



This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

Composer Help

Web Service Common Block

12/14/2025

Web Service Common Block

Contents

- **1 Web Service Common Block**
 - 1.1 Video Tutorial
 - 1.2 SOAP-Compliant Web Services
 - 1.3 Additional Information
 - 1.4 Web Service Block Security
 - 1.5 Testing the Web Service Block
 - 1.6 Name Property
 - 1.7 Block Notes Property
 - 1.8 Exceptions Property
 - 1.9 HTTPS SSL Authentication\Authentication Type
 - 1.10 HTTPS SSL Authentication\Trust Store Location
 - 1.11 HTTPS SSL Authentication\Trust Store Password
 - 1.12 HTTPS SSL Authentication\certificateStoreName
 - 1.13 HTTPS SSL Authentication\Certificate Alias
 - 1.14 HTTPS SSL Authentication\Certificate or Key Store Location
 - 1.15 HTTPS SSL Authentication\Key Algorithm
 - 1.16 HTTPS SSL Authentication\Key Store Password
 - 1.17 HTTPS SSL Authentication\Private Key Password
 - 1.18 Service URL Property
 - 1.19 Available Services Property
 - 1.20 Bindings Property
 - 1.21 Operations Property
 - 1.22 Service End Point Property
 - 1.23 Service End Point Variable Property
 - 1.24 Use Protocol Property
 - 1.25 Condition Property
 - 1.26 Logging Details Property
 - 1.27 Log Level Property

- [1.28 Enable Status Property](#)
- [1.29 Input Parameters Property](#)
- [1.30 Timeout Property](#)
- [1.31 Custom HTTP Headers Property](#)
- [1.32 Proxy Property](#)
- [1.33 Security\Authentication Type Property](#)
- [1.34 Login Name Property](#)
- [1.35 Password Property](#)
- [1.36 SOAP Digital Signature\Certificate Store Name Property](#)
- [1.37 SOAP Digital Signature\Certificate Alias Property](#)
- [1.38 SOAP Digital Signature\Certificate or Key Store Location Property](#)
- [1.39 SOAP Digital Signature\Key Algorithm Property](#)
- [1.40 SOAP Digital Signature\Key Store Password Property](#)
- [1.41 SOAP Digital Signature\Private Key Property](#)
- [1.42 SOAP Digital Signature\Private Key Password Property](#)
- [1.43 Custom Prefix Property](#)
- [1.44 Add Namespace Prefix Property](#)
- [1.45 Custom SOAP Envelope Property](#)
- [1.46 Output Result Property](#)
- [1.47 Map Output Values to Variables Property](#)
- [1.48 ORS Extensions Property](#)
- [1.49 Example Block Properties](#)
- [1.50 Web Services Description Language \(WSDL\) Support](#)
- [1.51 Errors in WSDL Parsing](#)
- [1.52 Note](#)
- [1.53 Microsoft Web Services Enhancements \(WSE\) Not Installed](#)

The Web Service block allows you to develop an application that supports secure mutual authentication and communication with a Web Service, through the use of both a digital client certificate and server certificate contained in a keystore file.

This functionality is supported for both callflows and workflows. In both cases, Composer generates all the necessary code based on the visual callflow or workflow, so you do not have to write any server-side code to supplement the communication with the external Web Service.

In version 8.1.450.20, the SSL certificate validation code is disabled as Composer does not utilize this code for HTTP requests. If required, this code can be enabled on a demand basis by setting the `web.legacyCertificateCode` option to true in the **composer.properties** file. In previous versions, this code was always enabled.

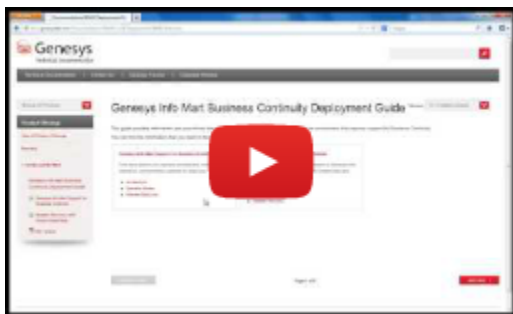
Beginning with version 8.1.560.15, a new validation mode for SSL server certificates is introduced. If enabled, Composer does not add the server certificate to the trust store and instead only performs an expiry date validity check and signature check for the certificate. Users can manually add the required server certificates to the trust store. This new mode is applicable only for Composer Java projects. To enable this mode, the `web.https.productionMode` property must be set to true in the **composer.properties** file. This property is set to false by default.

Video Tutorial

Below is a video tutorial on using the Web Service block.

Tip

While the interface for Composer in this video is from release 8.0.1, the steps are basically the same for subsequent releases.



SOAP-Compliant Web Services

This block can be used to invoke SOAP 1.1 compliant Web Services. It accepts and parses WSDL content for the WebService and collects input parameters based on this WSDL content.

- Uses common Web Services standards such as XML, **SOAP** and WSDL.
- You can pass parameters (as in subdialogs) and store the return values in variables.
- GET and POST methods are supported.
- Supports SOAP 1.1 and therefore requires a WSDL file to describe **endpoints** and services. The Web Service block will not work without this WSDL file.
- WSDL-based Web Services are supported with certain limitations. The WSDL is parsed and you are provided the option to select the service name, bindings type, operations, service end point, and mode (GET / POST). The Input and Output parameter list is pulled by default from the WSDL.

Data returned by the Web Service is converted to JSON format and made available in the application. (See an issue pertaining to JSON objects in **Troubleshooting**.)

Important

SOAP 1.2 is not supported.

Additional Information

For additional information, see:

- **Web Service Block and SignedSOAPRequests** and **Web Service SOAP Message Examples**.
- **WSDL_SOAP_XSD_WSSE_Support**

Web Service Block Security

For Java and .NET Composer projects, the Web Service Block supports secured SOAP communication using XML Digital Signature with a Client Certificate for Java Composer Projects. XML Digital Signature authentication is in compliance with the Second Edition of the **XML Signature Syntax and Processing Specification** and the **OASIS Web Services Security SOAP Messages Security Specification**. The **Authentication Type property** below allows you to select various types of authentication.

Testing the Web Service Block

When working with either a callflow or workflow, the Web Service block provides menu option to test the configured SOAP Web Service using the Web Services Explorer. Right-click the Web Services block and select **Test with Web Services Explorer**.

The Web Service block has the following properties:

Important

Some of the property names in this block are repeated under different categories. Such property names have been prefixed with the category name for readers to be able to easily differentiate the similar names on this page.

Name Property

Find this property's details under [Common Properties for Callflow Blocks](#) or [Common Properties for Workflow Blocks](#).

Block Notes Property

Find this property's details under [Common Properties for Callflow Blocks](#) or [Common Properties for Workflow Blocks](#).

Exceptions Property

Find this property's details under [Common Properties for Callflow Blocks](#) or [Common Properties for Workflow Blocks](#). You can also define [custom events](#). The Web Service block Exceptions dialog box has the following pre-set exceptions:

- Callflows: `error.badfetch` and `error.com.genesyslab.composer.webservice.badFetch`
- Workflows: `error.session.fetch` and `error.com.genesyslab.composer.webservice.badFetch`

Important

Server side exceptions from the Web Service block is optional from version 8.1.450.33. A new flag, `web.throwServerError`, is introduced. On adding the flag to the **composer.properties** file in the respective project and setting it to true, the Web Service block throws an `error.com.genesyslab.composer.servererror` error for failures. If the new flag is set to false, the the Web Service block updates the `errorMsg` entity in the JSON response. If the particular configuration option is not found in the **composer.properties** file, the Web Request or Web Service block updates the `errorMsg` entity in the JSON response (as when the flag is set to false).

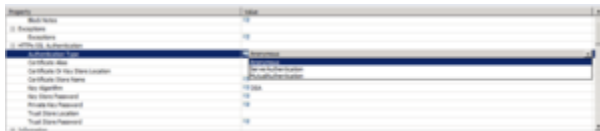
Note: For JAVA projects, the **composer.properties** file is found in the **WEB-INF** folder of the respective project. For .NET projects the **composer.properties** file is found in the **BIN** folder of the respective project.

The **composer.properties** is not created by default by Composer, and users must create one, if required.

HTTPS SSL Authentication\Authentication Type

Use this property to select the type of authentication you want to implement for establishing an HTTPS SSL connection, Mutual Authentication, Server Authentication, or Anonymous.

- Anonymous (selected by default, introduced in 8.1.530.17) - server certificate validation is not performed by the client.
- Server Authentication (introduced in 8.1.450.33) - the client authenticates the server using the server's Public Key Certificate (PKC).
- Mutual Authentication (introduced in 8.1.530.17) - both client and server authenticate each other's identities before actual communication occurs.



Important

If *Mutual Authentication* is selected here, the **certificateStoreName**, **Certificate Alias**, **Certificate or Key Store Location**, **Key Algorithm**, **Key Store Password**, **Private Key Password**, **Trust Store Location**, and **Trust Store Password** properties (all introduced in 8.1.530.17) are mandatory. The values specified in these properties are used during client certificate validation.

Important

Basic HTTP Authentication properties in the Web Service block are validated only during runtime in the server-side pages (ASPX/JSP). For design time, WSDL parsing authentication is not supported. You can copy the WSDL file to the **Include** folder within the required **Composer Project** folder and specify `include/<filename.wsdl>` in the **Service URL** property to parse the WSDL file and configure the block.

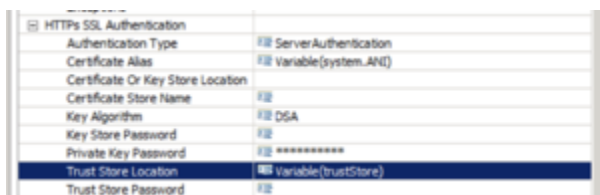
Important

For HTTPS URLs, only the **Server Authentication** and **Mutual Authentication**

types are recommended.

HTTPS SSL Authentication\Trust Store Location

Use this property to specify the path to the certificate store location. The path must point to the keystore file (*.jks) or **cacerts** (default trust store provided by JVM) that contains a list of certificate(s) that the Composer application trusts. The drop-down lists the Entry block variables by default.

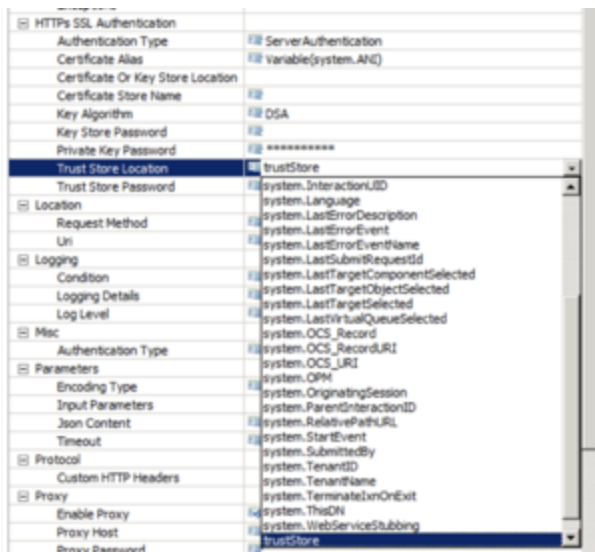


For Java projects, the drop-down is editable and the Java trust store can be customized and can override the default trust store location provided by the Java Virtual Machine.

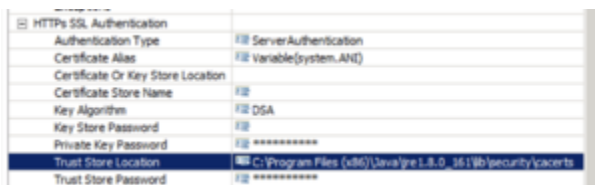
- You can point to the default CA certificate residing at \$JAVA_HOME/lib/security/cacerts, or
- Manually create a CA certificate file of their own (using the **keytool** utility).

Important

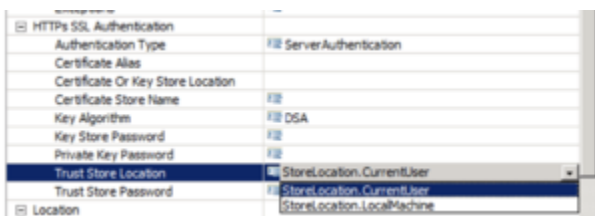
A forward slash denotes directory separation on the Linux OS. However, on Windows, a backward slash denotes directory separation. And since a backward slash is used for escaping characters in Java, you must include two backslashes when specifying the location for a Windows system. For example, C:\\JAVA_HOME\\jre\\lib\\security\\cacerts.



You can either select a variable that has the path or directly specify an absolute path.



For .NET projects, the drop-down is not editable. Windows has its own certificate store (StoreLocation.CurrentUser, StoreLocation.LocalMachine) and you cannot provide a new location or override the default location.



HTTPS SSL Authentication\Trust Store Password

Use this property to specify the password to access the specified trust store.

Important

This property is not applicable for .NET projects as access to the Windows store does not require a password.

HTTPS SSL Authentication\certificateStoreName

Use this property to specify the name of the key store that contains the client certificate entry. The supported key store types are jks and pkcs12.

Sample value for a Java project: `keystore.jks`.

HTTPS SSL Authentication\Certificate Alias

Use this property to provide an alias for the client certificate.

Sample value for a Java project: `keyAlias`.

HTTPS SSL Authentication\Certificate or Key Store Location

Use this property to specify the path to the key store location that contains the client certificate.

Sample value for a Java project: `C:\North\KeyStore_Certificates`.

Important

Windows has its own certificate store (`StoreLocation.CurrentUser` , `StoreLocation.LocalMachine`) and you cannot create or override this default store for .NET projects.

HTTPS SSL Authentication\Key Algorithm

Use this property to select the type of key algorithm (RSA or DSA) to be used during client certificate validation.

HTTPS SSL Authentication\Key Store Password

Use this property to provide the password to access the specified key store.

HTTPS SSL Authentication\Private Key Password

Use this property to specify the password to access the client certificate's private key.

Service URL Property

The Service URL property specifies the WSDL URL of the Web Service to invoke. To set the Service URL:

1. Select the **Service URL** row in the block's property table.
2. In the **Value** field, type a valid URL.

When you provide the WSDL URL in the Service URL property, Composer will try to access the URL and parse it to populate the drop-down lists for the remaining properties:

- **Available Services**
- **Bindings**
- **Operations**
- **Service End Point**
- **Use Protocol**

Note: When *upgrading older diagrams to 8.1.1 and higher, it is necessary to clear out the service URL* and specify it again. This is needed in newer versions to re-parse the WSDL obtained from the specified URL and not use the cached information stored in the diagram.

Available Services Property

When Composer accesses the Service URL, the available Web Services will populate the drop-down list of the Available Services property. To select an available service:

1. Click the **Available Services** row in the block's property table.
2. In the **Value** field, select an available Web Service from the drop-down list.

Bindings Property

When Composer accesses the Service URL, the available bindings will populate the drop-down list of the Bindings property. To select a binding:

1. Click the **Bindings** row in the block's property table.
2. In the **Value** field, select an available bindings setting from the drop-down list.

Operations Property

When Composer accesses the Service URL, the available operations will populate the drop-down list of the Operations property. To select an operation:

1. Click the **Operations** row in the block's property table.
2. In the **Value** field, select the desired operation from the drop-down list.

Service End Point Property

When Composer accesses the Service URL, the service end point options will populate the drop-down list of the Service End Point property. To select a service end point:

1. Click the **Service End Point row in the block's property table.**
2. In the **Value** field, select the service end point from the drop-down list.

Service End Point Variable Property

Use this property to parameterize the Service End Point in the Web Service block. This property will overwrite the above Service End Point property literal value. This enables you to move the workflows across different environments. Moving applications between a test environment and a production environment is same for both .NET and Java Projects. Only deployment procedures might change depending on the Web Server involved. The Web Service Block -> Service End Point Variable property can be used to externalize the Web Service URLs for different environments.

Use Protocol Property

When Composer accesses the Service URL, the protocol options (SOAP and HTTP) will populate the drop-down list of the Use Protocol property. To select a protocol:

1. Click the **Use Protocol** row in the block's property table.
2. In the **Value** field, select SOAP or HTTP from the drop-down list.

Condition Property

Find this property's details under [Common Properties for Callflow Blocks](#) or [Common Properties for Workflow Blocks](#).

Logging Details Property

Find this property's details under [Common Properties for Callflow Blocks](#) or [Common Properties for Workflow Blocks](#).

Log Level Property

Find this property's details under [Common Properties for Callflow Blocks](#) or [Common Properties for Workflow Blocks](#).

Enable Status Property

Find this property's details under [Common Properties for Callflow Blocks](#) or [Common Properties for Workflow Blocks](#).

Input Parameters Property

Note: The Web Service block won't work with IRD if the Web Service parameters are named double since URS considers it a reserved keyword. The same Web Service block will work fine in the voice application. After you have chosen the available service and operations which you want to invoke, along with bindings, service end point, and protocol, use the Input Parameters property to specify a list of required Name/Value pairs to pass as parameters to the Web Service URL. To specify input parameters:

1. Click the **Parameters** row in the block's property table.
2. Click the  button to open the Parameter Settings dialog box.

Add Button

Use the **Add** button to enter parameter details.

1. Click **Add** to add an entry to Web Service Parameters.
2. In the **Parameter Name** field, accept the default name or change it.
3. From the **Parameter Type** drop-down list, select **In**, **Out**, or **InOut**:

In	Input parameters are variables submitted to the web service.
Out	Output parameters are variables that the web service returns and will be reassigned back to the current callflow/workflow.
InOut	InOut parameters are parameters that act as both

	input and output.
--	-------------------

4. In the **Expression** drop-down list, select from among the **variables** shown, type your own expression, or click the button to use **Skill Expression Builder**.
5. In the **Definition** field, type a description for this parameter.
6. Click **Add** again to enter another parameter, or click OK to finish.

Delete Button

To delete a parameter:

1. Select an entry from the list.
2. Click **Delete**.

Timeout Property

Select the variable containing the number of seconds that the application will wait when fetching the result of the Web Service or the Web Request or keep the default of 90 (added in 8.1.440.18). If the requested resource does not respond in that time, then a timeout event will occur.

Custom HTTP Headers Property

Use this property to add Custom headers to be sent along with the HTTP request during the runtime execution of the Server Side block.

1. Click the row in the block's property table.
2. Click the  button to open the Custom HTTP Headers dialog box.
3. Click **Add** to open Configure Custom HTTP Headers dialog box.

Note: The list of headers is a standard list defined by the HTTP protocol. You can optionally specify a list of headers. For each header, the name can be selected from the drop down list or keyed in. The value can be specified as literal values or as variable. There is no special format.

1. Select a **Header type**.
2. Select **Literal** or **Variable**.
3. Type the literal value or select the variable that contains the value.

Proxy Property

You can specify a proxy server to act as an intermediary server when making requests for Web

Services from other servers. The proxy server evaluates the request as a way to simplify and control its complexity. Today, most proxies are web proxies, facilitating access to content on the World Wide Web and providing anonymity. To configure:

- Set **Enable Proxy** to true or false.
- Enter the IP address of the web proxy **Host**.
- Enter the **Password** for the web proxy.
- Enter the web proxy **Port**.
- Enter the web proxy **User Name**.

Security\Authentication Type Property

To assign a value to the Authentication Type property:

1. Select the **Authentication Type** row in the block's property table.
2. In the **Value** field, select from the following:
 - **Anonymous**--With the anonymous type of access, no user name/password is passed to Web service for client authentication in order to get data.
 - **HTTP Basic Authentication**--HTTP Protocol level Basic Authentication using Authorization header. If you select the basic type of access, you must supply the Login Name and Password properties.
 - **SOAP Message Level Basic Authentication**--SOAP Message level Basic Authentication for legacy Web Services using <BasicAuth> header.-- Rarely used but for compatibility.
 - **SOAP XML Signature Authentication**--SOAP Message level XML Digital Signature Authentication using Client Certificate.
 - **SOAP Signature with HTTP Basic Authentication**--SOAP Message Level XML Digital Signature Authentication using Client Certificate + HTTP Basic Authentication (for the Web Server level).

Login Name Property

Used when Security\Authentication Type = HTTP Basic Authentication. The Login Name property specifies the login name for the invoked web page. To provide a login name for the web request:

1. Select the **Login Name** row in the block's property table.
2. In the **Value** field, type a valid login name or select the name from a variable.

Password Property

Used when Security\Authentication Type = HTTP Basic Authentication. The Password property specifies the password for the invoked web page. To provide a password for the web request:

1. Select the **Password** row in the block's property table.
2. In the **Value** field, type a valid password that corresponds to the login name above or select the password from a variable.

SOAP Digital Signature\Certificate Store Name Property

Use this property to specify the name of the Windows Certificate Store. See [WebService Block and Signed SOAP Requests](#). To select a variable:

1. Select the **Certificate Store Name** row in the block's property table.
2. In the **Value** field, select one of the available [variables](#) from the drop-down list.

SOAP Digital Signature\Certificate Alias Property

Use this property to specify the Client Certificate Name. See [Web Service Block and Signed SOAP Requests](#). To select a variable:

1. Select the **Certificate Alias** row in the block's property table.
2. In the **Value** field, select one of the available [variables](#) from the drop-down list.

SOAP Digital Signature\Certificate or Key Store Location Property

Use this property to specify the location of the Certificate Store or Key Store. See [Web Service Block and Signed SOAP Requests](#). To select a variable:

1. Select the **Certificate or Key Store Location** row in the block's property table.
2. In the **Value** field, select one of the available [variables](#) from the drop-down list.

SOAP Digital Signature\Key Algorithm Property

Select DSA (default) or RSA to specify the Key Algorithm to sign the SOAP Digital Signature. See [Web Service Block and Signed SOAP Requests](#). Use this property to specify the Key Store Password. See [Web Service Block and Signed SOAP Requests](#). To select a variable:

1. Select the **Key Algorithm** row in the block's property table.
2. In the **Value** field, select one of the available **variables** from the drop-down list.

SOAP Digital Signature\Key Store Password Property

To select a variable:

1. Select the **Key Store Password** row in the block's property table.
2. In the **Value** field, select one of the available **variables** from the drop-down list. Does not need to match the variable name that is coming back as a result of the web request.

SOAP Digital Signature\Private Key Property

Use this property to specify private key of the Client Certificate. See Web Service Block and Signed SOAP Requests. To select a variable:

1. Select the **Private Key** row in the block's property table.
2. In the **Value** field, select one of the available **variables** from the drop-down list.

SOAP Digital Signature\Private Key Password Property

Use this property to specify the private key password. See Web & Service Block and Signed SOAP Requests. To select a variable:

1. Select the **Private Key Password** row in the block's property table.
2. In the **Value** field, select one of the available **variables** from the drop-down list.

Custom Prefix Property

Use this property to set custom namespace to the generated SOAP message tags. If this property is set it will overwrite the default / WSDL namespace prefix.

Note: To access this property, ensure that the **Show Advanced Properties** option is selected on the toolbar.

Add Namespace Prefix Property

Use this property to add Namespace prefix to the generated SOAP message. By default Composer

Web Service client doesn't generate namespace prefixes.

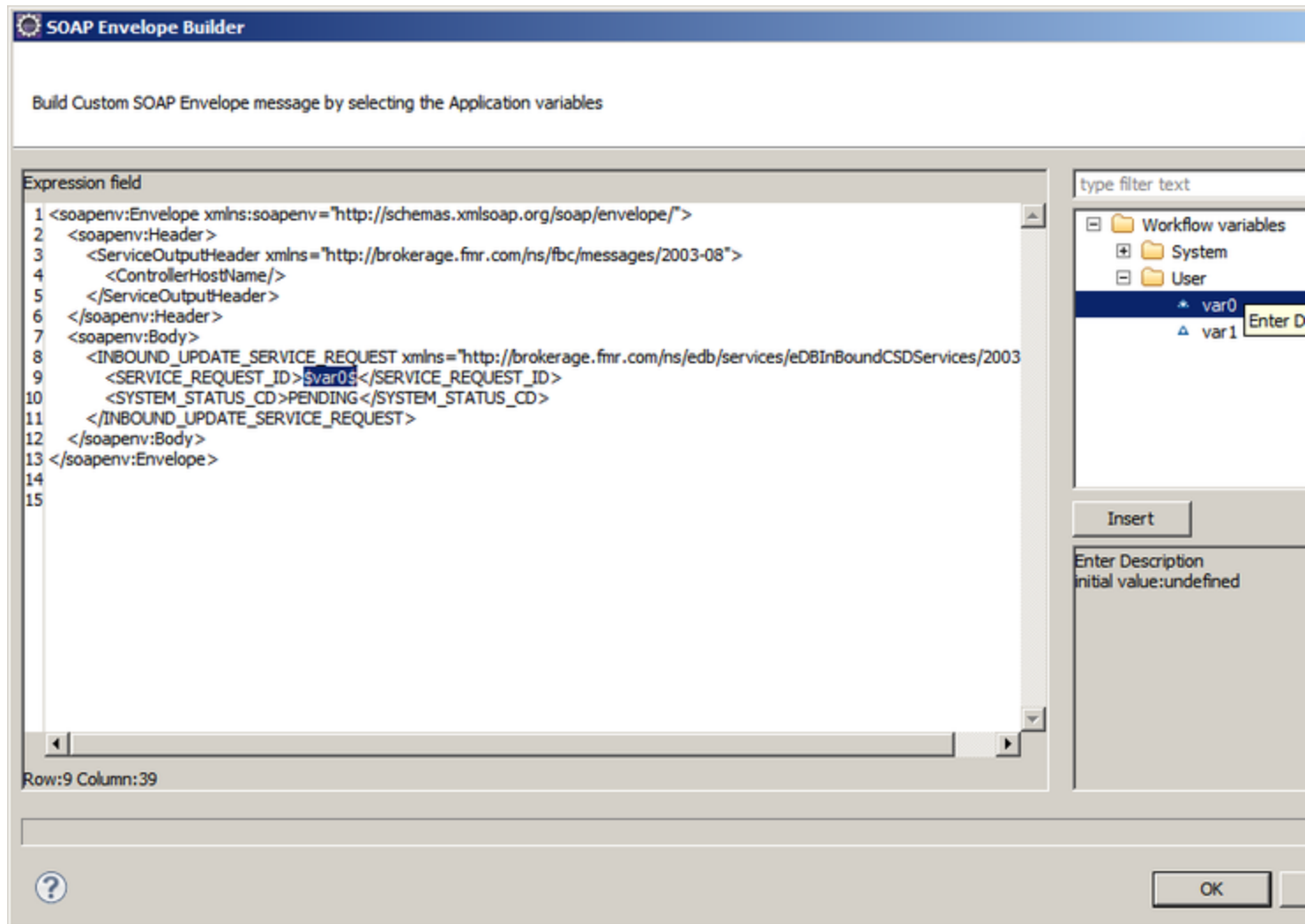
1. None - Do not add any namespace prefix to the SOAP:Body elements.
2. Method Name Tag Only - Add namespace prefix only to the Method Name tag (Operational name tag).
3. Method Name Tag and Child Tags - Add namespace prefix to all the tags in the SOAP message.

Note: To access this property, ensure that the **Show Advanced Properties** option is selected on the toolbar.

Custom SOAP Envelope Property

Use this property to set Custom SOAP Envelope messages. If this property is set, the Composer Web Service run-time client will use this message to get a Web Service response.

1. Click the **Custom SOAP Envelope** property under the **SOAP Message Generation** category. The Custom SOAP Envelope dialog is displayed.
2. Add the custom SOAP Envelope message (the message can be generated using any Web Service client tool).



Custom SOAP Envelope Dialog

The custom message is sent to the Web Service URL at run-time. Diagram application variables can be used to form dynamic contents. Variables can be used in the custom message with a **\$<Variable_Name>\$** notation.

Note:

1. To access this property, ensure that the **Show Advanced Properties** option is selected on the toolbar.
2. This property is supported for both Java Composer Projects and .NET Composer Projects.
3. Callflow-Root document variables and Workflow-Project variables are not supported in this property.

Output Result Property

When the **Map Output Values to Variables** property below is set to true, the Output Result property

maps the Web Service response keys to AppState variables. If Map Output Values to Variables is set to false, the entire Web Service response will be assigned to a variable. The Output Result property is the variable used to get back the invoked Web Service result. To select a variable:

1. Select the **Output Result** row in the block's property table.
2. In the **Value** field, select one of the available variables from the drop-down list. Does not need to match the variable name that is coming back as a result of the web request.

Map Output Values to Variables Property

The Map Output Values to Variables property indicates whether or not to map the Web Service response keys to AppState variables. To select a value for the Map Output Values to Variables property:

1. Select the **Map Output Values to Variables** row in the block's property table.
2. In the **Value** field, select true or false from the drop-down list.

ORS Extensions Property

Starting with 8.1.4, Composer blocks used to build routing applications (with the exception of the Disconnect and EndParallel blocks) add a new **ORS Extensions** property.

Example Block Properties

Example properties for a Web Service block are shown below.

Property	Value
[-] Alias	
Name	getGoldRate
[-] Information	
Service URL	http://www.webservicex.net/CurrencyConvertor.asmx?WSDL
[-] Invoking Configurations	
Available Services	CurrencyConvertor
Bindings	CurrencyConvertorSoap
Operations	ConversionRate
Service End Point	http://www.webservicex.net/CurrencyConvertor.asmx
Use Protocol	SOAP
[-] Parameters	
Input Parameters	Stubbed Parameter(FromCurrency,input), Stubbed Parameter(ToCurrency,input)
[-] Security	
Authentication Type	anonymous
Login Name	
Password	
[-] Web Service Result	
Map Output Values To Variables	true
Output Result	Stubbed Parameter(WebServiceResponseMessage,output)

ExampleWebService.gif

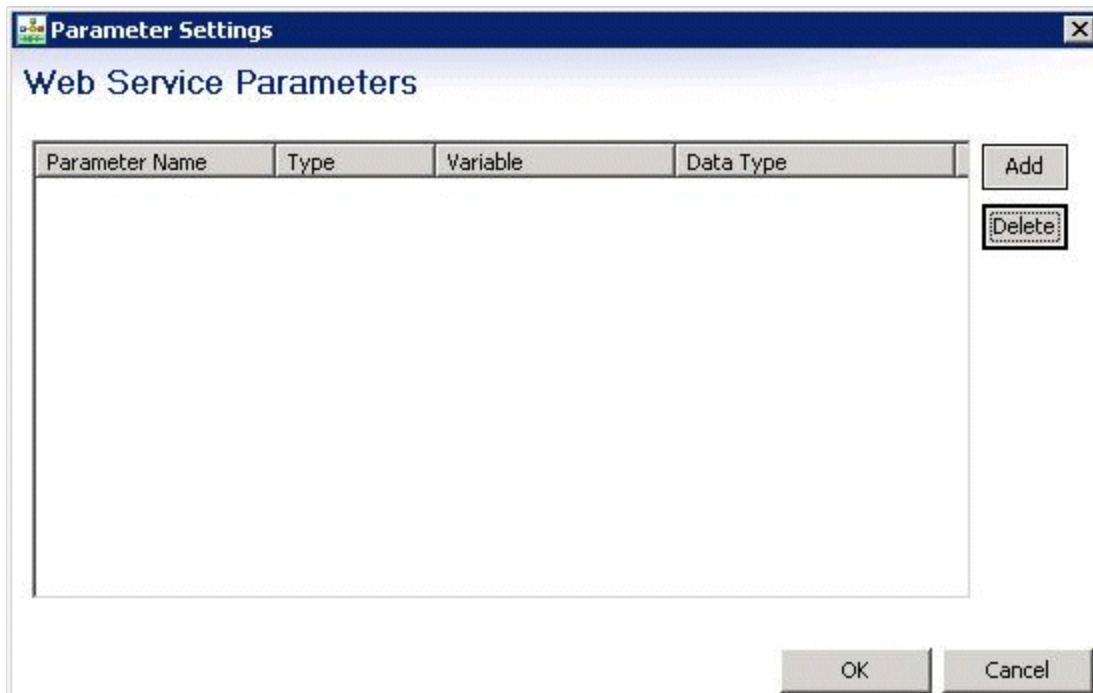
Web Services Description Language (WSDL) Support

Composer supports WSDL definitions conforming to the version WSDL 1.1 schema.

Errors in WSDL Parsing

The following Composer symptom may indicate errors in WSDL parsing:

- If the WSDL definition contains any of the unsupported types and elements, Composer may not be able to parse the WSDL correctly to identify the input and output parameters of the Web Service.
- If the Composer WSDL parser is unable to properly parse the WSDL definition for the Web Service, the input and output parameters fields in the Web Service Parameters dialog box might be empty, with no pre-configured parameters as shown below.



Workarounds

Currently, the following workarounds are available to change the schemas to work with Composer:

- Use qualified elementFormDefault (elementFormDefault="qualified") and define types with fully qualified namespace definitions.
- Define all wsdl types in one schema.
- Replace reference attributes with the actual types being referenced.
- Use the Add/Delete buttons to add or remove any parameters that may not have been automatically displayed. The SOAP request that will be generated at application runtime will take these changes into account.

Note

Composer Web Service block-generated SOAP messages do not have prefix in the SOAP elements. Web Services created using Metro / JAX-WS framework may return Null Pointer Exception or Unexpected Result due to the prefix limitation. Updating the JAX-WS API's / GlassFish server / Metro WS Framework to latest versions may help.

Microsoft Web Services Enhancements (WSE) Not Installed

See [.NET Troubleshooting](#) for steps on working with Composer .NET Projects when a machine does

not have WSE 3.0: