WS-Security

In this section:

To help you ensure that your security measures work flawlessly in terms of authentication, encryption, and access control, SOAtest contains a vast array of security tools and options that fully supports the industry standard WS-Security specification.

In the example given in the WS-Security test suite, examples of encryption/decryption, digital signature, and the addition of SOAP Headers are shown. The following are key security tools and options that SOAtest supports:

- **XML Encryption Tool:** The XML Encryption tool allows you to encrypt and decrypt entire messages or parts of messages using Triple DES, AES 128, AES 192, or AES 256. In WS-Security mode, Binary Security Tokens, X509IssuerSerial, and Key Identifiers are supported.
- **XML Signer Tool:** The XML signer tool allows you to digitally sign an entire message or parts of a message depending on your specific needs. In some cases it may be important to digitally sign parts of a document while encrypting other parts.
- **XML Signature Verifier Tool:** The XML verifier tool allows for the verification of digitally signed documents using a public/private key pair stored within a key store file.
- **Key Stores:** The use of key stores in SOAtest allows you to encrypt/decrypt and digitally sign documents using public/private key pairs stored in a key store. Key stores in JKS, PKCS12, BKS, PEM, and UBER format can be used.
- **Username Tokens, SAML Tokens, X509 Tokens, or Custom Headers:** SOAtest supports sending custom SOAP Headers and includes templates for Username Tokens and SAML tokens.

When you complete this section of the tutorial, your test suite should resemble the test suite entitled "WS-Security" in the `SOAtestTutorial.tst` file.

Unlimited Strength Java Cryptography Extension

In order to perform security tests using the XML Signature Verifier, XML Signer, or XML Encryption tools, or if using Key Stores, you will need to download and install the Unlimited Strength Java Cryptography Extension. For details, see JCE Prerequisite.

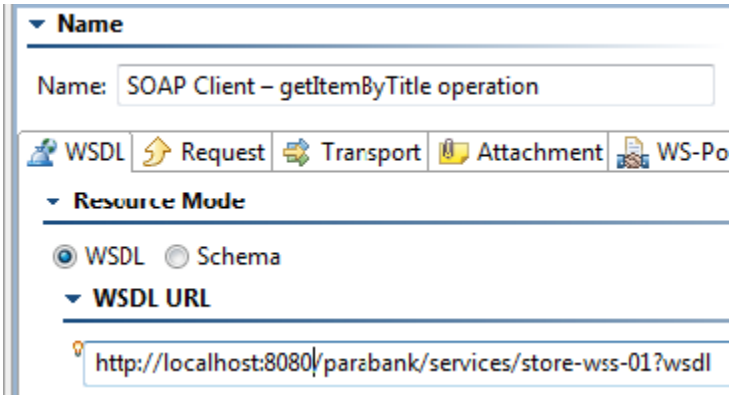
Message Layer Security with SOAP Headers

In this example we will use a book store web service, which requires a Username and Password to be submitted within the SOAP Header element according to the WS-Security specification. SOAtest provides the ability to add Custom Headers and also provides pre-defined templates for creating User-name Tokens and SAML Tokens. The following example uses a Username Token.

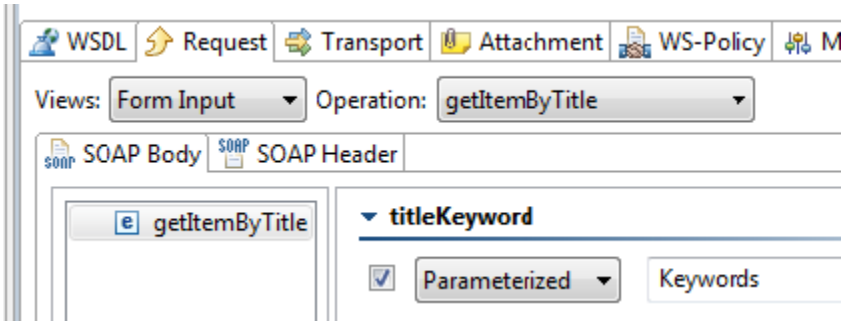
1. Right-click the project from the previous exercises, then choose **Add New> Test (.tst) File** from the shortcut menu.
2. Enter a name for the file, then click **Next**.
3. Select **Empty** and click **Finish**. An empty test suite folder is created.
4. Double-click the new **Test Suite: Test Suite** node that was added.
5. Type `WS-Security` into the **Name** field in the configuration panel on the right.
6. Click the **Save** button to save the WS-Security test suite.
7. Copy the **Excel: Books** data source that you added in the Functional Test lesson and paste it into this test suite.
8. Select the **Test Suite: WS-Security** node and click the **Add Test Suite** button.

9. Select **Empty** and click **Finish**. An empty test suite folder is created.
10. Double click the test suite to open its editor and name the suite `Username Tokens`.
11. Click the **Save** button to save the Username Tokens test suite.
12. Select the **Test Suite: Username Tokens** node and click the **Add Test or Output** button.

13. In the **Add Test** wizard, select **Standard Test** from the left pane, and **SOAP Client** from the right pane, and click **Finish**. A SOAP Client tool is added to the test suite.
14. Double-click the **Test 1: SOAP Client** node beneath the **Test Suite: Username Tokens** node.
15. Complete the SOAP Client tool's configuration panel as follows:
    1. Enter `SOAP Client – getItemByTitle operation` in the **Name** field.
    2. Open the **WSDL** tab and enter the following in the **WSDL URI** field:
       `http://localhost:8080/parabank/services/store-wss-01?wsdl`



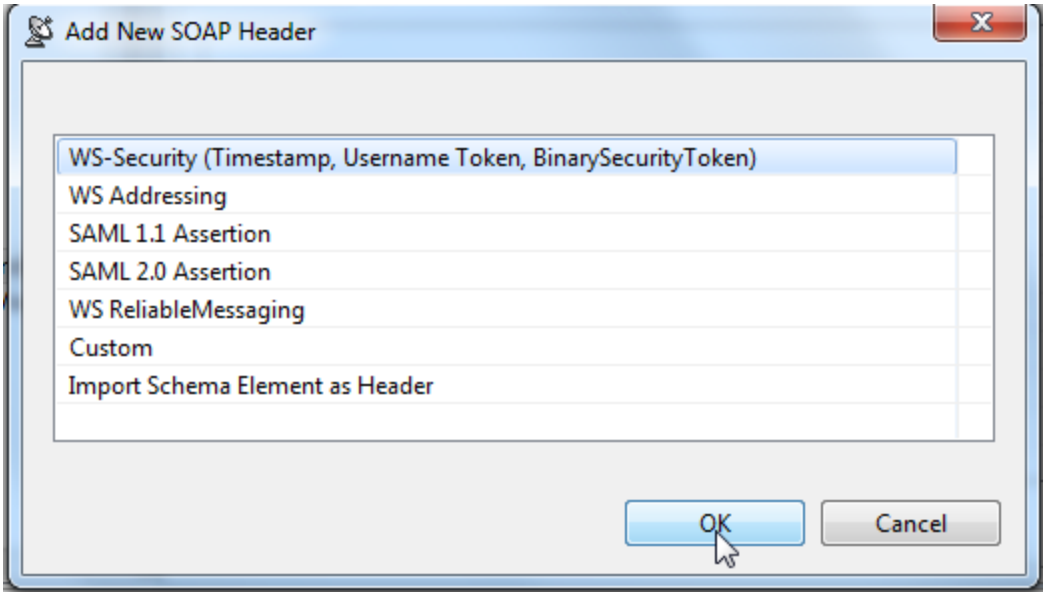
    3. Open the **Request** tab and select **getItemByTitle** from the **Operation** drop-down menu.
    4. Check the **titleKeyword** element, then select **Keywords** as its **Parameterized** value.



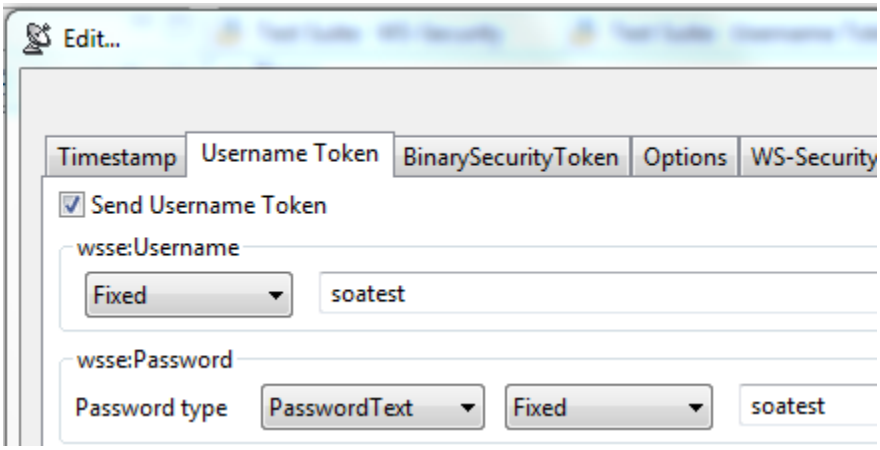
16. Click the **Save** toolbar button to save the modified test.
17. Run the test by clicking the **Test** toolbar button. Notice that the test fails because it did not have the required Security Header.

To add the required SOAP Header:

1. Double-click the **Test 1: SOAP Client - getItemByTitle operation** node.
2. Open the **SOAP Header** tab in the tool's configuration panel, then click the **Add** button. An Add New SOAP Header dialog opens.
3. Select WS-Security then click **OK**.



4. Double-click the new entry added to the SOAP Header table. A dialog will open.
5. In the **Timestamp** tab, clear the **Send Timestamp** check box.
6. Open the **Username Token** tab and complete the following:
    1. Enter `soatest` in the **wsse:Username** field.
    2. Enter `soatest` in the **wsse:Password** field.



7. Click **OK**.
8. Click the **Save** toolbar button to save the modified test.
9. Run the test by clicking the **Test** toolbar button. The test now succeeds.
10. Double-click the **Traffic Viewer** node to view the SOAP Header sent in the request and verify that the service returned information about the specified books.
11. To create a regression control that will alert you to any changes in the server response in the future, right click **Test 1: SOAP Client – getItemByTitleOperation** and choose **Create/Update Regression Control** from the shortcut menu.
12. In the Response Validation Wizard, select **Create Regression Control> Create Multiple Controls**, then click **Finish**.

If you run the test a few more times you will notice that it fails because the **price** element has changed. Follow the steps from previous exercises to ignore the dynamically changing **price** value.

Using the XML Encryption Tool

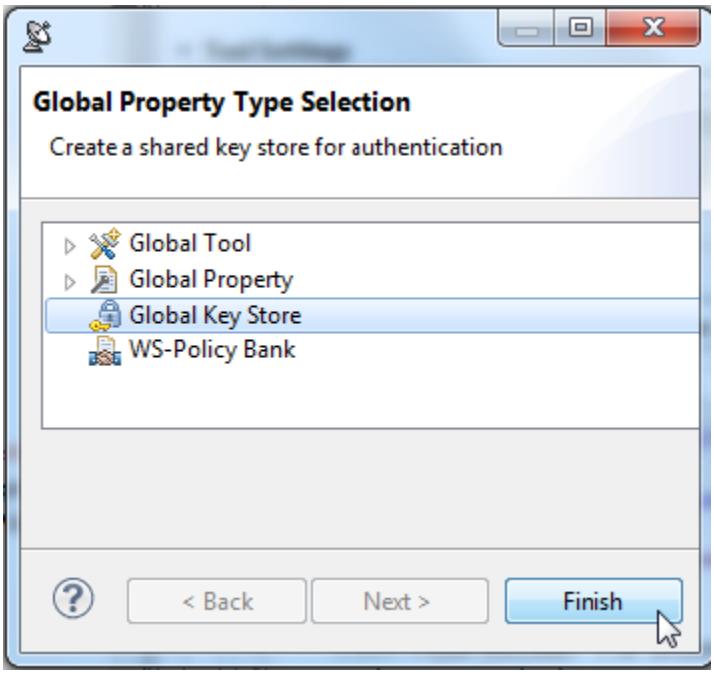
In this example, we will use a book store service similar to the service used in previous examples, except that:

- Request bodies must be encrypted using the key store `soatest.pfx` which is located in the `examples\keystores` directory.
- Responses are encrypted as well and can be decrypted using the same key store.

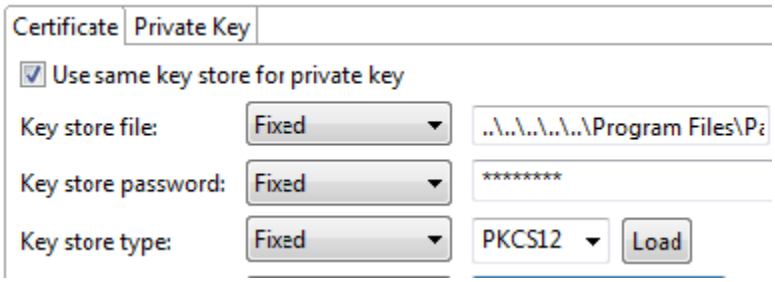
First you will need to set up the key store:

1. Select the **Test Suite: WS-Security** node and click the **Add Property** toolbar button.

2. In the Global Property Type Selection dialog, select **Global Key Store** and click **Finish**.



3. Complete the Key Store configuration panel as follows:
    1. Enter `PKCS12 Keystore` in the **Name** field in the GUI panel.
    2. Make sure the **Use same key store for private key** check box is selected.
    3. Click the **Filesystem** button and navigate to the location of the key store `soatest.pfx`.
        - **For Windows:** `C:\Program Files\Parasoft\SOAtest\[SOAtest version number]\examples\keystores`.
        - **For Others:** `[SOAtest installation directory]\examples\keystores`.
    4. Enter `security` in the **Key Store Password** field.
    5. Select **PKCS12** from the **Key Store Type** drop-down menu.
    6. Click the **Load** button.

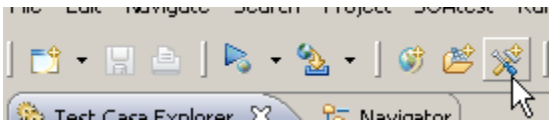


    The list of available certificate aliases within the keystore are populated into the **Certificate Alias** drop-down menu.
    7. Select **soatest** in the **Certificate Alias** field.
    8. Open the **Private Key** tab at the top of the Key Store configuration panel and click **Load**.
    9. Select **soatest** for the **Private Key Alias** and enter `security` for the **Private Key Password**.
4. Click the **Save** toolbar button.

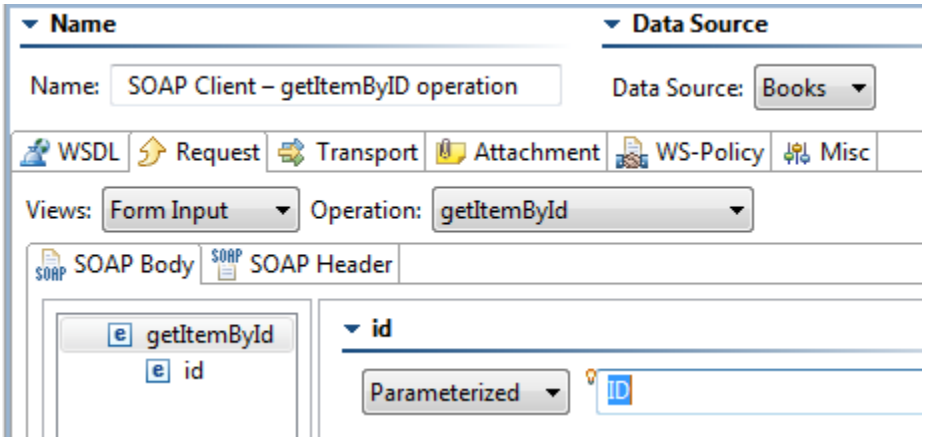
Now we are ready to set up a test using the XML Encryption tool. To better organize our security tests, we will create a new folder for the encryption test.

1. Select the **Test Suite: WS-Security** node and click the **Add Test Suite** button.

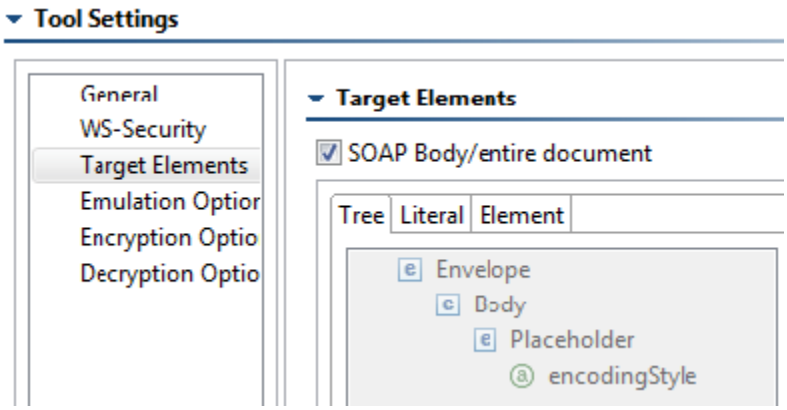
2. Select **Empty** and click **Finish**. An empty test suite folder is created.
3. Type `Encryption/Decryption` into the **Name** field in the right GUI panel.
4. Click the **Save** toolbar button.
5. Select the **Test Suite: Encryption/Decryption** node and click the **Add Test or Output** button.



6. Select **Standard Test** from the left pane, and select **SOAP Client** from the right pane, and click **Finish**. A SOAP Client tool is added to the test suite.
7. Complete the SOAP Client tool's configuration panel as follows:
    1. Enter `SOAP Client – getItemByID operation` in the **Name** field.
    2. Open the **WSDL** tab and enter the following in the **WSDL URI** field:
       `http://localhost:8080/parabank/services/store-wss-03?wsdl`
    3. Open the **Request** tab.
    4. Select **getItemById** from the **Operation** drop-down menu.
    5. For the **id** element, select **ID** as its **Parameterized** value.



8. Click the **Save** toolbar button.
9. Right-click the **Test 1: SOAP Client - getItemByID** operation node and select **Add Output** from the shortcut menu. The **Add Output** wizard displays.
10. Select **Request> SOAP Envelope** from the left pane, and select **XML Encryption** from the right pane, and click the **Finish** button. An Encryption Tool is chained to the SOAP Client.
11. Complete the Request SOAP Envelope -> XML Encryption tool's configuration panel as follows:
    1. Ensure that the **Encrypt** radio button is selected.
    2. Ensure that the **WS-Security Mode** box is checked.
    3. Select **AES 256** from the **Symmetric (Block Encryption)** drop down menu.
    4. Open the **WS-Security** page and ensure that **X509BinarySecurityToken** is selected in the **Form** box.
    5. Open the **Target Elements** page and verify that the **SOAP Body/entire document** check box is selected. This will encrypt the XML Body element.

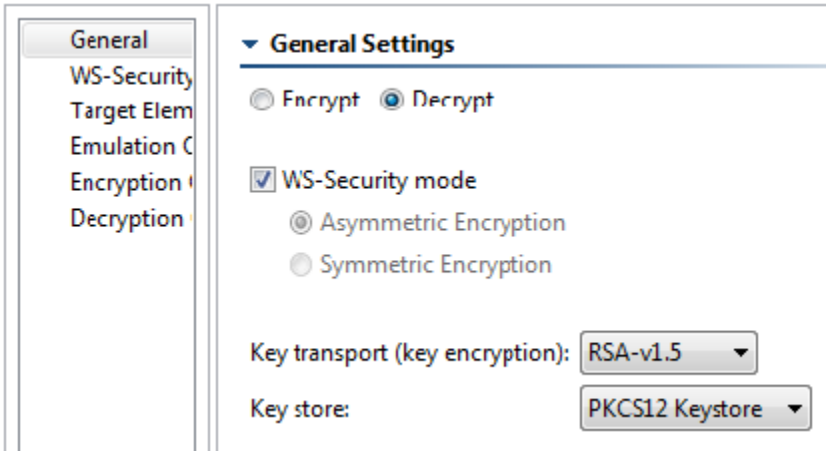


The XML Request is now set up to be encrypted when the request is sent to the service.

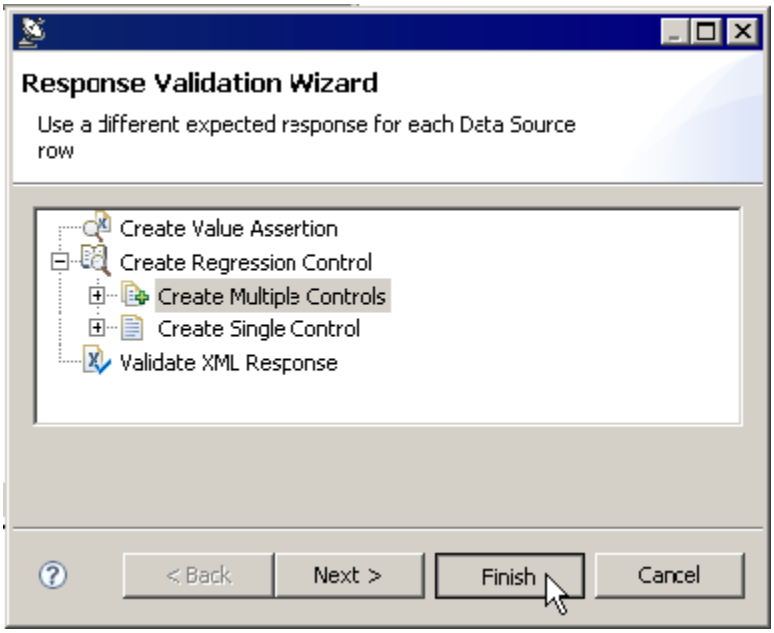
6. Click the **Save** toolbar button to save the modified test.

Now you can add an XML Encryption tool to the XML Response of the SOAP Client test to enable Decryption of the XML response.

1. Right-click the **Test 1: SOAP Client - getItemByID** operation node and select **Add Output** from the shortcut menu. The **Add Output** wizard displays.
2. Select **Response> SOAP Envelope** from the left pane, and select **XML Encryption** from the right pane, and click the **Finish** button. An Encryption Tool is chained to the SOAP Client.
3. Complete the Response SOAP Envelope -> XML Encryption tool's configuration panel as follows:
    1. Select the **Decrypt** radio button.
    2. Select the **PKCS12 Keystore** from the **Key Store** drop down menu.



4. Click the **Save** toolbar button to save the modified test.
5. Run the test by clicking the **Test** toolbar button.
6. Double-click the **Traffic Viewer** node to view the encrypted data.
7. Right-click the **Test 1: SOAP Client - getItemByID operation** node and select **Create/Update Regression Control**.
8. In the dialog that opens, select **Create Regression Control> Create Multiple Controls**, then click **Finish**.



Regression controls are created and automatically chained to the **Response SOAP Envelope -> XML Encryption**. Notice that the decrypted responses are shown in the Regression Control.

Finally, you want to ignore dynamic values from the XML Response so that the Regression Control does not fail each time.

1. Double-click the **XML Document -> Diff** node and complete the following in the right GUI panel:
    1. Set the **Diff Mode** to **XML**.

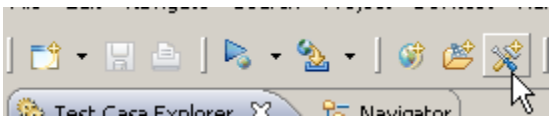
2. Select **Form XML** as the **Diff Mode**. When the Form XML tab is selected, a popup will appear asking whether to override with values from Literal XML view. Click **Yes**.
    1. Right-click the **price** element and select **Setup Ignored XPath** from the shortcut menu. An **Ignore XPath Setting** dialog appears. Click **OK** to ignore modifications to the **textcontent** of the **price** element.
    2. Repeat the previous step for the **CipherValue** element.
    3. Right click on the **DataReference** element and select **Setup Ignored XPath**. An **Ignore XPath Setting** Dialog appears. Select the **Attribute** check box to ignore changes to the attributes of the **DataReference** element. Click **OK**.
    4. Select the **Literal XML** button to switch back to Literal XML view.
    5. In the dialog that asks whether to override with values from Form XML view, click **Yes**.
3. Click the **Save** toolbar button to save the modified test.
4. Run the test by clicking the **Test** toolbar button.

Using the XML Signer Tool

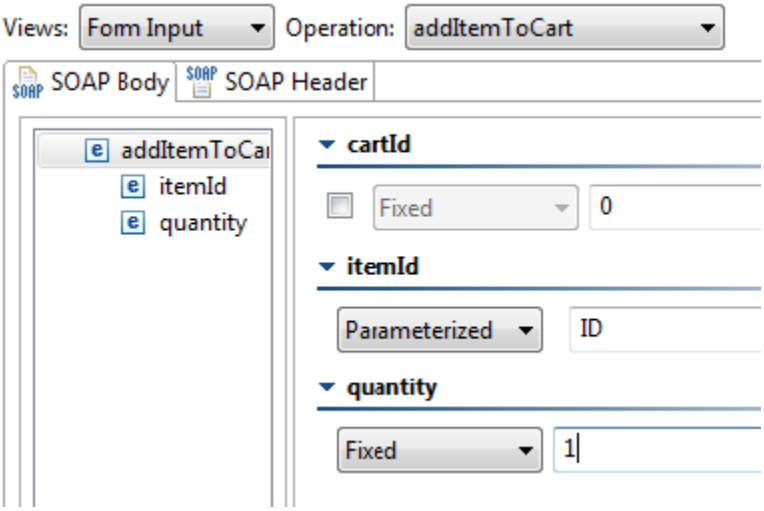
In the next example, we will use a book store service which requires request bodies to be signed with the certificate in the key store `soatest.pfx`. Responses from this service are signed as well and can be verified using the same key store. We will use the same key store settings from the previous example.

1. Select the **Test Suite: WS-Security** node and click the **Add Test Suite** button.

2. Select **Empty** and click **Finish**. An empty test suite folder is created.
3. Type `Sign/Verify` into the **Name** field in the right GUI panel, then click the **Save** toolbar button.
4. Select the **Test Suite: Sign/Verify** node and click the **Add Test or Output** button.

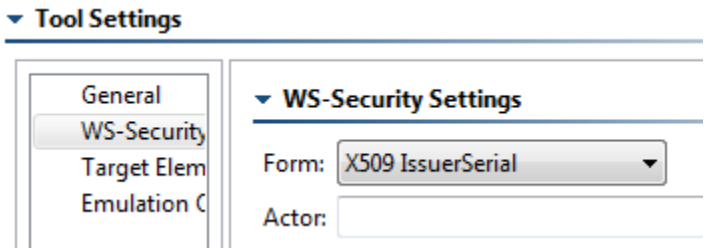


5. Select **Standard Test** from the left panel and **SOAP Client** from the right panel and click **Finish**. A SOAP Client tool is added to the test suite.
6. Complete the SOAP Client tool's configuration panel as follows:
    1. Enter `SOAP Client – addItemtoCart` operation in the **Name** field.
    2. Open the **WSDL** tab and enter the following in the **WSDL URI** field:
    `http://localhost:8080/parabank/services/store-wss-02?wsdl`
    3. Open the **Request** tab.
    4. Select **addItemtoCart** from the **Operation** drop-down menu.
    5. For the **itemId** parameter, select **ID** as its **Parameterized** value.
    6. For the **quantity** parameter, enter **1** as its **Fixed** value.
    7. Leave the **cartId** parameter unchecked. This will automatically generate a new cartId.



7. Click **Save to** save the modified test.
8. Right-click the **Test 1: SOAP Client - addItemtoCart** operation node and select **Add Output** from the shortcut menu. The **Add Output** wizard displays.

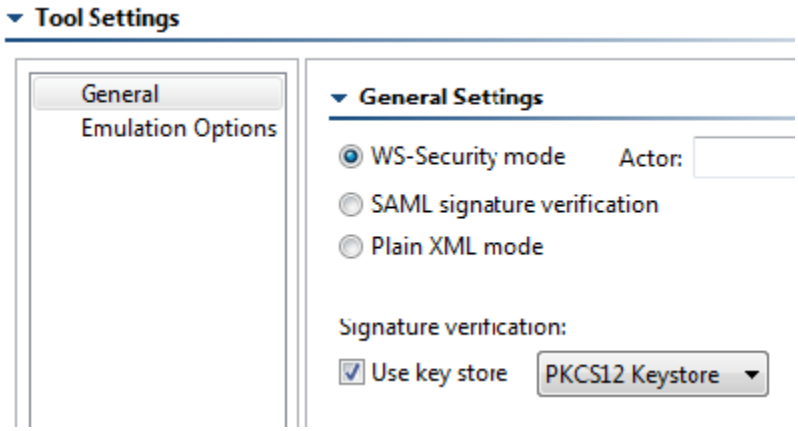
9. Select **Request> SOAP Envelope** from the left panel, select **XMLSigner** from the right panel, and click **Finish**. An XML Signer Tool is chained to the SOAP Client.
10. Complete the XML Signer tool's configuration panel as follows:
    1. Select **PKCS12 Keystore** from the **Key Store** drop down menu.
    2. Select **RSAwithSHA1 (PKCS1) – http://www.w3.org/2000/09/xmldsig#rsa-sha1** from the **Algorithm** drop down menu.
    3. Open the **WS-Security** page and select **X509IssuerSerial** from the **Form** box.



    4. Open the **Target Elements** page and verify that the **SOAP Body/entire document** check box is selected. The XML Request is now set up to be signed when the request is sent to the service.
    5. Click **Save** to save the modified test.

Now you can add an XML Verifier Tool to the XML Response of the SOAP Client test to enable Signature Verification of the XML response:

1. Right-click the **Test 1: SOAP Client - addItemtoCart** operation node and select **Add Output** from the shortcut menu. The **Add Output** wizard displays.
2. Select **Response> SOAP Envelope** from the left pane, and select **XML Signature Verifier** from the right pane, and click **Finish**. An XML Signature Verifier Tool is chained to the **Test 1: SOAP Client - addItemtoCart** operation node.
3. Complete the XML Signature Verifier tool's configuration panel as follows:
    1. Select the **Use Key Store** check box and choose **PKCS12 Keystore** from the drop-down menu.
    2. Ensure that the **WS-Security Mode** check box is checked.



4. Click the **Save** toolbar button to save the modified test.
5. Run the test by clicking the **Test** toolbar button.
6. Double-click the **Traffic Viewer** node to view the signed data. Since the test succeeds, this tells us that the server accepted our signed request and the server's signed response was successfully verified.

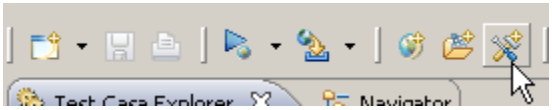
XML Encryption and Signature Combined

In this example, we will create a more complex test using a book store service which combines the security requirements of the previous two exercises. This service requires request bodies to be signed and encrypted using the key store **soatest.pfx**. The responses from this service are signed and encrypted as well and can be decrypted and verified using the same key store.

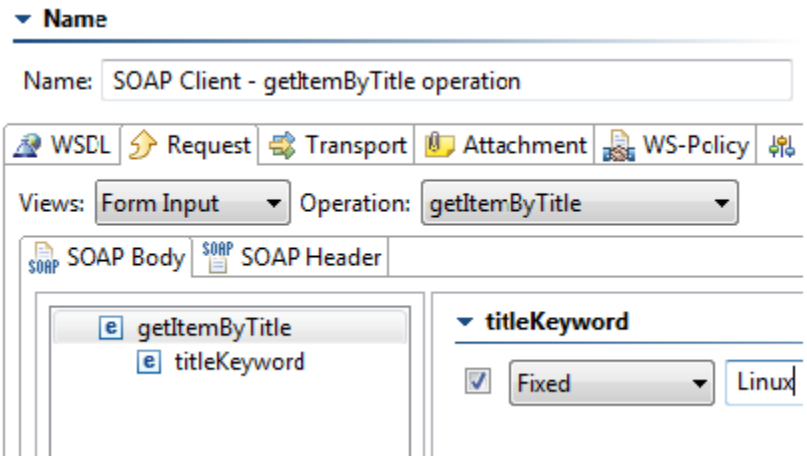
1. Select the **Test Suite: WS-Security** node and click the **Add Test Suite** button.

2. Select **Empty** and click **Finish**. An empty test suite folder is created.
3. Type `Encryption and Signature Combined` into the **Name** field in the right GUI panel, then click the **Save** toolbar button.

4. Select the **Test Suite: Encryption and Signature Combined** node and click the **Add Test or Output** button.



5. In the Add Test Wizard, select **Standard Test** from the left pane, and select **SOAP Client** from the right pane, and click **Finish**. A SOAP Client tool is added to the test suite.
6. Complete the SOAP Client tool's configuration panel as follows:
    1. Enter `SOAP Client - getItemByTitle` operation in the **Name** field.
    2. Open the **WSDL** tab and enter the following in the **WSDL URI** field:
    `http://localhost:8080/parabank/services/store-wss-04?wsdl`
    3. Open the **Request** tab.
    4. Select **getItemByTitle** from the **Operation** drop-down menu.
    5. Check the **titleKeyword** parameter and enter **Linux** as its **Fixed** value.



7. Right-click the **Test 1: SOAP Client - getItemByTitle** operation node and select **Add Output** from the shortcut menu. The **Add Output** wizard displays.
8. Select **Request> SOAP Envelope** from the left pane, and select **XML Signer** from the right pane, and click the **Finish** button. An XML Signer Tool is chained to the SOAP Client.
9. Complete the XML Signer tool's configuration panel as follows:
    1. Select **PKCS12 Keystore** from the **KeyStore** drop down menu.
    2. Select **RSAwithSHA1** from the **Algorithm** drop down menu.
    3. Open the **WS-Security** page and choose **X509BinarySecurityToken** from the drop down menu.
    4. Open the **Target Elements** page and ensure that **SOAP Body/entire document** is checked. The XML Request is now set up to be signed when the request is sent to the service.

Next you can add an XML Encryption Tool to the XML Response of the XML Signer Tool to encrypt the signed document.

1. Right-click the **Request SOAP Envelope> XML Signer** node and select **Add Output** from the shortcut menu. The **Add Output** wizard displays.
2. Select **XML Encryption** and click the **Finish** button. An XML Encryption Tool is chained to the XML Response of the XML Signer Tool.
3. Complete the XML Encryption Tool tool configuration panel as follows:
    1. Ensure that the **Encrypt** radio button is selected.
    2. Choose **PKCS12 Keystore** from the **KeyStore** drop down menu.
    3. Select **AES256** from the **Symmetric** drop down menu.
    4. Open the **WS-Security** page and select **X509BinarySecurityToken** from the **Form** box.
    5. Open the **Target Elements** page and verify that the **SOAP Body/entire document** check box is selected. The XML Request is now set up to be signed when the request is sent to the service.
    6. Click **Save** to save the modified test.
4. Run the test by clicking the **Test** toolbar button.
5. Double-click the **Traffic Viewer** node to view the server response.

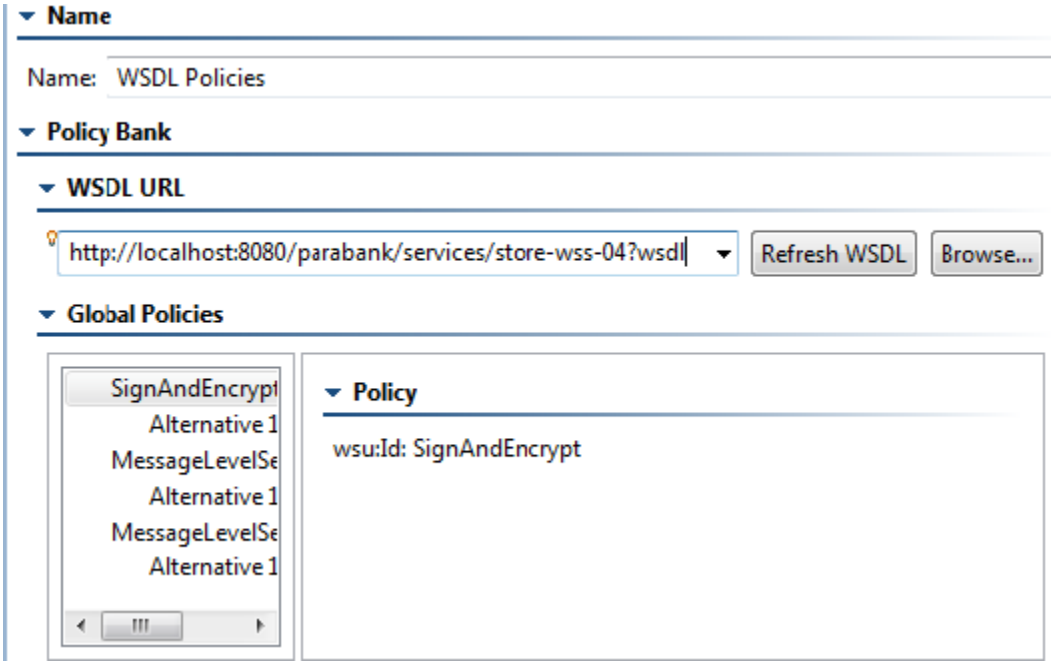
Automatically Generating WS-Security Tests with WS-SecurityPolicy

Parasoft enables automatic test creation to enforce runtime security policies. This helps you automatically generate the correct tests with the correct settings so the services can be invoked instantly. Furthermore, by managing the policies at the project test level, you can more easily create and manage various policy variations in order to test the services properly, both positive and negative.

SOAtest recognizes WS-SecurityPolicy assertions in the WSDL when using the WS-PolicyAttachment standard. In order to automatically generate tests from a WSDL with WS-SecurityPolicy assertions, complete the following:

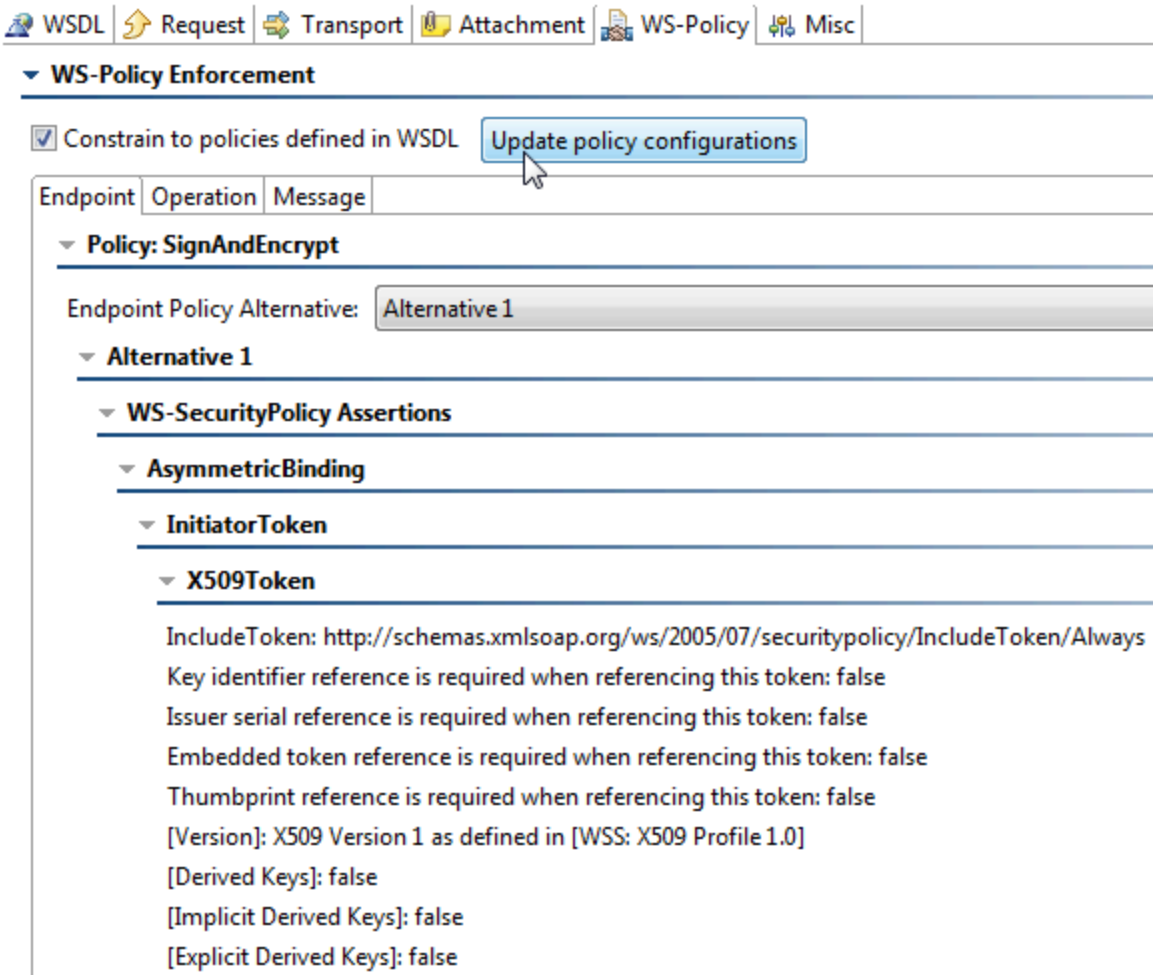
1. Select the **Test Suite: WS-Security** node and click the **Add Property** toolbar button.

2. In the **Global Property Type Selection** dialog, select **WS-Policy Bank** and click **Finish**. A **WS-Policy Banks** node is added to the Test Case Explorer.
3. In the WSDL Policies configuration panel on the right side of the GUI, enter `http://localhost:8080/parabank/services/store-wss-04?wsdl` in the **WSDL URL** field and click the **Refresh from WSDL** button. The Global Policies are populated.



Notice how there are policy nodes that include the WS-SecurityPolicy configuration that corresponds to the WS-SecurityPolicy assertions in the WSDL.

Notice that tests generated with the same WSDL now have WS-Policy properties that match the WSDL Policies. If you want SOAtest to automatically configure the signer and encryption tool on the request (because the policy dictates so), you could enable **Constrain to policies defined in WSDL** and click **Update policy configurations**.



Since a keystore has already been added to the test suite, the tests are ready to run. If you have not added a keystore, one needs to be configured. For more information on adding a keystore, see Using the XML Encryption Tool.