



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 Request Common Block

Web Request Common Block

Contents

- **1 Web Request Common Block**
 - 1.1 Name Property
 - 1.2 Block Notes Property
 - 1.3 Exceptions Property
 - 1.4 HTTPS SSL Authentication\Authentication Type
 - 1.5 HTTPS SSL Authentication\Trust Store Location
 - 1.6 HTTPS SSL Authentication\Trust Store Password
 - 1.7 HTTPS SSL Authentication\certificateStoreName
 - 1.8 HTTPS SSL Authentication\Certificate Alias
 - 1.9 HTTPS SSL Authentication\Certificate or Key Store Location
 - 1.10 HTTPS SSL Authentication\Key Algorithm
 - 1.11 HTTPS SSL Authentication\Key Store Password
 - 1.12 HTTPS SSL Authentication\Private Key Password
 - 1.13 Request Method Property
 - 1.14 Uri Property
 - 1.15 Condition Property
 - 1.16 Logging Details Property
 - 1.17 Log Level Property
 - 1.18 Misc\Authentication Type Property
 - 1.19 ORS Extensions Property
 - 1.20 Encoding Type Property
 - 1.21 Input Parameters Property
 - 1.22 JSON Content Property
 - 1.23 Timeout Property
 - 1.24 Custom HTTP Headers Property
 - 1.25 Proxy Property
 - 1.26 Login Name Property
 - 1.27 Password Property

- [1.28 Enable Status Property](#)
- [1.29 Verify JSON Response](#)
- [1.30 Result Property](#)

The Web Request block is used for both routing and voice applications. Use to invoke any supported HTTP web request or REST-style web Service.

- It supports PUT, DELETE, GET and POST methods.
- It is based on common Web Services standards such as XML, SOAP and WSDL instead of proprietary standards that are currently being replaced.

Representational State Transfer (REST) is an XML-based protocol for invoking Web Services over HTTP. REST is a lighter version of SOAP, which has evolved into a more complex protocol. REST-style web services offer a less coupled paradigm whereby simpler requests and responses are used. As an example, a simple HTTP request follows the REST methodology. The Web Request block allows the user to query "RESTful" Web services.

Important

Starting with version 8.1.450.33, Composer supports fetching HTTPS (HTTP over SSL) URLs in the Web Request and Web Service blocks.

The supported return formats for the Web Request block are:

- plain text. For workflows, the result will be returned in a JSON string with the key name result, e.g., {"result": this is a plain text result}.
- plain XML.
- JSON string. See an issue pertaining to JSON objects in [Troubleshooting](#).

Important

When converting from XML to JSON, any leading zeroes in an attribute's value are truncated. If there are leading zeroes in a number, the number is considered as an Octal value and must be enclosed in quotes to be interpreted as a string.

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.

The Web Request 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 Workflow Blocks](#) or [Common Properties for Callflow Blocks](#)

Block Notes Property

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

Exceptions Property

Find this property's details under [Common Properties for Workflow Blocks](#) or [Common Properties for Callflow Blocks](#). You can also define [custom events](#).

Important

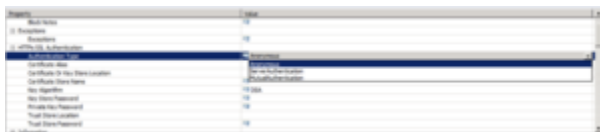
Server side exceptions from the Web Request block is optional from version 8.1.450.33. A new flag, `web.throwServerError`, is introduced. On adding the flag to the **composer.properties** file in the respective project and setting it to true, the Web Request block throws an `error.com.genesyslab.composer.servererror` for failures. If the new flag is set to false, the the Web Request block updates the `errorMsg` entity in the JSON response.

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

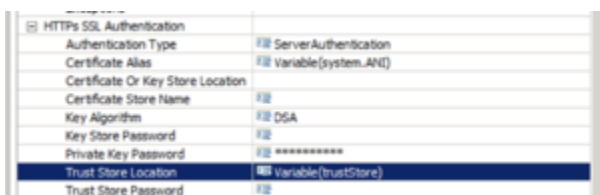
If *Mutual Authentication* is selected here, the **certificateStoreName**, **Certificate Alias**, **Certificate or Key Store Location**, **Key Algorithm**, **Key Store Password**, **Private Key Password**, **Trust Store Location**, and **Trust Store Password** properties (all introduced in 8.1.530.17)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

Important

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.

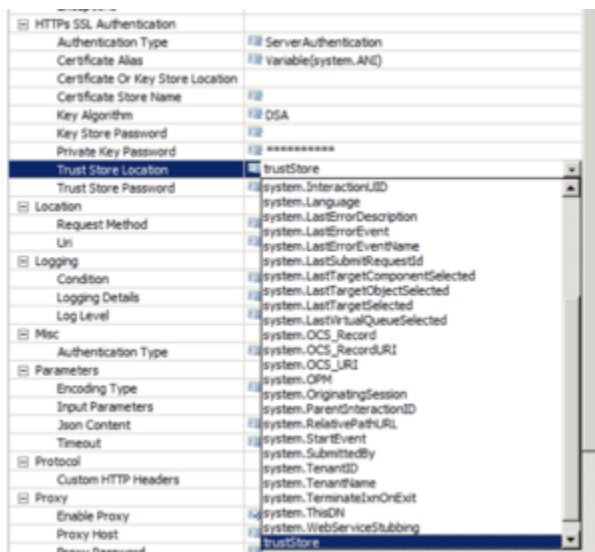


For Java projects, the drop-down is editable and the Java trust store can be customized and can override the default trust store location provided by the Java Virtual Machine.

- 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

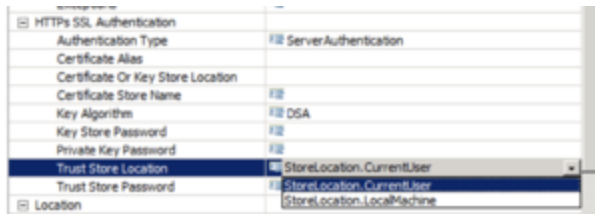
A forward slash denotes directory separation on the Linux OS. However, on Windows, a backward slash denotes directory separation. And since a backward slash is used for escaping characters in Java, you must include two backslashes when specifying the location for a Windows system. For example, `C:\\JAVA_HOME\\jre\\lib\\security\\cacerts`.



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.

Request Method Property

This property Indicates the method for invoking the web request:

- **get**--Invoked using HTTP Get.
- **post**--Invoked using HTTP Post.
- **put**--Invoked using HTTP Put.
- **delete**--Invoked using HTTP Delete.

To select a value for the Request Method property:

1. Select the **Request Method** row in the block's property table.
2. In the **Value** field, select get, post, put, or delete from the drop-down list.

Uri Property

The Uri property specifies the http:// page to invoke. To set a URL destination for the Uri property:

1. Select the **Uri** row in the block's property table.
2. In the **Value** field, click the down arrow and select the variable that contains URL.

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](#).

Misc\Authentication Type Property

The Authentication Type property specifies whether to use an anonymous or basic authentication for the web request. 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 anonymous (default) or basic from the drop-down list. With the anonymous type of access, no user name/password is passed to Web service for client authentication in order to get data. If you select the basic type of access, you must supply the Login Name and Password properties.

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.

Encoding Type Property

The Encoding Type property (used for callflows only) indicates the media encoding type of the submitted document. GVP 8.1 supports two encoding types:

- application/x-www-form-urlencoded
- multipart/form-data

To select a value for the Encoding Type property:

1. Select the **Encoding Type** row in the block's property table.
2. In the **Value** field, select one of the following:
 - **application/x-www-form-urlencoded (default)**
 - **application/json**

Input Parameters Property

Use the Input Parameters property to specify a list of required Name/Value pairs to pass as parameters to the http:// page. 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 Request 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 request.
Out	Output parameters are variables that the web request returns and will be reassigned back to the current callflow/workflow.
InOut	InOut parameters are parameters that act as both input and output.

1. In the Expression drop-down list, select from among the variables shown, type your own expression, or click the  button to use **Expression Builder**.
2. In the Definition field, type a description for this parameter.
3. 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**.

JSON Content Property

If the HTTP request to be invoked expects JSON content, this property can be used to specify that input. It expects a variable whose content will be sent to the API specified in the HTTP URI property of the block. Set the Encoding Type property of the block to `application/json`. In this case, the Input Parameters property will not be used.

The variable selected in this property should contain a JavaScript object. The object can be built from a JSON string, or using the ECMAScript block.

For example, if you would like to pass a JSON content to the HTTP URI, using a variable named "content", the variable can be initialized in the following ways:

- If you have a JSON string, you can use the Assign block to assign the following value to "content":

```
JSON.parse('{ "abc": "def", "xyz": 3 }')
```

- Alternately, you can build a JavaScript object using an ECMAScript block with code like the following:

```
var content = new Object(); content['abc'] = 'def'; content['xyz'] = 3;
```

In both cases, set the JSON Content property of the Web Request block to the variable named "content".

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 starting with 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 Configuration Custom HTTP Headers dialog box.

4. Select a Header type.
5. Select **Literal** or **Variable**.
6. 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**.

Login Name Property

Used when Misc\Authentication Type = basic. 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.

Password Property

Used when Misc\Authentication Type = basic. 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, starting with 8.1.440.18, you can select the password from a variable.

Enable Status Property

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

Verify JSON Response

Composer 8.1.400.35 adds this property. When set to false, the Web Request block will not parse the JSON response and will return it as-is. The default is true. Add this property to the Properties view by clicking the **Show Advanced Properties** button.



Result Property

The Result property is the variable used to get back a result from the web request. To select a variable:

1. Select the **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.