Bare SOAP-UI for WS-Security

Paul Glezen, IBM

Abstract

This document is a member of the Bare Series of WAS topics distributed in both stand-alone and in collection form. The latest renderings and source are available on GitHub at http://pglezen.github.io/was-config.

Table of Contents

If you lock down your service provider too tightly, not even your testers can invoke it with SOAP-UI. This section explains how to configure SOAP-UI to invoke a web service that only accepts payloads with timestamps signed by certain parties. Of course, one of those parties needs to be the tester.

1. Generating the Keystore

These steps describe how to create a tester key store for use in signing requests to the provider. There is no requirement that this be done on WAS. One could use the JDK keytool if desired. The WAS admin console is just easier. Since there is no requirement to manage this keystore as part of a cell, it is not created within the cell configuration and synchronized across nodes.

The public certifiate associated with the tester's key is exported so that it can be included within the provider's trust store.

1. In the WAS admin console, navigate to Security → SSL Certificate and key management → Key stores and certificates.
2. Click the Add button.
    a. For Name, enter `Tester`.
    b. For Description, enter `Key store used by testers, not by WAS.`
    c. For Path, enter a fully qualified path name outside the cell configuration directory. The filename extension should be `.jks`.
    d. Enter a password for the key store.
    e. For Type, select JKS.
    f. Click OK.

    The result is a new empty key store named Tester.

3. For the Tester key store, click the Personal certificates link to create a new public/private key pair.

4. Click the Create button and select self-signed Certificate from the dropdown list.

    a. For Alias, enter `tester1`.

    b. For Common name, enter `Acme Tester 1`.

    c. For Validity period, enter a suitable validity period.

    d. For Organization, enter `Acme`.

    e. Enter any of the other fields you like and click OK.

    Tester1 is now in your list of personal certificates.

5. Extract the Tester1 public certificate.

> **Extract vs Export**
>
> The Personal Certificates list has two buttons that appear similar in meaning: Extract and Export. The Extract button refers to a certificate associated with the key. The result is a certificate file for use in importing into a trust store.
>
> The Export button refers to a public/private key pair. Its destination is either a new or existing key store.

    a. In the personal certificates list, check the box next to tester1 alias and click the Extract button.

    b. In the Certificate file name field, enter the fully qualified file name of the certificate file with a `.cer` extension. This will be the location from which it will be imported into the service provider's trust store.

    c. Click OK.

6. Import the tester1 certificate into the provider trust store.

    a. Navigate to the Signer certificates section of the CC Provider trust store.

    b. Click the Add button.

    c. The alias is not so important. It's good to enter something related to the certificate to be imported. The `tester1` alias used for creating the certificate is a good choice.

    d. Enter the fully qualified file name of the tester certificate you extracted in the previous step.

    e. Click OK.

    f. Save the configuration.

The provider trust store should look similar to the screen shot in Figure 1.

Figure 1. CC Provider Trust Store with Tester Certificate

ccconsumer	CN=CC Consumer Application, O=Acme, C=US	4D:FA:60:89:C0:76:A6
root	CN=WIN-UCDAOI2236T, OU=Root Certificate, OU=WIN-UCDAOI2236TNode07Cell,	2A:D5:8C:5A:2A:71:A1
tester1	CN=Acme Tester 1, OU=CC, O=Acme, C=US	A7:FD:D6:D7:C2:2F:24

The root entry is the WAS cell root that is created automatically for each new key store. It does not play a role here. The ccconsumer certificate belongs to the consumer application and tester1 is the tester certificate that was just imported. It will allow the service provider to accept requests signed by this tester.

The next section will explain how to configure the tester's SOAP-UI installation to sign requests with the new key.

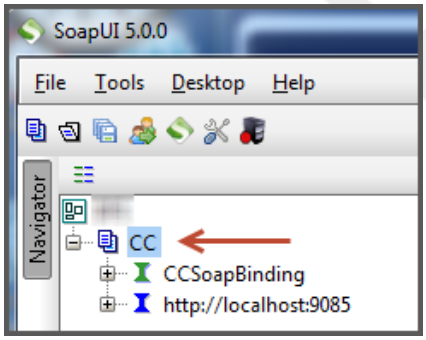
2. Configure SOAP-UI

In SOAP-UI we start with a SOAP project that invokes a service provider. The SOAP-UI has no WS-Security configured. They keystore and its passwords from the previous step are readily available.

2.1. Add Keystore to SOAP-UI

1. Double-click on your SOAP project to bring up the project configuration panel.

Figure 2. Double-click CC SOAP-UI Project

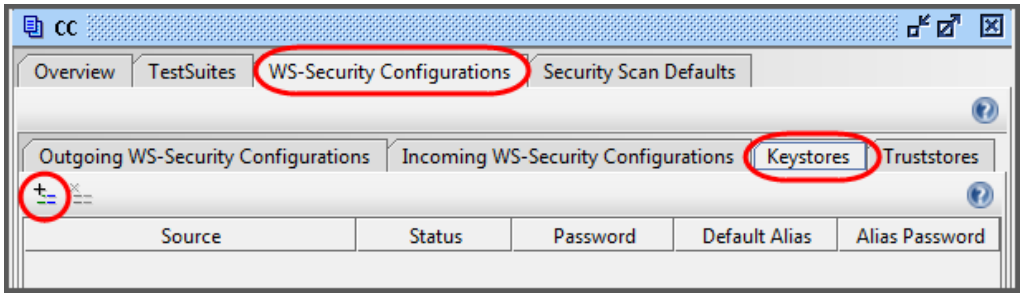


2. Select the WS-Security Configurations tab.

3. Select the Keystore tab.

4. Select the + button to add a new key store.

Figure 3. Project Keystore Configuration

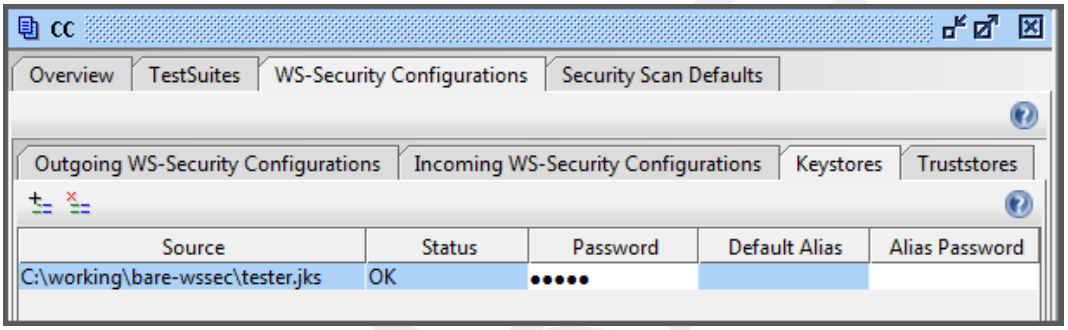


5. Select the tester key store (with the `.jks` extension).

6. Enter the key store password. (There shouldn't be an alias password since we didn't password-protect the key.)

This completes the configuration of the key store within SOAP-UI. The key store panel should look like Figure 4.

Figure 4. Project Keystore Configuration



2.2. Add WS-Security to SOAP-UI

This section continues from configuration panel of the previous section.

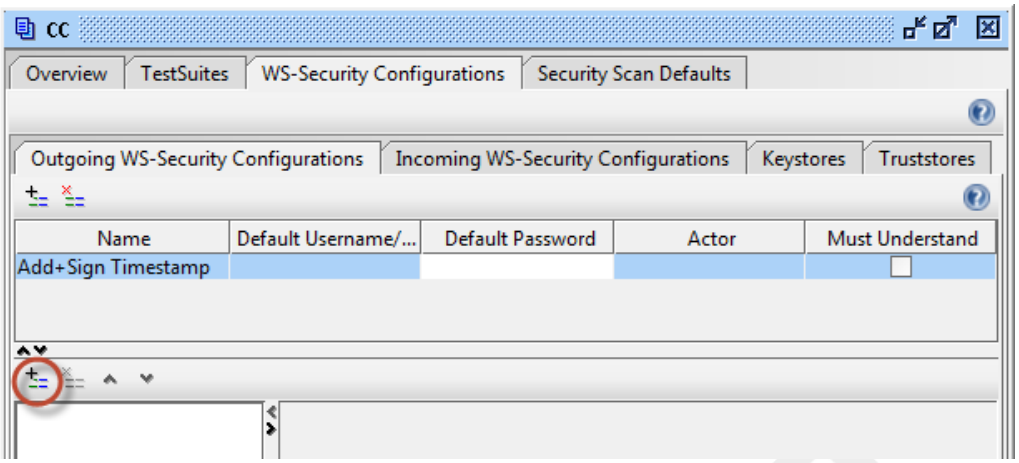
1. Select the Outgoing WS-Security Configurations sub-tab to the left of the Keystores sub-tab from the previous section.

2. Click the + button on the upper-left to add a new outgoing WS-Security configuration.

3. For the name of the configuration, enter `Add+Sign Timestamp`.

4. In the lower panel, click the + button to add a new WSS entry.

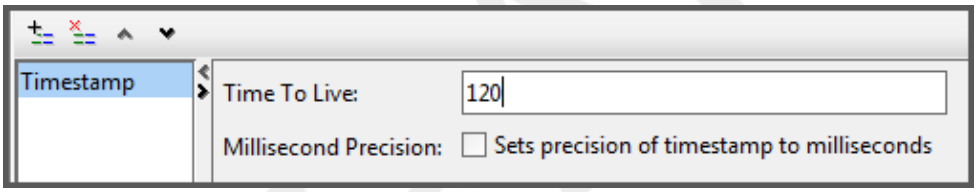
Figure 5. Add WS-Sec Timestamp



a. For the first entry type, select Timestamp.

b. Set Time to Live to a liberal value like `120` (seconds) to avoid any problems with clock skew between the test machine and the service provider. Uncheck the box for Millisecond precision.

Figure 6. Configure WS-Sec Timestamp



5. Click the + again to add another WSS entry element.

6. Select Signature for the entry type.

7. Add the following settings to the signature configuration panel.

   a. For Keystore, select tester.jks. This entry comes from the SOAP-UI key store configuration in the previous section.

   b. For Alias, chose tester1.

   c. For Password, enter the key store password.

   d. For the next five fields choose the options shown in Figure 7.

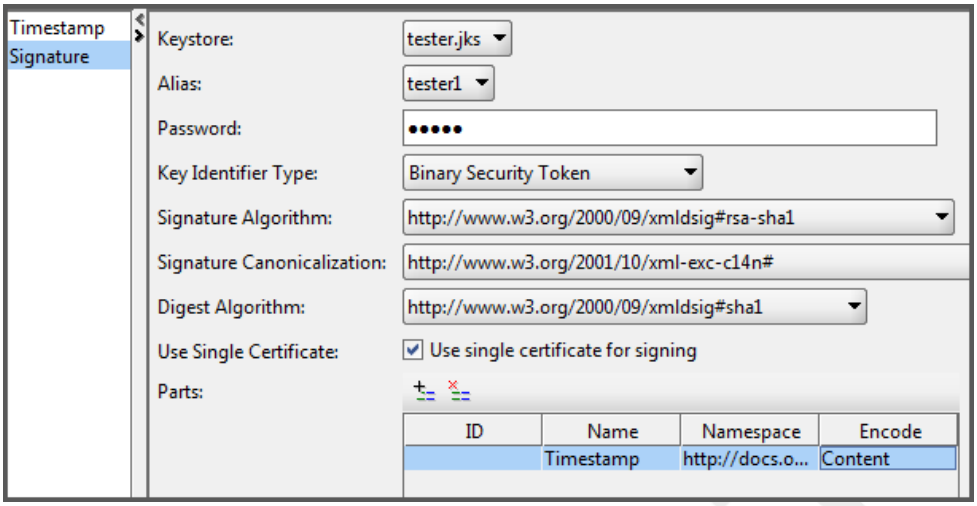
   e. Click the + button in the Parts section. This creates a new row in the table.

      i. In the Name column, enter `Timestamp`.

      ii. In the Namespace column, enter `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd`.

      iii. In the Content column, select Content.

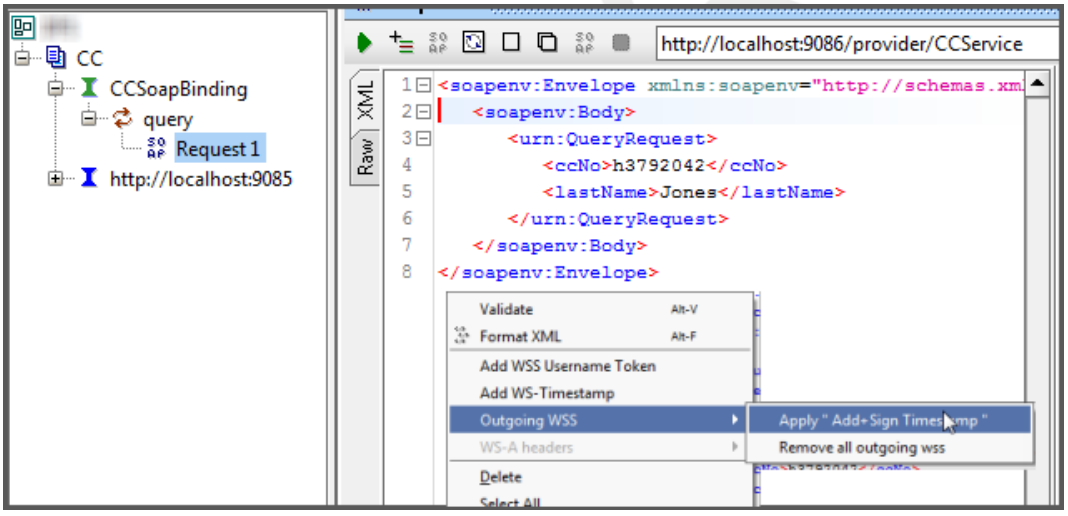
Figure 7. Configure WS-Sec Signature



2.3. Invoke CC Provider with Signature

This is the easy part. Start with any sample request payload (without the WS-Security header). Right-click in the request payload editor and select Outgoing WSS → Apply "Add+Sign Timestamp" as shown in Figure 8.

Figure 8. Add WS-Sec Signature



This will slap the WS-Security header right into the payload.

Caution

Do not alter the WS-Security header after generating the timestamp and signature (such as reformating it). In most cases this will invalidate the signature.

After generating the timestamp and signature, be sure to invoke the CC provider within 120 seconds (before the timestamp expires). After the timestamp expires, simply generate a new WS-Security header using the same menu option shown in Figure 8. It will replace the existing WS-Security header with a new one automatically. There is no

need to manually delete the old one. If you wish to simply remove the old WS-Security header without adding a new one, right-click into the request editor and select Outgoing WSS → Remove all outgoing wss.

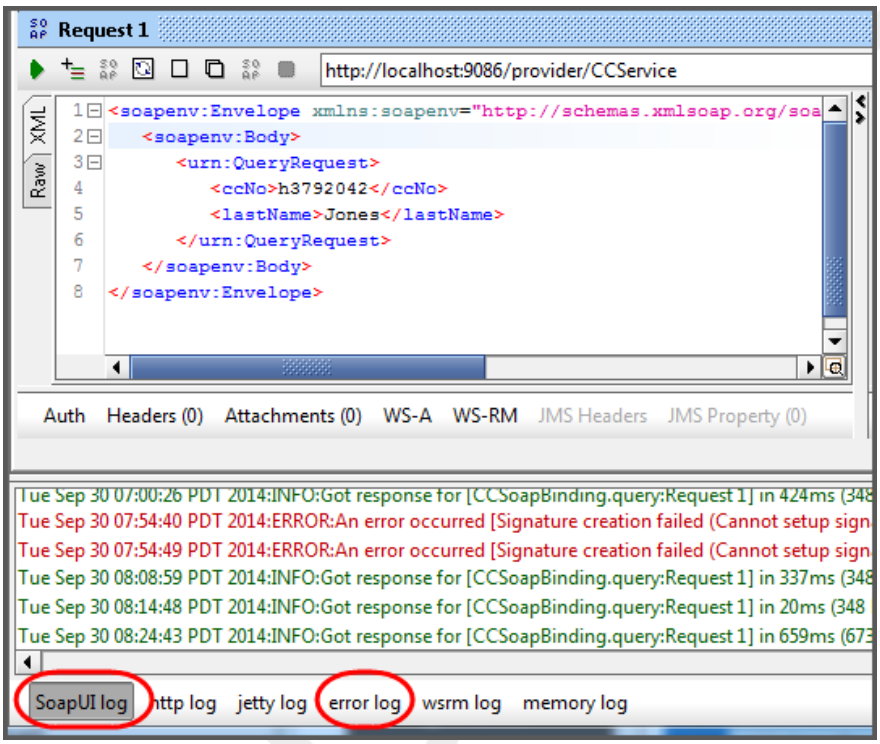
2.4. Troubleshooting

Here are some troubleshooting tips for when things go wrong.

2.4.1. Logs

Check the logs on both sides. For WAS, there is always the `SystemOut.log` file. For SOAP-UI there the SoapUI log and error log tabs at the bottom of the window as shown in Figure 9.

Figure 9. SOAP-UI Logs



2.4.2. Format Request

As mentioned above, formatting the request payload corrupts the signature. But if you've already invoked the request (for better or worse) and you wish to view the security header, go ahead and reformat using by right-clicking in the request panel and selecting Format XML. This will allow you to see the header elements more clearly for troubleshooting.