

Oracle® Fusion Middleware

Oracle API Gateway Deployment and Promotion Guide
11g Release 2 (11.1.2.3.0)

April 2014

ORACLE®

Oracle API Gateway Deployment and Promotion Guide, 11g Release 2 (11.1.2.3.0)

Copyright © 1999, 2014, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services. This documentation is in prerelease status and is intended for demonstration and preliminary use only. It may not be specific to the hardware on which you are using the software. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to this documentation and will not be responsible for any loss, costs, or damages incurred due to the use of this documentation.

The information contained in this document is for informational sharing purposes only and should be considered in your capacity as a customer advisory board member or pursuant to your beta trial agreement only. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in this document remains at the sole discretion of Oracle.

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle Software License and Service Agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced, or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

30 April 2014

Contents

1. Introduction to API Gateway Promotion and Deployment	1
Overview	1
Environment Topology	1
Promotion and Deployment	3
API Gateway Configuration	3
API Gateway Configuration Packages	5
2. API Gateway Deployment and Promotion Tasks	7
Overview	7
Deploying in a Development Environment	7
Environmentalizing Configuration	7
Promoting Upstream (First Cycle)	8
Promoting Upstream (Subsequent Cycles)	9
Rolling Back Configuration	10
Multi-Site High Availability Environments	10
Passphrase-Protected Configuration	11
3. Configuring Package Properties	12
Overview	12
Configuring Package Properties	12
Customizing Package Properties in the Topology View	13
4. Example Promoting from Development to Testing Environment	15
Overview	15
Step 1—Policy Developer Edits Configuration and Deploys in Development Environment	15
Step 2—Policy Developer Environmentalizes the Environment-Specific Settings	16
Step 3—Policy Developer Saves Policy Package in Policy Studio for Promotion	20
Step 4—API Gateway Administrator Creates Testing Environment Package in Configuration Studio	21
Step 5—API Gateway Administrator Deploys Configuration to Testing Environment Group	24
Step 6—Further Configuration Updates in Testing Environment	24
5. Promoting and Deploying Using Scripts	26
Overview	26
Running Sample Scripts	26
Scripts for Environmentalizing Configuration	26
Scripts for Promoting Configuration	26
6. Externalizing API Gateway Instance Configuration	28
Overview	28
Configuring Environment Variables	28
Configuring Certificates as Environment Variables	29
7. Manage certificates and keys	31
Overview	31
View certificates and private keys	31
Configure an X.509 certificate	31
Create a certificate	31
Import certificates	32
Configure a private key	32
Global options	33
Manage certificates and keystores	33
Export certificates to a keystore	33
Configure key pairs	33
Add a key pair	33
Manage OpenSSH keys	34
Configure PGP key pairs	34
Add a PGP key pair	34
Manage PGP keys	34
8. Manage API Gateway users	36

Overview	36
API Gateway users	36
Add API Gateway users	36
API Gateway user attributes	36
API Gateway user groups	37
Add API Gateway user groups	37
Update API Gateway users or groups	37

Introduction to API Gateway Promotion and Deployment

Overview

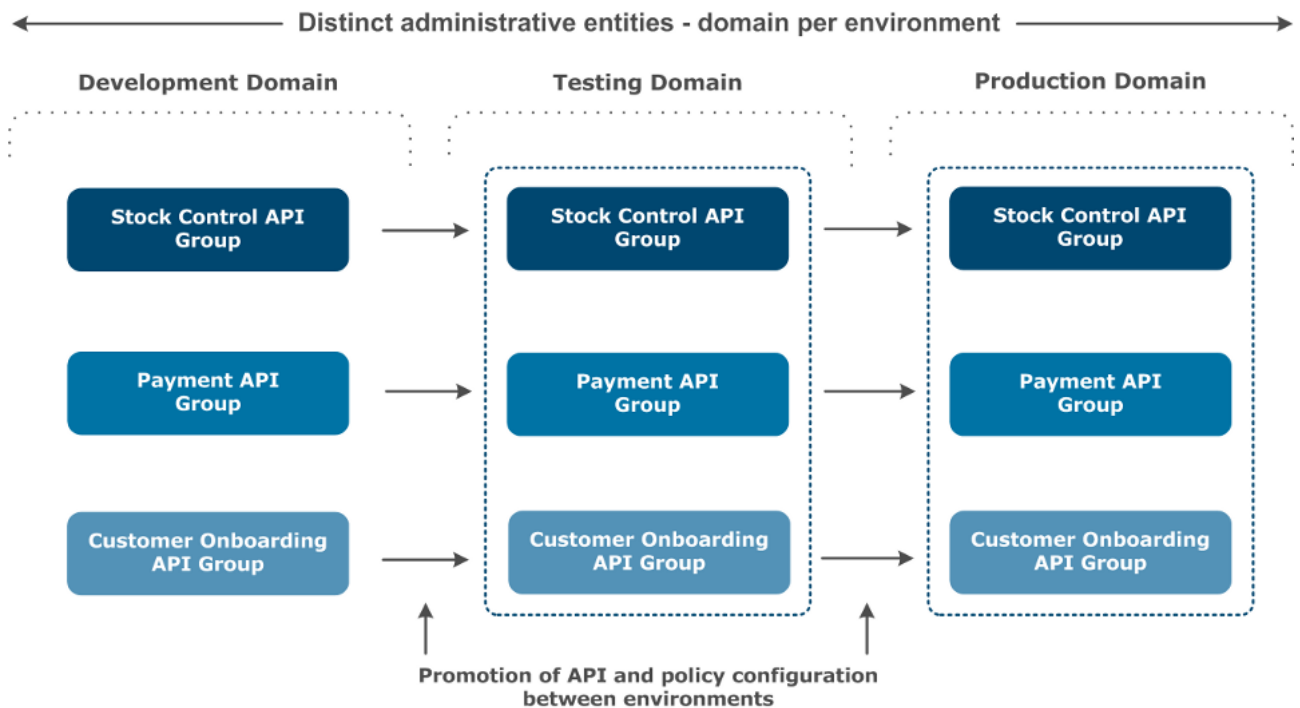
This topic introduces the concepts in the deployment and promotion of API Gateway configuration. A typical enterprise-level customer will have several environments through which an API Gateway configuration will move from development to production. For example, this typically includes completely separate development, testing, and production domains. Promotion refers to the act of moving API Gateway configuration from one environment to another, and configuring environment-specific values so that the configuration can be deployed in each environment. For details on general API Gateway concepts, see the *API Gateway Concepts Guide*.

Environment Topology

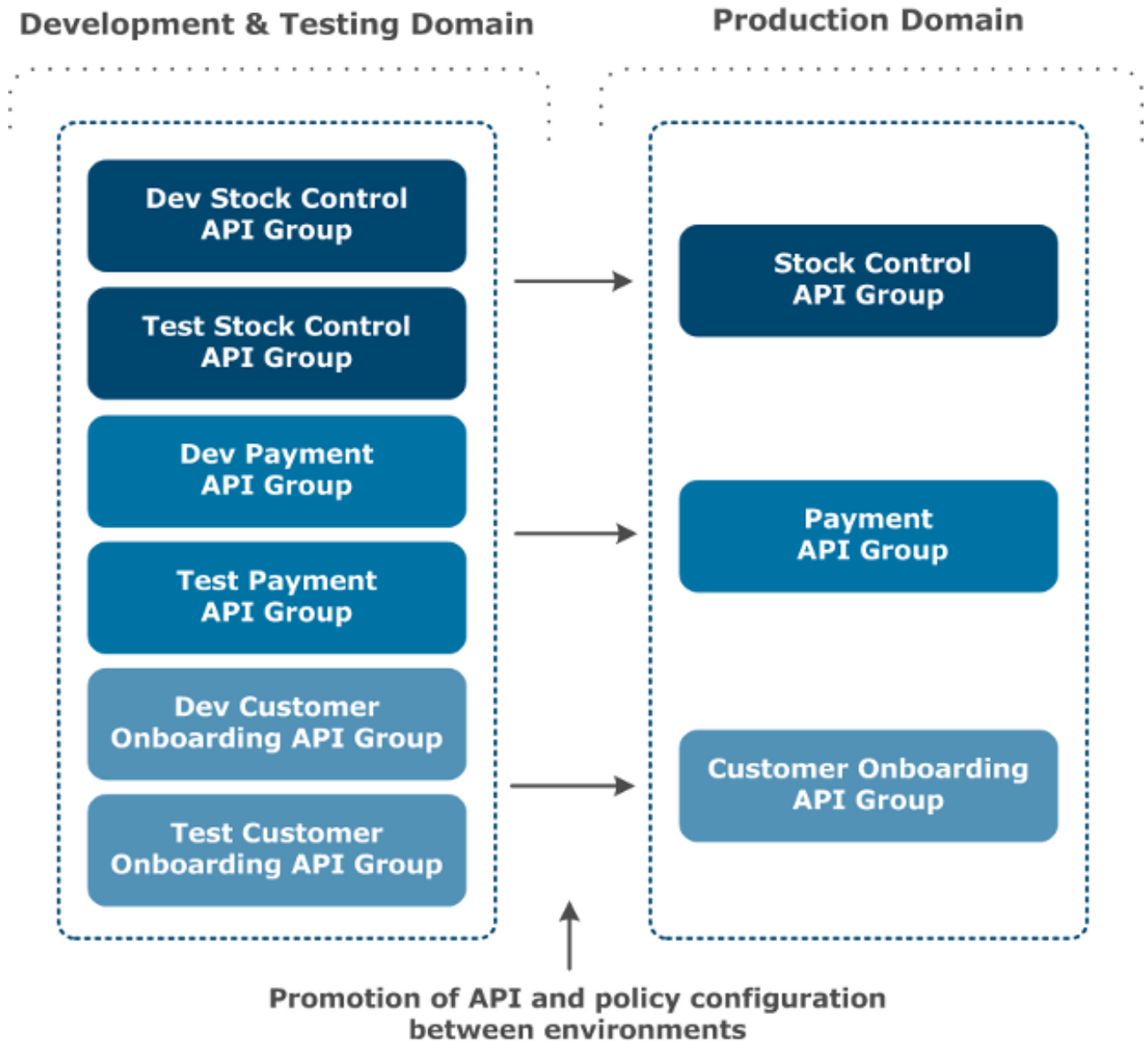
In a typical environment topology, each environment is implemented as a completely separate API Gateway domain. The exact mapping of environments to domains is determined by how each environment is administered, and which users have access rights.

Environments are distinct administrative entities in which only certain users have the privileges to perform operations. For example, only Production Operations staff have access to the Production environment. In the API Gateway architecture, a domain is a distinct administrative entity for managing groups of API Gateways. For example, the Production environment is implemented as a distinct Production domain to which only Production Operations staff have access.

In the following diagram, each environment is implemented as a distinct API Gateway domain. Developers work in their own Development environments, and then promote their API Gateway configurations to a central Testing team that performs testing in a single Testing environment. When testing is complete, the Testing team promotes the API Gateway configurations to the Production Operations team for deployment in the Production environment. Development, Testing, and Production Operations teams have access to their respective environments only. Therefore, each environment should be implemented as a distinct domain.



The API Gateway configuration is deployed to a group of API Gateways. Therefore, each domain consists of the required API Gateway groups to run the configurations. In the following diagram, the Development and Testing teams work in the same environment with common access to all API Gateway configurations. Therefore, in this case, there is a single domain for all the Development and Testing API Gateway groups.



Note

The API Gateway does not mandate a specific environment-to-domain configuration, and is flexible enough to work with any architecture. However, you should manage your environments in an environment and domain topology. Implementing each environment as a distinct API Gateway domain is a good starting point.

Promotion and Deployment

In a multi-environment topology, *promotion* refers to physically moving an API Gateway configuration between environments. For example, this may involve using FTP to transfer a configuration file, or loading and retrieving the file in a Configuration Management (CM) repository. In addition, promotion involves configuring environment-specific settings for the target environment (for example, users, certificates, and external connections to third-party systems). This is known as *environmentalization*.

Promotion typically involves two distinct tasks performed by different user types:

- In the downstream Development environment, the policy developer prepares the configuration for promotion to upstream environments (for example, Testing and Production). This involves deciding what settings are environment-specific, and assumes expertise in policy development and configuration tools such as Policy Studio.
- The upstream user takes the configuration prepared by the policy developer, creates the environment-specific configuration, and deploys it. This is typically performed by an API Gateway administrator in upstream environments. The Configuration Studio tool used for this promotion step is designed for the skills of upstream administrators, and does not assume expertise in policy development and configuration.

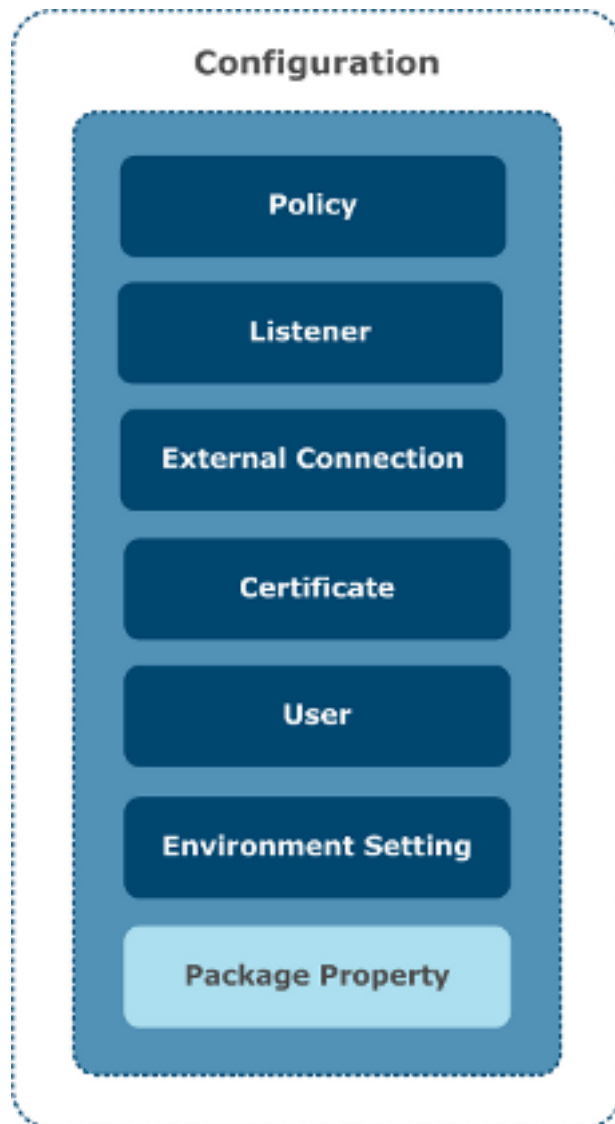
Deployment refers to deploying configuration to an API Gateway group in a local domain. For example, you can deploy using the following tools:

- Policy Studio in a Development environment
- API Gateway Manager in a Testing environment
- Scripts in a Production environment (for example, `managedomain` or a custom script)

For more details, see [API Gateway Deployment and Promotion Tasks](#).

API Gateway Configuration

API Gateway configuration consists of the following types of information:



These component configuration types are described as follows:

Configuration Type	Description	Environment Specific
Policy	Policy rule definitions and configuration	Some may be environment specific
Listener	Configuration for listeners used by policies (for example, for HTTP, JMS, or TIBCO)	Some environment specific
External Connection	Settings for external configuration (for example, Database Connection or Authentication Repository)	Some environment specific
Certificate	Security certificates and keys used by policies	All environment specific
User	Local username/password store for	All environment specific

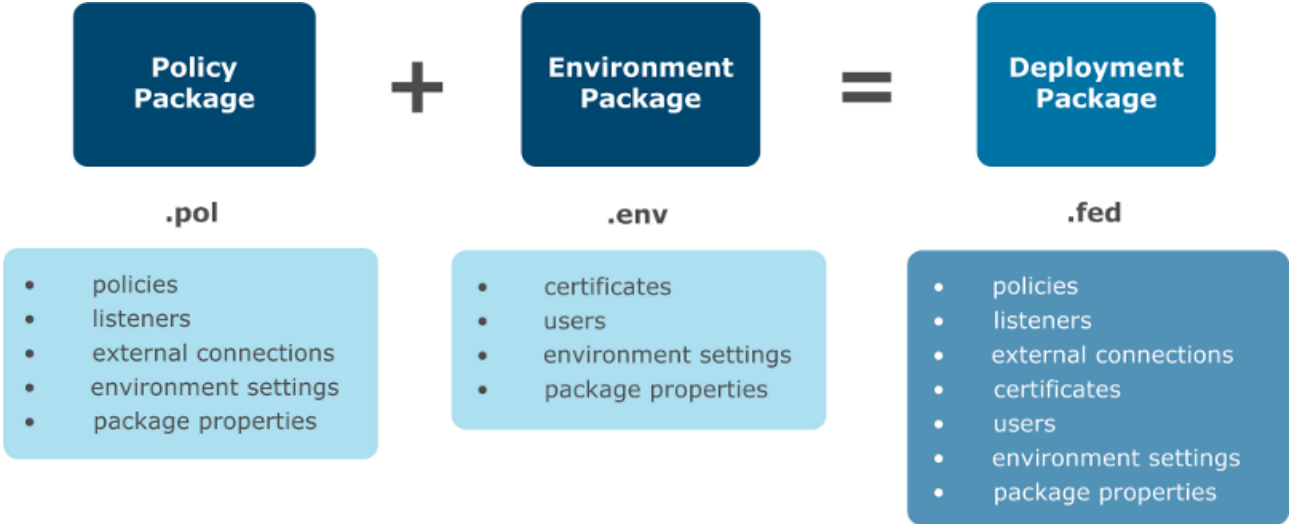
Configuration Type	Description	Environment Specific
	API client authentication	
Environment Setting	Environment-specific settings for environmentalized configuration in the policy, listener, and external connection configuration	All environment specific
Package Property	Name-value pairs used to describe the contents of the configuration package (for example, Version, ID, or Timestamp).	Some environment specific

API Gateway Configuration Packages

The API Gateway deployment and promotion tools bundle the API Gateway configuration into the following configuration package files:

Package	Description	Used When
Deployment Package (.fed)	Contains all API Gateway component configuration (policy, listener, external connection, user, certificate, and environment setting). Implemented as a .fed file. This contains all the data that would be contained in separate policy and environment packages combined.	Used by the policy developer in Policy Studio during the iterative development cycle to deploy all configuration. For more details, see the section called "Deploying in a Development Environment" .
Policy Package (.pol)	Contains the policy, listener, external connection, and environment setting configuration. Implemented as a .pol file. The environment settings in the .pol file contain a list of what has been environmentalized in the policy, listener, and external connection configuration. It does not contain the environment-specific values.	Used when promoting APIs and policy configuration to upstream environment for example, testing or production). Its contents remain unchanged in the upstream environment. For more details, see the section called "Environmentalizing Configuration" .
Environment Package (.env)	Contains the user, certificate, and environment setting configuration. Implemented as an .env file. The environment settings in the .env file contain a list of what has been environmentalized in the policy, listener, and external connection configuration, along with the environment-specific values.	Environment-specific settings used in upstream environments only (for example, testing or production). For more details, see the section called "Promoting Upstream (First Cycle)" .

The combined contents of the policy package and environment package are equivalent to the contents of the deployment package, which contain all API Gateway configuration:



For more details on package properties, see [Configuring Package Properties](#).

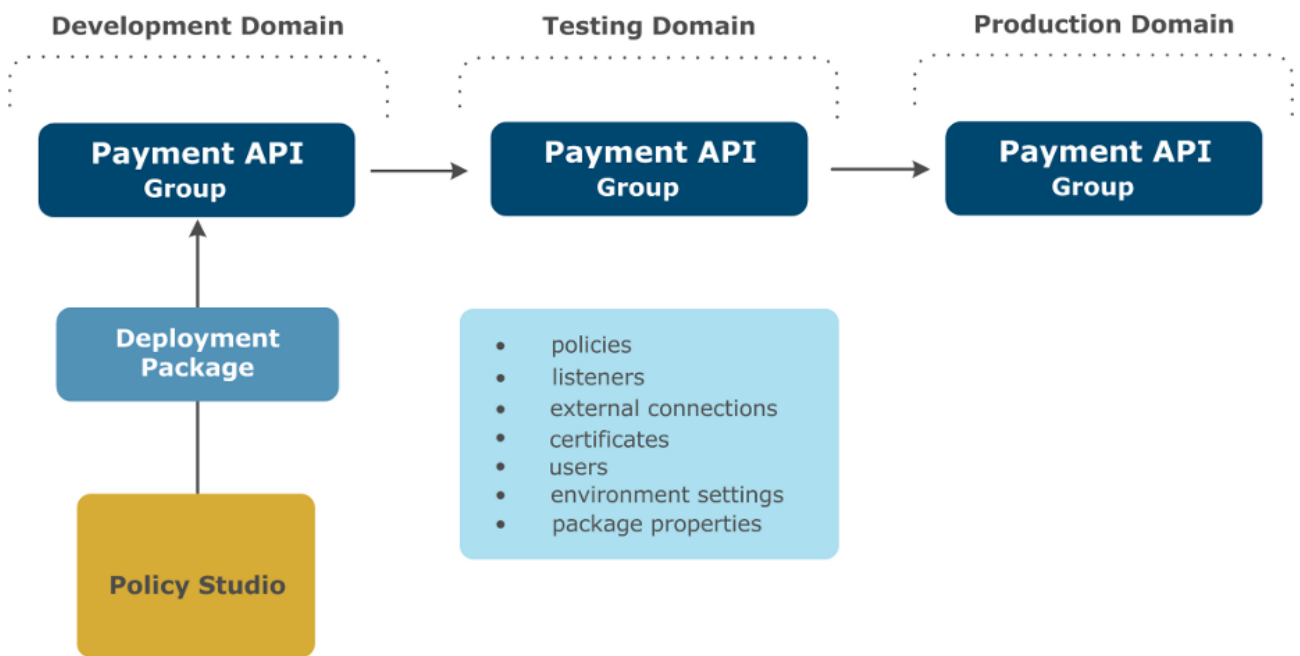
API Gateway Deployment and Promotion Tasks

Overview

This topic describes the tasks, tools, and architecture used in API Gateway deployment and promotion. It explains the breakdown of tasks performed by a policy developer in a development environment, and the tasks performed by an API Gateway administrator in an upstream environment (for example, testing or production).

Deploying in a Development Environment

In a development environment, the policy developer works in a continuous cycle of iterative development, deployment, and testing. In this environment, it makes sense to keep all API Gateway configuration in a single package. This enables the policy developer to deploy the API Gateway configuration directly from Policy Studio in a single *deployment package*. The following diagram shows an example environment topology:



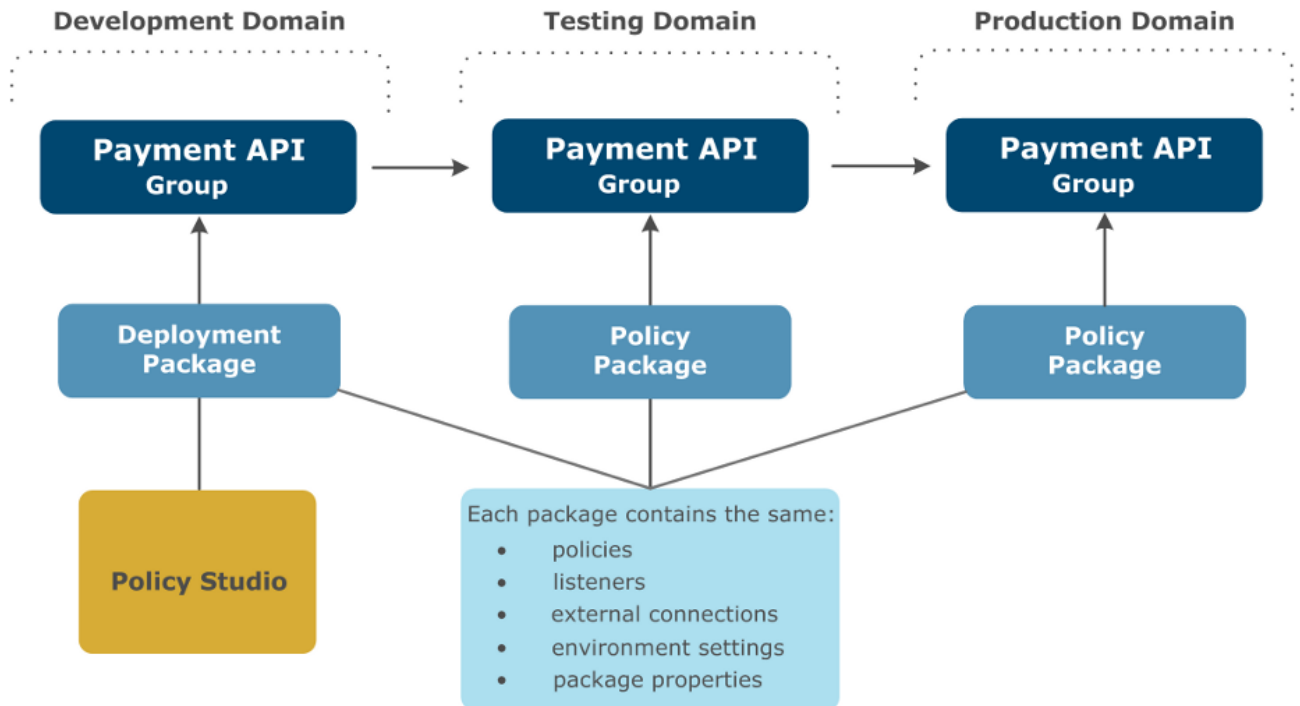
The deployment package contains the entire API Gateway configuration, and is implemented as a `.fed` file.

Environmentalizing Configuration

When development is complete, the policy developer must prepare the configuration for promotion to upstream environments. This involves *environmentalizing* the configuration that will require environment-specific settings in upstream environments. The policy developer performs the following tasks in Policy Studio:

- Selects the policy, listener, and external connection configuration settings that are environment specific.
- Enters values for these environment-specific settings to ensure that the configuration remains deployable in the Development environment. These environment-specific settings are contained in the environment settings in the deployment package. If you have already entered values for these settings, these are used so you do not have to manually re-enter them.
- Exports a *policy package* (`.pol` file) on disk for promotion. For example, this enables you to FTP the file to the upstream environments, or to load the file into a CM repository.

The following diagram shows an example environment topology:



The policy package that is exported for promotion is implemented as a `.pol` file. This file should remain unchanged when it is promoted to upstream environments.

Promoting Upstream (First Cycle)

A *first cycle* promotion refers to promoting to an upstream environment in which no previous promotions have been performed. This means that the upstream environment is still running the default factory configuration that is installed with the API Gateway. In this case, there is no existing upstream *environment package* (`.env`) to load into the Configuration Studio at promotion time.

Creating an Environment Package

In an upstream environment (for example, Testing), the API Gateway administrator uses the Configuration Studio to create an environment package that is specific to their environment. Because this is the first promotion cycle, the administrator opens the policy package (`.pol`) received from the Development environment, and performs the following tasks:

- Specifies values for the environment-specific settings selected in the Development environment (for example, policy, listener, and external connections).
- Imports or creates certificates and keys.
- Defines users and user groups.
- Exports the environment package to a file on disk. The environment package is implemented as an `.env` file. For version history and rollback, you could also load the file into a CM repository.

Deploying the Policy and Environment Packages

When the environment package has been created, the administrator can use the API Gateway Manager web console or scripts to deploy both the policy package from the Development environment and the newly created environment package. Each environment will have its own version of the `.env` file containing environment-specific settings, certificates,

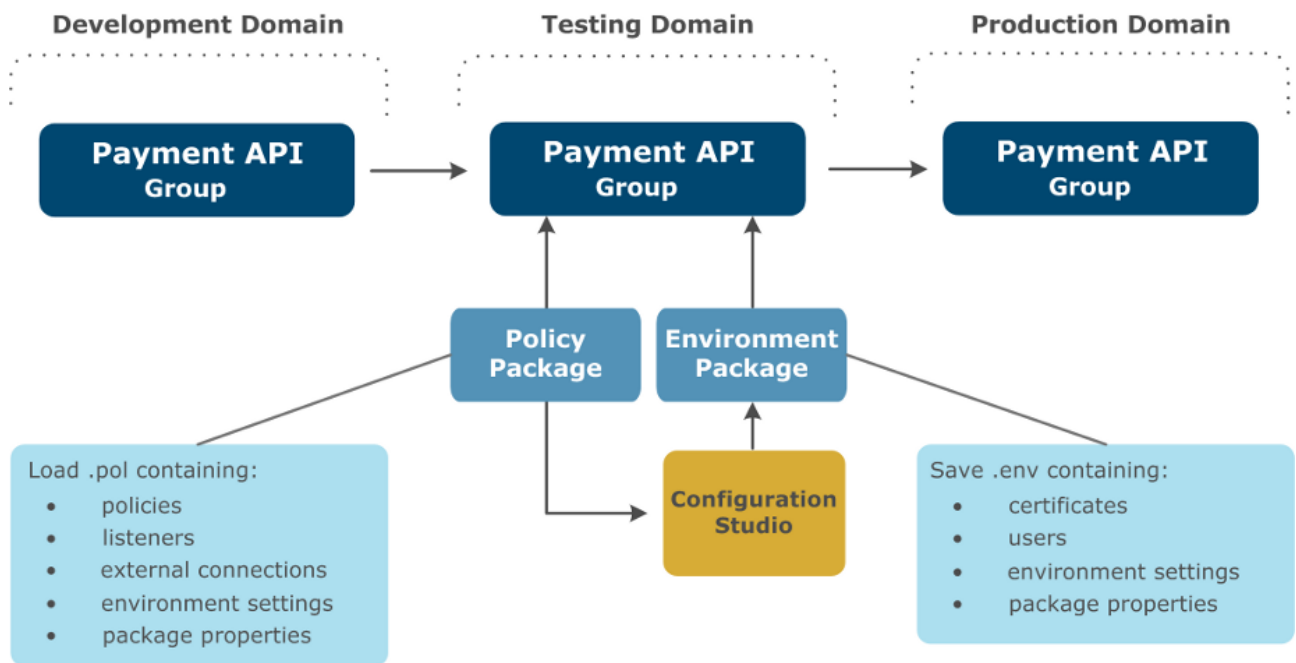
users, and so on. This constitutes a full deployable configuration when combined with the unmodified `.pol` file from the Development environment.



Note

Alternatively, the administrator can save a deployment package (`.fed`) from the Configuration Studio, which merges the policy and environment package data. If you are not concerned with moving an unmodified policy package from the Development environment to all upstream environments, you can save a single `.fed` file, and deploy this using the API Gateway Manager or scripts (for example, if you want a single file for convenience).

The following diagram shows an example environment topology:



Note

The environment settings in the environment package (`.env`) override the environment settings in the policy package (`.pol`). The environment settings in the policy package indicate settings for which you need to specify environment-specific values.

Promoting Upstream (Subsequent Cycles)

A *subsequent cycle* promotion refers to promoting to an upstream environment that has already had configuration promoted to it (any number of times). In this case, there is an existing version of the upstream environment package (`.env`) to load into the Configuration Studio at promotion time.

Creating the Environment Package

In the upstream environment, the API Gateway administrator uses the Configuration Studio to create an environment package specific to their environment that contains all the environment-specific settings, certificates, and so on. This is required for the new policy package received from the Development environment. Because this is not the first promotion cycle, there is already an environment package deployed in this environment. The administrator must merge this with the

new policy package from the Development environment, which enables reuse of environment-specific settings already entered.

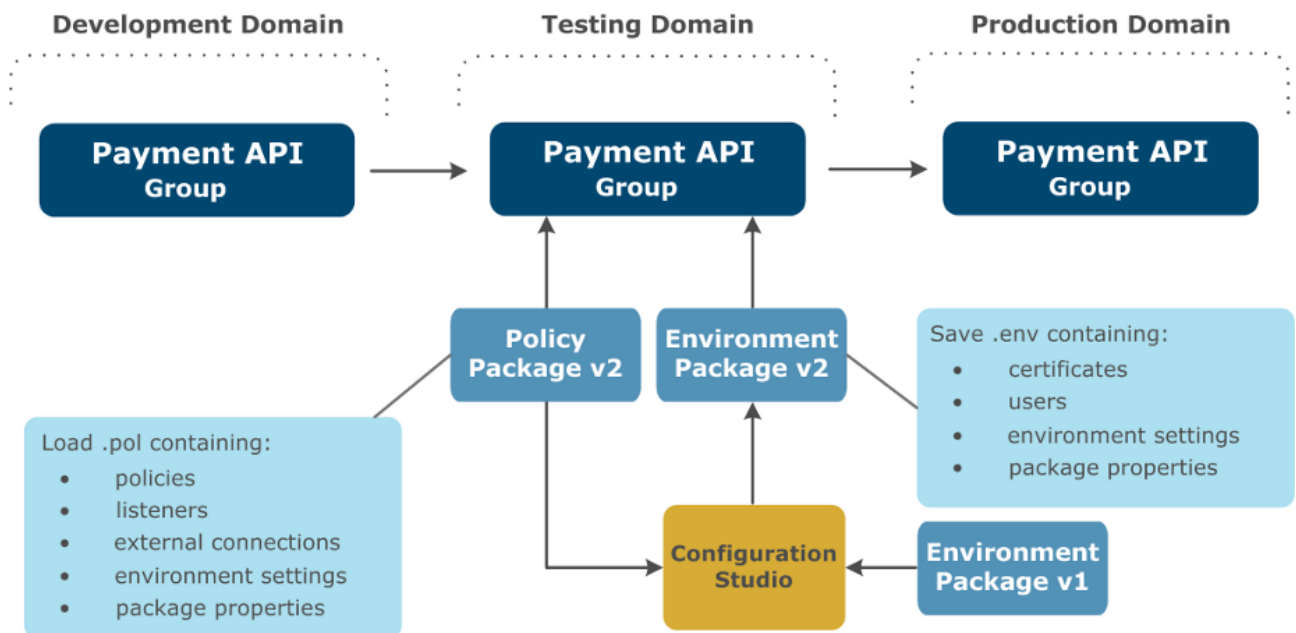
The administrator opens the new policy package from the Development environment, and the environment package currently deployed in their environment. Opening these `.pol` and `.env` files displays a merged view of the environment settings. The administrator then performs the following tasks:

- Specifies values for new environment-specific settings required by the new policy package from the Development environment.
- Updates values for environment-specific settings that previously existed (if necessary).
- Adds or removes certificates and keys.
- Adds or removes users and user groups.
- Exports the environment package to a file on disk. Alternatively, for version history and rollback, you could load the file into a CM repository.

Deploying the Policy and Environment Packages

When the environment package has been created, the API Gateway administrator can then deploy both the policy package received from the Development environment, and the new environment package using the API Gateway Manager web console, or using scripts.

The following diagram shows an example environment topology:



Rolling Back Configuration

You must ensure to maintain a copy of previous configuration versions (policy and environment packages) in case you need to roll back and deploy an earlier configuration version. For example, you could use a Configuration Management (CM) repository to manage and roll back configuration package versions.

Multi-Site High Availability Environments

Some environments may require different environment values for connections, certificates, and so on (for example, a remote High Availability (HA) site for a production environment in an active/passive configuration). In this scenario, the

primary site is active processing requests. The remote site is the backup passive configuration, deployed but not processing requests, and only becomes active if the primary site goes down. The same API Gateway configuration is deployed in both sites. Each site could be a separate domain, or one domain with different groups for each site. But specific environment values could be different for each site. For example, the remote site may connect to a different backup authentication server.

When the administrator receives the policy package (`.pol`) from the downstream environment, they can use Configuration Studio to create separate environment packages (`.env`) for the primary site and the remote site. The only difference between both environment packages is in the environment values required. In the primary site, the administrator deploys the policy package and the primary site environment package. In the remote HA site, the administrator deploys the same policy archive and the remote site environment package.

Passphrase-Protected Configuration

When promoting encryption 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 details on setting encryption passphrases, see the *API Gateway Administrator Guide*.

Configuring Package Properties

Overview

The API Gateway configuration package files include property files that contain name-value pairs describing the package contents, and which are known as *package properties*. This topic describes these properties, and explains how to configure default and custom package properties using the Policy Studio and Configuration Studio tools. It also shows how to customize the package properties that are displayed in the **Topology View** in Policy Studio.

The API Gateway bundles its configuration in the following package formats:

- Deployment package (.fed)
- Policy package (.pol)
- Environment package (.env)

For a description of each package, see [the section called "API Gateway Configuration Packages"](#).

Configuring Package Properties

All three API Gateway configuration package formats (.fed, .pol, and .env) contain property name-value pairs, which you can use to describe the package contents. These package property values are stored in package property files (.mf). A deployment package (.fed) has two sets of package properties, one associated with the policy-related configuration, and one associated with the environment-related configuration. Policy packages (.pol) and environment packages (.env) have a single set of properties each.

Default Properties

The default set of package properties that can be edited includes the following:

Property	Description
Name	Name associated with the configuration (for example, Payment API Configuration)
Description	Description associated with the configuration (for example, API Gateway configuration settings for the Payment API)
Version	Configuration version (for example, v3)
VersionComment	Comment relating to the configuration version (for example, Added SSL)

These fields are all free format text fields. You can set them to an empty value, or remove them completely, as required. The set of properties is completely customizable. You can add your own custom properties if required.

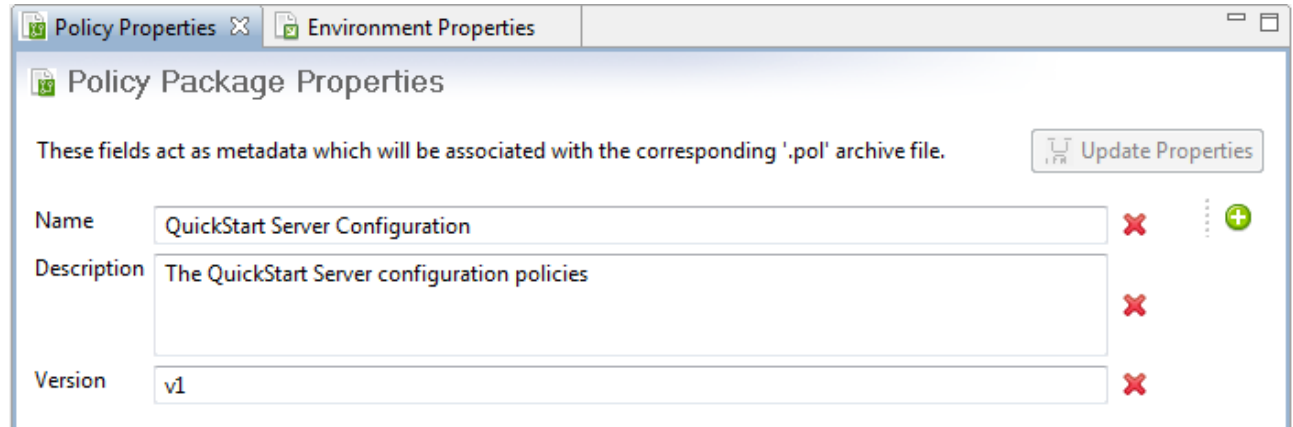
Read-Only Properties

The package also includes the following read-only, system-controlled package properties:

Property	Description
Id	A unique ID for the package
Timestamp	The time that the package was written

Configuring Properties in Policy Studio

When editing an API Gateway configuration in Policy Studio, you can add, edit, or remove the policy properties and environment properties in the **Package Properties** tree node. For example, the following screen is displayed when you select **Policies**:



To add a new package property, click the green button (+ icon) on the right of the screen. Similarly, to delete package property, click the red button (x icon).

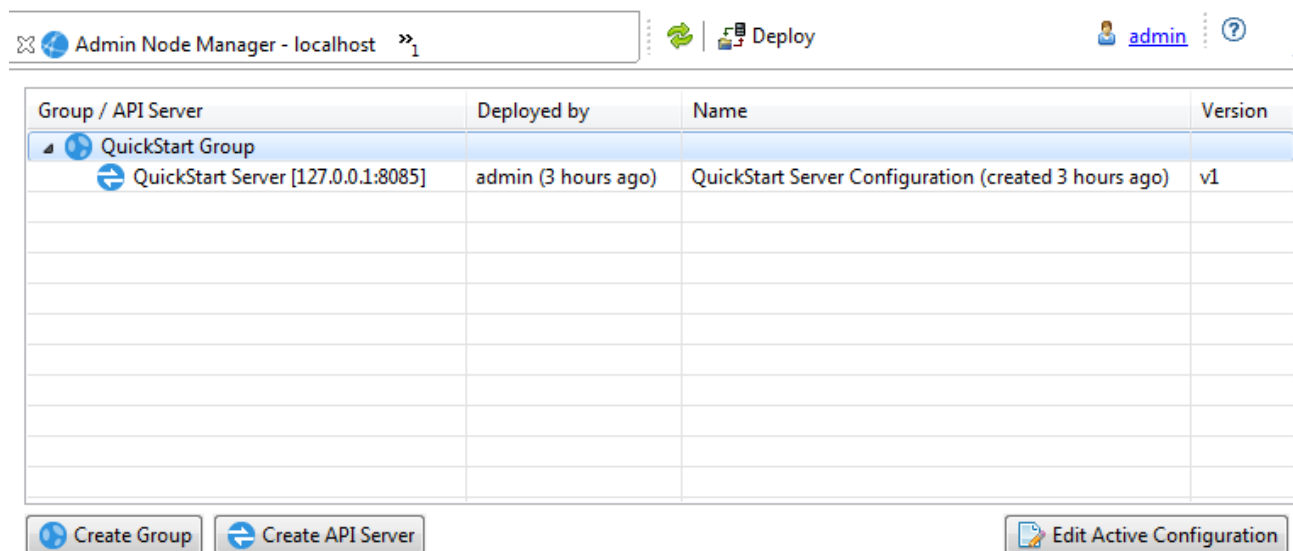
Configuring Properties in Configuration Studio

You can edit environment properties in the Configuration Studio using a similar screen. You can only view policy properties because these are read-only.

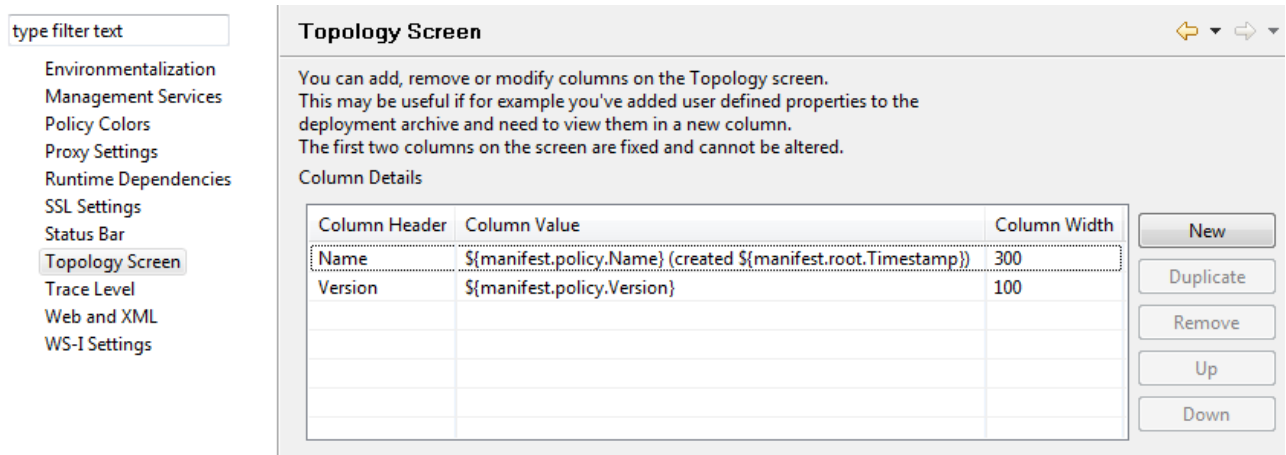
Package property values are deployed to an API Gateway along with the entire configuration in the relevant configuration package structure.

Customizing Package Properties in the Topology View

This section explains how to display package properties in the **Topology View** in Policy Studio. The default view is displayed as follows:



The **Group / API Gateway** and **Deployed by** columns in the table are read only. You can customize all other columns to show package property values by selecting **Window > Preferences > Topology Screen** from the main menu. The following screen shows the default customization settings:



Specifying Default Column Values

You can specify the following default package property values in the **Column Value** column:

- `${manifest.policy.Name}`
- `${manifest.policy.Description}`
- `${manifest.policy.Version}`
- `${manifest.policy.VersionComment}`
- `${manifest.env.Name}`
- `${manifest.env.Description}`
- `${manifest.env.Version}`
- `${manifest.env.VersionComment}`
- `${manifest.root.Id}`
- `${manifest.root.Timestamp}`

Adding Custom Column Values

You can also add custom package property values in the **Column Value** column. For example, perform the following steps:

1. Click **New**.
2. Double-click the value in **Column Header**, and enter `MyCustomPolicyField`.
3. Double-click the value in **Column Value**, and enter `${manifest.policy.MyCustomPolicyField}`.

Similarly, to add a custom environment package property, add a property with a **Column Header** of `MyCustomEnvField`, and a **Column Value** of `${manifest.env.MyCustomEnvField}`.

Customizing the Topology View Table

You can add, edit, remove, or reorder the columns displayed in the **Topology View** using the **Topology Screen** preferences. You can also specify the **Column Width** displayed.

Example Promoting from Development to Testing Environment

Overview

This topic describes a step-by-step example of promoting configuration from a Development environment to a Testing environment. If further promotions to more upstream environments are required, you can repeat steps 4 and 5 only.



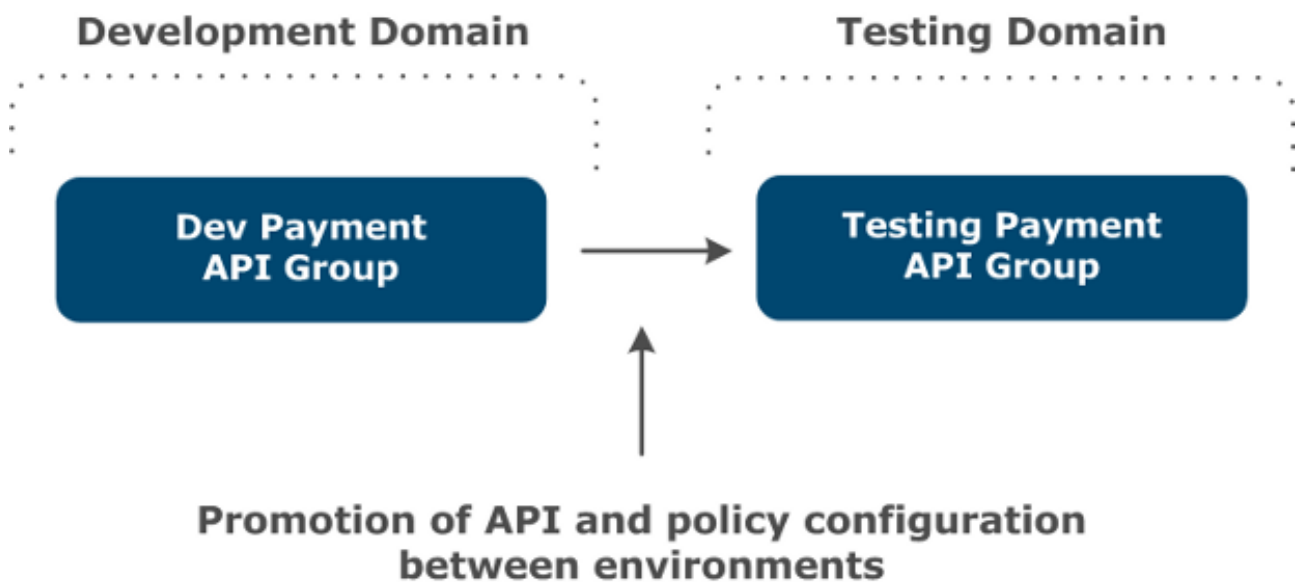
Note

Some environments (for example, Testing and Production) may be exact copies of each other, which enables you to deploy the same environment package to both environments. In these cases, repeat step 5 only.

Example Topology

This example assumes the following simple environment topology:

- A domain is configured in the Development environment with a group of API Gateways named **Dev Payment API Group**.
- A domain is configured in the Testing environment with a group of API Gateways named **Testing Payment API Group**. The configuration developed in the Development environment must be promoted to the servers in this group.



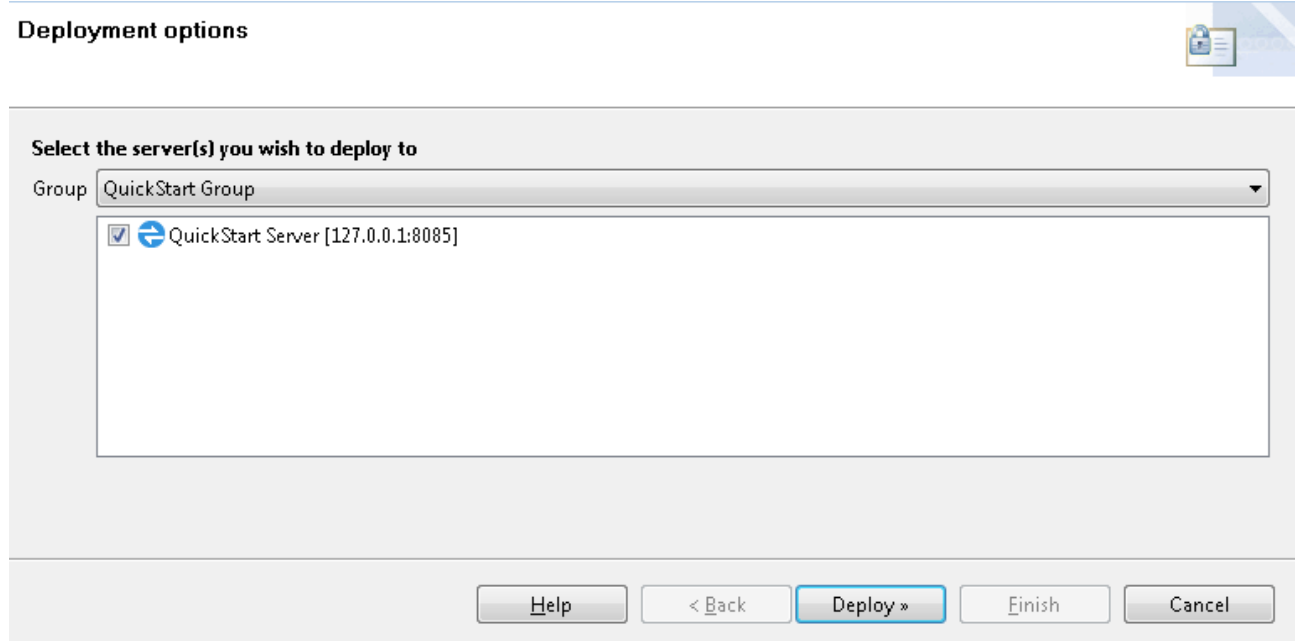
Step 1—Policy Developer Edits Configuration and Deploys in Development Environment

The policy developer in the Development environment uses Policy Studio to create policies, users, certificates, listeners, and so on as required for the business solution they are developing. The policy developer will most likely edit and deploy the configuration to the **Dev Payment API Group** repeatedly until they are finished with the configuration.

Deploying in Policy Studio

The policy developer deploys the configuration in Policy Studio by clicking the **Deploy** button in the toolbar when editing

the configuration. This displays the following screen:



Select the **Group** and API Gateway instance(s) to which you wish to deploy, and click **Deploy**. This uploads the configuration to the Admin Node Manager for the group, and then deploys it to the API Gateway instance(s) on the host(s).



Note

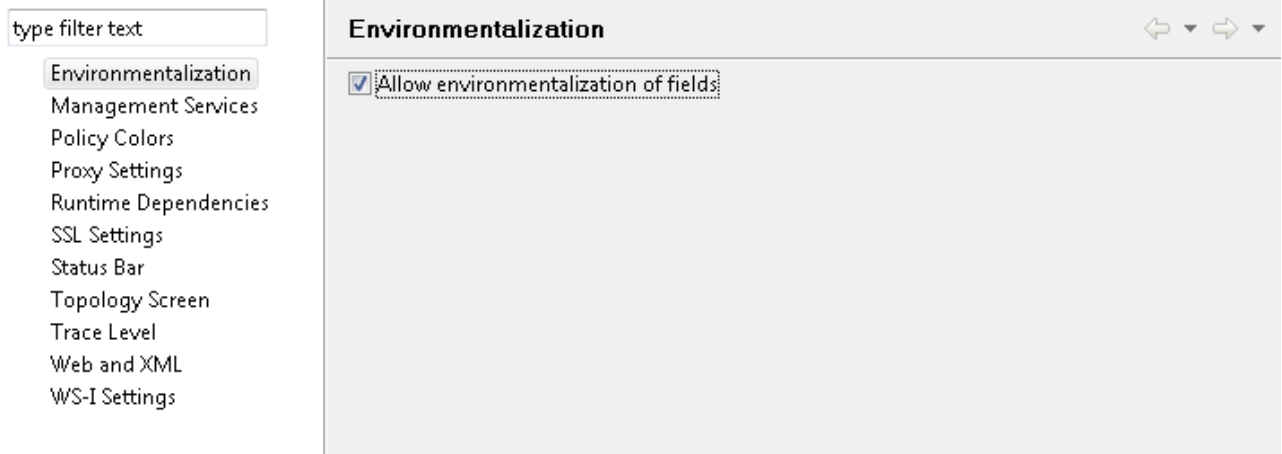
This simple example shows a group with a single API Gateway instance. Groups will typically have multiple API Gateway instances. If some Node Managers in the group are not running, do not select the API Gateways on those hosts, and you can still deploy to the other hosts in the group.

Step 2—Policy Developer Environmentalizes the Environment-Specific Settings

When the policy developer is developing policies in an iterative manner as described in Step 1, they may choose not to consider what settings are environment-specific yet, or they may choose to environmentalize these settings as they go along. Either way, before promotion can occur, all settings that are environment-specific must be environmentalized to prepare the configuration for promotion to upstream environments.

Displaying Environmentalized Configuration

You must first enable the display of configuration settings that are assigned for environmentalization in Policy Studio. Select **Window > Preferences > Environmentalization** in the main menu, and select **Allow environmentalization of fields**.



Environmentalizing Configuration Settings

For example, the developer chooses to environmentalize the following settings in the configuration:

- **URL, User Name, and Password** fields in a **Default Database Connection**
- **URL** field in a **Connect to URL** filter in a policy named **GetProducts**
- **X.509 Certificate** field in an HTTPS Interface named **OAuth 2.0 Interface**
- **URL, User Name, Password, and Signing Key** fields in a **Sample Active Directory Connection**

The policy developer edits the **Database Connection, Connect to URL** filter, **HTTPS Interface, and LDAP Connection**. You can select the **Environmentalize** button (🌐 globe icon) on the right of the fields shown in the following examples. Alternatively, you can environmentalize a selected field by pressing Ctrl-E.



Tip

You must give the field focus before the **Environmentalize** button is displayed.

For example, select **External Connections > Database Connections > Default Database Connection > Edit**, and click **Environmentalize** in the appropriate fields:

Select **Policies > QuickStart > Virtualized Services > REST > Get Products > Connect to Heroes' REST Service**, and click **Environmentalize** in the **URL** field:

Basic	Request Details
Name:	Connect to Heroes' REST Service
URL:	

Select **Listeners > API Gateway > OAuth 2.0 Services > Ports > OAuth 2.0 Interface**, and click **Environmentalize** in the **X.509 Certificate** field:

Network	Mutual Authentication	Traffic Monitor	Advanced	Advanced (SSL)
Name:	OAuth 2.0 Interface			
Port:	\${env.PORT.OAUTH2.SERVICES}			
Address:	*			
Protocol:	IPv4			
Trace level:	From System Settings			
<input type="checkbox"/> Enable interface				
X.509 Certificate				

Select **External Connections > LDAP Connections > Sample Active Directory Connection > Edit**, and click **Environmentalize** in the appropriate fields:

Name:	Sample Active Directory Connection
URL:	
Cache Timeout:	300000
Cache Size:	8
Authenticate LDAP Requests	
Type:	Simple
User Name:	
Password:	
Realm:	
<input checked="" type="checkbox"/> SSL Enabled	
Signing Key:	

When configuration settings have been environmentalized, the corresponding node in the Policy Studio tree is displayed with a globe icon and bold text.

Viewing Environment Settings

After environmentalizing the fields, the following nodes are available under the **Environment Settings** tree in Policy Studio:



Updating Environment Settings

Assuming the policy developer has already entered values for the fields that they have selected to be environmentalized, these values are automatically specified in the **Environment Settings** tree. If the developer wishes to update the setting values for the Development environment, they must do so using the **Environment Settings** tree.

For example, using the example environmentalized settings, the following screen is displayed when you select **Environment Settings > External Connections > Database Connections > Default Database Connection**:

Deselecting Environment Settings

If the policy developer no longer wants to environmentalize a setting, they can right-click its node in the **Environment Settings** tree, and select **Remove**. This also deselects the field in the screen used to edit the configuration setting (for example, Database Connection). The value configured before environmentalization is displayed again.

Alternatively, you can click the **Jump to configuration** link, and return to the screen used to edit the configuration setting, and deselect the **Environmentalize** button on this field, or press Ctrl-E. This also removes the field as a setting to be configured under the **Environment Settings** tree. The value configured before environmentalization is displayed again.

Deploying the Configuration

After all environment-specific fields have been selected, and appropriate values set for the Development environment, the policy developer should deploy and test the updated configuration. For details on deploying to the group, see Step 1. The deployment package (.fed) deployed to the **Dev Payment API Group** will contain entries in the Environment Set-

tings store, and the associated values suitable for the Development environment.

Environmentalizing Reference Fields

Configuration fields that point to other fields are known as *reference fields*. For example, in an **HTTPS Interface** or **XML Signature** filter, you environmentalize a reference to an X.509 certificate. You can also environmentalize references to complex types such as Authentication Repositories. If a reference to an Authentication Repository is environmentalized, you could set the repository to the **Local User Store** in the Development environment, and to an **LDAP Repository** in the Testing environment.

The standard way to environmentalize a certificate at group level is to click **Environmentalize** on its configuration screen. Environmentalizing a certificate, or any other reference field, is the same as all other fields. For example, when you environmentalize the signing certificate in an **XML Signature** filter. The **Environment Settings** tree where you enter environment-specific values displays a node for the **XML Signature** filter. The screen on the right includes a **Signing Key** button to display a list of available certificates. You must select one of these certificates in Configuration Studio or Policy Studio. This field will most likely be prepopulated in Policy Studio if you already selected a certificate before clicking **Environmentalize**.

Alternatively, you can environmentalize a certificate using an alias. For example, in the Development environment, the **XML Signature** filter could use a certificate named `MySigningCert`. The policy package (`.pol`) created from the Development environment must be merged with an environment package (`.env`) that contains a certificate with the same alias.



Note

You can also environmentalize certificates using an alias at the API Gateway instance level as described in the [Externalizing API Gateway Instance Configuration](#). However, certificates are normally environmentalized at the API Gateway group level as described in this topic.

Step 3—Policy Developer Saves Policy Package in Policy Studio for Promotion

The policy developer finishes editing and environmentalizing the configuration that they are running with, and deploys it to the API Gateway. They must then save the policy package in Policy Studio to enable promotion to the Testing environment. To save the policy package, perform the following steps:

1. When the active configuration is loaded, select **File > Save > Policy Package**.



Important

Before creating the policy package, Policy Studio automatically detects any unenvironmentalized certificate references, and enables you to automatically environmentalize these settings before proceeding.

2. Browse to the directory in which you wish to save the package, and enter its filename (for example, `c:\temp\payment.pol`).
3. Click **Save**.

A policy package (`.pol`) file is created on disk. The policy developer must transfer this file to the Testing environment using some external mechanism (for example, FTP or email).



Note

The steps described so far are the same for first and subsequent cycle promotions. For the first cycle, the policy developer will most likely use the default factory configuration as their starting point for editing the configuration. In subsequent cycles, the starting point will most likely be the existing configuration currently deployed to the **Dev Payment API Group**.

Step 4—API Gateway Administrator Creates Testing Environment Package in Configuration Studio

This step depends on whether this is a first cycle promotion or a subsequent cycle promotion.

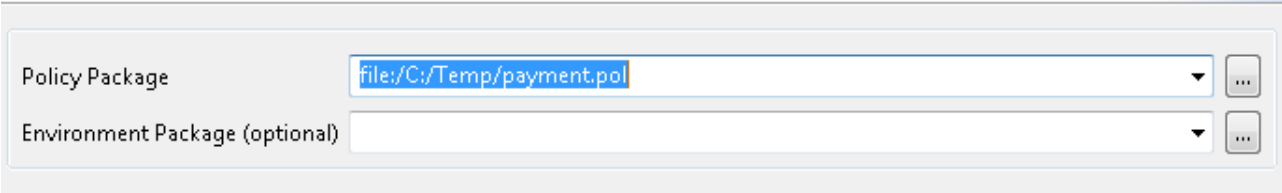
First Cycle Promotion—Open the Policy Package

If this is a first cycle promotion, the Testing API Gateway administrator uses the Configuration Studio to open the policy package created in the Development environment by the policy developer in Step 3. The administrator does not need to open an environment package for a first cycle promotion. To open the policy package, perform the following steps:

1. Open a command prompt, and change to your Configuration Studio installation directory (for example, `INSTALL_DIR\configurationstudio`).
2. Start `configurationstudio`.
3. Select **File > Open File**.
4. Enter or browse to the location of the **Policy Package** (for example, `c:\temp\payment.pol`).
5. Click **OK**.

Open Policy Package (.pol) file

Select the Policy package and the optional Environment package you wish to open




Note

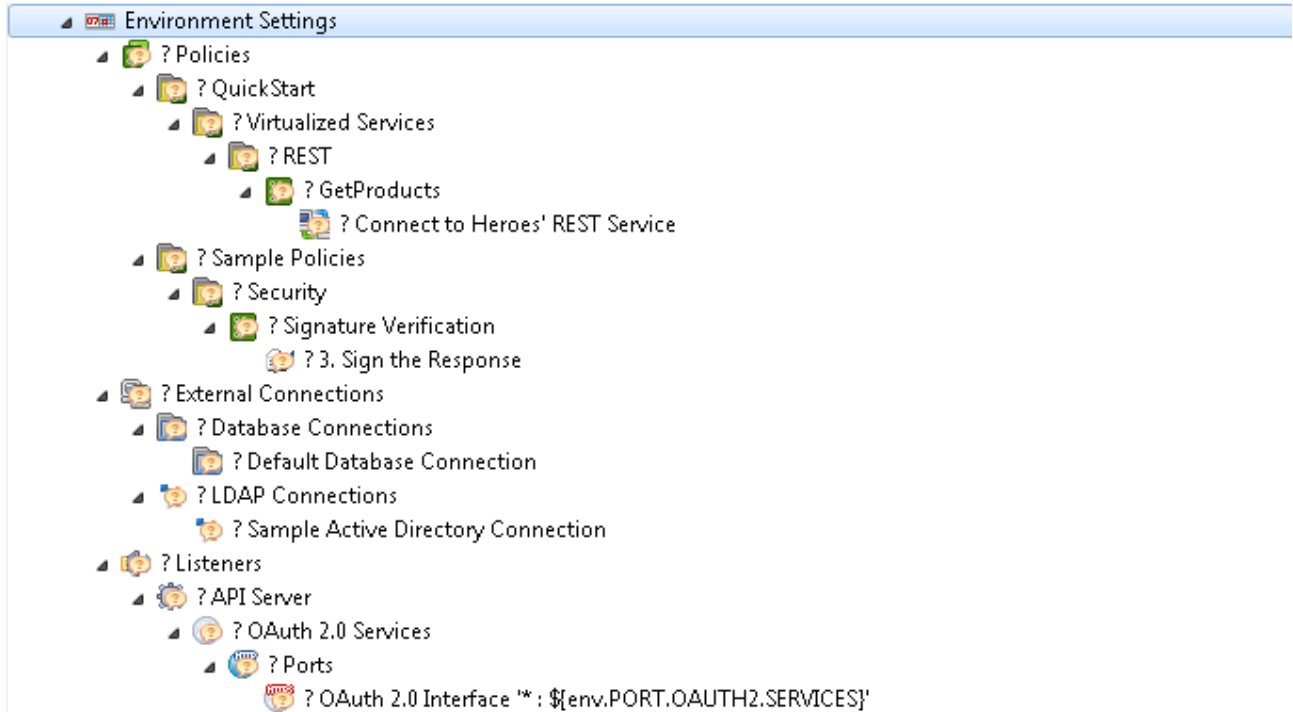
The Configuration Studio opens policy packages and environment packages by opening files available on disk. The administrator must ensure that the required files are available to the application.

Subsequent Cycle Promotion—Open the Policy and Environment Packages

If this is a subsequent cycle promotion, the Testing API Gateway administrator uses the Configuration Studio to open the policy package created in the Development environment by the policy developer in step 3. You must also open the currently deployed environment package for the Testing environment. If you do not open the currently deployed environment package at this point, you may need to re-enter certificates and settings that you entered for the previous promotion.

Specifying Environment Settings

The administrator must navigate the **Environment Settings** tree in the Configuration Studio, and enter values that are specific to the Testing environment. Values from the Development environment are not displayed. The **Environment Settings** tree displays any settings without values with a question mark indicator on the tree node. For a first cycle promotion, initially all environment setting values will be empty. Assuming a first cycle promotion with the sample data from Step 1, the **Environment Settings** tree is displayed in the Configuration Studio as follows:



For subsequent cycle promotions, settings that are still required by the new policy package from the Development environment, and that existed in previously promoted policy packages, will have values configured. Any certificates, keys, user, and user groups previously created will also be shown. Environment settings that existed in previously promoted configuration but are no longer required will be removed. New settings in the new policy package are listed with no value.

For example, assuming a first cycle promotion using the example environmentalized settings, the following screen is displayed when you select **Environment Settings > External Connections > LDAP Connections > Sample Active Directory Connection**:



Note

You cannot add or delete environment settings using Configuration Studio. These are predetermined by the policy developer in the Development environment.

Updating Certificates and Users

The administrator can add, edit, or remove new certificates, keys, users, and user groups in Configuration Studio in the same way as in Policy Studio. For more details, see the following topics:

- [Manage certificates and keys](#)
- [Manage API Gateway users](#)



Note

If a certificate reference has been environmentalized like in the **Sample Active Directory Connection**, you must create or import a Testing environment certificate in Configuration Studio. This makes the certificate available for selection when the environmentalized settings are edited in the **Environment Settings** tree in Configuration Studio.

Updating Package Properties

At any time, the API Gateway administrator can edit the environment package properties by selecting the **Package Properties > Environment** tree node in Configuration Studio. For example:

Environment Package Properties
Save

These fields act as metadata which will be associated with the corresponding '.env' package file.

Name	Default Factory Configuration (Environment)	✘	+
Description	Default factory configuration for API Server (Environment)	✘	
Version	v1 (Environment)	✘	
VersionComment	Original factory configuration (Environment)	✘	

If the API Gateway administrator selects the **Package Properties > Policy** tree node, this displays a read-only view of the policy package properties on a similar screen. For example:

Policy Package Properties
Save

These fields act as metadata which will be associated with the corresponding '.pol' archive file.

Name	QuickStart Server Configuration	+
Description	The QuickStart Server configuration policies	
Version	v1	



Note

You cannot edit the contents of the policy package file in Configuration Studio.

Saving the Environment Package

When you have entered all the environment-specific settings for the Testing environment, select **File > Save > Environment Package** in the Configuration Studio. An environment package (.env) file is saved to disk.

Step 5—API Gateway Administrator Deploys Configuration to Testing Environment Group

The Testing API Gateway administrator takes the policy package unchanged from the Development environment created in Step 3, and the environment package created using Configuration Studio for the Testing environment created in Step 4, and deploys them to the **Testing Payment API Group** using API Gateway Manager or scripts.

For example, to deploy using API Gateway Manager, perform the following steps:

1. Enter the following URL in your browser to launch API Gateway Manager:

```
https://127.0.0.1:8090/
```

2. On the **Dashboard** tab, select the API Gateway group in the **TOPOLOGY** section.
3. Click the **Edit** button on the right of the group, and select **Deploy Configuration**.
4. Choose the **I wish to deploy configuration contained in a Policy Package and Environment Package**, and browse to the .pol and .env files.
5. Click **Deploy**.

Deploy Configuration

Select configuration you wish to deploy

I wish to deploy configuration contained in a single Deployment Package

Browse for .fed

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

Browse for .pol

Browse for .env

Step 6—Further Configuration Updates in Testing Environment

This section describes how to update environment-specific settings using the Configuration Studio, and if necessary, using Policy Studio.

Updating Environment Settings Using Configuration Studio

If further updates are required to the environment-specific settings in the Testing environment, the Testing API Gateway administrator can open the policy package and environment package files in Configuration Studio at any time, and update the contents for the environment package file. The administrator can then deploy the policy package and updated environment package files to the **Testing Payment API Group** using API Gateway Manager or scripts.

Updating Environment Settings Using Policy Studio

Normally the policy package will be promoted through to upstream environments without any updates. However, in some cases, a single policy package for all environments will not be possible. For example, you may wish to use different Authorization filters in Development and Testing environments. But the policy developer may not have sufficient knowledge to create the necessary configuration for all upstream environments in the policy package. In this case, the API Gateway administrator in the upstream environment must use Policy Studio to make the required changes.

The administrator will open a policy package from the Development environment and the current Testing environment package (if one exists) in Policy Studio, before making the Testing environment-specific updates to the configuration. The administrator can save a policy package (.pol) and an environment package (.env) from Policy Studio. They can deploy them as usual to the **Testing Payment API Group** using API Gateway Manager or scripts. Alternatively, they can save a single deployment package (.fed), and deploy this package.

Promoting and Deploying Using Scripts

Overview

The API Gateway provides a collection of sample scripts to enable you to automate various common administration tasks. These scripts are based on the Jython Java scripting interpreter (see <http://www.jython.org>). You can extend these scripts to suit your needs by using the Jython language syntax. All Jython sample scripts are found in the following directory in your API Gateway installation:

```
INSTALL_DIR/samples/scripts
```

Running Sample Scripts

To run a sample script, call the run shell in the `/samples/scripts` directory, and specify the script that you wish to run. For example:

Windows

```
run.bat config\getEnvSettings.py
```

Linux/Unix

```
sh run.sh config/getEnvSettings.py
```

Scripts for Environmentalizing Configuration

You can use the following scripts in the `/sample/scripts/environmentalize` directory to environmentalize API Gateway configuration:

Script Name	Description
<code>addEnvSettings.py</code>	Downloads a deployment package (<code>.fed</code>) from an API Gateway. Marks the Traffic HTTP Interface port field to be environmentalized. Creates an environment settings entry for the port, and sets it to 7878.
<code>getEnvSettings.py</code>	Connects to an API Gateway and lists all the fields that have been marked for environmentalization. The associated values in environment settings are output.
<code>mergeEnvSettings.py</code>	Offline script that does not connect to an API Gateway. Merges a policy package (<code>.pol</code>) from downstream with an environment package (<code>.env</code>) from upstream, and merges them to create a deployment package (<code>.fed</code>).
<code>removeEnvSettings.py</code>	Downloads a deployment package (<code>.fed</code>) from an API Gateway. Removes the Traffic HTTP Interface port field from being environmentalized (opposite of the <code>addEnvSettings.py</code> script).

Scripts for Promoting Configuration

You can use the following scripts in the `/sample/scripts/migrate` directory to environmentalize API Gateway con-

figuration:

Script Name	Description
<code>archive.py</code>	Downloads the current API Gateway deployment package (<code>.fed</code>), policy package <code>.pol</code> , and environment package <code>.env</code> files from the Node Manager.
<code>createDeploymentPackage.py</code>	Creates a deployment package (<code>.fed</code>) from policy (<code>.pol</code>) and environment (<code>.env</code>) packages. Where the policy and environment packages are obtained from is out of scope for this script. For example, they could of been obtained from a running server (see <code>archive.py</code>), a source code repository (CVS, Git, SVN, and so on), or an FTP or USB connection.
<code>envMigrate.py</code>	Demonstrates the promotion of configuration from a development environment to a staging environment.

Externalizing API Gateway Instance Configuration

Overview

When API Gateway configuration is deployed to group, the configuration package settings are applied to all API Gateway instances in the group. You can also specify API Gateway configuration values on a per-API Gateway instance basis using environment variables in the `envSettings.props` file. For example, you can specify the port on which the API Gateway listens for HTTP traffic with different values depending on the environment in which the API Gateway is deployed.

The environment variable settings in the `envSettings.props` file are external to the API Gateway core configuration. The API Gateway runtime settings are determined by a combination of external environment variable settings and core configuration. This mechanism provides a simple and powerful approach to configuring specific API Gateway instances in the context of API Gateway group configuration defined in policy and environment packages.

The `envSettings.props` file is located in the `conf` directory of your API Gateway installation, and is read each time the API Gateway starts up. Environment variable values specified in the `envSettings.props` file are displayed as environment variable selectors in the Policy Studio (for example, `${env.PORT.TRAFFIC}`). For more details on selectors, see the *API Gateway User Guide*.



Important

Environment variables in the `envSettings.props` file apply to the API Gateway instance only. Configuration packages (`.fed`, `.pol`, and `.env` files) apply to the API Gateway group.

Configuring Environment Variables

The `envSettings.props` file enables you to externalize configuration values and set them on a per-server environment basis. This section shows the configuration syntax used, and shows some example values in this file.

Environment Variable Syntax

If the API Gateway configuration contains a selector with a format of `${env.X}`, where `X` is any string (for example, `MyCustomSetting`), the `envSettings.props` file must contain an equivalent name-value pair with the following format:

```
env.MyCustomSetting=MyCustomValue
```

When the API Gateway starts up, every occurrence of the `${env.MyCustomSetting}` selector is expanded to the value of `MyCustomValue`. For example, by default, the HTTP port in the server configuration is set to `${env.PORT.TRAFFIC}`. Specifying a name-value pair of `env.PORT.TRAFFIC=8080` in the `envSettings.props` file results in the server opening up port 8080 at start up.

Example Settings

The following simple example shows some environment variables set in the `envSettings.props` file:

```
# default port the API Gateway listens on for HTTP traffic
env.PORT.TRAFFIC=8080

# default port the API Gateway listens on for management/configuration HTTP traffic
env.PORT.MANAGEMENT=8090
```

The following example screen shows the corresponding `${env.PORT.TRAFFIC}` selector displayed in the **Configure HTTP Interface** dialog. At runtime, this is expanded to the value of the `env.PORT.TRAFFIC` environment variable specified in the `envSettings.props` file:

Network	Traffic Monitor	Advanced
Name:	Traffic HTTP Interface	
Port:	\${env.PORT.TRAFFIC}	
Address:	*	
Protocol:	IPv4	
Trace level:	From System Settings	
<input checked="" type="checkbox"/> Enable interface		



Important

All entries in the `envSettings.props` file use the `env.` prefix, and the corresponding selectors specified in the Policy Studio use the `${env.*}` syntax. If you update the `envSettings.props` file, you must restart or deploy the API Gateway for updates to be applied to the currently running API Gateway configuration.

Configuring Certificates as Environment Variables

You can also use the `envSettings.props` file to bind a reference to a server host-specific SSL certificate to a specific deployment.

Example Syntax:

The following entry shows an example of the environment variable syntax used to specify a server host-specific certificate:

```
env.serverCertificate=/[Certificates]name=Certificate Store/[Certificate]dname=
CN=MY_HOST_NAME
```

Alternatively, the following entry shows the syntax when the alias is the same as the Distinguished Name:

```
env.serverCertificate=/[Certificates]name=Certificate Store/[Certificate]dname=
MY_ALIASED_CERT_NAME
```

Example Settings

When the `env.serverCertificate` variable is specified in the `envSettings.props` file, the **X.509 Certificate** field in the **Configure HTTPS Interface** dialog can then reference its value using the `${env.serverCertificate}` selector. The following example screen shows the corresponding `${env.serverCertificate}` selector specified at the bottom of the **Select Certificate** dialog, which is displayed by pressing the **X.509 Certificate** button:

Choose a specific Certificate

Private Keys in Trusted Store

Alias	Certificate Name	Expiry
<input type="checkbox"/> CN=Change this for production		
<input type="checkbox"/> CN=Change this for production	CN=Change this for production	Thu Oct 01 12:32:00 BST 2037
<input type="checkbox"/> CN=Samples Test CA		
<input type="checkbox"/> Samples Test CA	CN=Samples Test CA	Sat Jan 10 10:56:00 GMT 2037
<input type="checkbox"/> Samples Test Certificate	CN=Samples Test Certificate	Sat Jan 10 10:56:00 GMT 2037

Bind the Certificate at runtime

Binding variable

The following example screen shows the `{env.serverCertificate}` selector then referenced in **X.509 Certificate** field:

Network

Name:

Port:

Address:

Protocol:

Trace level:

Enable interface



Important

In the `envSettings.props` file, when setting the value, you must specify escape characters for commas using `\\`. For example:

```
env.serverCertificate=/[Certificates]name=Certificate Store/[Certificate]dname=CN=linux-test-desktop\\,OU=QA\\,O=Saturn Inc.\\,L=Dublin\\,ST=Dublin\\,C=IE
```

Manage certificates and keys

Overview

The **Certificates and Keys** node in the Configuration Studio tree enables you to manage the X.509 certificates and keys trusted by API Gateway. These settings are environment-specific, and typically need to be configured during promotion to an upstream environment.

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 Configuration 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).

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,

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 [ht-
`tp://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 users

Overview

The **Users and Groups** node in the Configuration Studio tree enables you to manage API Gateway users and groups, which are stored in the API Gateway user store. These settings are environment-specific, and typically need to be configured during promotion to an upstream environment.

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 the *API Gateway User Guide*.

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.

Adding 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).

Adding 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=niiall@oracle.com`
- `dept=eng`
- `company=oracle`

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 in 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.