Gateway

Overview

The 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 the API Gateway on the left, and the services that it provides on the right:



API Gateway Features

The Oracle API Gateway supports the following main services:

API Transformation
This includes 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
This includes 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
This includes 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
This includes 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
This includes the following:

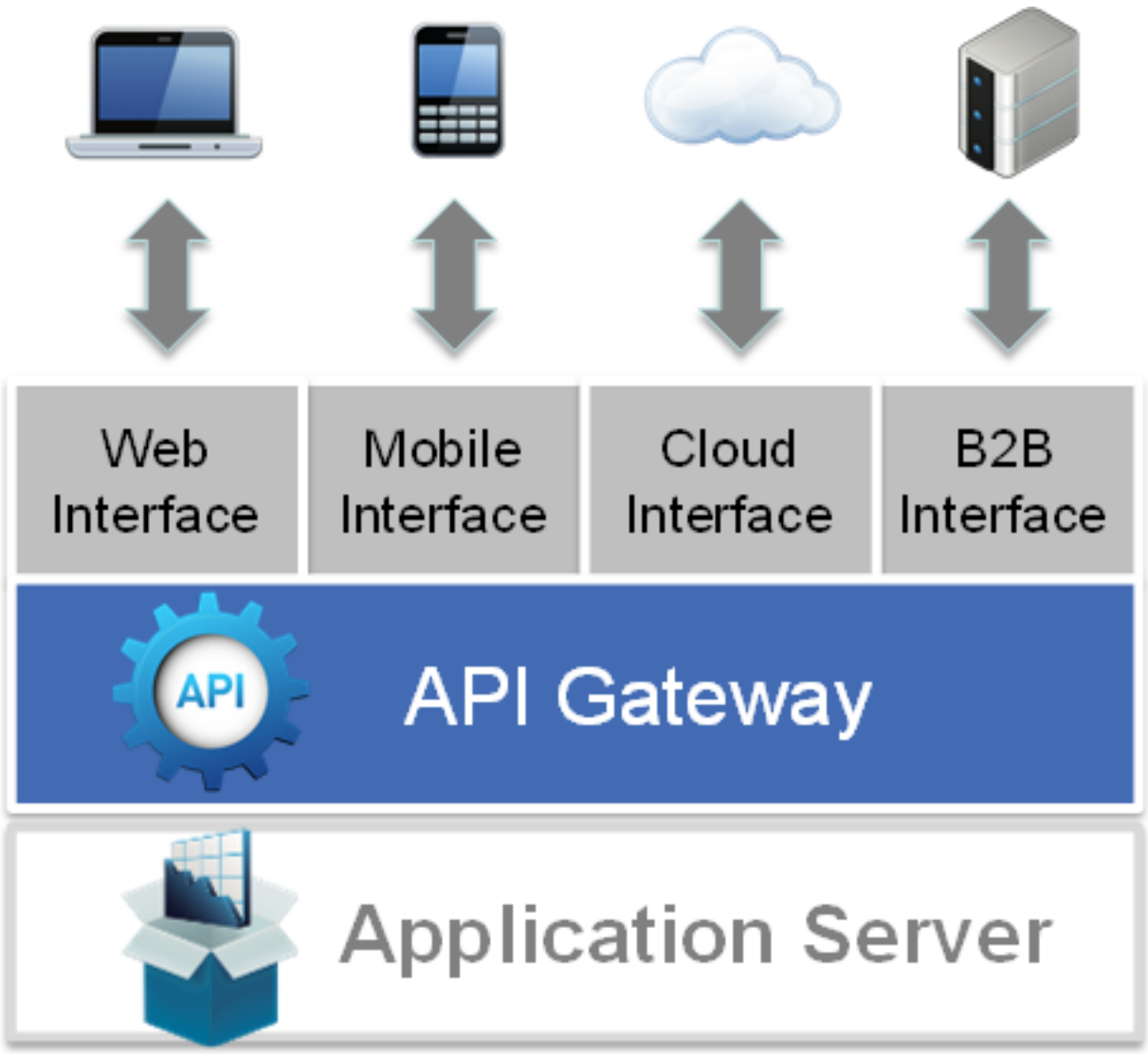
- Manage API lifecycle from creation to end-of-life
- Drag-n-drop policy creation with intuitive flow chart metaphor
- Extensive library of pre-built policy rules
- Interactive API testing tool
- Promotion between environments

API Administration
This includes the following:

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

API Gateway is Core Infrastructure

The 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:

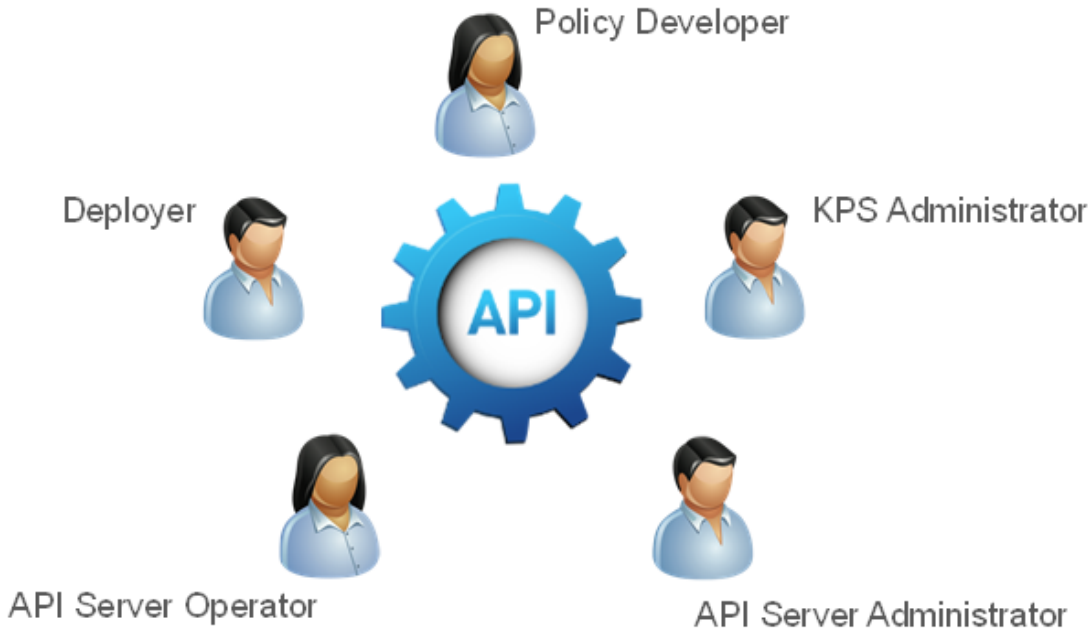
- Security (for example, authentication and authorization)
- Connectivity with a range of different protocols
- Virtualization
- Scalability and elasticity
- High availability
- Manageability (for example, using API Gateway Manager)
- Development simplicity

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

The API Gateway provides the following 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.

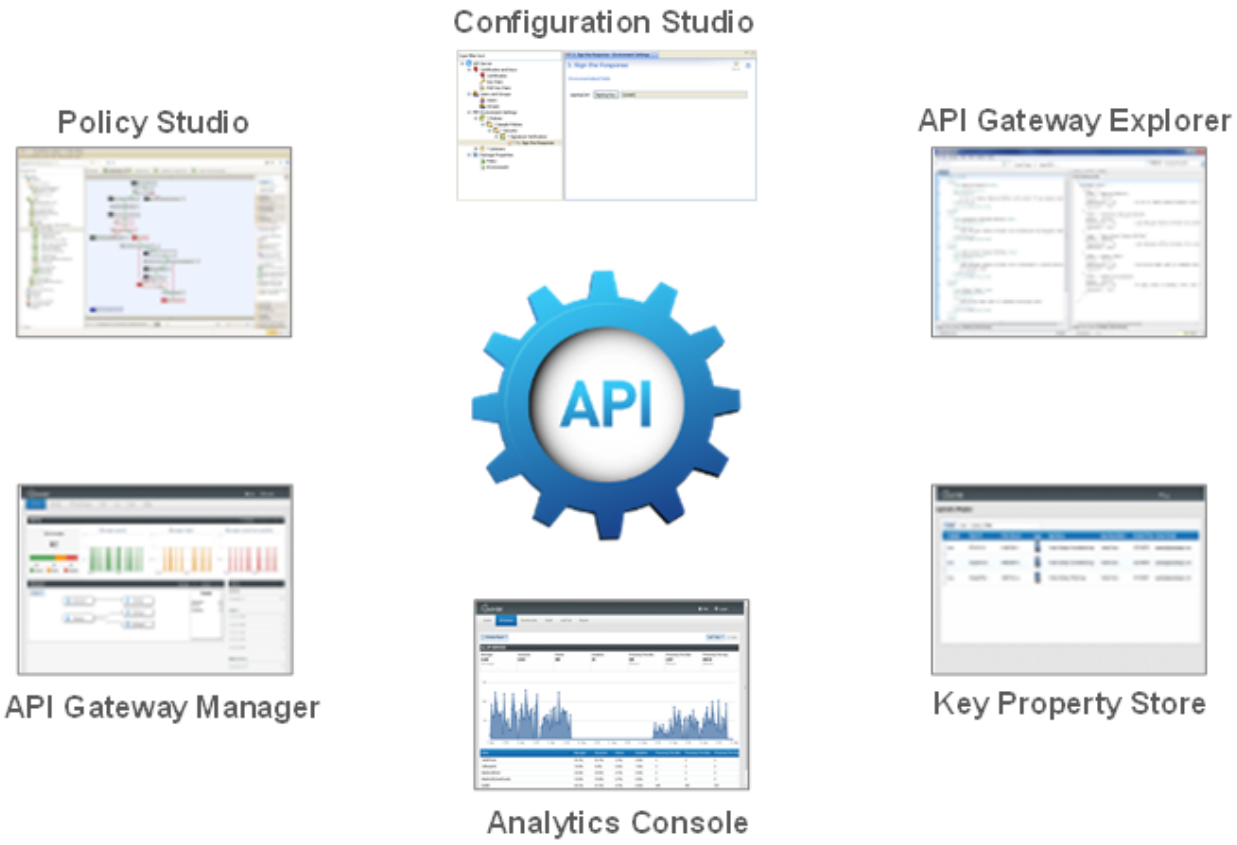
Deployer
This role deploys API Gateway configurations using scripts. It has a restricted deployment role, and is typically used in

production environments.

API Gateway Tools

Overview

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



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
  - Virtual appliance—VMWare and Amazon Machine Image (AMI)

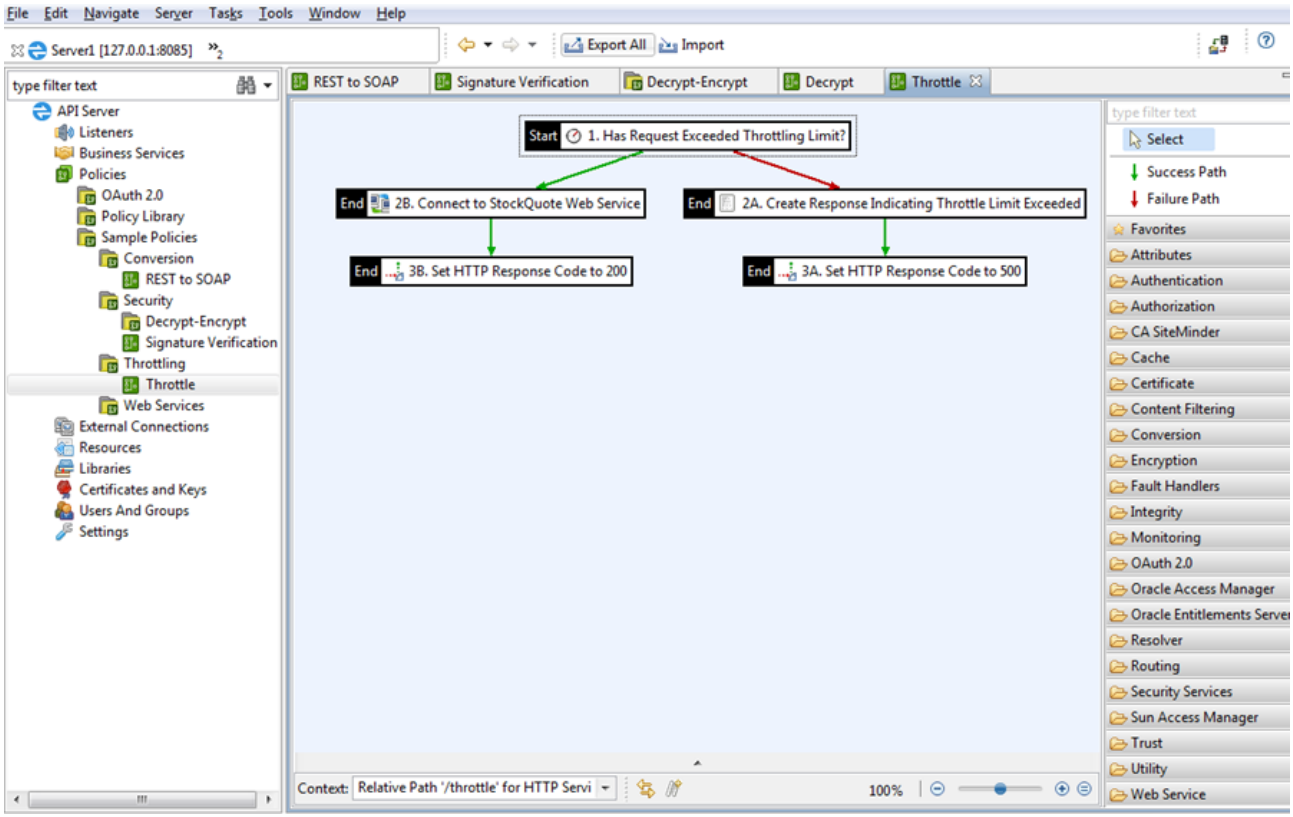
In enterprise organizations, the API Gateway is typically deployed in the DMZ between the public Internet and private intranet.

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 screenshot shows the policy canvas at the center and the filter library on the right:

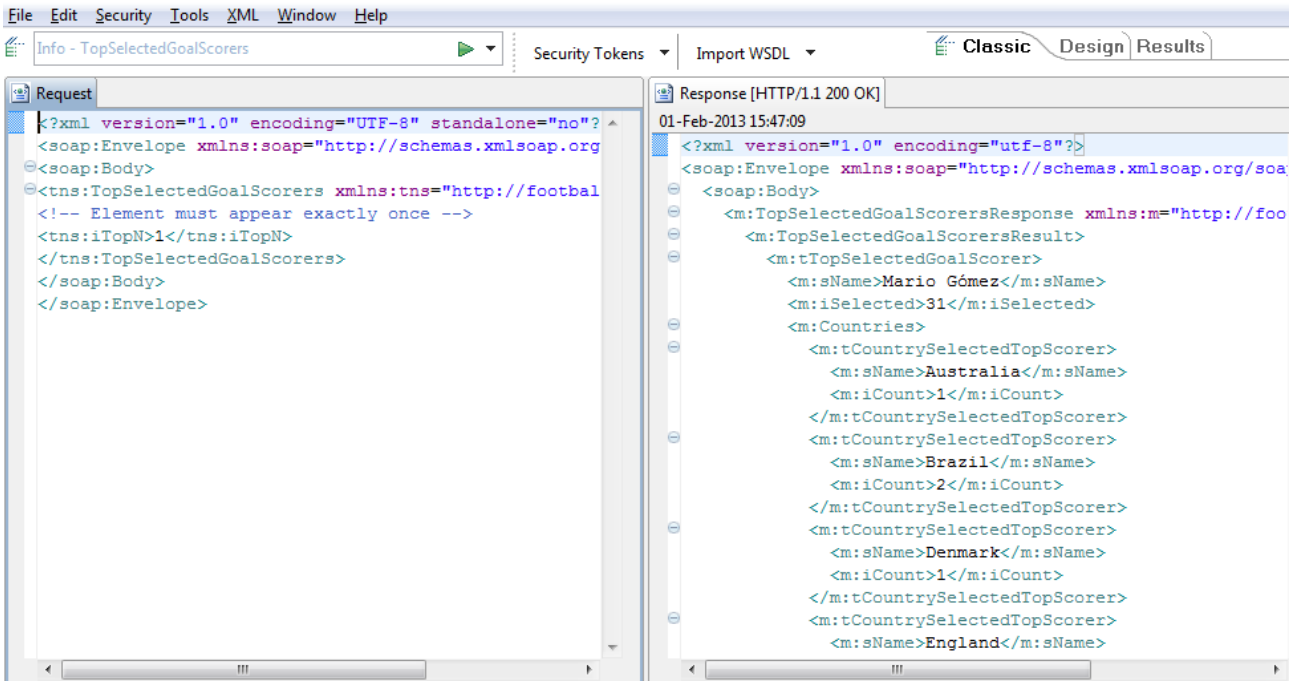


A *filter* is an executable rule that performs a specific type of processing on a message. For example, the **Message Size** filter rejects messages that are greater or less than a specified size. There are many categories of message filters available with the API Gateway (for example, Authentication, 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 screen.

API Gateway Explorer

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

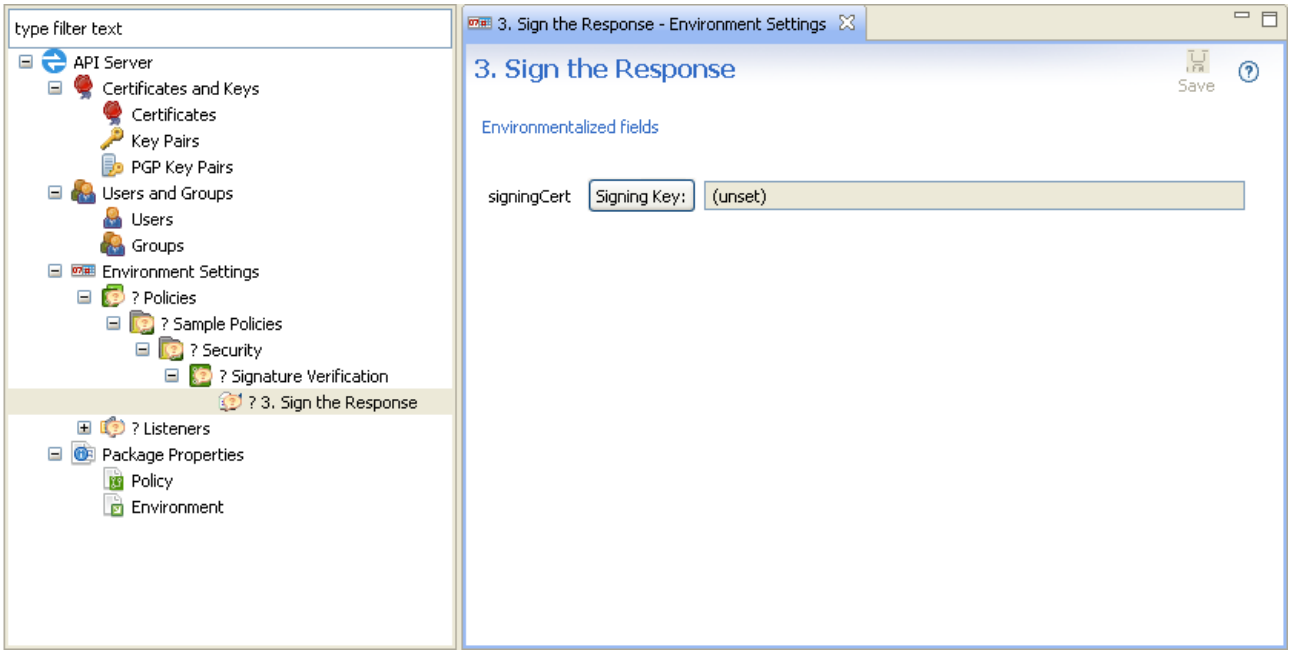


API Gateway Explorer includes the following features:

- 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

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.

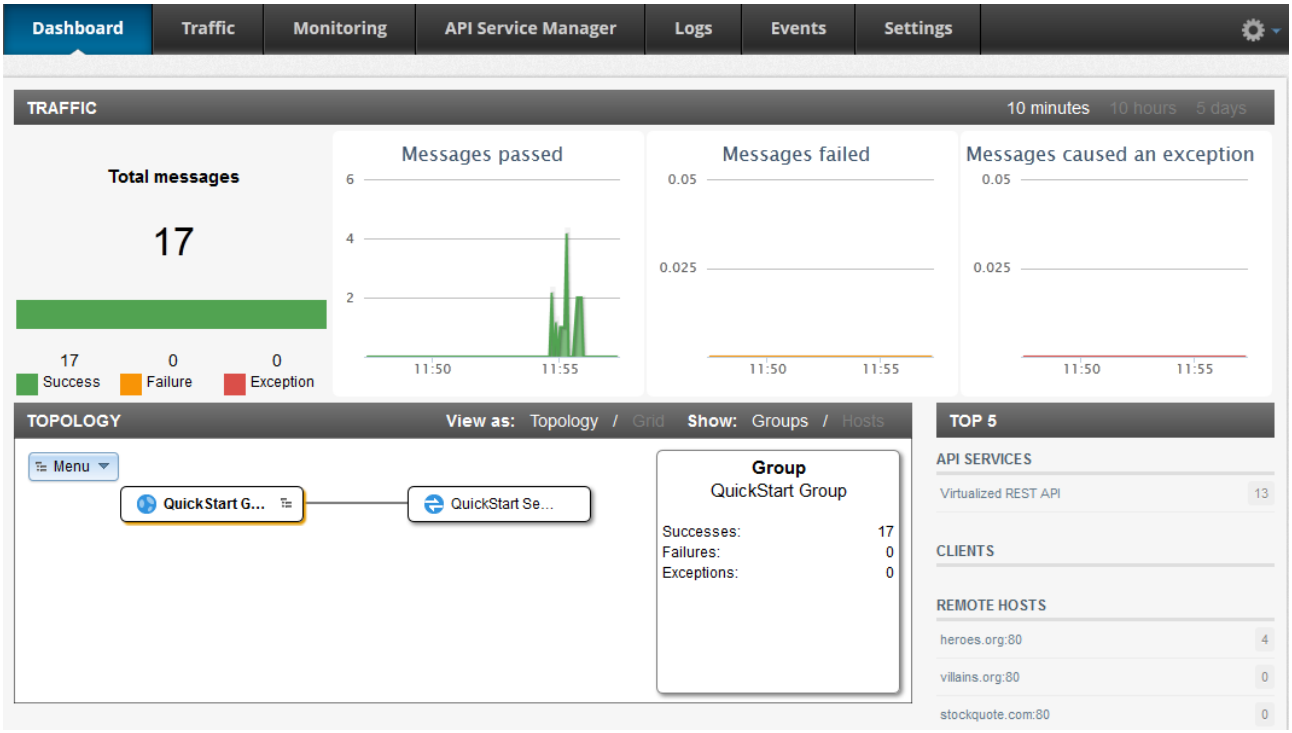


Configuration Studio enables administrators to perform tasks such as the following:

- 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.

API Gateway Manager

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

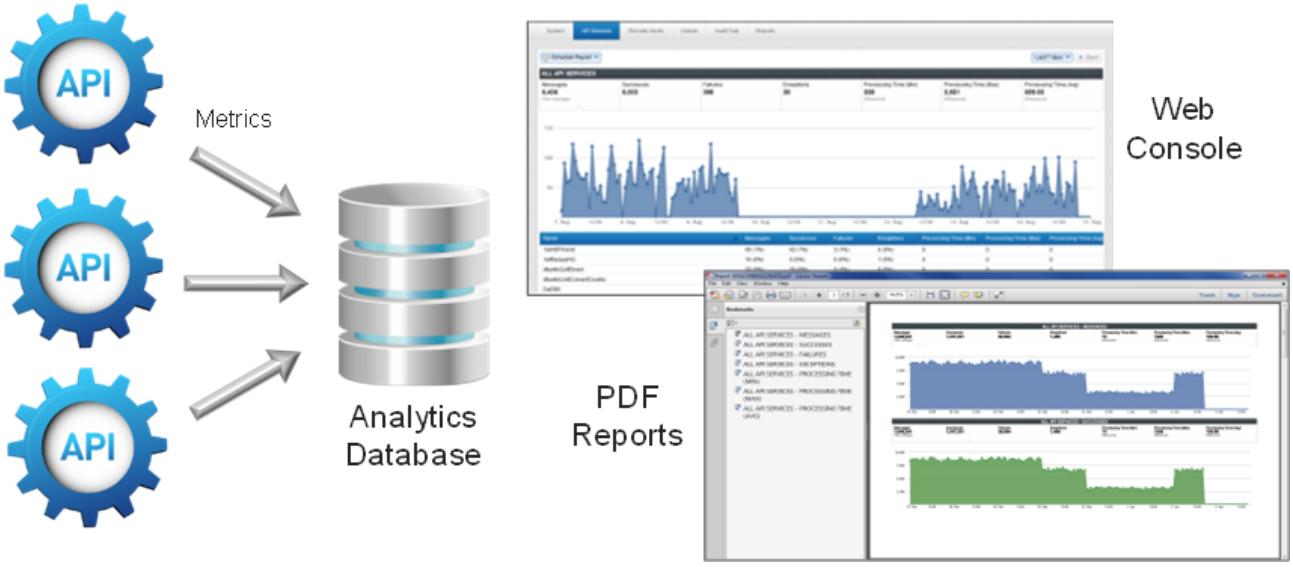


API Gateway Manager includes the following features:

- 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

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.

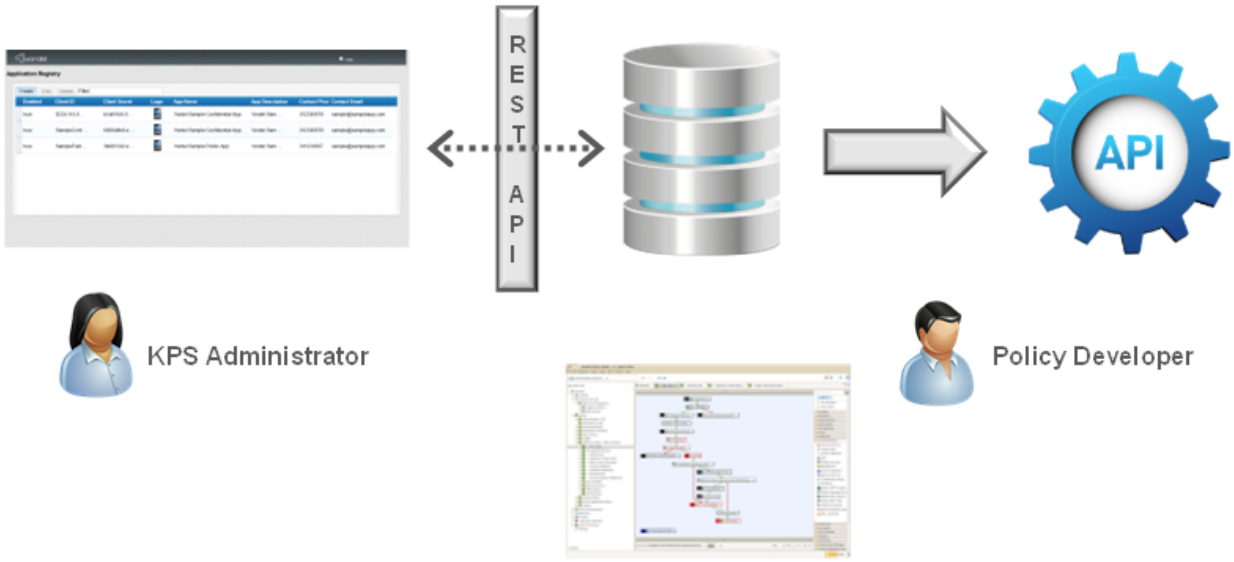


API Gateway Analytics includes the following features:

- 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

Key Property Store

An Oracle Key Property Store (KPS) is used to store 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

API Gateway Groups and Domains

Overview

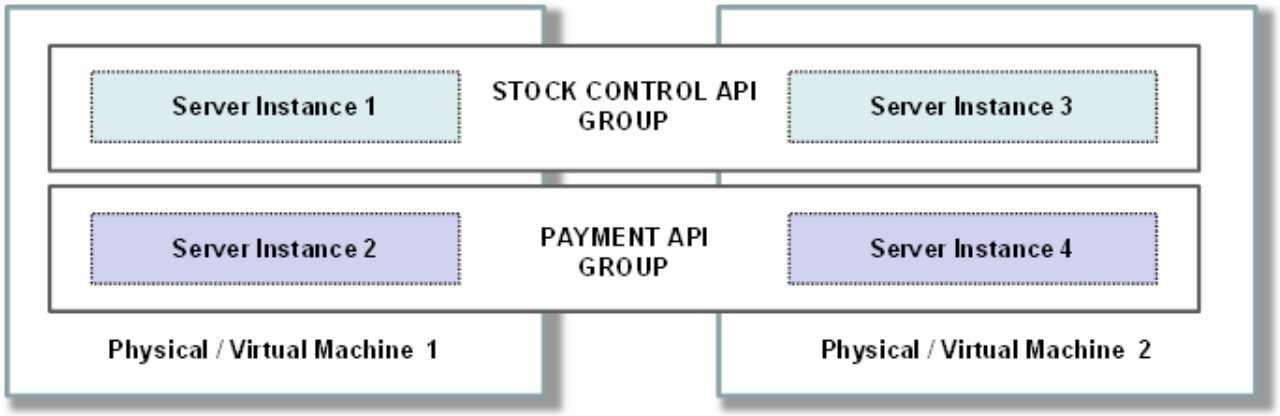
The API Gateway supports a distributed architecture based on groups of API Gateways in an administrative domain. The benefits of this architecture include the following:

- 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

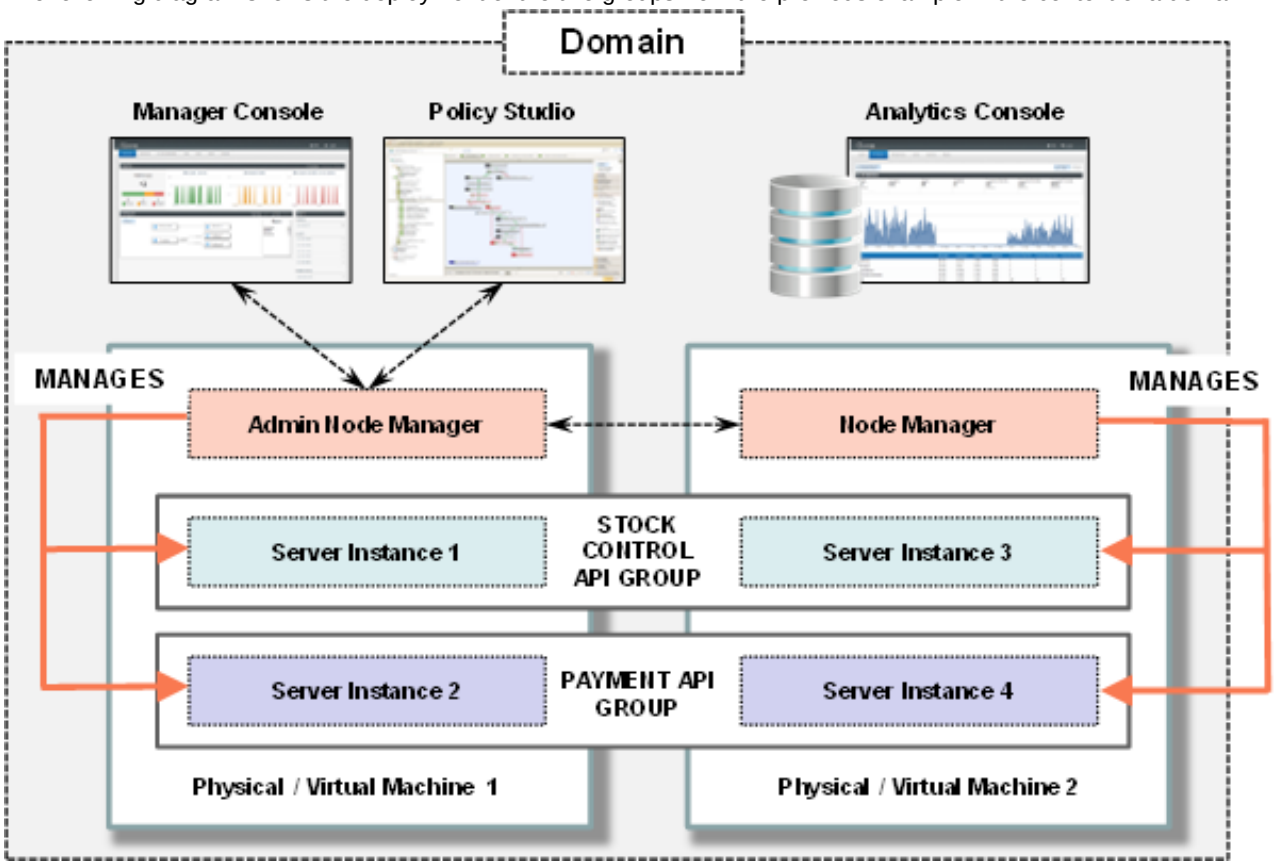
Note

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.

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

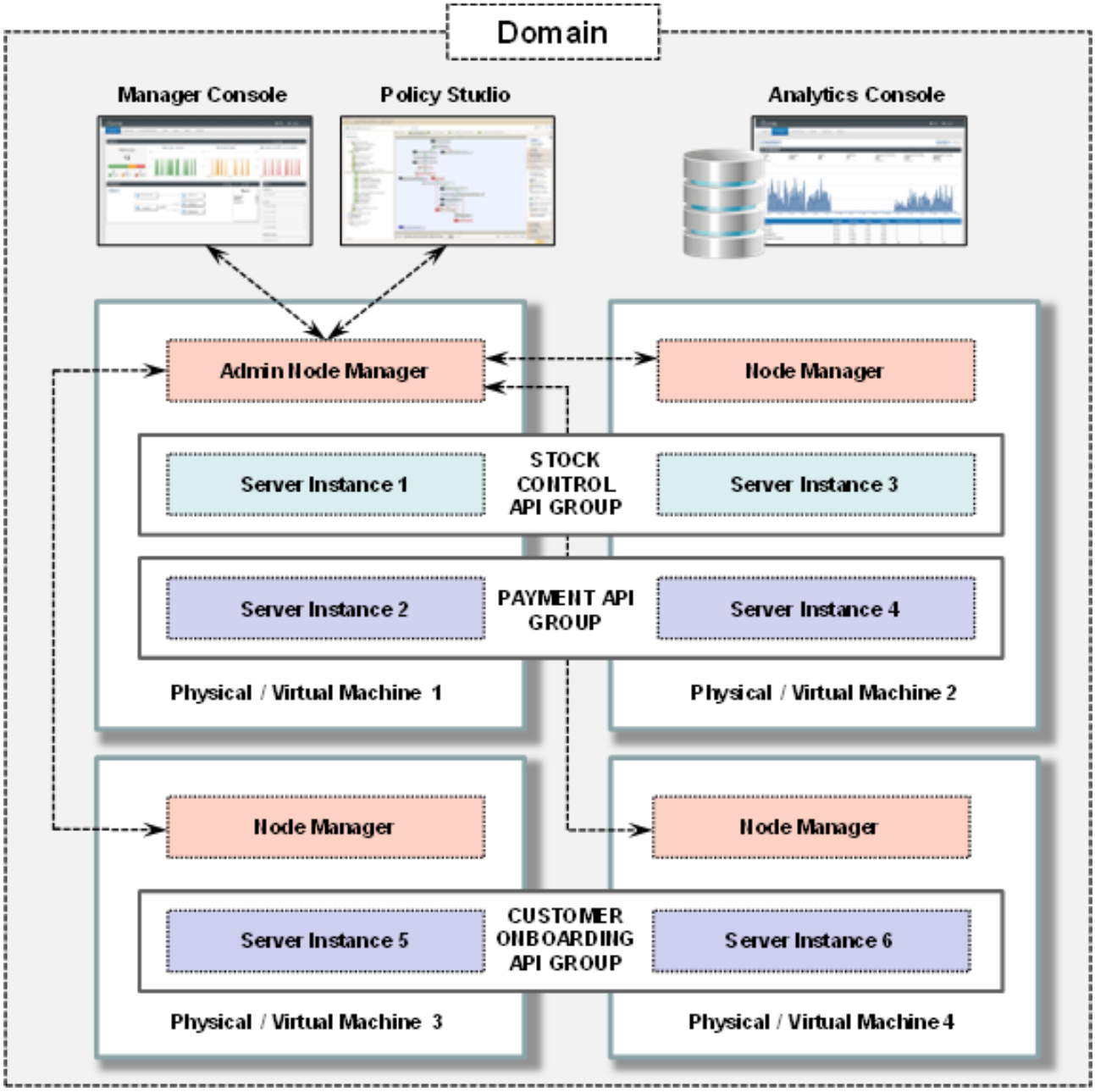


This domain-based architecture is described as follows:

- A single 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.
- A single 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.

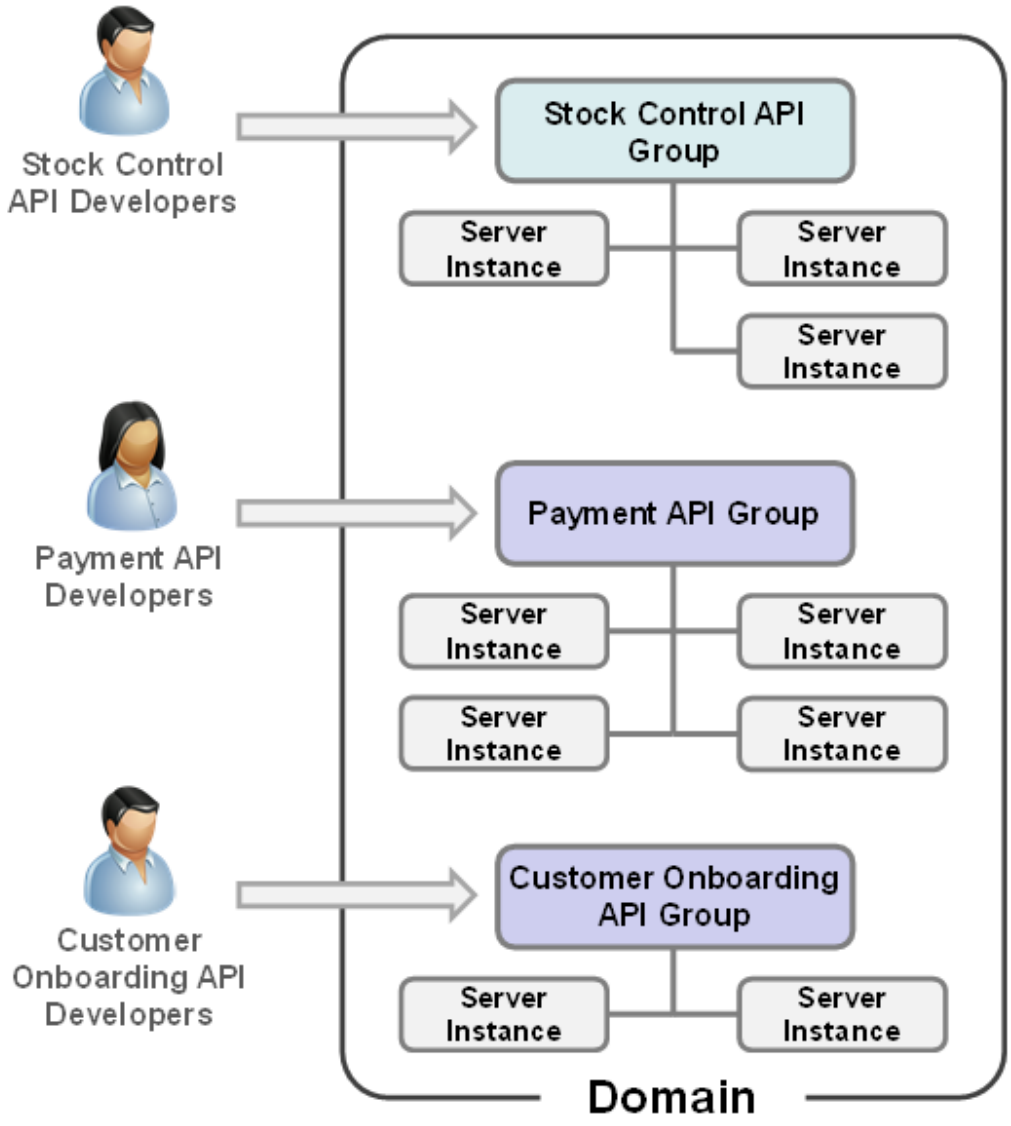
The following diagram shows a more complex domain with three groups distributed across four host machines:



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:

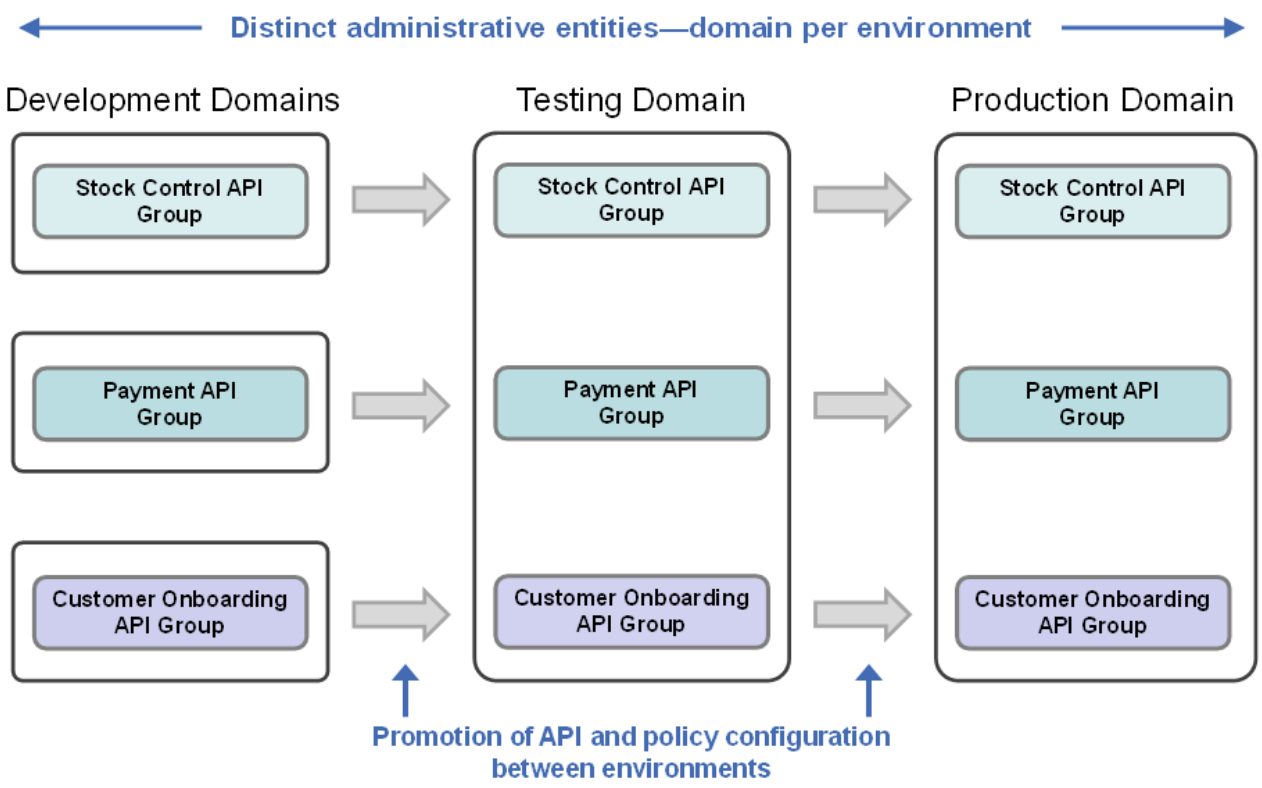


Enabling Virtualization

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:



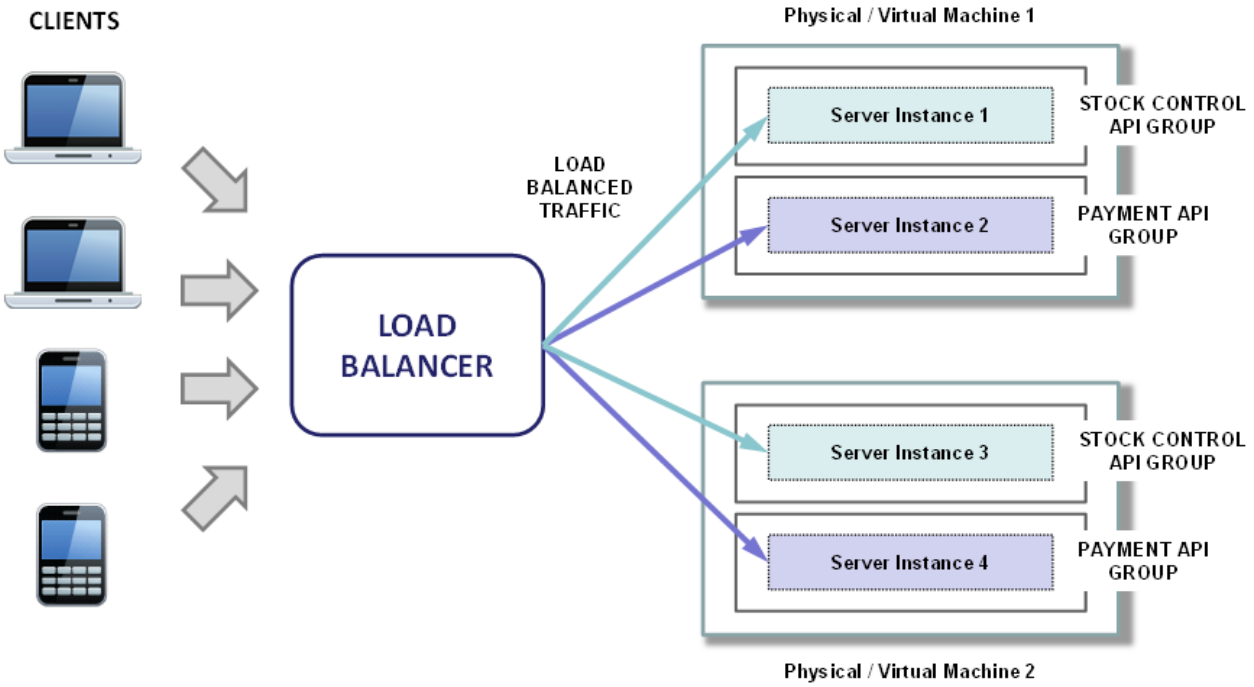
In this context, *promotion* refers to moving API Gateway configuration between environments and ensuring that environment-specific settings are properly configured. *Deployment* refers to the physical act of pushing configuration to an API Gateway instance (for example, using Policy Studio).

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 User Guide*.

Further Information

Overview

This chapter shows where to look for further information in the API Gateway documentation library.

API Gateway Documentation Library

The API Gateway documentation library includes the following user guides:

Document	Description
API Gateway Concepts Guide	Provides an overview of the API Gateway components, tools, and architecture.
API Gateway Installation and Configuration Guide	Describes how to install API Gateway components on all platforms, how to configure a domain and API Gateway instances, and how to upgrade API Gateway versions.
API Gateway User Guide	Describes the main API Gateway features (for example, all policies, filters, configuration options and so on), 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 Explorer User Guide	Describes how to use the API Gateway Explorer graphical tool to test REST-based APIs and SOAP-based Web Services.

Glossary of Terms

This Glossary explains terms used throughout the Oracle API Gateway documentation library.

A

Admin Node Manager
: The Admin Node Manager is an Oracle API Gateway component that is 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. There must be only one Admin Node Manager in each domain. 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).
See Also Node Manager , Domain.

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 clients.

API Gateway
: A server-side application that manages, delivers, and secures APIs. The Oracle API Gateway provides services such as the following:

  - API transformation
  - API control and governance
  - API security
  - API monitoring
  - API development lifecycle
  - API administration

B

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 then split to form 4 groups of 6-bits each. Each 6-bits gets translated to an encoded printable ASCII character, using a table lookup. The specification is described in RFC 2045.

C

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

cacert
: A file used to keep the root certificates of signing authorities. The default password is `changeit`. It is typically stored in `c:\jdk1.6\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, whereas certificate entries consist of just a certificate.

  Because you implicitly trust all the Certificate Authorities 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.

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.

D

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.
See Also Admin Node Manager .

DTD
: A Document Type Definition (DTD) defines a formal grammar for specifying the structure of an XML document. An XML document is said to be valid if it conforms to the syntax rules specified in the DTD.

F

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

H

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, thus adding the security capabilities of SSL/TLS to standard HTTP communications. For more details, see http://en.wikipedia.org/wiki/HTTP_Secure.

I

ISO
: The International Organization for Standardization (ISO) is a worldwide consortium of national standards bodies from more than 140 countries. The goal of ISO is to promote standardization in the world with a view to facilitating the international exchange of goods and services, and to develop cooperation in scientific, technological and economic activity.

J

JMS
: Java Message Service (JMS) is a messaging standard that enables application components based on the Java 2 Enterprise Edition (J2EE) to create, send, receive, and read messages. It enables the 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, Standard ECMA-262 3rd Edition, December 1999. 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.

K

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

L

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.

N

Node Manager
: A Node Manager is an Oracle 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.
See Also Admin Node Manager , Domain.

O

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.

P

PEM
: Privacy Enhanced Mail (PEM) was originally intended for securing Internet mail through authentication, message integrity, and confidentiality using various encryption techniques. Its scope was widened in later years 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
: PKCS#12 is a standard for storing private keys and certificates securely. It is used in (among other things) Netscape and Microsoft Internet Explorer with their import and export options.

Policy
: A policy is a network of message 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 above example.

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

Public Key
: The publicly-disclosable component of a pair of cryptographic keys used for asymmetric cryptography.

R

RBAC
: Role-Based Access Control (RBAC) restricts system access to authorized users based on their 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 assigned 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.

S

SAML
: Security Assertion Markup Language (SAML) is an XML standard for establishing trust between entities. SAML assertions can contain identity information about users (authentication assertions), and also information about the access permissions of users (authorization assertions). The basic idea is that when a user is authenticated at one site, that site issues a SAML authentication assertion and gives it to the user. The user can then use this assertion in requests at other affiliated sites. These sites need only check the details contained within the authentication assertion in order to authenticate the user. In this way, SAML allows authentication and authorization information to be shared between separate sites.

Signature
: A value computed with a cryptographic algorithm and appended 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 through corporate firewalls. SOAP defines the use of XML and HTTP to access services, objects and servers in a platform-independent manner. SOAP provides a way to access services, objects, and servers in a completely platform-independent manner. SOAP is a wire protocol that can be used to facilitate highly ultra-distributed architecture.

  SOAP is simple. It is nothing more and nothing less than a protocol that defines how to access services, objects, and servers in a platform-independent manner using HTTP (also SMTP) and XML. See the Simple Object Access Protocol Specification [http://www.w3.org/TR/SOAP/] for more details.

SSL
: Secure Sockets Layer (SSL) is an encrypted communications 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.

T

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

U

UDDI
: Universal Description, Discovery, and Integration (UDDI) is an XML-based lookup service for locating Web Services in an Internet scenario. See the Universal Description Discovery Integration (UDDI) standard [http://www.uddi.org/] for more details.

URI
: Uniform Resource Identifiers (URIs) are a platform-independent way to specify a file or resource somewhere on the

web. Strictly speaking, every URL is also a URI, but not every URI is also a URL. Two RFCs specify the format of a URI:

- RFC 2396: Uniform Resource Identifiers (URI): Generic Syntax [http://www.faqs.org/rfcs/rfc2396.html]
- RFC 2732: Format for Literal IPv6 Addresses in URIs [http://www.faqs.org/rfcs/rfc2732.html]

W

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. The operations and messages are described abstractly, and then 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 their messages regardless of what message formats or network protocols are used to communicate, however, the only bindings described in this document describe how to use WSDL in conjunction with SOAP 1.1, HTTP GET/POST, and MIME. See the Web Services Description Language Specification [http://www.w3.org/TR/wsdl] for more details.

X

X.509
: X.509 is the standard that defines the contents and data format of a public key certificate.

XKMS
: XML Key Management Specification (XKMS) uses the relative simplicity of 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. See the XML Key Management Specification [http://www.w3.org/TR/xkms/] for more details.

XML
: eXstensible 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 the way that is now possible with HTML. See the XML Specification [http://www.w3.org/TR/xml] for more details.

XPath
: XML Path Language (XPath) is a language that describes how to locate and process specific parts of an XML document. See the XML Path Language Specification [http://www.w3.org/TR/xpath] for more details.

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 will reference an XSL stylesheet, which will define how the XML elements of the document should be displayed as HTML. Therefore, a clear separation of content and presentation is achieved.

XSLT
: Extensible Stylesheet Language Transformations (XSLT) are used to convert XML documents into other formats.