An Oracle White Paper
May 2011

Configuring Transport and Message Level Security for Ensuring Integrity and Confidentiality using Oracle Enterprise Gateway

Disclaimer

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. 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 for Oracle's products remains at the sole discretion of Oracle.

1 Introduction

This guide will demonstrate how to configure OEG Gateway to provide integrity and confidentiality at the transport and message level.

- At the transport level the guide will show how to establish both one-way and two-way (mutual) SSL connections from a client to the Gateway and from the Gateway to a Web Service.
- At the message level the guide will show how to configure the Gateway to both process and generate messages containing XML Signature and XML Encrypted elements.

Setup Used for this Guide:

- OEG Gateway 11.1.1.x
- OEG Service Explorer 11.1.1.x test client
- OEG Gateway 11.1.1.x acting as Web Service

2 Gateway Certificate Store

Overview:
The Certificate Store is a certificate manager where certificates and optionally associated private keys can be loaded for use by the Gateway. The Certificate Store contains private keys, certificates, and trusted issuers of certificates. The OEG Gateway will access the Certificate Store to retrieve certificates, keys, and trusted issuers of certificates
In the Certificate Store it is possible to either create or import existing public and private keys. The Gateway ships with a number of trusted Certificate Authority (CA) certificates. Normally, one the first tasks for providing confidentiality/integrity is to create or import an existing key-pair.
For this tutorial it will be demonstrated how to create, import and export certificates and keys using the Gateway Certificate Store.

Scenario:
For this tutorial three parties will be used to demonstrate a typical scenario where secure connections will be used:
The scenario is based on an **insurance broker** looking for insurance information. The broker will send a request to the **insurance provider** (Gateway) containing confidential information. The provider will then contact the **credit check authority** (web service). As all these transactions contain confidential information, they will be transmitted over secure connections using SSL and/or mutual SSL). For these secure connections to take place all the parties requires certificates and keys.

The Client: The Insurance Broker
Client certificates:
- BrokerCA Certificate (Certificate Authority)
- BrokerClient Certificate

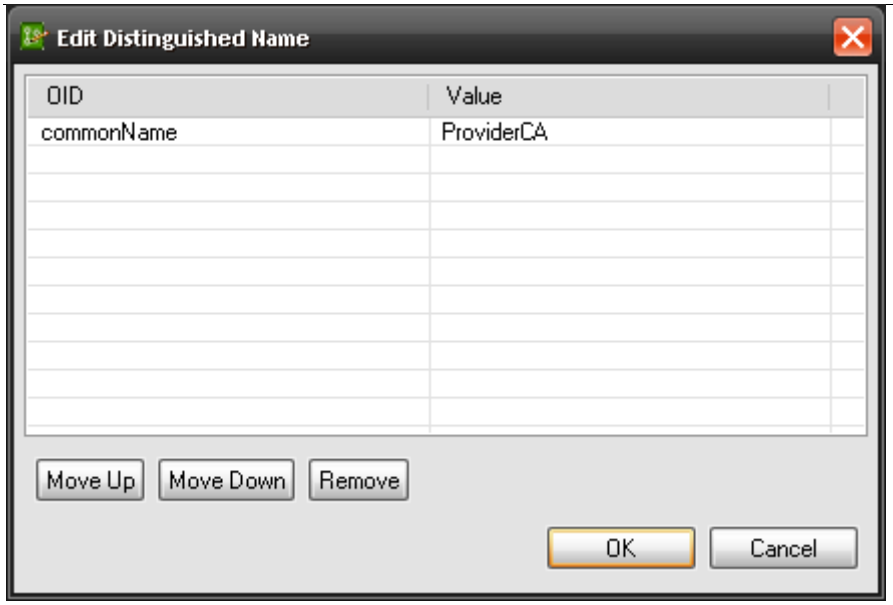
The Gateway: The Insurance Provider
Gateway certificates:
- ProviderCA Certificate (Certificate Authority)
- ProviderServer Certificate

The Web Service: Credit Check Authority
Web Service certificates:
- CreditAuthorityCA Certificate (Certificate Authority)
- CreditAuthorityServer Certificate

Creating, Importing and Exporting Certificates

For this tutorial a number of certificates and keys must be preloaded into the Certificate store of the Gateway, the Web Service and OEG Service Explorer.
Normally a company will go to a commercial certificate authority to purchase a CA certificate. For this demonstration Policy Studio will be used to create certificates for purpose of this tutorial.

1. How to Create a Certificate/Key in Policy Studio
To demonstrate how a certificate is created using OEG Policy Studio, the "ProviderCA" certificate that will act as the Certificate Authority for the Gateway will be created below.
- Start Policy Studio by running **"policystudio.exe"** (Windows) or **"policystudio.sh"** (Unix/Solaris) from the Policy Studio root directory.
- Click on the URL for the Gateway or Policy Director if the Gateway/s is managed via Policy Director.
- Click on the OEG Gateway process listed to open the configuration window in a new tab.
- Click on the **"Certificates"** module then click on the **"Certificates"** object.
- On the right hand side of the window click on **"Create/Import"** button.
- In the **"Configure Certificate and Private Key"** window click on **"Edit"** button next to the **"Subject"** field.
- Under the **"OID"** column click in the empty field and select **"commonName"** from the list.
- Under the **"Value"** section enter **"ProviderCA"** then click on **"OK".**

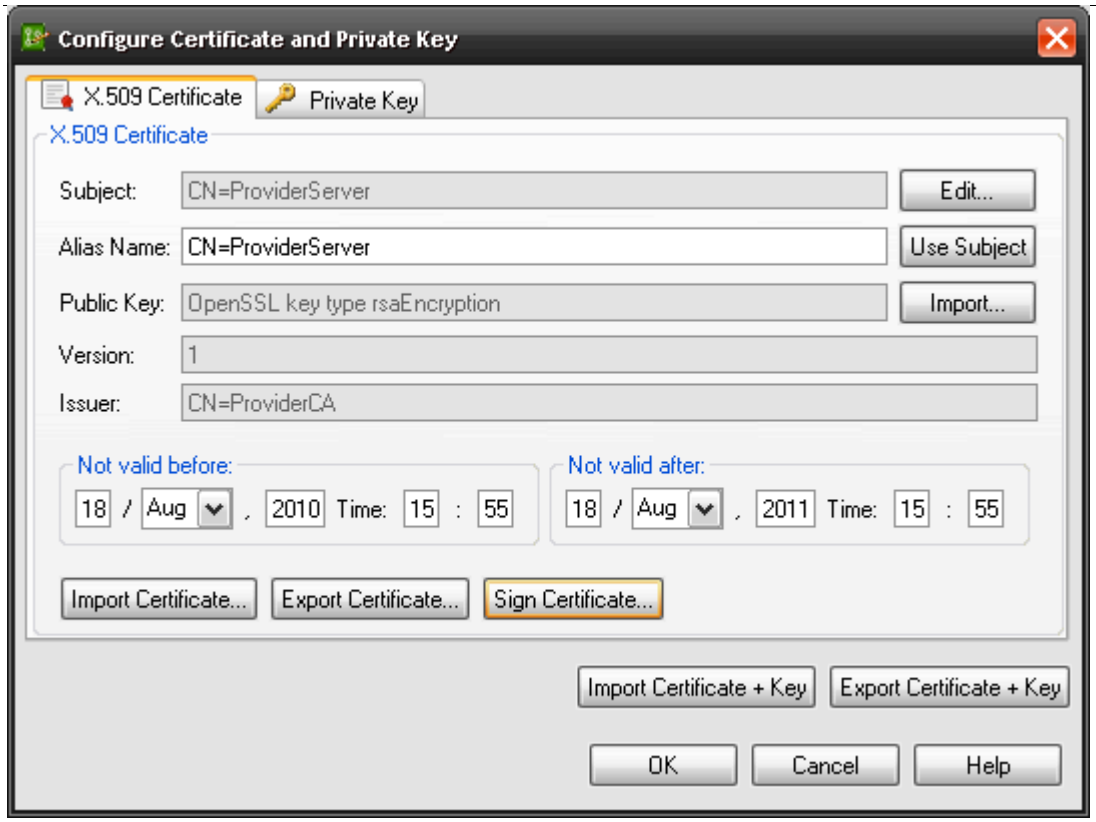


- Make sure that valid dates are entered in the **"Not Valid Before' and "Not Valid after"** fields.
- Click on **"Sign Certificate".**
- When prompted **"Do you want to self-sign the certificate"** select **"Yes".**
- Next to **"Certificate Alias"** click the **"Use Subject"** button.
- Click on **"OK".**
- This **"ProviderCA"** certificate will act as the Certificate Authority that will sign/issue the **"ProviderServer"** certificate

The **"ProviderServer"** certificate is going to be signed by the **"ProviderCA"** certificate above and used for the Gateway to authenticate itself to the clients and the web service during the SSL handshake.

- Repeat the steps above with two exceptions:
- The **"commonName"** for the Certificate will be **"ProviderServer"**. It is recommended that the **"commonName"** be given the name of the host that the Gateway is running on as some systems are sensitive in cases where the common name of a certificate is not the same as the host.
- When prompted **"Do you want to self-sign the certificate"** select **"No"** which in turn will present the list of certificates to be used to sign the certificate, choose the **"ProviderCA"** certificate from the list.

The "ProviderServer" Certificate:



2. How to Import a Certificate/Key in Policy Studio

- Click on the **"Certificates"** module in Policy Studio.
- Select **"Certificates"** then click on **"Create"** on the right hand side.
- Click on **"Import Certificate"** or **"Import Certificate and Key"** if the certificate contains a private key.
- Browse to the location of the certificate.
- Click on **"Open"** to import the client certificate into the Gateway Certificate Store. If the certificate contains a private key it will prompt for a password.
- Click on **"OK".**

3. How to Export a Certificate/Key in Policy Studio

- Click on the **"Certificates"** module in Policy Studio.
- Select **"Certificates"** then click on **"Create"** on the right side.
- Click on **"Export Certificate"** or **"Export Certificate and Key"'** depending on whether the certificate has a private key associated with it
- Browse to the location where the certificate is to be saved to.

- Click on **"Open"** to export the certificate to a chosen location. If a certificate containing a private key was chosen it will prompt for a password.
- Click on **"OK".**

4. How to Create a Certificate in OEG Service Explorer:

To demonstrate how a certificate can be created using OEG Service Explorer the "BrokerCA" certificate that will act as the Certificate Authority for the Broker client will be created below:

- Start **OEG Service Explorer** by running **"OEG Service Explorer.exe"** (win32) or **"OEG Service Explorer.sh"** (UNIX) located in the OEG Service Explorer root directory.
- Click on **"Security"** on the top menu then select **"View Certificates".**
- Click on **"Create".**
- The **"Configure Certificate and Private Key"** window will open.
- In this window click on **"Edit"** button next to the **"Subject"** field.
- Under the **"OID"** section click in the empty field and it should automatically populate **"commonName".**
- Under the **"Value"** section enter **"BrokerCA"** then click on **"OK".**
- When prompted **"Do you want to self-sign the certificate"** select **"Yes".**
- Under the **"Certificate Alias"** section tick the **"Use Distinguished Name"** box
- Click on **"OK".**
- This **"BrokerCA"** certificate will act as the Certificate Authority that will sign/issue **the "BrokerClient"** certificate.

NOTE: Refer to the appendix of this guide to using OpenSSL to create a certificate and key pair.

For the **"BrokerClient"** certificate that is going to be signed by the **"BrokerCA"** certificate above and used for the Broker Client (OEG Service Explorer) to authenticate itself to the Gateway during Mutual SSL handshake the following steps apply.

- Repeat the steps above with two exceptions:
- The commonName for the Certificate will be **"BrokerClient**
- When prompted **"Do you want to self-sign the certificate"** select **"No"** which in turn will present the list of certificates to be used to sign the certificate, choose the **"BrokerCA"** certificate from the list.

5. How to Import a Certificate/Key into OEG Service Explorer

- Click **"Settings"** on the top menu then select **"View Certificates"** option.
- Click on **"Import Certificate"** or **"Import Certificate and Key"** depending on whether the certificate has a private key associated with it.
- Browse to the location of the certificate.

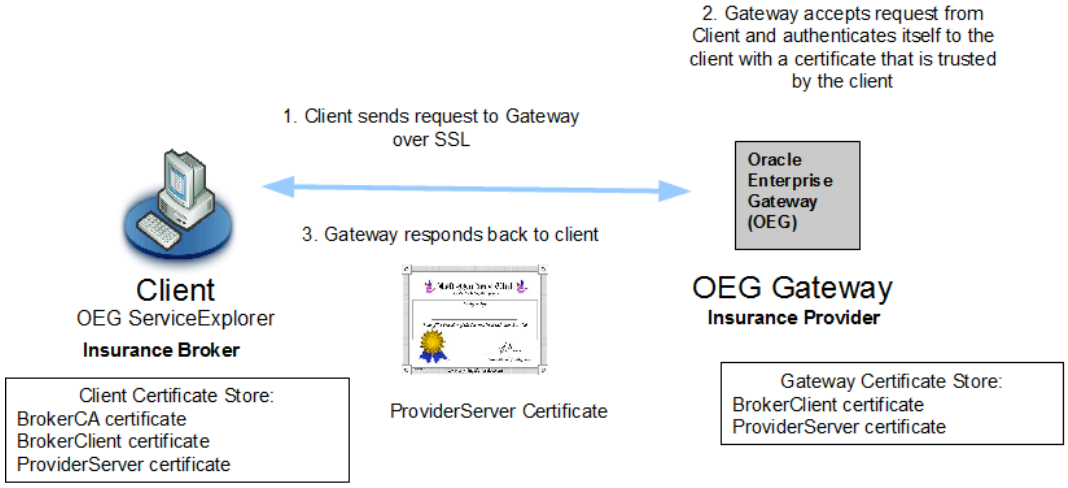
- Click on **"Open"** to import the client certificate into the Gateway Certificate Store.
- Click on **"OK"** (if a certificate and key was chosen it will prompt for a password first)

NOTE: The CA certificate that issued the client certificate would normally be the certificate that is trusted. This means that the "BrokerCA" certificate needs to be imported into the Gateway Certificate Store. To do this, please see step 2 above.

3 Transport Level Security (TLS / SSL)

The Gateway can be configured to handle one way or two-way (mutual) SSL connections. For incoming connections a HTTPS interface is configured. For outgoing SSL connections from the Gateway, the connection filters in the policy is configured to allow for SSL connections.

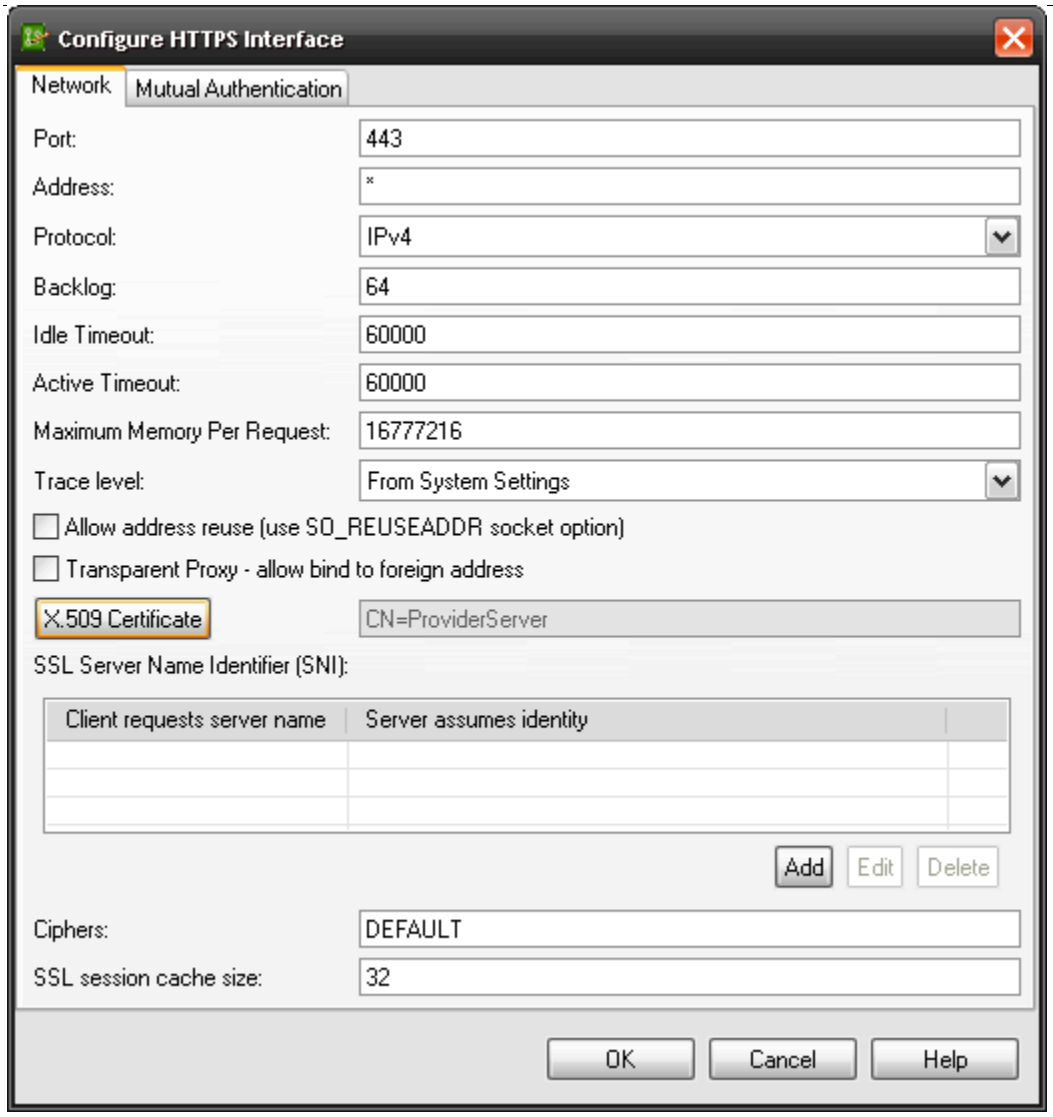
Inbound SSL



Configuring a HTTPS Interface in the Gateway

- Click on the **"Services"** module in the left side of **"Policy Studio".**
- Browse to **"OEG Gateway"** in the tree by expanding the **"Processes"** node.
- Expand the **'OEG Gateway'** node to show the **"Default Services"** object.
- Right click on the **"Default Services".**

- Choose **"Add Interface"** and select **"HTTPS".**
- Explanation of some of the fields. Click on the help button for more information:
- **Port:** Insert a port number that the HTTPS interface will listen on. In this example it has been configured to 443.
- **Address:** Enter the IP address or host of the network interface on which the Process will listen. Entering * indicates any IP Address.
- **X.509 Certificate**: Select the server certificate that will be used for SSL. This certificate will be presented by the Gateway during the SSL handshake with a client, the certificate enables server authentication. It verifies the server's identity to the client. The client would need to have an access to the server certificate. The list of certificates with associated private keys currently stored in the **Certificate Store** will be presented, from which a single certificate must be selected. In this example the **"ProviderServer"** certificate that was created earlier and signed by the **"ProviderCA"** certificate, has been selected.
- The rest of the values are automatically filled in and can be left default. Please refer to **"Help"** on the window as shown above for more information on these values.
- Click on **"OK".**
- The HTTPS Interface is listed underneath **"Default Services"** and has been configured to listen on **port 443.**



Testing an Incoming Connection using SSL:

Create a policy to demonstrate a connection being made from a client to the Gateway over SSL.

STEP 1: Configuring the Policy

- In Policy Studio click on the **"Policies"** module.
- Right click on **"Policies"** module in the tree node on the left and select **"Add Policy".**
- Name Policy **"Inbound SSL".**

- Drag a **"Set Message"** filter from **'Conversion"** category on the right of Policy Studio from the filters palette.
- Configure the filter as follows:
  - Name: Choose a name for the filter
  - Content-Type: text/xml
  - Message Body:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Message>
This message was received by the SSL service.
    </Message>
  </soap:Body>
</soap:Envelope>
```

- Drag a **"Reflect"** Filter from the **"Utilities"** category.
- Configuration can be kept default.

STEP 2: Configuring the relative path

- In Policy Studio click on the **"Services"** module.
- Expand **"Processes"**, then **'OEG Gateway'** in the left tree node, right click on **"Default Services"** and select **"Add Relative Path".**
- The relative path will be: **/ssl**
- Right click on the **/ssl** relative path and point to the **"Inbound SSL"** policy created above.
- Deploy the configuration by pressing the **"F6"** key or select **"Settings"** located in the top menu of Policy Studio and click on **"Deploy".**

STEP 3: Test it via a Web Browser:

- Open a Web Browser of choice.
- Enter the URL : https://gateway_ip:443/ssl
- The browser might display a warning that the certificate is not trusted. This is due to the fact that a self signed certificate is being used and cannot be verified by the browser. The certificate authority for the certificate is not in the browsers' trusted certificate store.
- Trust the certificate and continue to the website.

- It is also possible to export the certificate from the Browser certificate store
- For **Firefox** click on **"Tools"**, **"Options"** and **"View Certificates".**
- Click on the relevant tab, select the certificate and click on **'Export'**
- The default format for the export will be X.509 Certificate PEM format.
- This certificate can then be imported into another certificate store as required.
- For **Internet Explorer** click on **"Tools"**, **"Internet Options"** and **"Connections"** tab.
- Click on **"Certificates"** tab and select the relevant certificate from on of the tabs displayed.
- Click on **"Export"** and follow the prompts displayed.
- This certificate can then be imported into another certificate store as required.

STEP 4: Send Request via OEG Service Explorer

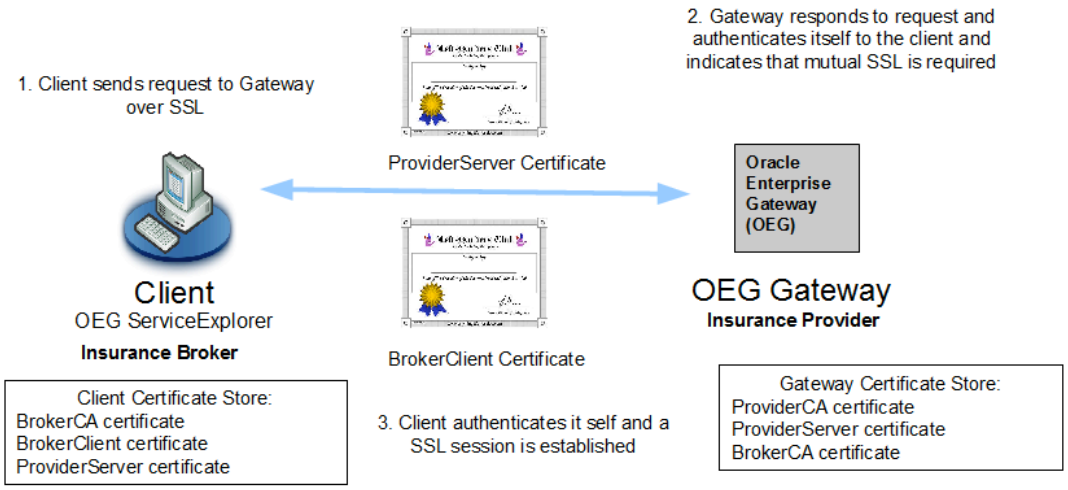
OEG Service Explorer test client will be used to test the connections

- Start **OEG Service Explorer** by running **"OEG Service Explorer.exe"** (win32) or **"OEG Service Explorer.sh"** (UNIX) located in the OEG Service Explorer root directory.
- Click on **"Security"** on the top menu then select **"View Certificates".**
- Click on **"Create"** and then click on **"Import Certificate".**
- Select the **"ProviderCA"** certificate that was exported earlier from the Gateway or from the browser.
- Add a SOAP message in the SOAP Request window.
- In the URL field enter the URL of the Gateway for example: https://localhost:443/ssl
- Click on the **"Security"** tab followed by the **"Trusted Certificates"** tab.
- Select **"ProviderCA"** certificate from the list.
- Click on the **"Run"** button.

Inbound Mutual SSL

It is also possible to configure mutual or two-way SSL authentication for inbound requests to the Gateway.

The flow of the Mutual SSL Request to the Gateway:



It is also possible to configure mutual or two-way SSL authentication for inbound requests to the Gateway.

- Click on the **"Services"** module in the left side of **"Policy Studio".**
- Browse to the **"OEG Gateway"** in the tree by expanding the **"Processes"** node.
- Expand the **'OEG Gateway'** node show the **"Default Services".**
- Right click on the configured HTTPS Interface and select **"Edit".**
- If no HTTPS Interface has been configured yet, please see section 3.1 on how to do so.
- For SSL a certificate has already been selected for the Gateway to authenticate itself to the clients.
- For Mutual SSL authentication, click on the **"Mutual Authentication"** tab at the top of the configuration window.

- Clients can be configured to authenticate to the Gateway on the **"Mutual Authentication"** tab. There are 3 options available on this tab:

1. **Ignore Client Certificates:** The Gateway will ignore client certificates if they are presented during the SSL handshake.
2. **Accept Client Certificates:** Client certificates will be accepted when presented to the Gateway, but clients that do not have to present certificates will not be rejected.
3. **Require Client Certificates:** The Gateway will only accept connections from clients that present a certificate during the SSL handshake.

Client certificates are issued by a Certificate Authority (CA). In most cases, the CA will include a copy of its certificate in the client certificate so that consumers of the certificate can decide whether or not to trust the client based on the issuer of the certificate.

It is also possible that a chain of CA's were involved in issuing the client certificate. For example, a top-level organization-wide CA (e.g. Company CA) may have issued department-wide CA's (e.g. Sales CA, QA CA, etc), and each department CA would then be responsible for issuing all department members with a client certificate.

In such cases, it is possible that the client certificate will contain a chain of 1 or more CA certificates. The **"Maximum depth of client certificate chain"** field on the Mutual Authentication tab can be used to configure how many CA certificates in a chain of 1 or more CA certificates are present in a client certificate that will be trusted when validating the client certificate.

By default, only 1 issuing CA certificate is used, and this certificate must be checked in the list of "Trusted Root Certificates". If more than 1 certificate is to be used, then only the top-level CA must be considered trusted, while the intermediate CA certificates are not.

Finally, select the root CA (Certificate Authority) certificates that the Gateway will consider trusted when authenticating clients during the SSL handshake. Only certificates signed by the CA selected here will be accepted.

NOTE: The CA certificate that issued the client certificate needs to be imported into the OEG Gateway Certificate store. See section 2.1 for an explanation how to import a certificate into the Gateway certificate store using Policy Studio.

Testing Mutual SSL:

Create a policy to demonstrate a SSL connection requiring mutual SSL.

For mutual SSL to be successful the CA certificate that issued the client certificate needs to be imported into the Gateway Certificate Store and the HTTPS interface needs to be configured.

STEP 1: Import Client Certificate Issuer Certificate (BrokerCA) into the Gateway Certificate store.

- Select the **"Certificates"** module in Policy Studio, click on the **"Certificates"** node then click on **"Create".**
- Click on the **"Import Certificate"** button.
- Browse to the **"BrokerCA2** certificate that was created and exported before (if has not been exported yet, see section 2 how to export certificates).
- Click on **"Open"** to import the client CA certificate into the Gateway Certificate Store.
- Click on **"OK".**
- Click on the "Services" module then right click on the HTTPS Interface under **"Default Services"** and click on **"Edit".**
- Click on the **"Mutual Authentication"** tab then select the **"Require Client Certificates"** option.
- Select the client certificate to be trusted. In this case **"BrokerCA".**
- Click on **"Ok".**
- Refresh the Gateway by pressing the **"F6"** key or select **"Settings"** located in the top menu of Policy Studio and click on **"Deploy".**

STEP 2: Test via a Web Browser:

- Open a Web Browser of choice.
- Enter the URL : https://gateway_ip:443/ssl
- The Gateway expects the browser to present a certificate during the connection, but as the browser does not contain the required certificate that the Gateway trusts, the connection will fail.
- To be able to initiate a successful connection, the trusted client certificate needs to be imported into the certificate store of the browser.
- Import the **"BrokerCA"** certificate into the browser of choice and test the connection again. Please refer to browser help on how to import certificates.

STEP 3: Send Request via OEG Service Explorer

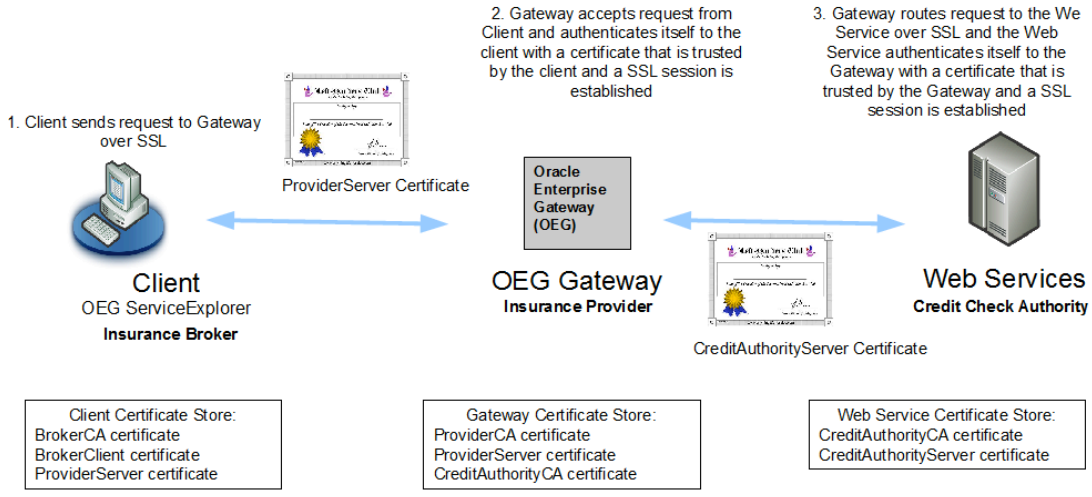
- Start "**OEG Service Explorer"** by running **"OEG Service Explorer.exe"** (win32) or **"OEG Service Explorer.sh"** (UNIX) located in the OEG Service Explorer root directory.
- Add a SOAP message in the SOAP Request window.

- In the URL field enter the URL of the Gateway for example: https://localhost:443/ssl
- Click on the **"Security"** tab followed by the **"Trusted Certificates"** tab.
- Select **"ProviderCA"** certificate from the list
- Click the **"Client SSL Authentication"** tab and select the **"BrokerClient"** certificate in the list and click on **"Finish".**
- Click on the **"Run"** button.

Outbound SSL

OEG Gateway can be configured to route messages to services that require SSL connections and mutual SSL connections.
For the Gateway to be able to do this it has to trust the issuer of the certificate of the service that it will be connecting to, requiring the issuer of the remote service certificate to be imported into the Certificate store and trusted via the "Connection" filter.

The flow of a SSL session from the Gateway to the Web Service:



STEP 1: Import the Web Service Issuer's Certificate into Gateway Certificate Store

- Select the **"Certificates"** module in "**Policy Studio"**, click on the **"Certificates"** node then click on **"Create".**
- Click on the **"Import Certificate"** button.
- Select the "**Certificates"** node then click on **"Create"** on the right side of Policy Studio.

- Click on **"Import Certificate".**
- Browse to the **"CreditAuthorityCA"** certificate that was created and exported before.
- Click on **"Open"** to import the **"CreditAuthorityCA"** certificate into the Gateway Certificate Store.
- Click on **"OK".**

STEP 2: Create Route to SSL Service Policy

The **"Connect to URL"** filter is configured to route the request to the web service and the required certificate to trust for the one-way SSL connection is selected.

Configuring the Route to SSL Service Policy:

- Click on the **"Policies" module** in Policy Studio.
- Right click on the **"Policies"** node and select **"Add Policy".**
- Create a policy named **"Route to SSL Service"**
- Drag a **"Connect to URL"** filter from the **"Routing"** category onto the policy canvass.
- In the **"Name"** field enter any appropriate name for the filter.
- In the **"URL"** field enter the full URL of the service that is being routed to for example: https://IP_OF_WebService:443/secureservice
- Under the **"Trusted Certificates"** tab select the certificate that the Gateway should trust. Select the **"CreditAuthorityCA"** certificate that issued the "CreditAuthorityServer" certificate.
- Click on **"Finish".**

STEP 3: Change the relative path to point to the Route to SSL Service Policy

Change the /ssl relative path to point to the **"Route to SSL Service"** policy:

- Click on the **"Services"** module then expand the **"Processes"** node followed by the **"OEG Gateway"** node and right click on **"Default Service".**
- Right click on the relative path **/ssl**
- Click on **"Edit"** and select the **"Route to SSL Service"** policy from the list.
- Click on **"OK".**
- Deploy the configuration by pressing the **"F6"** key or select **"Settings"** located in the top menu of Policy Studio and click on **"Deploy".**

STEP 4: Send Request using OEG Service Explorer

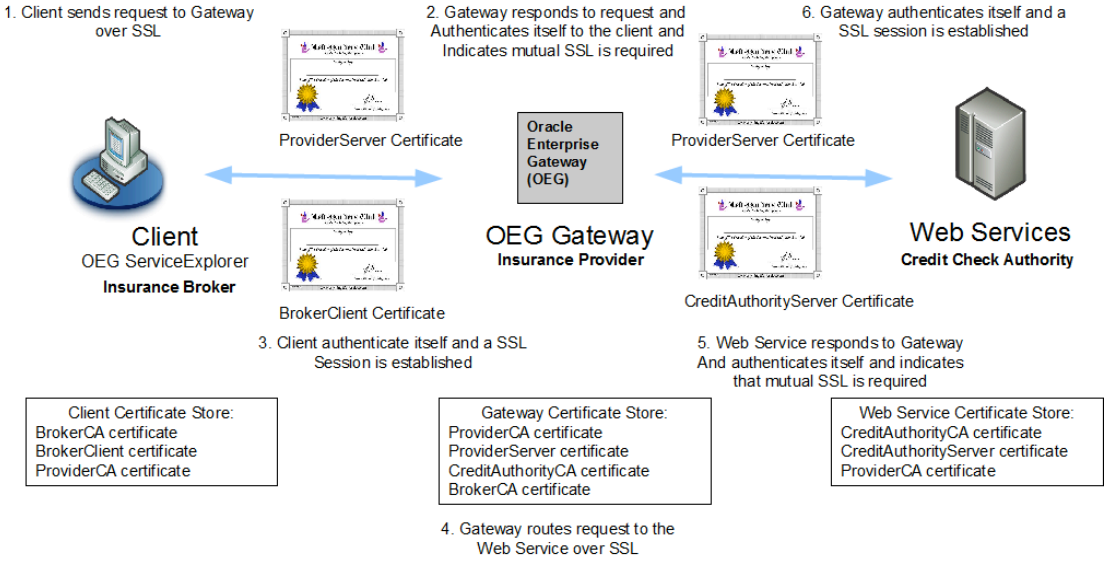
OEG Service Explorer test client will be used to test the connections.

- Start "**OEG Service Explorer"** by running **"OEG Service Explorer.exe"** (win32) or **"OEG Service Explorer.sh"** (UNIX) located in the OEG Service Explorer root directory.
- Add a SOAP message in the SOAP Request window.

- In the URL field enter the URL of the Gateway for example: https://localhost:443/ssl
- Click on the **"Security"** tab followed by the **"Trusted Certificates"** tab.
- Select **"ProviderCA"** certificate from the list.
- Click the **"Client SSL Authentication"** tab and select the **"BrokerClient"** certificate in the list and click on **"Finish"**
- Click on the **"Run"** button.

Outbound Mutual SSL

The Gateway can also route requests to a Web Service using mutual SSL.
The flow of a Mutual SSL session from the Gateway to the Web Service:



The **"Route over SSL Service"** policy above will be modified.

- Click the **"Policies"** module and expand the **"Policies"** tree and click on policy named **"Route to SSL Service".**
- Double click on the **"Connect to URL"** filter.
- Under the **"Trusted Certificates"** tab select the certificate that the Gateway should trust. It should be the **"CreditAuthorityCA"** certificate that issued the **"CreditAuthorityServer"** certificate
- Click on the **"Client SSL Authentication"** tab.
- Select the **"ProviderServer"** certificate in the list
- Click on **"Finish".**

NOTE: The Web Service the Gateway is connecting to needs to have the **"ProviderCA"** certificate imported and trusted for the mutual SSL connection to be successful.

4 Message Level Security

The Gateway can be configured to ensure the confidentiality and integrity of requests passing through and being forwarded on to services. In this section it will be demonstrated on how to achieve this.
The Gateway can decrypt an XML encrypted message on behalf of its intended recipients. XML Encryption is a W3C standard that allows data to be encrypted and decrypted at the application layer of the OSI stack, thus ensuring complete end-to-end confidentiality of data.

Confidentiality

XML Encryption

XML Encryption facilitates the secure transmission of XML documents between two application end-points. Whereas traditional transport-level encryption schemes, such as SSL and TLS, can only offer point-to-point security, XML Encryption guarantees complete end-to-end security. Encryption takes place at the application-layer and so the encrypted data can be encapsulated within the message itself. The encrypted data can therefore remain encrypted as it travels along the multiple-hop paths to the target Web Service.

When a message is encrypted, it is encrypted in such a manner that only the intended recipient(s) of the message can decrypt it. By encrypting the message with the recipient's public key, the sender can be guaranteed that only the intended recipient can decrypt the message through the use of his private key, to which he has sole access.

For the encryption example below the Gateway will encrypt a request intended for the Web Service (Credit Check Authority). The request will be encrypted using the public key of the Web Service (Credit Check Authority) using the trusted "**CreditAuthorityCA"** certificate which has already been imported into the Gateway certificate store. This request can only be decrypted by the Web Service's private key.

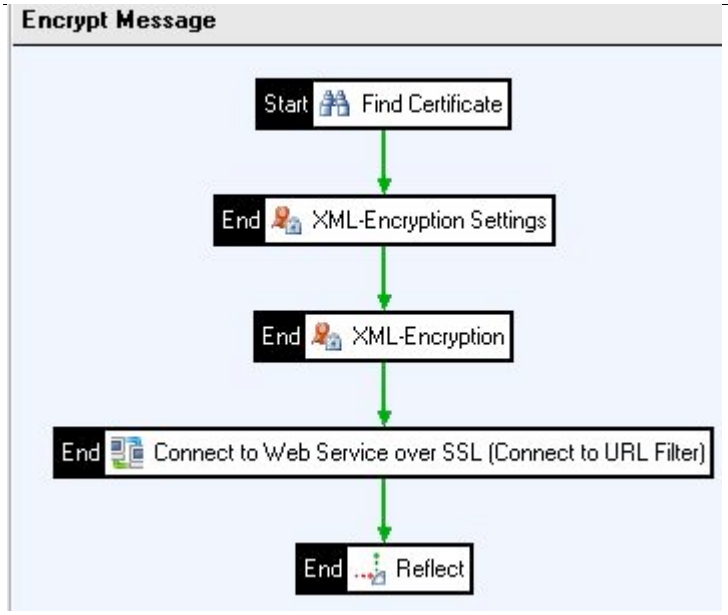
STEP 1: Create Encryption Policy

- Click on the **"Policies"** module in Policy Studio then right click on the **"Policies"** node and select **"Add New Policy".**
- Name the policy **"Encrypt Message".**

- Drag a **"Find Certificate"** filter from the **"Certificates"** category.
- Under the **"Attribute Name"** section select **"Certificate".**
- Under the **"Retrieve certificate from the following location"** section choose **"Certificate Store"** and select the desired certificate to be used. (In this case the **"CreditAuthorityCA"** certificate will be used)
- Click on **"Finish".**
- Drag a **"XML-Encryption Settings"** filter from the **"Encryption"** category and connect to the **"Find Certificate"** filter via a success path.
- Double click on the **"XML-Encryption Settings"** filter to configure it.
- Under the **"Node(s) to Encrypt"** section choose **"Encrypt SOAP Method"** option from the drop down list. (Alternatively an XPath expression can be used to select the exact node/s for the message that needs to be encrypted)
- Select the **"Encrypt Node"** radio button.
- Under the **"Recipient"** tab click on **"Add".**
- Enter a name in the **"Recipient Name"** field.
- In the **"Actor"** drop down menu select **"Current actor/role only".**
- Click on **"OK".**
- Drag a **"XML-Encryption"** filter from the **"Encryption"** category and connect it to the **"XML-Encryption Settings"** filter via a success path.
- Drag a **"Connect to URL"** filter from the routing category and configure to forward request to the Web Service.
- Drag a **"Reflect"** filter from the **"Utilities"** category and connect to the **"Connect to URL"** filter via a success path.

NOTE: For more information on the configuration options for the "XML-Encryption Settings" filter please refer to OEG Gateway User Guide located in the /docs directory in the OEG Gateway install directory or click on the "Help" button in the filter itself.

The 'Encrypt Message' policy after it has been configured



STEP 3: Create new relative path

Create a new relative path to point to this policy:

- Click on the **"Services"** module in Policy Studio and right click on **"Default Service"** node.
- Select **"Add Relative Path".**
- Create a relative path: **/encrypt**
- Select the **"Encrypt Message"** policy in the list.
- Refresh the Gateway by pressing the **'F6'** key or select **"Settings"** located in the top menu of Policy Studio and click on **"Deploy".**

STEP 4: Testing the Encryption Policy using OEG Service Explorer

- Start **OEG Service Explorer** by running **"OEG Service Explorer.exe"** (win32) or **"OEG Service Explorer.sh"** (UNIX) located in the OEG Service Explorer root directory.
- Add a SOAP message in the SOAP Request window
- In the URL field enter the URL of the Gateway for example: http://localhost:8080/encrypt
- Click on the **"Run"** button.

XML Decryption

The XML-Decryption Settings should be used in conjunction with the XML-Decryption filter, which actually performs the decryption. The XML-Decryption Settings generates

the decryption.properties message attribute, which is required by the XML-Decryption filter.
It is important to note that the output of a successfully executed decryption filter is the original unencrypted message. Depending on whether or not the Remove EncryptedKey used in decryption has been enabled, all information relating to the encryption key can be removed from the message.
For the Decryption example below, the Gateway will decrypt a message that was encrypted in OEG Service Explorer using the ProviderCA (Gateway) certificate that has already been added to the certificate store of OEG Service Explorer.

STEP 1: Create Decryption Policy

- Click on the "Policies" module in Policy Studio then right Click on the **"Policies"** node and select **"Add New Policy".**
- Name the policy **"Decrypt Message".**
- Drag a **"XML-Decryption Settings"** filter from the **"Encryption"** category.
- Under the **"Node(s) to decrypt"** section, select **"All Nodes".**
- Under the **"Decryption Key"** section select **"Find via KeyInfo in Message".**

Note: More information on the Decryption Key Section:
This section allows you to specify what key to use to decrypt the encrypted nodes. Data encrypted with a public key can only be decrypted with the corresponding private key. In this section, you can elect to take the private (i.e. decryption) key from the <KeyInfo> element of the XML Encryption block or else, the certificate stored in a OEG message attribute can be used to lookup the private key of the intended recipient of the encrypted data in the Certificate Store.
Find via KeyInfo in Message:
Select this option if you want to extract the decryption key from the XML Encryption block in the message. The client that encrypted the message may or may not have added its private key (to use to decrypt the data that was encrypted with its public key) to the XML Encryption block within the message. So this option should only be used in cases where the private key has been included in the <KeyInfo> element within the XML Encryption block in the message.
Find via Certificate in Attribute:
In cases where the client has opted not to include its private key in the XML Encryption block, the key must be extracted from an alternative source so that the encrypted data can be decrypted.
Typically, a **Find Certificate** filter would be used in a policy to locate an appropriate certificate and store it in the certificate message attribute. Once the certificate has been stored in this attribute, the XML Decryption Settings filter can use this certificate to lookup the Certificate Store for a corresponding private key for the public key stored in the certificate. To do this, simply select the certificate attribute from the dropdown.

- Click on **"Finish".**
- Drag a **"XML-Decryption" filter** from the **"Encryption"** category and connect it to the **"XML-Decryption Settings"** filter via a success path.

- Drag a **"Reflect"** filter from the **"Utilities"** group and connect to the **"XML-Decryption"** filter via a success path.

NOTE: For more information on the configuration options for the 'XML-Decryption Settings' filter please refer to OEG Gateway User Guide located in the docs directory in the OEG Gateway install directory.

STEP 2: Create new relative path:
Create a new relative path to point to this policy:

- Click on the **"Services"** module in Policy Studio and right click on **"Default Service"** node.
- Select **"Add Relative Path"**
- Create a relative path: **/decrypt**
- Select the **"Decrypt Message"** policy in the list.
- Deploy the configuration by pressing the **"F6"** key or select **"Settings"** located in the top menu of Policy Studio and click on **"Deploy".**

STEP 3: Testing the Decryption Policy using OEG Service Explorer

- Start **OEG Service Explorer** by running **"OEG Service Explorer.exe"** (win32) or **"OEG Service Explorer.sh"** (UNIX) located in the OEG Service Explorer root directory.
- Add a SOAP Message in the SOAP Request window.
- Click on **"Security"** then **"Encrypt Request"** in the top menu of OEG Service Explorer.
- Select the **"ProviderCA"** certificate from the certificate store.
- In the next screen and for XPath choose **"Encrypt the SOAP Method"** from the drop down list
- For the **"Recipient"** click on **"Add"** and enter a Recipient Name and choose **"Current Actor/Role Only"** from the drop down list for Actor.
- Under the **"Basic"** tab leave the **"Attribute containing Public Key"** as **"certificate".**
- Under the **"Key Info"** tab select the **"Embed Public key information in key info section"** and have the **"Include Certificate"** box selected.
- Click **"OK".**
- The SOAP request is now encrypted
- In the URL field enter the URL of the Gateway for example: http://localhost:8080/decrypt
- Click on the **"Run"** button.

Integrity

OEG Gateway can be configured to ensure the integrity of the messages it receives by means of signing and performing verification on messages. It will be demonstrated how to sign and verify a message.

Digital Signatures

The Gateway can sign both SOAP and non-SOAP XML messages. Attachments to the message can also be signed. The resultant XML signature is inserted into the message for consumption by a downstream Web Service or client as any encryption / signing etc. can be performed on the request and response, it largely depends on where the filters are placed. At the Web Service, the signature can used to verify that the message has not been tampered with during transit

STEP 1: Creating a Sign Message Policy:

- Click on the **"Policies"** module in "**Policy Studio"** then right click on the **"Policies"** node and select **"Add New Policy".**
- Name the policy **"Sign Message".**
- Drag a **"Sign Message"** filter from the **"Integrity"** category.
- On the **"Signature/What to Sign"** tab under the **"XPath"** section select **"User XPath".**
- From the XPath dropdown list select **"Sign SOAP Body".**
- For the **"Signing Key"** select the desired certificate from the list. (Create one if it is necessary either using **"Policy Studio"** as referenced earlier or using OpenSSL as documented in the Appendix)
- The rest of the options can be left default.
- Click on **"Finish".**
- Drag a **"Reflect"** filter from the **"Utilities"** category and connect it to the **"Signature Verification"** filter with a success path.

NOTE: For more information on the configuration options for the **"Sign Message"** filter please refer to OEG Gateway User Guide located in the docs directory in the OEG Gateway install directory

STEP 2: Create new relative path:

Create a new relative path to point to this policy:

- Click on the **"Services"** module in **"Policy Studio"** and right click on **"Default Service"** node.
- Select **'Add Relative Path'**
- Create a relative path **/sign**
- Select the **'Sign Message'** policy in the list.

- Deploy the configuration by pressing the **"F6"** key or select **"Settings"** located in the top menu of **"Policy Studio"** and click on **"Deploy".**

STEP 3: Testing the Sign Message Policy using OEG Service Explorer

- Open "**OEG Service Explorer".**
- Add a SOAP message in the SOAP Request window.
- In the URL field enter the URL of the Gateway for example: http://localhost:8080/sign
- Click on the **"Run"** button.

Verification

In addition to validating XML Signatures for authentication purposes, the Gateway can also use XML Signatures to prove message integrity. By signing an XML message, a client can be sure that any changes made to the message will not go unnoticed by the Gateway. Therefore by validating the XML Signature on a message, the Gateway can guarantee the integrity of the message.

STEP 1: Create Signature Verification Policy

- Click on the **"Policies"** module in "**Policy Studio"** then right click on the **"Policies"** node and select **"Add New Policy".**
- Name the policy **"Signature Verification".**
- Drag a **"Signature Verification"** filter from the **"Integrity"** category.
- Under the **"Signature Location"** section choose **"SOAP Message Header"** from the drop down list.
- Set the **"Signature Position"** to **1.**
- Under the **"What Must Be Signed"** section select **"All Contents of SOAP Body (SOAP 1.1)"** for the XPath Expression.
- Under the **"Signer's Public Key/Certificate"** section select **"Certificate in Message".**
- Click on **"Finish".**
- Drag a **"Reflect"** filter from the **"Utilities"** category and connect it to the **"Signature Verification"** filter with a success path.

NOTE: For more information on the configuration options for the 'Signature Verification' filter please refer to OEG Gateway User Guide located in the docs directory in the OEG Gateway install directory

STEP 2: Create new relative path

Create a new relative path to point to this policy:

- Click on the **"Services"** module in **"Policy Studio"** and right click on **"Default Service"** node.
- Create a relative path **/verification**
- Select the **"Encrypt Message"** policy in the list.
- Deploy the configuration by pressing the **"F6"** key or select **"Settings"** located in the top menu of **"Policy Studio"** and click on **"Deploy".**

STEP 3: Testing the Encryption Policy using OEG Service Explorer

- Open "**OEG Service Explorer".**
- Add the SOAP Message from the previous Sign Message test in the SOAP Request window
- In the URL field enter the URL of the Gateway for example: http://localhost:8080/verification
- Click on the **"Run"** button.

5 Certificate Validation

OEG Gateway can also do validation and authenticity checks on certificates. Validating a certificate ensures that applications that use the certificate for secure communications, signing objects or encryption and decryption are unlikely to encounter problems when using the certificate. Part of the validation process is the checks of CRL or Certification Revocation List.

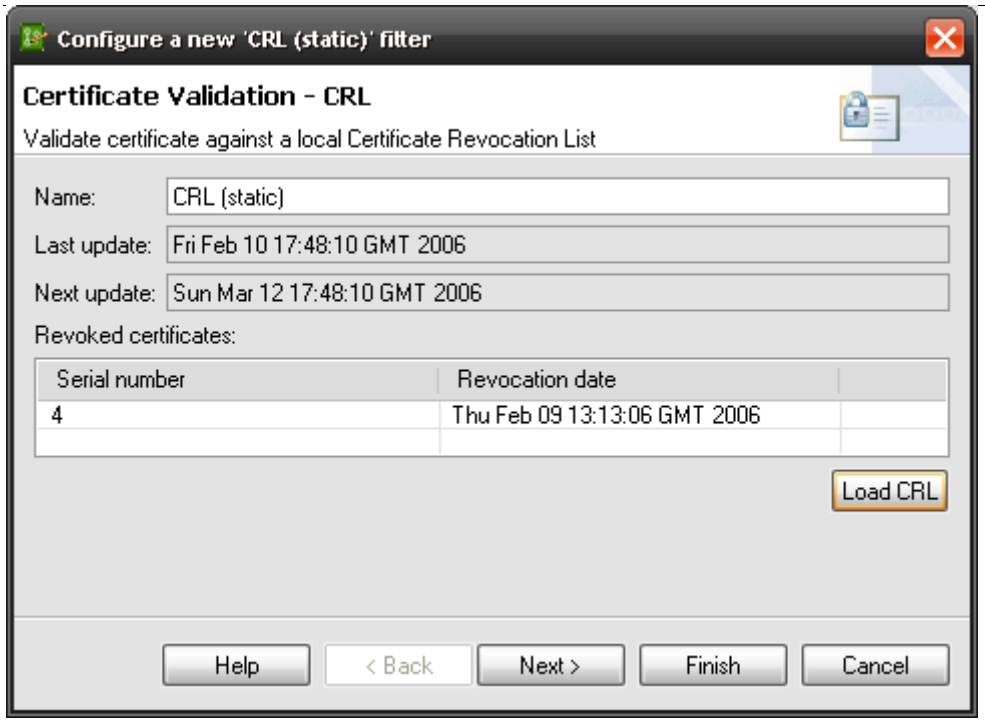
A CRL (Certificate Revocation List) is a signed list indicating a set of certificates that are no longer considered valid (i.e. revoked certificates) by the certificate issuer. The Gateway can query a CRL to find out if a given certificate has been revoked - if the certificate is present in the CRL, it should not be trusted.

In order to validate a certificate using a CRL lookup, the certificate's issuing CA's certificate should be trusted by the Gateway. This is because for a CRL lookup, the CA's public key is needed to verify the signature on the CRL. The issuing CA's public key is not always included in the certificates that it issues, so it is necessary to retrieve it from the Gateway's certificate store instead.

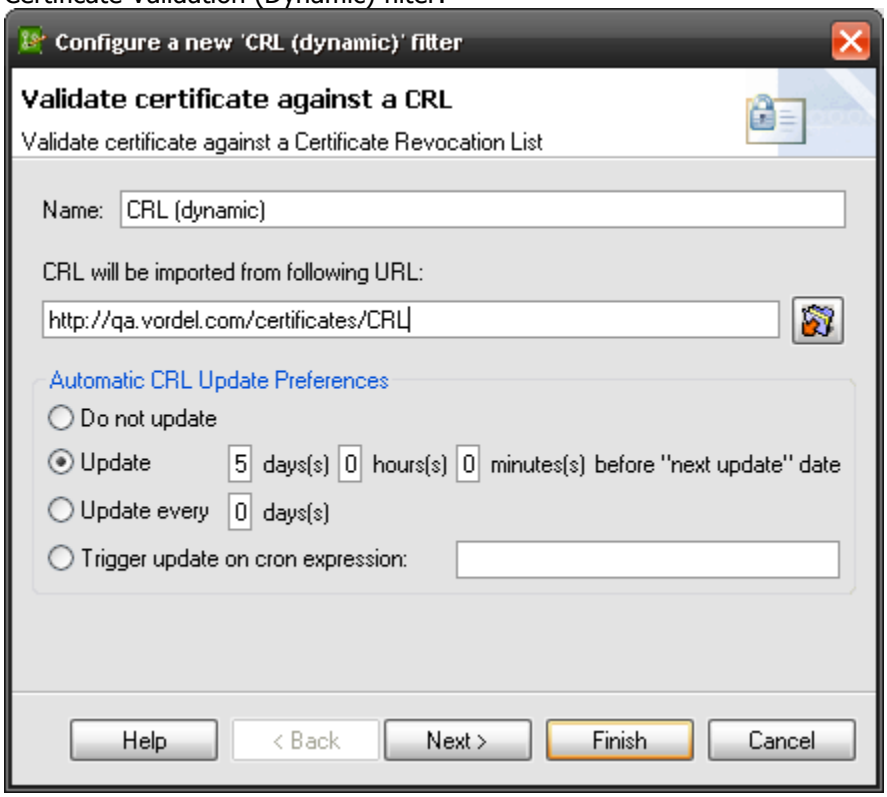
CRL Static and Dynamic Filter:

A CA may wish to publish a CRL (Certificate Revocation List) to a file. In such cases, the Gateway can load the revoked certificates from the file-based CRL and validate user certificates against it. The certificate of the CA that issued the CRL must be imported into the Certificate Store before this filter can work correctly. The Certificate CRL - Static requires the certificates message attribute to be set by a predecessor.

The Certificate Validation (Static) filter:



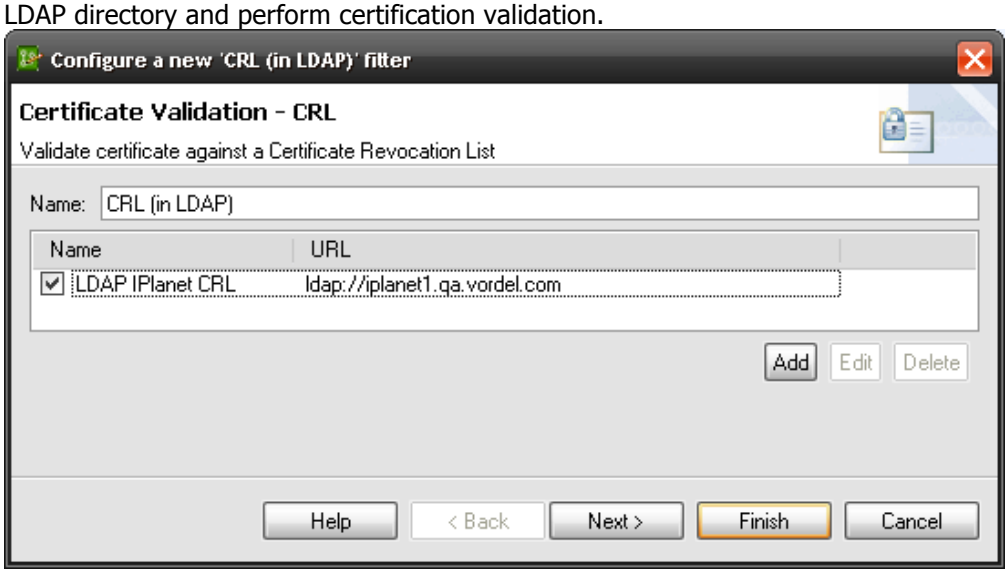
Certificate Validation (Dynamic) filter:



Typically a CA will publish a new CRL, containing the most up-to-date list of revoked certificates at regular intervals. However, the Static CRL Certificate Validation filter does not automatically update the CRL when it is loaded from a local file. To automatically retrieve updated CRL's from a particular URL, use the Dynamic CRL Certificate Validation filter.

Certificate Validation (LDAP) Filter:

CRL lists can also be stored in a LDAP directory. This filter can be configured to connect to the LDAP directory and perform certification validation.

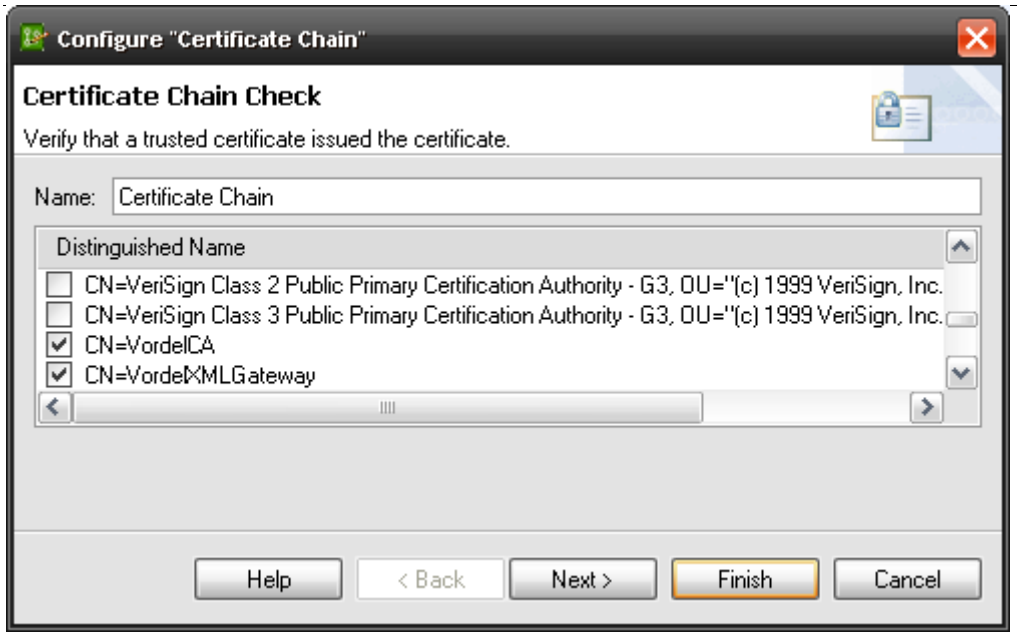


Certificate Chain Check Filter:

The Gateway can establish the authenticity of the client certificate by ensuring that the certificate originated from a trusted source. To do this a server can perform a certificate chain check on the client certificate.

The main purpose of certificate chain validation is to ensure that a certificate has been issued by a trusted source. Typically, in a Public-Key Infrastructure (PKI), a Certificate Authority (CA) is responsible for issuing and distributing certificates. The whole infrastructure is based on the premise of transitive trust - if everybody trusts the CA, then everybody transitively trusts the certificates issued by that CA. If entities only trust certificates that have been issued by the CA, they can then reject certificates which have been self-generated by clients.

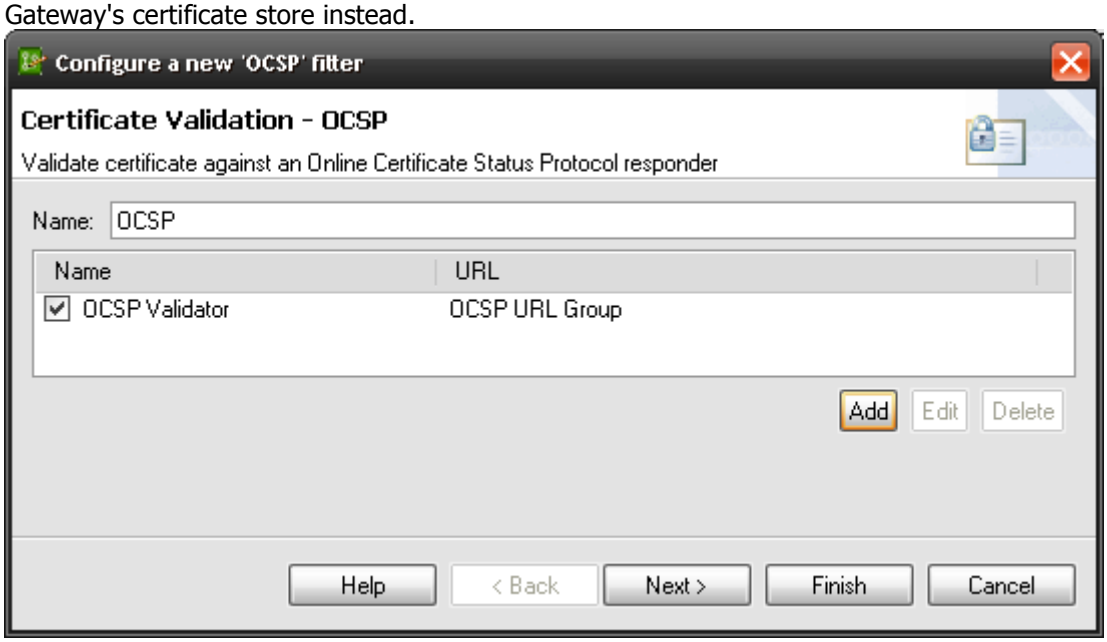
When a CA issues a certificate, it digitally signs the certificate and inserts a copy of its own certificate into it. This is called a certificate chain. Whenever an application (such as the Gateway) receives a client certificate it can extract the issuing CA's certificate from it, and run a certificate chain check to determine whether or not it should trust the CA. If it trusts the CA, it will also trust the client certificate.



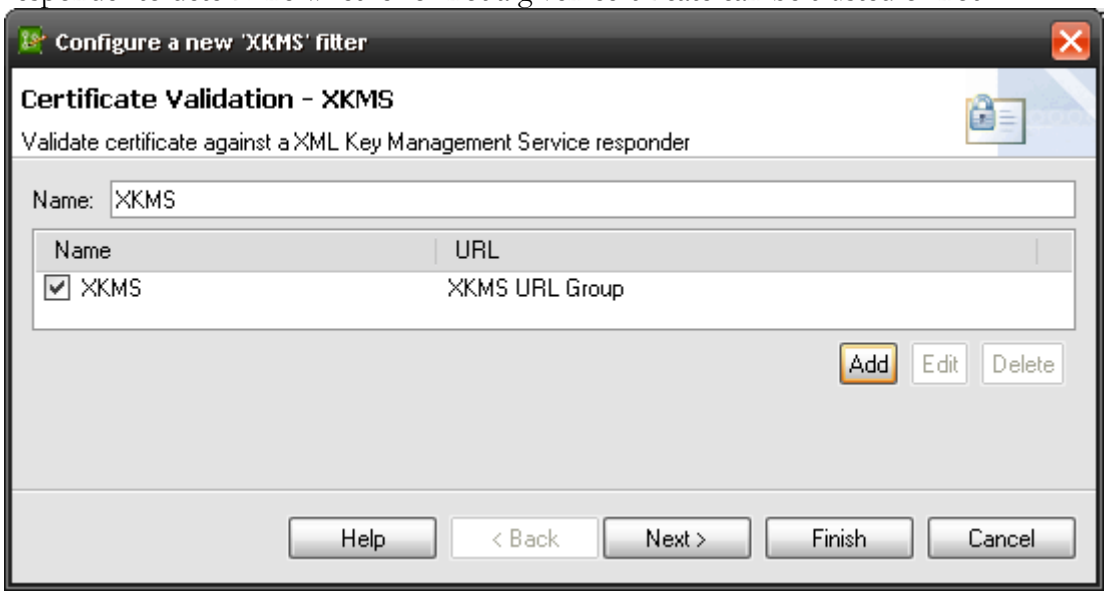
OCSP (Online Certificate Status Protocol) Filter:

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

In order to validate a certificate using an OCSP lookup, the issuing CA's certificate should be trusted by the Gateway. This is because for an OCSP request, the protocol stipulates that the CA's public key must be submitted as part of the request. The issuing CA's public key is not always included in the certificates that it issues, so it is necessary to retrieve it from the Gateway's certificate store instead.

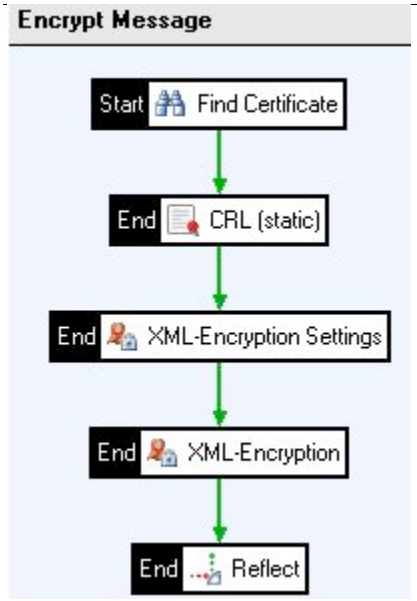


XKMS Filter:
XKMS is an XML-based protocol for (amongst other things) establishing the trustworthiness of a certificate over the Internet. The Gateway can query an XKMS responder to determine whether or not a given certificate can be trusted or not.



Sample Policy containing a CRL filter:
The encryption policy created earlier modified below to contain a CRL (Static) Filter:

1. The certificate is looked up and stored in an attribute by the 'Find Certificate' Filter
2. The CRL (static) filter checks for the validity of the certificate using a file based CRL list. This filter can be replaced by any of the certificate validation filters above.
3. The message is encrypted process using the certificate that has been validated.


Encrypt Message
Start
Find Certificate
End
CRL (static)
End
XML-Encryption Settings
End
XML-Encryption
End
Reflect

6 Conclusion

This document demonstrated how to configure the OEG Gateway to receive and forward requests via SSL and mutual SSL. It has also been demonstrated how message integrity and validation can be enforced using the OEG Gateway.
This configuration can be part of a larger policy, including features such as XML threat detection and conditional routing, features which are out of the scope of this document but are covered in other documents which can be obtained from Oracle at http://www.oracle.com.

7 Appendix

Generating a Self Signed Certificate using OpenSSL:

OpenSSL is a free, popular and robust open source toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols as well as a full-strength general purpose cryptography library. It is available on multiple platforms (Linux, BSD & Windows). OpenSSL can be used to easily create certificate signing requests (csr file) for servers to request certificate from certification authorities for example Verisign. It can also be used to create self-signed certificates to use on SSL enabled services for testing purposes.

STEP 1: Creating a Private Key
Create a private key which will be used to generate the CSR or self-signed certificate.

For example create a private key file named mycompanyca.key of strength 1024 (very strong):

```
openssl genrsa -out mycompanyca.key 1024
```

This creates the private key in the file mycompanyca.key.

STEP 2: How to Create a CSR to Request Certificate from External Certification authorities

Use the private key file mycompanyca.key to create a CSR which can be used with external certification authorities:

```
openssl req -new -key mycompanyca.key -out mycompanyca.csr
```

This creates a CSR file named mycompanyca.csr using the mycompanyca.key key file. You can submit this file to a certification authority. The data in this file will be used to create a certificate for the requester.

Note: Provide the necessary information as per example below:
Country Name (2 letter code) [IE]:
State or Province Name (full name) []:
Locality Name (e.g., city) [DUBLIN]:
Organization Name (e.g., company) [My Company Ltd]:
Organizational Unit Name (e.g., section) []:
Common Name (e.g., server's hostname) [myserver]:
Email Address []:
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
Specify the domain name of your server as the Common Name. For example to generate a CSR for the domain https://qa.vordel.com I must use qa.vordel.com as the common name.

STEP 3: How to create self-signed certificate
You can create a self-signed certificate for your own servers using the procedure below:

```
openssl x509 -req -days 365 -in mycompanyca.csr -signkey mycompanyca.key -out
mycompanyca.crt
```

Notes:
1. Replace 365 with the number of days the certificate should be valid.
2. A CSR (see above) must first be created before this command is run
The certificate will be saved in the file mycompanyca.crt.
Self-signed certificates will not be recognized by browsers. When accessing websites or services using such certificate user will be asked to accept / reject the certificate. Certificates signed by recognized external certification authority are automatically accepted by browsers.

Oracle Enterprise Gateway
May 2011
Author:

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

Oracle is committed to developing practices and products that help protect the environment

SOFTWARE. HARDWARE. COMPLETE.