Bureau of Long Term Services and Supports (BLTSS)

Utah Electronic Visit Verification (UEVV)

Configuring Secure Messaging

Through SOAP UI

SOAP API Testing

Version 1.1

Utah Department of
Health & Human Services
Integrated Healthcare

July 2020

Contact

Please send all inquiries and questions to dmhf_evv@utah.gov

Table of Contents

1. Overview

SOAP connections used by composites are secured against public access through the use of X.509 certificates. SOAP UI, an application that aids in testing SOAP connections, can be configured to utilize X.509 certificates and allow SOAP messages to pass through the EVV security layer.

The following steps outline how to configure SOAP UI for this purpose:

1. Download or update SOAP UI, if necessary
2. Generate the client keystore
3. Export the client certificate and provide it to BLTSS
4. Setup the initial request in SOAP UI
5. Obtain the server certificate via SOAP UI
6. Import the server certificate into the client keystore
7. Configure SOAP UI to use the keystore information
    a. Add client keystore to configuration
    b. Add outgoing security entry
    c. Add basic authentication to request

Note: This is one possible approach out of many. You may find an implementation that better suits your architecture. This is meant only to help establish an initial connection, so that you have a clearer target to aim towards with your organization's particular configuration.

2. Ensure SOAP UI Is Updated Correctly

A free version of SOAP UI can be downloaded from https://www.soapui.org/downloads/soapui.html

For the purposes of this documentation, you want at least version 5.5.0. SOAP UI is a Java-based program, so you will want to ensure that you have Java installed as well. This guide assumes you are using Java 1.8. More recent versions may work as well.

3. Generate The Client-Side Keystore And Certificate

A client-side keystore needs to be generated, if one does not already exist. This is used to hold client-side encryption keys used by SOAP UI to handle the setup of a secure connection to the EVV application. A certificate is produced from this keystore, which is loaded onto the server in order to allow the server to recognize a valid connection attempt.

Note: If you already have a means of creating SSL certificates, you may skip to section 4. Section 3 is referred to throughout this document, however, so review it as needed.

3.1. Creating The Keystore

To create such a keystore, we will use the “keytool” application that comes with Java. This application can be found in the `bin` subfolder of the Java directory.

For example:

```
C:\Program Files\Java\jdk1.8.0_171\bin\keytool.exe
```

Once the keytool application has been identified, open a command prompt at that location, and execute the following command:

```
keytool -genkey -keyalg RSA -alias <<alias>> -keystore <<keystore filename
with path>> -storepass <<password>> -validity 3600
```

Note: Depending on your configuration, you may need to use elevated permissions on your command prompt in order to use the Java keytool program.

Replace the following fields in that command with the correct information:

- `<<alias>>` : a name for the particular key. For example, `soapclient`
- `<<keystore filename with path>>` : a name and location for the keystore itself. Where you want the keystore to be created, and what you want it to be called. For example, `c:\Users\your_username\Desktop\myfirstkeystore.jks`
- `<<password>>` : a password to secure the contents of the keystore.

Upon executing the command, you will be prompted with a series of questions. The table below shows these questions:

Table 1: Keystore identity information

Question
What is your first and last name?
What is the name of your organizational unit?
What is the name of your organization?
What is the name of your City or Locality?
What is the name of your State or Province?
What is the two-letter country code for this unit?
Is the provided information correct?

Finally, you will be asked for a password that corresponds to the chosen alias. You can leave this blank to have it default to the keystore password, instead.

3.2. Export The Certificate

With the client-side keystore created, you must export a certificate from it to load onto the server. Using the same command prompt from section 3.1, enter the following command to export the certificate:

```
keytool -export -rfc -keystore <<keystore filename with path>> -alias
<<alias>> -file <<certificate filename with path>>
```

Replace the following fields in that command with the correct information:

- `<<keystore filename with path>>` : This is the path and filename of the keystore created in section 3.1. Use the same information, unless the keystore was moved or renamed between steps.
- `<<alias>>` : This is the name of the particular key that you want the certificate to be created from. Use the same alias chosen from section 3.1, when the keystore was created.
- `<<certificate filename with path>>` : This is the path and filename of the certificate being created. For example, `c:\Users\your_username\Desktop\soapclient.cer`

Finally, provide this newly created certificate file to BLTSS, so that they can have the certificate imported into the server-side keystore.

Note: If your certificate is not provided to BLTSS for installation, you will not be able to successfully send records to the EVV application!

4. Set Up The Initial Request In SOAP UI

Once your certificate has been installed by BLTSS, you can begin configuring SOAP UI to communicate with the server. The first step is to create the initial request by using the proper WSDL.

Open SOAP UI and create a new SOAP project by clicking the "SOAP" button on the toolbar.

This will bring up a prompt to configure the project. Enter the WSDL URL into the "Initial WSDL" field. The URL for the WSDL is as follows:

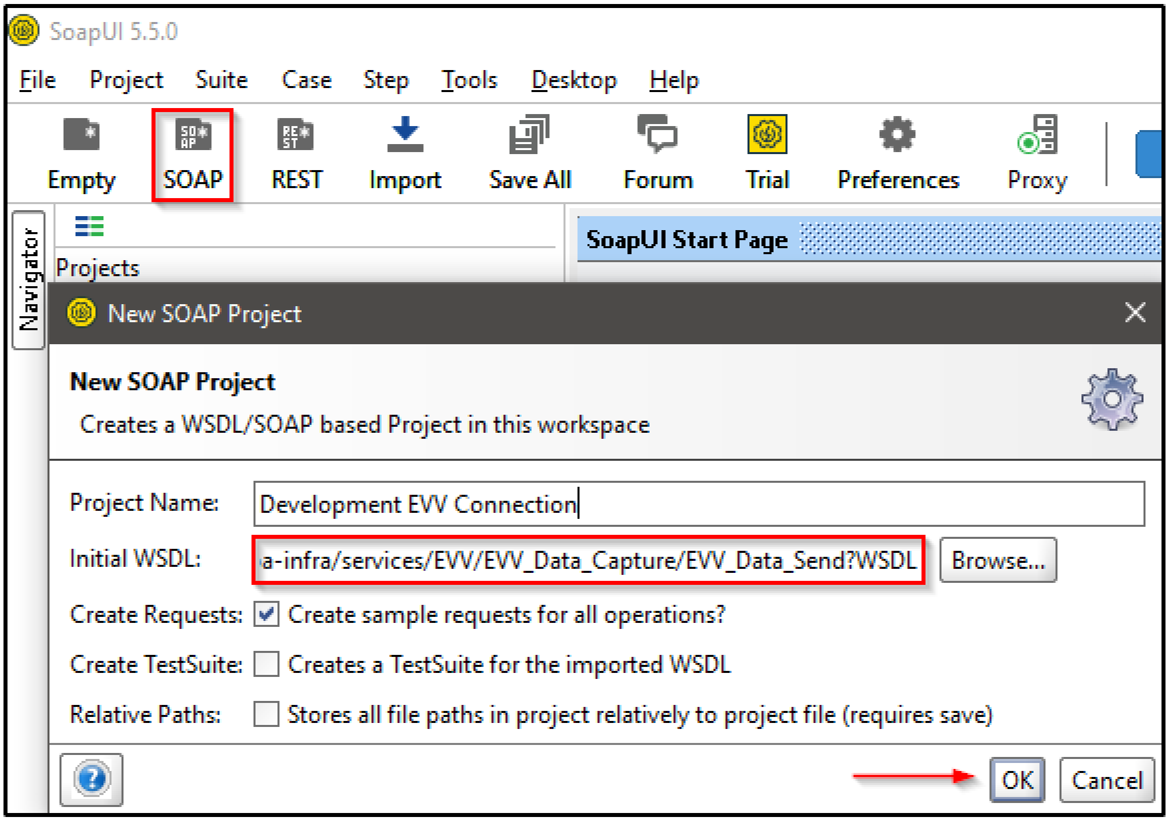
https://evv.medicaid.utah.gov/soa-infra/services/EVV_BATCH/EVV_Data_Capture/EVV_Data_Send?WSDL

This will populate the "Project Name" field with the server name from the WSDL URL.

Note: You can replace the project's name with something more descriptive.

Ensure that the "Create Requests" checkbox is selected, and then press "Ok" to generate the initial SOAP project from the WSDL you provided. See Figure 1 for an example of the project creation dialog.

Figure 1: Creating a new SOAP project



New SOAP Project

New SOAP Project

Creates a WSDL/SOAP based Project in this workspace

Project Name: Development EVV Connection

Initial WSDL: a-infra/services/EVV/EVV_Data_Capture/EVV_Data_Send?WSDL

Create Requests: ☑ Create sample requests for all operations?

Create TestSuite: □ Creates a TestSuite for the imported WSDL

Relative Paths: □ Stores all file paths in project relatively to project file (requires save)

This will produce a new project in SOAP UI.

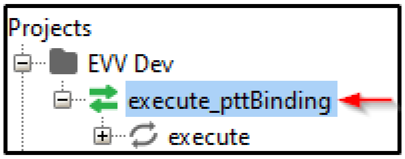
5. Get The Server-Side Certificate

The WSDL used to generate the SOAP UI project (see section 4) contains certificate information about the server-side of the connection we are working to establish.

The WSDL URL can be found inside the SOAP UI project, by viewing the "Overview" tab of the Interface Viewer for the SOAP UI project. This can be brought up with the following steps:

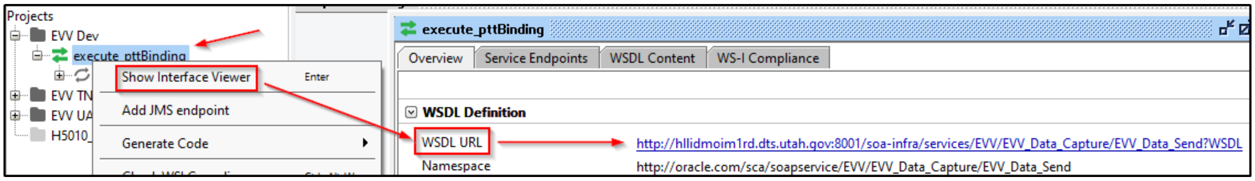
- Expand the project in SOAP UI, and right-click on the Service node.
  - A Service node in SOAP UI is indicated by a pair of vertically-stacked green horizontal arrows pointing in opposite directions.

Figure 2: A Service node in a SOAP UI project



- Select the "Show Interface Viewer" option
- If necessary, select the "Overview" tab in the "Interface Viewer"
- If necessary, expand the WSDL Definition section on the "Overview" tab

Figure 3: The WSDL URL is available in the interface viewer

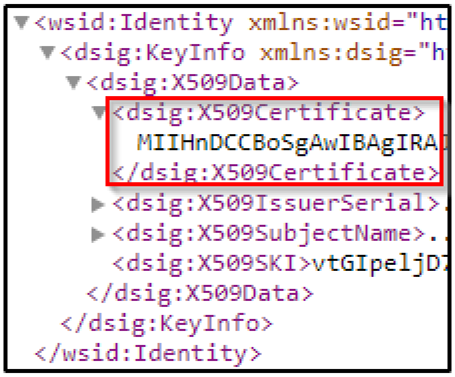


WSDL URL http://hllidmoim1rd.dts.utah.gov:8001/soa-infra/services/EVV/EVV_Data_Capture/EVV_Data_Send?WSDL

Namespace http://oracle.com/sca/soapservice/EVV/EVV_Data_Capture/EVV_Data_Send

The WSDL URL is displayed as a clickable link. Clicking the link opens a page in your default web browser. Near the bottom of the WSDL is a segment titled “dsig:X509Certificate”, which contains a base-64 encoded string. This string is the needed server-side certificate.

Figure 4: The X509 certificate is available in the WSDL



```
▼<wsid:Identity xmlns:wsid="ht
  ▼<dsig:KeyInfo xmlns:dsig="h
    ▼<dsig:X509Data>
      ▼<dsig:X509Certificate>
         MIIHnDCCBoSgAwIBAgIRA
       </dsig:X509Certificate>
      ▶<dsig:X509IssuerSerial>
      ▶<dsig:X509SubjectName>.
       <dsig:X509SKI>vtGIpeljD
     </dsig:X509Data>
   </dsig:KeyInfo>
 </wsid:Identity>
```

Copy the string (only the base-64 encoded text itself, not the surrounding XML tags) into an empty text document. Surround the certificate with the following lines:

```
-----BEGIN CERTIFICATE-----
<<base64 cert from the WSDL>>
-----END CERTIFICATE-----
```

Note: The lines in question are simply “BEGIN CERTIFICATE” or “END CERTIFICATE” in all caps, with 5 leading and trailing hyphens.

Save this file (the server-side certificate). It will be used in section 6 to load the information into the client-side keystore.

6. Add Server-Side Certificate To Client-Side Keystore

The server-side certificate, obtained in section 5, needs to be loaded into the client-side keystore. To do so, open a Command Prompt (or PowerShell terminal) and navigate to the keytool utility within your Java installation. See section 3.1 for details.

Use the following command to import the server-side certificate into the client-side keystore:

```
keytool -importcert -file <<filename and path to server-side certificate>> -
keystore <<filename and path to client-side keystore>> -alias <<server-side
certificate alias>>
```

Replace the following fields in that command with the correct information:

- `<<filename and path to server-side certificate>>` : This is the certificate file that was created in section 5.
- `<<filename and path to client-side keystore>>` : This is the client-side keystore that was created in section 3.1.
- `<<server-side certificate alias>>` : This is the name to use when looking up the server-side certificate once it is loaded into the keystore. It is a name of your own choosing. For example, evvserver.

This process may ask for a password to confirm the changes. Use the password for the keystore itself (if it differs from any alias passwords – see section 3.1 for more information).

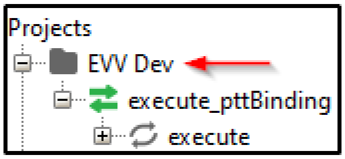
Note: You may have some other means of managing certificates at your disposal. For the purposes of using SOAP UI, as long as you are able to point SOAP UI to the client and server certificates, the particulars of how you store them are up to you.

At this point, both client and server have the information they need to mediate a secure connection. The remaining sections are about configuring SOAP UI to utilize this information.

7. Add The Keystore To The Configuration

To configure a SOAP UI project to use the keystore, right-click the project and select "Show Project View" from the resulting menu.

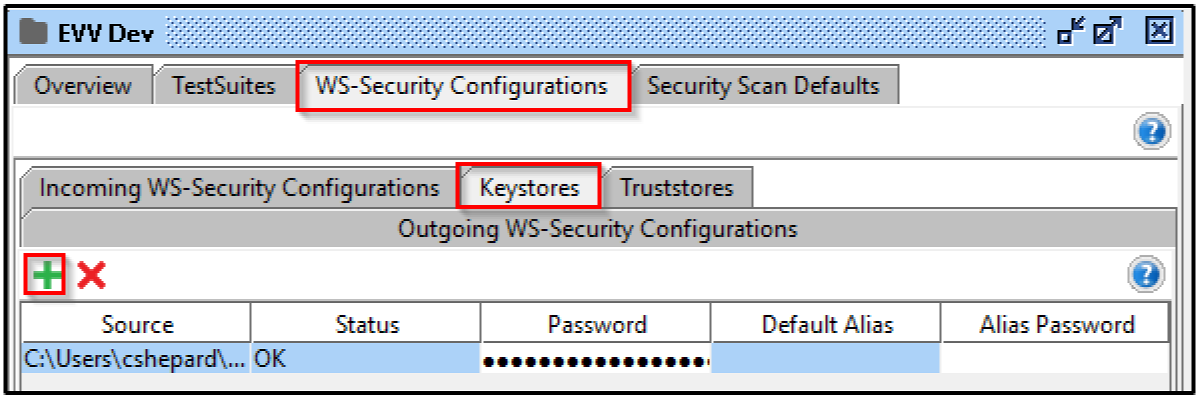
Figure 5: A Project node in a SOAP UI project



With the project view open, select the tab labeled "WS-Security Configurations". From here, select the "Keystores" tab. This will present a list of all known keystores that are currently tied to this particular project.

Click the green "plus" button to attach the client-side keystore that was created in section 3.1.

Figure 6: Attaching a keystore to a SOAP UI project


EVV Dev
Overview
TestSuites
WS-Security Configurations
Security Scan Defaults
Incoming WS-Security Configurations
Keystores
Truststores
Outgoing WS-Security Configurations

Source	Status	Password	Default Alias	Alias Password
C:\Users\cshepard\...	OK	●●●●●●●●●●●●●●●●●		

Select the keystore file when prompted, and then enter the keystore password. This is the password for the keystore itself, as it was created in section 3.1.

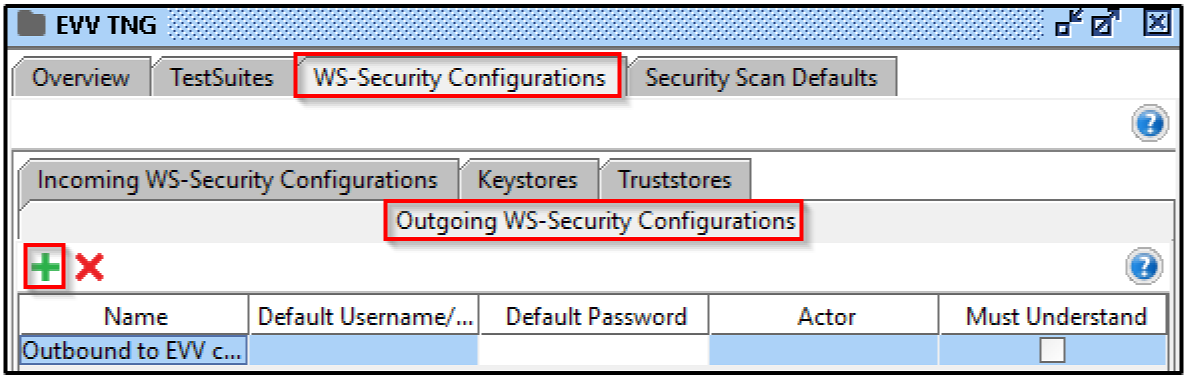
This will add the keystore to the list of keystores for the project, as in Figure 6.

8. Add Outgoing Entry

The SOAP UI project needs to be configured to have outgoing messages encrypted. Without this, the server will reject the message due to misconfigured security.

To configure outgoing messages for a SOAP UI project, navigate to the WS-Security Configurations tab of the Project View (see section 7 for instructions on navigating to this tab), and select the "Outgoing WS-Security Configurations" tab.

Figure 7: Configuring security for outgoing messages


EVV TNG
Overview
TestSuites
WS-Security Configurations
Security Scan Defaults
Incoming WS-Security Configurations
Keystores
Truststores
Outgoing WS-Security Configurations

Name	Default Username/...	Default Password	Actor	Must Understand
Outbound to EVV c...				□

Click the green "plus" button to create a new entry. At the name prompt, enter something that makes it clear which project is being configured.

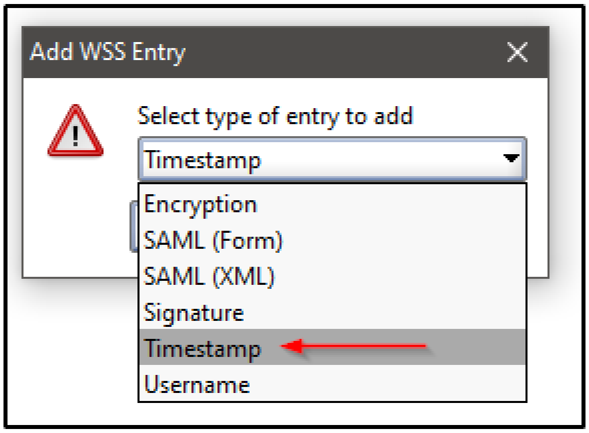
With this newly created entry selected, a new subsection will appear. This is where the specific nature of the outgoing message security will be configured. This consists of setting up the following:

- Timestamp information
- Signature configuration
- Ecryption configuration

8.1. Add Timestamp Information

Select the outgoing security configuration entry that was created in section 8. Click the green "plus" button in the panel below the entry. At the resulting popup, select "Timestamp" from the list.

Figure 8: Selecting "timestamp" when configuring outbound message security


Add WSS Entry

Select type of entry to add

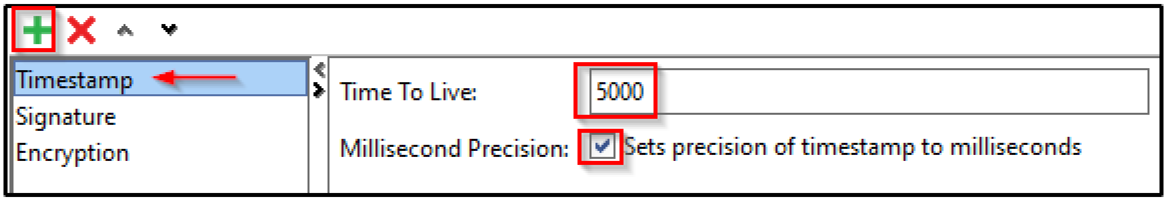
Timestamp

This will add a “Timestamp” entry to this sublist. With this entry selected, a form will be displayed. Update the fields as found in Table 2 below.

Table 2: Timestamp settings

Variable	Value
Time To Live	5000
Millisecond Precision	Checked

Figure 9: Configuring the timestamp settings



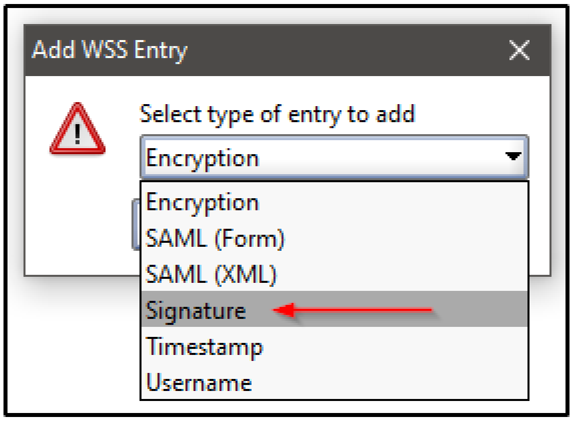
Time To Live: 5000

Millisecond Precision: ☑ Sets precision of timestamp to milliseconds

8.2. Add Signature Information

Select the outgoing security configuration entry that was created in section 8. Click the green “plus” button in the panel below the entry. At the resulting popup, select “Signature” from the list.

Figure 10: Selecting "signature" when configuring outbound message security



This will add a “Signature” entry to the sublist. With this entry selected, a form will be displayed. Update the fields as found in Table 3 below:

Table 3: Signature settings

Variable	Value
Keystore	The keystore with client-side information
Alias	The client-side alias from section 3.1
Password	The password for the chosen alias
Key Identifier Type	Binary Security Token
Signature Algorithm	http://www.w3.org/2000/09/xmldsig#rsa-sha1
Signature Canonicalization	http://www.w3.org/2001/10/xml-exc-c14n#
Digest Algorithm	http://www.w3.org/2000/09/xmldsig#sha1
Use Single Certificate	Checked
Prepend Signature Element	Checked
Custom Key Identifier	(Left blank)
Custom Key Identifier ValueType	(Left blank)

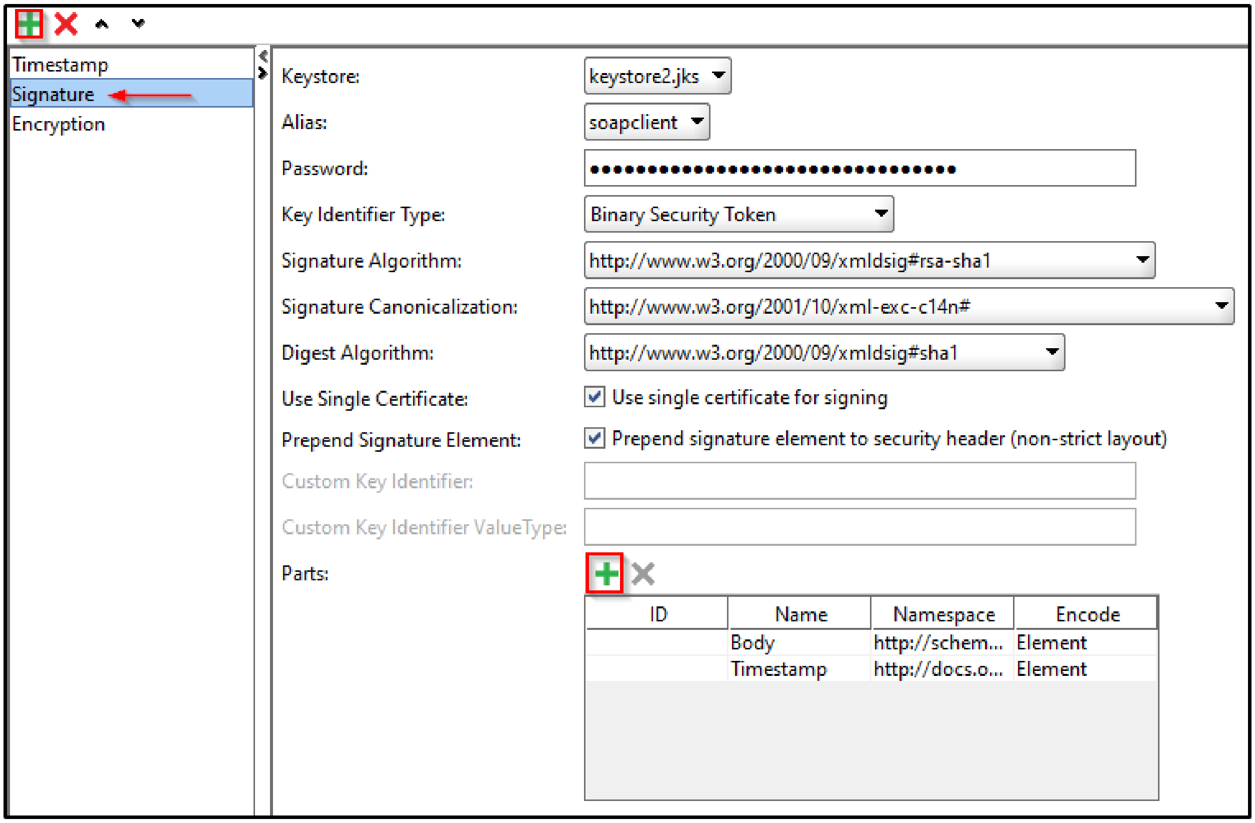
Under the “Parts” subsection, add two rows with the following information:

Table 4: Signature Parts settings

Name	Namespace	Encode
Body	http://schemas.xmlsoap.org/soap/envelope/	Element
Timestamp	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd	Element

Note: Leave the "ID" column of the "Parts" entries blank

Figure 11: Configuring signature settings



Timestamp
Signature
Encryption

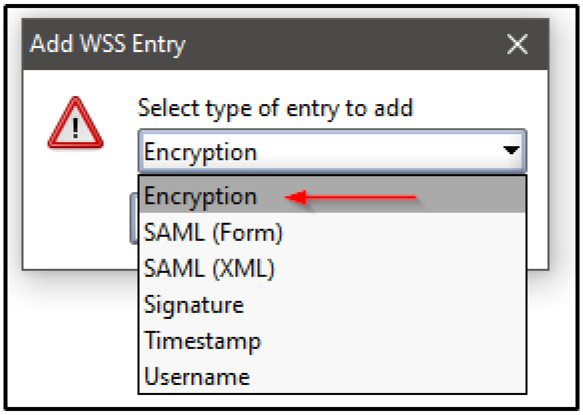
Keystore: keystore2.jks
Alias: soapclient
Password: ••••••••••••••••••••••••••••••••
Key Identifier Type: Binary Security Token
Signature Algorithm: http://www.w3.org/2000/09/xmldsig#rsa-sha1
Signature Canonicalization: http://www.w3.org/2001/10/xml-exc-c14n#
Digest Algorithm: http://www.w3.org/2000/09/xmldsig#sha1
Use Single Certificate: ☑ Use single certificate for signing
Prepend Signature Element: ☑ Prepend signature element to security header (non-strict layout)
Custom Key Identifier:
Custom Key Identifier ValueType:
Parts:

ID	Name	Namespace	Encode
	Body	http://schem...	Element
	Timestamp	http://docs.o...	Element

8.3. Add Encryption Information

Select the outgoing security configuration entry that was created in section 8. Click the green "plus" button in the panel below the entry. At the resulting popup, select "Encryption" from the list.

Figure 12: Selecting "Encryption" when configuring outbound message security



Add WSS Entry

Select type of entry to add

Encryption

Encryption
SAML (Form)
SAML (XML)
Signature
Timestamp
Username

This will add an “Encryption” entry to the sublist. With this entry selected, a form will be displayed. Update the fields as found in Table 5 below:

Table 5: Encryption settings

Variable	Value
Keystore	The keystore with server-side information
Alias	The server-side alias from section 6
Password	The password for the chosen alias
Key Identifier Type	Binary Security Token
Embedded Key Name	(Left blank)
Embedded Key Password	(Left blank)
Symmetric Encoding Algorithm	http://www.w3.org/2001/04/xmlenc#aes128-cbc
Key Encryption Algorithm	http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p
Create Encrypted Key	Checked

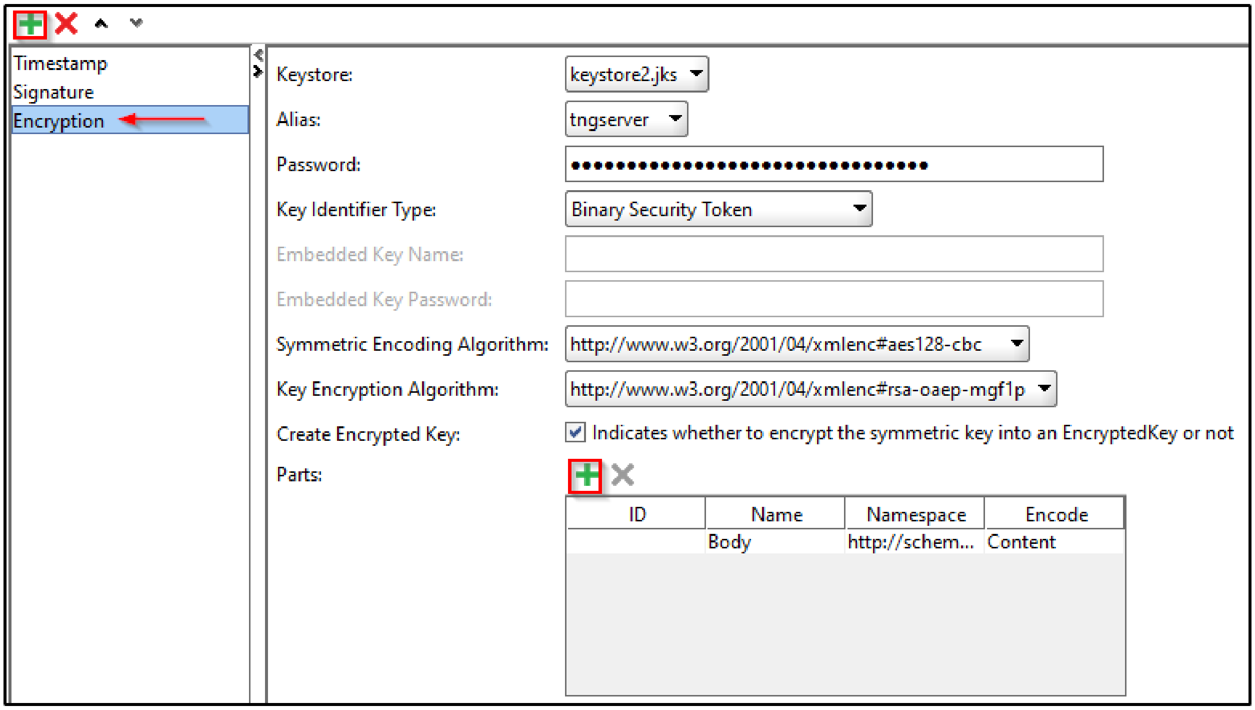
Under the “Parts” subsection, add a row with the following information:

Table 6: Encryption Parts settings

Name	Namespace	Encode
Body	http://schemas.xmlsoap.org/soap/envelope/	Content

Note: Leave the “ID” column of the “Parts” entry blank

Figure 13: Configuring encryption settings



Timestamp
Signature
Encryption

Keystore: keystore2.jks
Alias: tngserver
Password: •••••••••••••••••••••••••••••••
Key Identifier Type: Binary Security Token
Embedded Key Name:
Embedded Key Password:
Symmetric Encoding Algorithm: http://www.w3.org/2001/04/xmlenc#aes128-cbc
Key Encryption Algorithm: http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p
Create Encrypted Key: ☑ Indicates whether to encrypt the symmetric key into an EncryptedKey or not
Parts:

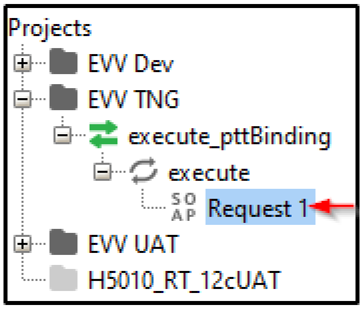
ID	Name	Namespace	Encode
	Body	http://schem...	Content

9. Add Basic Authentication To The Request

The previous sections in this document revolved around configuring project-wide security settings. With all of those set up appropriately, the remaining step is to configure the specific request actions to make use of those settings.

To begin, double-click on a request inside the project that you have been working with. Such a request has a "SOAP" icon, in SOAP UI.

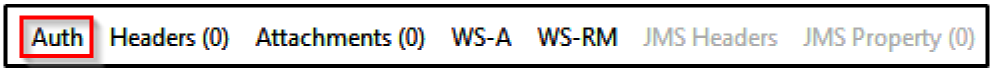
Figure 14: A SOAP UI request node



Projects
EVV Dev
EVV TNG
execute_pttBinding
execute
Request 1
EVV UAT
H5010_RT_12cUAT

Double-clicking a request node will bring up the "Request Editor" for that node. At the bottom of the editor are a series of tabs, including one labeled "Auth". Click on the "Auth" tab.

Figure 15: The "Auth" tab inside the request editor



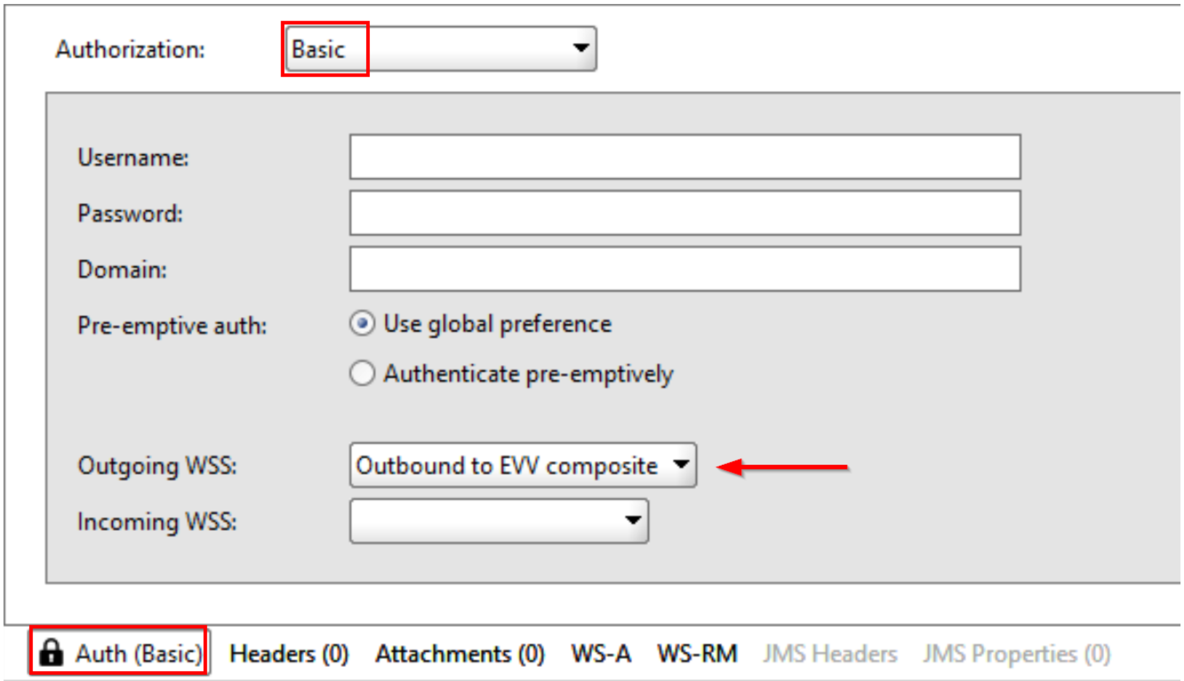
Clicking on the “Auth” tab opens up a new section where the request’s authorization can be configured. Configure this section according to the table below:

Table 7: Request authorization settings

Variable	Value
Authorization	Basic
Username	(Left blank)
Password	(Left blank)
Domain	(Left blank)
Pre-emptive auth	Check “Use global preference”
Outgoing WSS	The outgoing entry created in section 8
Incoming WSS	(Left blank)

Once this section is populated correctly, the “Auth” tab will be updated to indicate the kind of authorization chosen (basic, in this case), and a lock icon will be prepended, indicating that authorization has been configured.

Figure 16: Configuring request authorization



Authorization: Basic

Username:

Password:

Domain:

Pre-emptive auth: ● Use global preference ○ Authenticate pre-emptively

Outgoing WSS: Outbound to EVV composite

Incoming WSS:

At this point, you should be able to send secure messages, and receive secure responses.

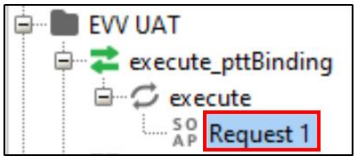
10. Testing The Connection

At this point, you should have the following pieces in order:

- An up-to-date installation of SOAP UI
- A SOAP project based on the EVV WSDL
- Outgoing WSS authentication correctly configured for the SOAP UI project

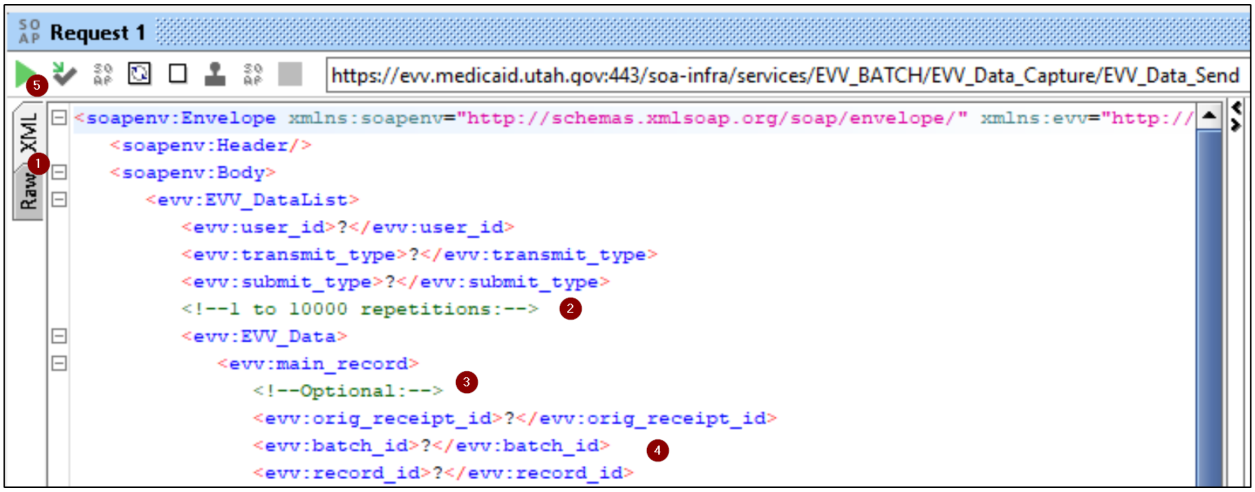
With these items in place, a message can be configured and sent to the EVV application. The SOAP project built from the WSDL has a request with the expected XML syntax defined automatically.

Figure 17: The request is nested several layers deep inside the SOAP project



Double-click the request to open the request editor. You should see an XML template with the available fields presented.

Figure 18: The request view is where the actual message is configured



https://evv.medicaid.utah.gov:443/soa-infra/services/EVV_BATCH/EVV_Data_Capture/EVV_Data_Send

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:evv="http://
   <soapenv:Header/>
   <soapenv:Body>
      <evv:EVV_DataList>
         <evv:user_id>?</evv:user_id>
         <evv:transmit_type>?</evv:transmit_type>
         <evv:submit_type>?</evv:submit_type>
         <!--1 to 10000 repetitions:-->
         <evv:EVV_Data>
            <evv:main_record>
               <!--Optional:-->
               <evv:orig_receipt_id>?</evv:orig_receipt_id>
               <evv:batch_id>?</evv:batch_id>
               <evv:record_id>?</evv:record_id>
```

Take note of the highlighted points:

1. By default, you will see the unencrypted XML message. You can instead choose to see the raw message content, as seen by the server. This raw view will only populate once you have actually sent a message.
2. The EVV_Data element allows up to 10,000 records to be sent in a single batch. We advise, however, to limit batch sizes to 5,000 records or less.
3. Some fields in the message are optional. These elements can be deleted entirely if you do not need them.
4. By default, all fields are populated with question marks (?). These should be replaced with your actual testing data. Failing to do so may result in an error as the question mark is not valid input for several fields.
5. To send your message, click the green “play” button in the upper left corner of the request pane.

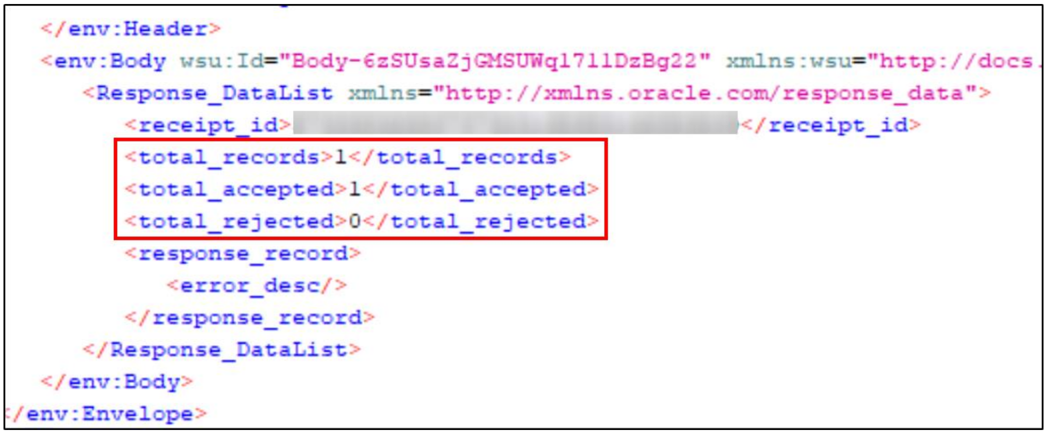
When configuring your XML message for testing purposes, we recommend the following:

- The "user_id" field should represent your organization. If you aggregate records from multiple organizations, the "user_id" is used for the aggregator, while "member_id" and "provider_npi" are used to represent the organizations whose data you send.
- When testing, the "member_id", "first_name", and "last_name" should be set to "Test" in order to distinguish this data from valid data.

Note: The connection that is being created here is to the production EVV system. Failing to properly flag inserted records as "Test" when testing your connection may cause unexpected problems.

Once you have sent your message, the right-side pane of the Request panel will show the response. A successful response will provide a clear indication of how many records were accepted or rejected. Any rejected messages will be accompanied by an error explaining the rejection.

Figure 19: The request response gives a summary of the work done and a receipt ID



```
  </env:Header>
  <env:Body wsu:Id="Body-6zSUsaZjGMSUWq1711DzBg22" xmlns:wsu="http://docs.
     <Response_DataList xmlns="http://xmlns.oracle.com/response_data">
        <receipt_id>[redacted]</receipt_id>
        <total_records>1</total_records>
        <total_accepted>1</total_accepted>
        <total_rejected>0</total_rejected>
        <response_record>
           <error_desc/>
        </response_record>
     </Response_DataList>
  </env:Body>
</env:Envelope>
```

Responses will also include a receipt ID. This value is used when sending corrected records, and must be preserved alongside information about the batch ID and record ID for each record sent.

Note: It is important to preserve the receipt ID, and its association to the batch and record IDs. Without these IDs, you will be unable to correctly update records in the system!

Errors received when attempting to send a message may mean that your WS security settings are misconfigured. Double-check that they are correctly configured, that your client-side and server-side certificates are correctly identified, and that your XML payload matches the WSDL specification.

Finally, SOAP UI is a versatile tool in its own right, complete with comprehensive documentation. If you have further questions about the functionality of SOAP UI itself, refer to its documentation. The URL for SOAP UI's documentation on SOAP messaging is provided below for your convenience:

https://www.soapui.org/soap-and-wsdl/getting-started/