EDI Client

This topic covers the EDI Client tool, which sends messages in EDI format. It can be used in concert with the Enhanced Call Back and XML Tools.

In addition to EDI, the EDI Client and Enhanced Call Back tools support CSV, Fixed Length, Lines, and Plain Text. In these cases, the tool name changes to reflect the appropriate format (i.e., CSV Client and CSV Call Back, Fixed Length Client and Fixed Length Call Back, Lines Client and Lines Call Back, and Plain Text Client and Plain Text Call Back).

The XML Converter tool supports these formats as well. It changes names depending on the format and direction—for instance, to Convert Fixed Length to XML, Convert XML to Fixed Length, Convert EDI to XML, Convert XML to EDI.

Sections include:

Message Packs License Required

The EDI format requires the Message Packs license feature to be enabled. The Message Packs license feature enables the EDI format in the EDI Client, Enhanced Call Back Took, and XML Converter.

Basic EDIFACT Support

The EDIFACT support that was available in SOAtest 9.1 is still available as the "Basic EDIFACT" format in EDI Client, Enhanced Call Back, and XML Converter. See Basic EDIFACT Support for details.

Using EDI with SOAtest

Business and organizations can exchange messages using EDI (Electronic Data Interchange), which uses several standardized message formats. One such format is EDIFACT (Electronic Data Interchange For Administration, Commerce and Transport).

Most EDI message formats, including EDIFACT, are difficult to parse and difficult for humans to read.

The EDI Client, Enhanced Call Back and XML Converter tools simplify the use of EDIFACT in SOAtest. By seamlessly converting from XML to native formats and from native formats to XML, SOAtest allows you to easily leverage SOAtest's existing XML functionality for sending, verifying, and extracting data from EDI messages.

The EDI format support covers the following standards:

- AL3 - Also known as ACORD AL3
- CARGO - Also known as IATA Cargo-IMP
- EANCOM
- EDIFACT
- EDIFACT Basic - Described in Basic EDIFACT Support
- EDIGAS - Also known as Edig@s
- HIPAA
- HL7
- IATA - Also known as IATA PADIS
- NCPDP - Also known as NCPDP SCRIPT
- TELCO - Also known as NCPDP Telecommunication
- TRADACOMS
- X12

For details on what versions and message types within each standard are supported, see EDI Support Details.

Using EDI Client

EDI Client is designed for modelling EDI messages in a form view that allows you to easily change the values of specific fields within the message, or that allows you to parameterize fields within the message.

EDI Client offers a superset of the Message Client tool's functionality. It can do anything that the Messaging Client can—and it can also automatically convert the request payload from XML to EDI (and the response payload from EDI to XML).

It simplifies the use of EDI, which is often difficult to read, by allowing you to utilize XML. You can model your EDI request payload as an XML document; the client then automatically converts the XML to EDI before sending the message. If the server returns an EDI message, the client can convert the message to XML so that you can attach tools such as the XML Assertor and XML Data Bank to the response.

If you simply want to send a literal EDI message, you might want to use the Messaging Client instead of this tool. However, if you are expecting an EDI message in return, the EDI Client has the following advantage over a Messaging Client: the Response Payload modeled as XML output. This output will allow you to chain an XML Assertor, XML Data Bank, or Diff Tool in XML mode to it so you can validate the message easily. With a Messaging Client, you would need one extra step: add a Text/XML Transformer to the Response Traffic output, and then add the XML Assertor, XML Data Bank, or Diff Tool.

Workflows

There are two expected workflows that can be used to configure the EDI Client...

Existing Literal EDI Message

(Recommended) If you already have a literal EDI message:

1. Create a new EDI Client.
2. Switch to Literal mode and paste in your EDI message.
3. Switch to Form Input or Form XML mode, which will then get populated based on the message you entered. You can now configure and parameterize your message as you wish. See Specifying the Request for details.
4. Set the appropriate transport settings. See Specifying Transport Settings for details.
5. Add appropriate validation tools to the outputs. See Validating EDI Messages for details.
6. Run the test.

Creating an EDI Message From Scratch

If you want to create an EDI message from scratch:

1. Create a new EDI Client.
2. Select a message type using the **Dialect**, **Version**, and **Message** type fields, which will populate the Form Input view.
3. Manually specify the message using Form Input. See Specifying the Request for details.

Form Input will constrain your message to the schema for the selected message type, and will not allow you to add fields to the message that are not part of the message. It will also warn you when you are attempting to use value types that are not supported in a given field (like trying to put a string into an integer field). Form XML does not perform either of these checks.

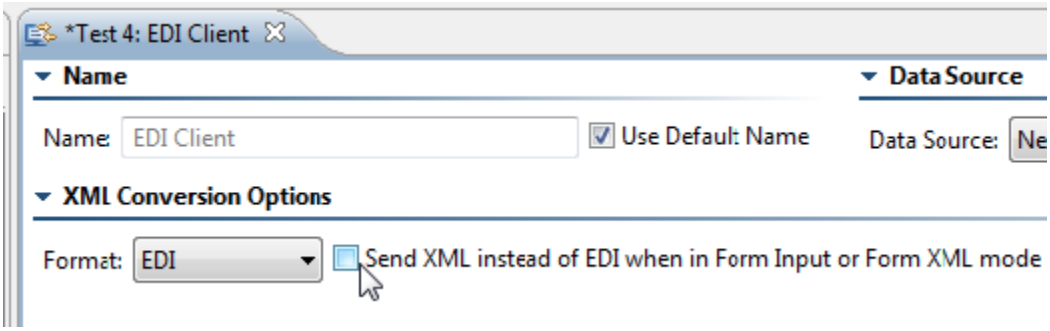
Note that you will likely need to manually add fields that your message uses because SOAtest generates a minimal valid EDI message based on the message type you selected. The message will include default values, which you might want to modify. Using the right-click **Populate** option (described in Populating and Parameterizing Elements with Data Source Values) in Form Input is generally discouraged. It will add all optional fields to the message, but it will not necessarily add valid default values. After using **Populate**, you would need to manually review and enter the value of each field. For most EDI messages, this can be quite tedious.

Tool Configuration Details

Sending in Native or XML Format

Using default options, the EDI Client always sends the native format that is selected in the **Format** combo box. However, in Form Input and Form XML modes, the EDI Client can be configured to send SOAtest's XML format rather than the native format.

To do this, check the box **Send XML instead of EDI when in Form Input or Form XML mode** in the **XML Conversion Options** section of the EDI Client editor.



- **If disabled (default):** SOAtest will automatically convert the XML model into the native format before sending the message. In other words, SOAtest sends the native format to the server—so that is what you will see in the Traffic Viewer.
- **If enabled:** SOAtest will send the message you configured in the UI for the request. You can create the message payload in the native format by using the Literal or Scripted views. If you have this option disabled and you create XML in the UI (e.g., using the Form Input view), then SOAtest will send that XML unmodified. You typically would not want to do this for the formats currently supported, but SOAtest makes this option available.

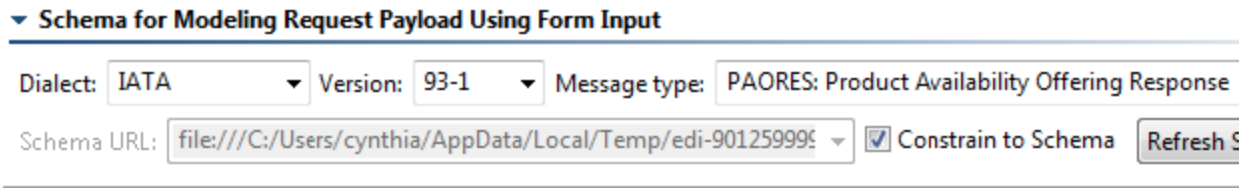
Note that this option has no relevance to Literal or Scripted views—the client will always send the exact text that appears in Literal view or that is returned by a script in Scripted view.

Specifying the Request

For recommended workflows for specifying the request, see Workflows.

Note that when you switch between Form Input or Form XML and Literal, the content in Form Input or Form XML is automatically converted to the native format, which is shown in Literal view. Content in Literal view is automatically converted to an XML format that is then shown in Form Input or Form XML.

In EDI format mode, if you paste in an EDI message into Literal view and then switch to Form Input or Form XML, SOAtest will automatically detect the message type and automatically populate the **Dialect**, **Version**, and **Message Type** combo boxes.



For details on the available views, see Message Tool and Responder Options.

Sending custom EDI messages

If you want to send a custom EDI message, Form XML view will let you add custom fields that Form Input will not allow you to enter.

What if the view is not populated after I switch views?

If you paste a message into Literal view, and switch to Form Input but Form Input does not get populated, then switch from Literal view to Form XML instead. Form Input will not be populated in the cases where SOAtest is not able to find a matching schema for the EDI message that was pasted in. However, with Form XML, SOAtest can often still support those messages.

Specifying Transport Settings

See Testing Through Different Protocols for details on how to configure settings for the available transports.

Note that when the transport is HTTP, the Content-Type HTTP header gets set to one of the following:

- **Form Input and Form XML and Scripted modes -** The default mimetype for the format type. If you would like to send a different mimetype than the default, then you need to override the mimetype by adding a new Content-Type header in the HTTP Headers section of the **Transport** tab.
- **Literal mode -** The mimetype set in the MIME Type field in Literal view.

The default mimetype for each format is:

- CSV - text/csv
- EDI - application/EDI-X12
- Basic EDIFACT - application/EDIFACT
- Fixed Length - text/plain
- Lines - text/plain
- Plain Text - text/plain

Configuring Miscellaneous Options

Click the **Misc** tab to configure the following options:

- **Valid HTTP Response Codes:** Allows you to customize the tool behavior so that it succeeds with HTTP response codes outside the `2xx` range. Specify single codes and/or code ranges as a comma-separated list. For example, if you use `"302, 500-599"`, a `302` code or any code in the `5xx` range will be accepted. If you're using a parameterized value, be sure that the value in the data source uses this same format (e.g., `"302, 500-599"`).
- **Timeout after (seconds):** Specifies the length of delay (in seconds) after which SOAtest should consider your FTP, telnet, or HTTP requests to be timed out. The Default setting corresponds to the timeout set in the Preferences panel. The Custom setting allows you to enter a timeout. A non-positive timeout value can be entered to specify an infinite timeout.
    - **Fail the test on timeout:** Select this option to fail the tool on the specified timeout.
    - **Pass the test only if a timeout occurred:** Select this option to pass the tool if the specified timeout occurred (i.e. tool did not finish execution within the specified time).
- **Outgoing Message Encoding**: Choose **Custom** from the drop-down menu and choose an encoding for the outgoing message. The default is to use the encoding configured in the immediate parent test suite (see Specifying Client Options). You can also specify this option globally in the Parasoft Preferences Misc settings (see Additional Preference Settings).

Specifying Conversion Options

In EDI format, when an option is blank, the conversion will use the default option for that option. Otherwise, if a value is selected or typed in, the conversion will attempt to use that value. Note that if an invalid value is entered manually, you may get an error when trying to switch between Form Input /Form XML and Literal views, or when running the test.

For EDI, available conversion options are:

- **Treat all segments and elements as optional -** If set to yes, all mandatory segments and mandatory data elements are treated as optional. If set to no (the default), a missing mandatory segment or mandatory data element triggers an error. Enabling this option can be useful if your provider declines to provide segments and elements that are considered mandatory according to the EDI specification.
- **Enable validation -** When set to yes (the default), the version, release, messages and segments of the EDI file (input or output) are compared to the relevant EDI repository. If the EDI file contains a value that is not in the EDI repository, an error is thrown.
  If validation is disabled (the value is set to "no"), processing continues even if the EDI file contains a version, release, message, or segment that is not in the repository. When processing an unknown version, release, message, or segment, some checks cannot be performed because the structure of the required data is unknown.
  For example, if a file is of a known version but contains an unknown segment, data type checking for that segment is not performed, but checks on the remainder of the file are performed as usual. Similarly, if a message does not exist in the EDI repository, the file is still processed, but segment order checking is not performed.
- **Component value separator -** When an element is a composite element, the character that is used to separate the component elements from each other within the composite element. This option affects EDI to XML conversion and vice versa.
- **Line continuation character -** Character appended to the segment terminator when each segment in an EDI message is split onto a new line. This character indicates to the host system that the end of the interchange has not been reached. Appended to all segments in an interchange but the last one. This option affects EDI to XML conversion and vice versa. This option is supported for all EDI dialects except CARGO.
- **Decimal character -** Symbol used for the decimal character in the converted file. Usually a period or comma. This option affects EDI to XML conversion and vice versa.
- **Element separator -** The character used to separate elements in a segment. This option affects EDI to XML conversion and vice versa.
- **Segment name/content separator (TRADACOMS) -** In TRADACOMS data streams, the default character used to separate the segment name and segment contents is the equal sign (=). You can use this option to override the default character. This option affects EDI to XML conversion and vice versa. This option only applies to the TRADACOMS dialect.
- **Invalid character replacement -** Used with the REPLACE value for the Character repertoire override (EDIFACT) option to specify the character that should be used to replace invalid characters. The default (if this option is not specified) is an underscore ("_")). Valid values are:

  \u#### – Use to specify a Unicode value (substitute the #### for the appropriate value).

  \d#### – Use to specify a decimal value, (substitute the #### for the appropriate value).

  This option affects EDI-to-XML conversions and vice versa.
- **Release (escape) symbol -** The release, or escape, character. It turns off special processing of the next character. Suppose your text message uses the same character that is also used to separate elements. The specified character is used to instruct the EDI processor to treat that character as a normal character and not the end of the text. This option affects EDI to XML conversion and vice versa.
- **Repeat symbol -** The repeat symbol for EDI dialects that use it. This option affects EDI to XML conversion and vice versa.
- **Segment separator -** The character you want to use for segment separators. This option affects EDI to XML conversion and vice versa.
- **Stop processing of EDI content character -** Specifies a character that stops all processing of EDI content when that character is encountered. For example, if an EDI stream is terminated when it encounters a ^Z (0x1A), specifying terminate=26 (26 is the decimal equivalent of 0x1A) causes EDI parsing to end when the ^Z is encountered and any remaining file content is ignored.
- **Subcomponent (tertiary separator) -** The character you want to use for subcomponent separators. This option affects EDI to XML conversion and vice versa.
- **Character repertoire override (EDIFACT) -** Allows the conversion to override and alter character encodings for EDIFACT-based documents (like EANCOM and IATA).
  You can use one or more of the following values, concatenated with a "+" symbol (FINN-ISH+REPLACE, for example):

  DEFAULT – The encoding specified in the file is used. This value cannot be used with any others.

  EANCOM – Support for these extra EANCOM characters to UNOA and UNOB are added: #, @, [,], {, }, \, |, ‘, and ^.

  SYMBOL – Forces all characters, including special characters such as element and segment separators that might otherwise be permitted, to be validated against the encoding.

  REPLACE – Replaces any invalid characters with the character specified by the invalid property. An underscore ("_") is used if the invalid property is not specified. If REPLACE is not specified, the conversion reports an error.

  FINNISH – Changes the meaning of certain characters in the Finnish character set for UNOA and UNOB (and adds UNOY and UNOZ as

synonyms for UNOA and UNOB respectively).

- **Extension map file -** Allows you to enter the path to an extension map file that describes how the custom/proprietary EDI message type you are using differs from the EDI standard on which it is based. Extension map files can use the EDI Standard Exchange Format (SEF), an open standard that allows you to describe proprietary EDI message types. The ${ENV_VAR} format can be used to specify the file path.

Note that no conversion options are applicable for Basic EDIFACT.

Validating EDI Messages

To validate EDI messages, you typically will attach a XML Assertor, XML Data Bank, or Diff tool in XML mode to the Response Payload Modeled as XML output. However, you can attach Extension tools or other kinds of tools to that output to do whatever you need to do. If you would like to validate the literal EDI content, you can attach any tool to the Response Payload output; this will send the literal EDI content to whatever tools are attached to it.

The EDI Client offers two outputs to the response payload:

- **Payload:** The response that SOAtest received from the server. Typically this will be an EDI message. This is analogous to the "Traffic" output of the Messaging Client.
- **Payload Converted to XML:** The result of converting the response payload to XML. The selected EDI format, such as EDIFACT, determines how SOAtest converts the message to XML.

For example, if you use the Plain Text format and the server returns the string "delta", SOAtest will provide the value "delta" as input to tools attached to the "Payload" output. To the tools attached to the "Payload Converted to XML" output, the EDI Client will provide the value " <root>delta</root>".

Working with the Request Payload

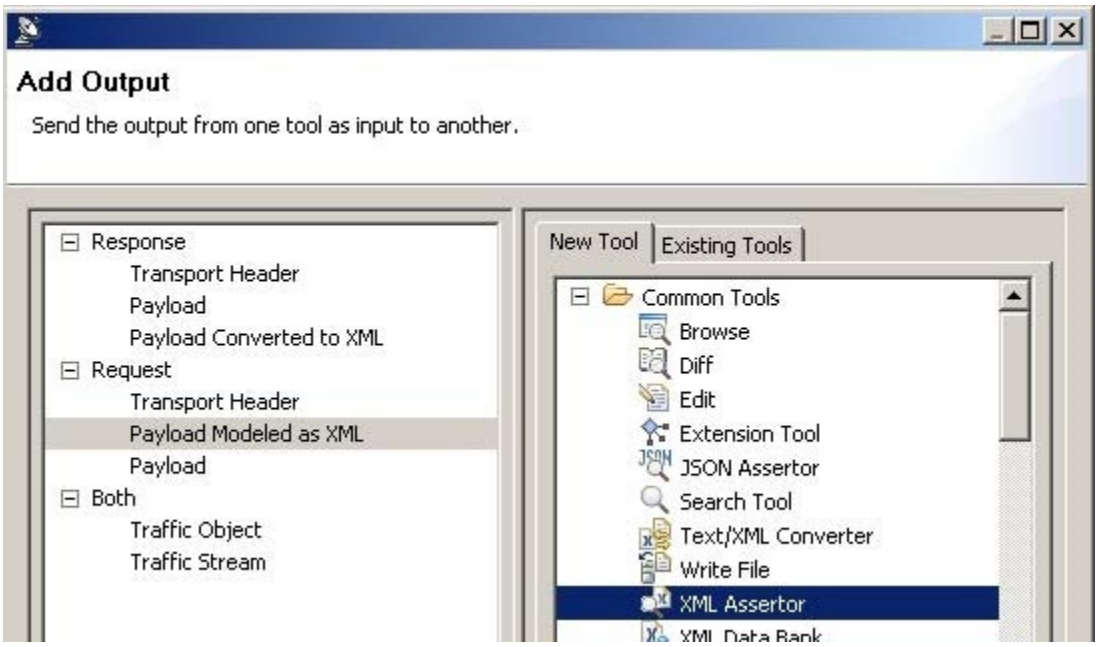
Attaching Request Payload Outputs

The EDI Client offers two outputs to the request payload. In contrast, the Messaging Client offers a single output (simply named "Traffic"). The EDI Client has two outputs because of its support for automatically converting the request from XML to an EDI format.

The two request outputs of the EDI Client are:

- **Payload Modeled as XML:** This is the XML model of the request payload—typically what you created using the Form Input or Form XML views. If you have selected the **Send XML instead of {format} when in Form Input or Form XML** option, then this output provides the request payload before it is converted from XML to EDI.
- **Payload:** The payload that SOAtest sends to the server. This is analogous to the "Traffic" request output of the Messaging Client. If you have selected the **Send XML instead of {format} when in Form Input or Form XML** mode to EDI option, then this output will be the result of converting the XML to EDI.

If you have selected the option to send XML instead of EDI (Send XML instead of EDI when in Form Input or Form XML mode), then the outputs attached to the output "Payload Modeled as XML" will not run.



Modifying the Request Payload

As with the Messaging Client, you can modify the request payload by attaching tools to the request output. For example, if you attach to the Payload output an Extension tool that returns a value, then SOAtest will use that value as the request payload. In other words, SOAtest will send that value to the server and you will see that value in the Traffic Viewer. Additionally, you can modify the XML model of the message by attaching tools to the "Payload Modeled as XML" output.

Running the Tool

The EDI Client runs the "Payload Modeled as XML" outputs before the "Payload" outputs.

For example, assume you have the following tool:

- EDI Client
  - Request Payload Modeled as XML->Extension Tool 1
  - Request Payload->Extension Tool 2

In the EDI Client UI, you have chosen to convert from XML to EDI and have chosen to use the Plain Text format. You have modeled the request payload as the following XML:

```
<root>alpha</root>
```

When the EDI Client runs, it first runs Extension Tool 1, which is attached to the request payload modeled as XML. The Extension Tool 1 contains the following Jython:

```
from com.parasoft.api import Application
def main(input, context):
   Application.showMessage("input: " + str(input))
   return str(input).replace("alpha", "beta")
```

The Extension Tool 1 prints "input: <root>alpha</root>" to the console and returns "<root>beta</root>".

The Extension Tool 2 contains the following Jython:

```
from com.parasoft.api import Application
def main(input, context):
    Application.showMessage("input: " + str(input))
    return str(input).replace("beta", "gamma")def main(input, context):
```

The tool prints "input: beta" to the console because the EDI Client internally converted the value

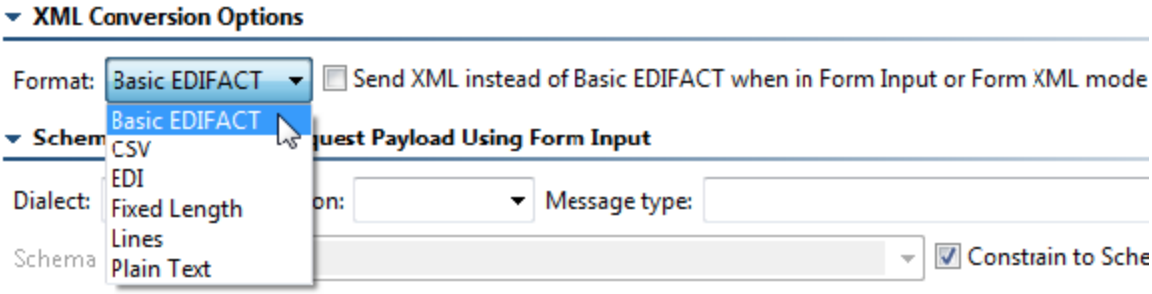
returned by Extension Tool 1, "<root>beta</root>", to text by removing the "root" tags. Extension Tool 2 returns the string "gamma". You will see the string "gamma" in the request payload in the Traffic Viewer.

Tutorial

See Working with EDI Messages.

Basic EDIFACT Support

The basic EDIFACT support that was available in SOAtest 9.1 is still available as the "Basic EDIFACT" format:



It is available in EDI Client, Enhanced Call Back, and XML Converter.

Basic EDIFACT support does not require a Message Packs license. It supports a subset of the dialects and versions that the Message Packs license covers. For example, EDIFACT is the only available dialect, and only versions 10B, 96B, S3, S4, and S41 are supported.

Note that:

- Load Test does not support the "Basic EDIFACT" format. Load Test does support all other text/XML formats, including the "EDI" format that requires the Message Packs license.
- Basic EDIFACT and EDI conversions generate different XML representations of the same EDI data. As a result, you cannot convert from EDI to XML using "Basic EDIFACT" and then convert back to EDI using the "EDI" conversion. Likewise, you cannot convert from EDI to XML using "EDI" and then convert back to EDI using "Basic EDIFACT."