Message Responder Overview

This topic explains how to configure the various Message Responder tools that send response messages correlated to incoming request messages. Message Responders can be created from traffic files that capture actual behavior or they can be generated from definitions such as WSDLs, OpenAPI /Swagger/RAML definitions, and XML schemas. They can also be added/defined manually.

Sections include:

Introduction

The term "Message Responder" refers to all tools that send responses over HTTP, MQ, JMS, or other/custom protocols. Custom Message Responders can support any message format that you are working with—for example, mainframe message formats, binary formats, or any other kind of proprietary custom message.

Message Responders can receive and respond with any content over one of the supported virtual asset protocol/API deployment options—namely, HTTP, JMS and WebSphere MQ, or through a protocol/API that is handled by a custom extension (see Using Custom Transports, Message Formats, and Tools). By customizing Message Responder options, you can customize the behavior of the virtual assets—with different request/response use cases specified manually or via data sources, error conditions, delays, and so forth.

> **SQL Responders are different than Message Responders**
>
> Whereas Message Responders send response messages that correlate to incoming requests, SQL Responders send results sets that correlate to SQL queries.

Locating and Storing Message Responders

Message Responders are created and saved within .pva files.

The **VirtualAssets** project is the default and recommended location for .pva files. Any .pva files added to this project will be automatically deployed to the local Virtualize server. If a .pva is not in this project, it will not be deployed.

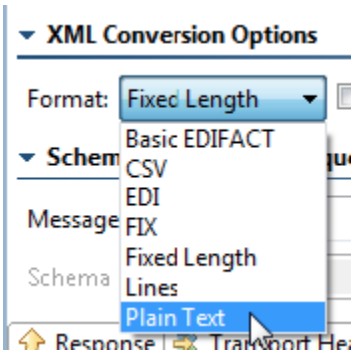
This project contains a **VirtualAssets.xml** file that stores the deployment configuration for each virtual asset—including the location of each virtual asset's .pva file, name and HTTP endpoint path, global reporting settings, and JMS and WebSphere MQ transport settings. This file is automatically updated when virtual asset deployment settings are modified.

Configuring Responders for the Applicable Message Format(s)

Message Responders for different message formats are created and configured using a common Message Responder framework. Configuration options are shared across all types of Message Frameworks whenever feasible. In addition, special configuration options are available for certain message formats when applicable (for example, there might be custom conversion options for a custom mainframe or binary message format extension that your team developed). Thus, Message Responders for different formats will sometimes offer different configuration options.

Message Responders can be configured to receive a message in one format and respond in another (for example, receive an EDI message and respond with a Fixed Length message). This is configured via Request Handling options.

If you ever want to change the message format that a Message Responder uses (e.g., if you want to change from Fixed Length to Plain Text), just change the selected format.



Configuration Options

General Configuration

The following general option is available at the top of the tool:

- Data Source: Specifies the Data Source to be used for providing tool values. This menu is only available if a Data Source was added to the project (as described in Parameterizing Tools with Data Source Values, Variables, and Extracted Values).

XML Conversion Options

These options available only for formats that are converted to/from XML.

- **Format:** Select the format for the response payload.
- **Respond with XML instead of [native format] when in Form Input or Form XML mode:** If disabled (default), Virtualize will automatically convert the XML model into the native format before sending the message. In other words, Virtualize sends the native format to the server—so that is what you will see in the Traffic Viewer.
    - If this option is enabled, Virtualize will send the message you configured in the UI for the response. 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 Virtualize will send that XML unmodified. You typically would not want to do this for the formats currently supported, but Virtualize makes this option available.

Schema for Modeling Request Payload Using Form Input

These options available only for formats that are converted to/from XML.

Message type:

- **Message type:** Specifies the message type you want to use. This prompts Virtualize to populate the Form Input view.
- **Schema URL:** Describes the Schema URL where this service can be accessed. You can either enter a value or click the **Browse** button. If you do not have a schema, you can leave this field empty.
- **Constrain to Schema:** Determines whether certain tool parameters obtain their values from the Schema rather than from manual entry. If this option is enabled, certain parameters are disabled and get their values from the schema. If this option is disabled, the **Refresh Schema** button will also be disabled.

Definition Tab

This tab is designed specifically for JSON, SOAP, and Plain XML Message Responders

Specifying options in the **Definition** tab allows Virtualize to populate the **Response** tab with items that make it easier for you to specify the response message. You can specify the following settings related to the WSDL or schema that defines the expected request and outgoing response:

- **Service Definition:** Specifies if the responder has an associated service definition (RAML, OpenAPI/Swagger, WSDL or Schema). For plain XML, choose **None**. When a service definition is specified and the tool is constrained to WSDL or schema, the applicable form view will be automatically populated with appropriate field values and controls (e.g., radio buttons for booleans, different controls for numbers vs. integers, drop-downs for enumerations). It will also prevent the entry of invalid messages (e.g., messages missing required values, a value of the incorrect type, a method/resource/response code not specified in a schema, etc.). Editing will be restricted to ensure that the message complies with the associated schema (e.g., you will not be able to insert, delete, rename, copy, or paste tree nodes).
- **For RAML Service Definition Mode**
    - **RAML URL:** A specific RAML URL, or a variable referencing a RAML URL.
- **For OpenAPI/Swagger Service Definition Mode**
    - **OpenAPI/Swagger URL:** A specific OpenAPI/Swagger URL, or a variable referencing a OpenAPI/Swagger URL.
- **For WSDL Service Definition Mode**
    - **WSDL URL:** Describes the WSDL URL where this service can be accessed. You can either enter a value or click the **Browse** button. If you do not have a WSDL, you can leave this field empty.

    - **Constrain to WSDL** Determines whether certain tool parameters obtain their values from the WSDL rather than from manual entry. If this option is enabled, certain parameters (e.g. router endpoint, SOAP action, SOAP body and header parameters) are disabled and get their values from the WSDL. If this option is disabled, the **Refresh WSDL** button will also be disabled.
    - **WSDL Documentation:** (Automatically completed if available): Describes the service at the given WSDL URI.
- **For Schema Service Definition Mode**
    - **Schema URL:** Describes the Schema URL where this service can be accessed. You can either enter a value or click the **Browse** button. If you do not have a schema, you can leave this field empty.
    - **Constrain to Schema:** Determines whether certain tool parameters obtain their values from the Schema rather than from manual entry. If this option is enabled, certain parameters are disabled and get their values from the schema. If this option is disabled, the **Refresh Schema** button will also be disabled.
    - **XML Message Type:** Determines whether the Response tab's Form Input controls are geared for SOAP messages or plain XML messages. If Plain XML is selected, Form Input mode will directly represent the message. If SOAP is selected, the XML configured in Form Input will be wrapped with a SOAP Envelope and the view will provide a SOAP Header section that allows you to configure SOAP headers if desired.

Response Tab

The **Response** tab allows you to configure the response values that you want the responder to deliver when correlated requests are received.

The options available vary depending on what option is selected in the Views menu.

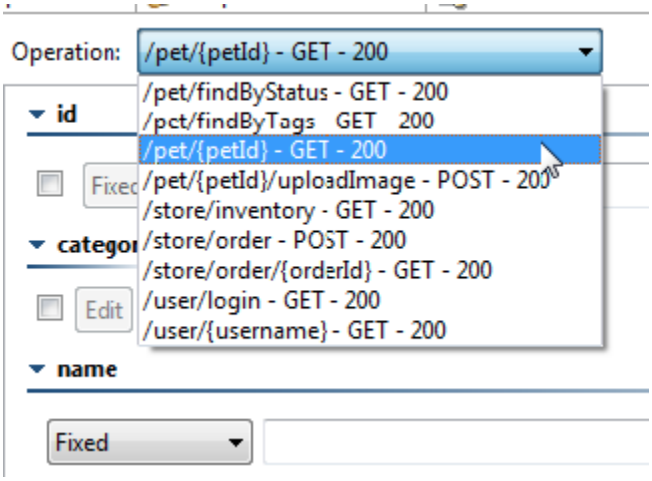
Form Input, Form XML, Literal, Multiple Response, Scripted, and Form JSON Views

These views are similar across multiple Parasoft tools; they are described in the following areas:

- Form Input View Options
- Form XML View Options
- Literal View Options
- Scripted View Options
- Multiple Responses View Options
- Form JSON View Options
- Form ISO 8583 View Options

Notes

- The form-based views allow you to build the response and parameterize values from a data source.
- If an operation selector is enabled in the Form Input or Form JSON view, you can use it to select the operation for your response. When you select an operation, the Form Input/JSON view will be populated with appropriate values and UI controls (e.g., radio buttons for booleans, different controls for numbers vs. integers, etc.).
    - In Form Input mode, the operation selector will show all operations for which the associated service definition 1) defines the response as XML and 2) specifies an XML schema.
    - In Form JSON mode, the operation selector will show all operations for which the associated service definition 1) defines the response as JSON and 2) specifies a JSON schema.



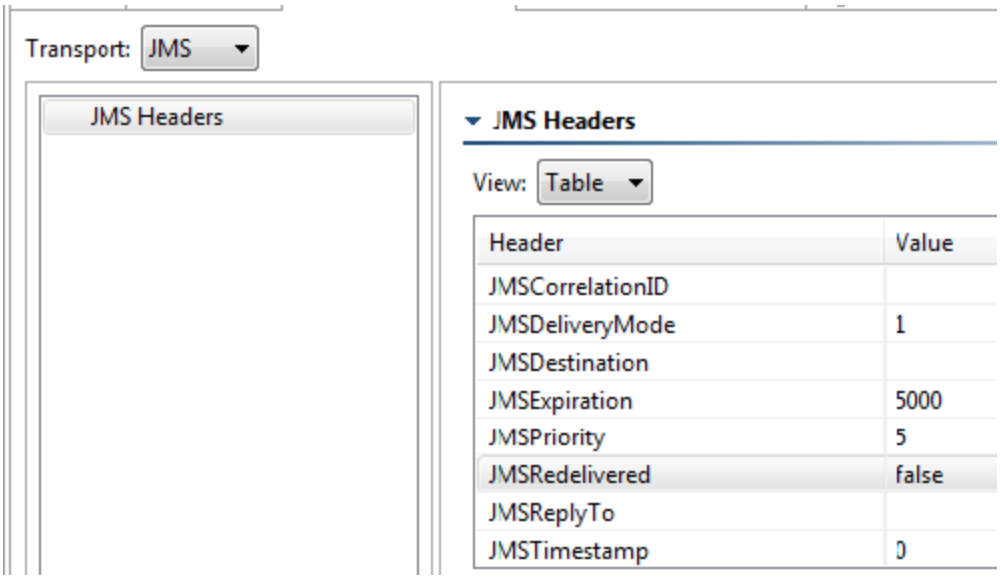
- In Multiple Responses mode, each possible response has its own correlation to the incoming request, static response message, and performance (time) options.

Transport Header Tab

The **Transport Header** tab allows you to specify the HTTP, JMS, or MQ message header that will be returned with the response message. You can use table view or literal view.

In the case of HTTP, these header values will be appended to the standard HTTP headers that are generated in the responses. In the case of JMS, they are set as String message properties into the outgoing response messages. Or, if the provided header name matches one of the existing message property names, then it will be overwritten (example: JMSCorrelationID). In the case of WebSphere MQ, there is a section for RFH2 headers that allows for configuring these as well.

For JMS, you can define any miscellaneous property values to be set into the javax.jms.Message object before it gets sent to a queue or published to a topic. For instance, you could define JMSPriority, JMSDeliveryMode, JMSExpiration, and other properties as follows:



Responder Correlation Tab

The **Responder Correlation** tab allows you to specify which messages this Message Responder tool accepts and processes. Various messages sent to the virtual asset URL are routed to specific Message Responder tools (each of which handle different operations) based on the settings here. For example, one Message Responder tool might respond to customer registration messages, another might respond to payment messages, and another might function as a default "catch all" that is used when none of the others match.

You can specify which messages a Message Responder accepts by entering values in the following **Responder Correlation** tab areas:

- **Transport:** Allows you to specify HTTP Headers, JMS message properties, or MQ message fields within the message that will determine whether or not the message is processed by this particular responder. See Transport Correlation for details.
- **Request Body:** Allows you to specify XPaths within the message that will determine whether or not the message is processed. See Request Body Correlation for details.
- **URL Parameters (for RESTful services):** Allows you to specify URL parameters that will determine whether or not the message is processed. You can configure the correlation to match all messages that include specific parameters (no matter what value they are set to), or only match messages where specific parameters are set to specific values). URL parameters can be repeated (you can have the same parameter set to different values). See URL Parameter Correlation for details.
- **URL Path:** Allows you to specify URL paths that will determine whether or not the message is processed. See URL Path Correlation for details.
- **HTTP Methods:** Allows you to specify HTTP methods that will determine whether or not the message is processed. See HTTP Method Correlation for details.
- **Custom:** Allows you to specify a custom responder correlation that is based on the return value of a custom method written using Java or scripting languages. See Custom Correlation for details.
- **ISO 8583 Correlations:** Allows you to specify correlations for ISO 8583 messages. See Using ISO 8583 for details. *These options are designed specifically for SOAP, Plain XML, and Literal Message Responders.*

You can configure one type of correlation, multiple types of correlation, or no correlations. If no correlations are configured, everything in the message will be processed.

Transport Correlation

To configure a transport correlation:

1. Select the **Enable correlation** check box.
2. Click the **Add** button. A new entry row displays.
3. Enter the **Header Name**.
4. Do one of the following:
    - If you want to match messages where this header is set a specific value: Enter that value under **Value of Header**.
    - If you want to match all messages that include this header (no matter what value it is set to): Check **Correlate whenever the named header is present regardless of value**.

Request Body Correlation

To configure a request body correlation:

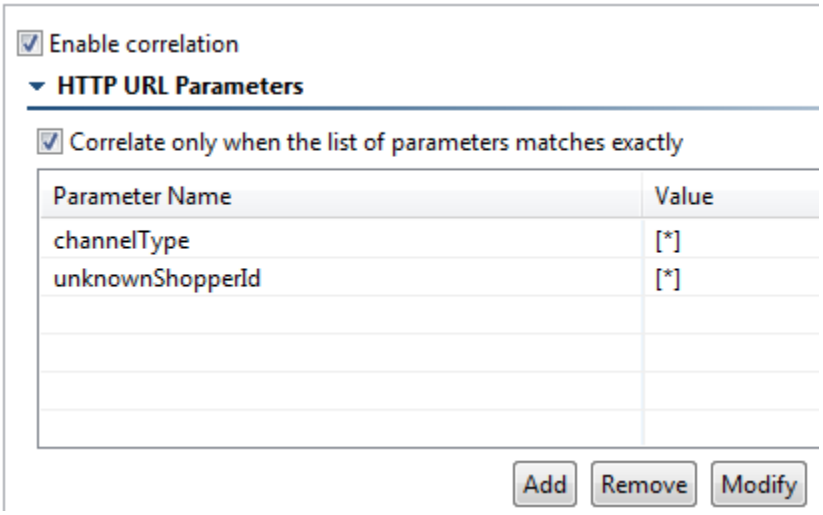
1. Select the **Enable correlation** check box.
2. Click the **Edit** button. An Edit XPath Function dialog displays.
3. Select an element from the Element tree, select a function from the Function drop-down menu, and click **OK**.
4. Do one of the following:
    - If a tree representation of the incoming request message is displayed, then select an element from the tree and select a function from the Function drop-down menu.
    - If a tree representation of the requests is not available—or if the tree does not fully display the elements or attributes that are of interest for extraction and evaluation—then provide an XPath expression manually. The XPath expressions supported by Virtualize for correlation purposes is based on the standard XPath 1.0 standard by W3C.
5. If you want to specify any additional XPaths, add them in the same manner.

URL Parameter Correlation

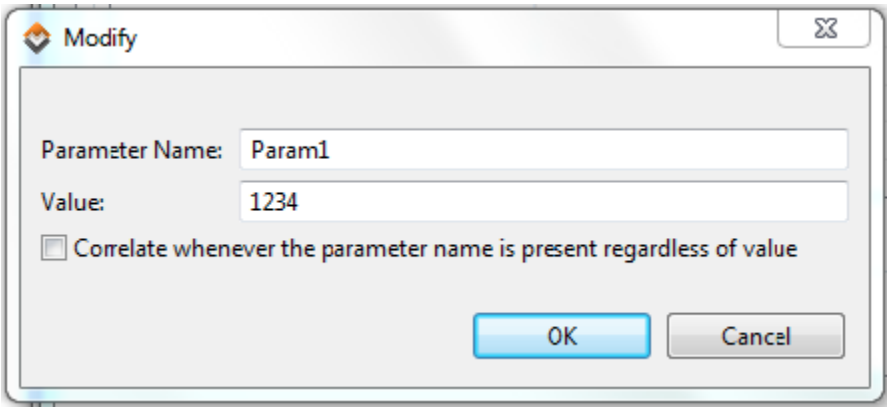
URL parameter correlations apply to virtual assets that will be accessed over HTTP/HTTPS.

To configure a URL parameter correlation:

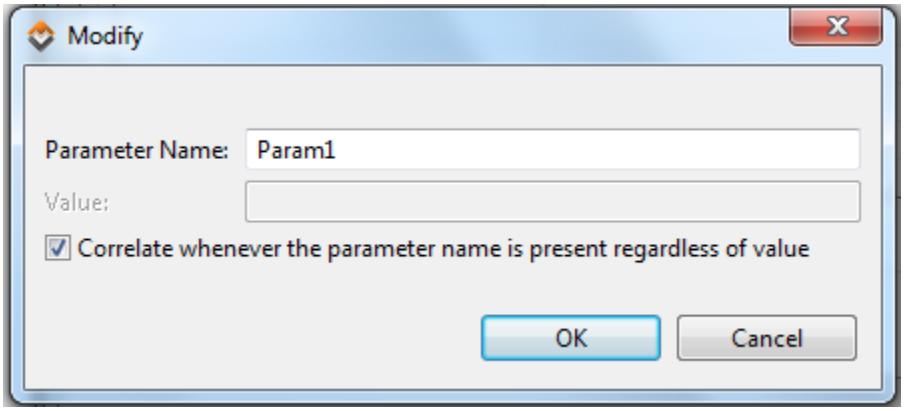
1. Select the **Enable correlation** check box.
2. If you want the correlation to be applied only if the request has the *exact* URL parameters specified in this table—no more and no less—check the **Correlate only when the list of parameters matches exactly**. Otherwise, any request that contains listed parameters as well as other parameters will correlate. Whether this option is enabled or disabled, a message that lacks one of the specified parameters will not match.



3. Click the **Add** button. A new entry row displays.
4. Enter the **Parameter Name**.
5. Do one of the following:
    - If you want to match messages where this parameter is set a specific value: Enter that value under **Value**.



- If you want to match all messages that include this parameter (no matter what value it is set to): Check **Correlate whenever the parameter name is present regardlessof value**.



For example, assume that an asset is deployed under the path `http://myvirtserver:9080/MyAsset/MyPath` and the following URL Parameters are specified in a responder:

Parameter Name	Value
param1	value1
param2	value2

The following request URLs will match this correlation criteria:

- http://myvirtserver:9080/MyAsset/MyPath?param1=value1&m2=value2
- http://myvirtserver:9080/MyAsset/MyPath?param1=value1&m2=value2&m3=value3
- http://myvirtserver:9080/MyAsset/MyPath?param2=value2&m1=value1&m3=value3

The following URLs will NOT map to this responder:

- http://myvirtserver:9080/MyAsset/MyPath?param1=value1&param2=someothervalue&param3=value3
- http://myvirtserver:9080/MyAsset/MyPath?param1=value1&m3=value3

URL Path Correlation

URL path correlations apply to virtual assets that will be accessed over HTTP/HTTPS. *Note that the matching strategy has changed since Virtualize 9.7; existing virtual assets will automatically be reconfigured to use the current paradigm when they are modified.*

To configure a URL path correlation:

1. Select the **Enable correlation** check box.
2. Specify the path you want to use for correlation.

For example, assume that a virtual asset is deployed under the path `http://myvirtserver:9080/MyAsset/MyPath` and the path `/segment1/segment2/**` is specified in the responder correlation.

The following URLs will match this correlation criteria:

- http://myvirtserver:9080/MyAsset/MyPath/segment1/segment2
- http://myvirtserver:9080/MyAsset/MyPath/segment1/segment2/
- http://myvirtserver:9080/MyAsset/MyPath/segment1/segment2/segment3?param1=value1&m2=value2

The following URLs will NOT match:

- http://myvirtserver:9080/MyAsset/MyPath/segment1/
- http://myvirtserver:9080/MyAsset/MyPath/segment3/segment1/segment2

Ant-style wildcards can be used, where * matches zero or more characters and ** matches zero or more directories. Using this format:

- `/**/abc` matches `/abc` or `/this/that/abc`
- `/abc/**` matches `/abc` or `/abc/this/that/theother`
- `/ab**` DOES NOT match `/abc/d` (it is interpreted as if you used a single asterisk `/ab*`); it does match `/absolutely`
- `/**bc` DOES NOT match `/0/abc` (it is interpreted as if you used a single asterisk `/*bc`); it does match `/abc`

Example 1: **/service/*

Matches	Does Not Match
service/Repository org/web/service/Entries org/something/else/tools/stiff/service/Entries	org/web/service/foo/bar/Entries

Example 2: org/parasoft/virtualize/**

Matches	Does Not Match
org/parasoft/virtualize/tools /service org/parasoft/virtualize/stuff	org/parasoft/somethingelse

Example 3: org/parasoft//EM/***

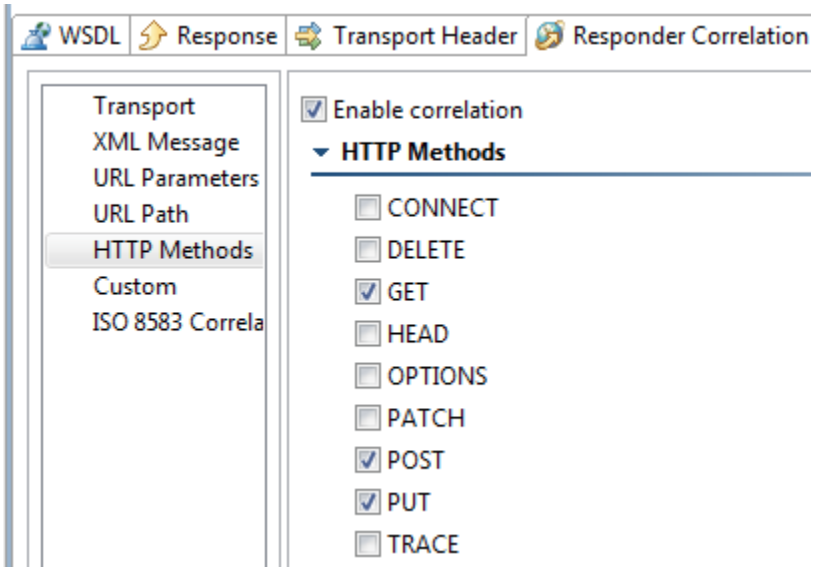
Matches	Does Not Match
org/parasoft/EM/Entries org/parasoft/virtualize/soatest/tools/EM/Entries	org/parasoft/EM/foo/bar/Entries

HTTP Method Correlation

HTTP method correlations apply to virtual assets that will be accessed over HTTP/HTTPS.

To configure an HTTP method correlation:

1. Select the **Enable correlation** check box.
2. Specify the HTTP methods you want to use for correlation.



For example, if GET, POST, and PUT are checked, the responder will match HTTP requests with the method GET, POST, or PUT. HTTP requests with CONNECT, DELETE, HEAD, etc. will not match.

Custom Correlation

A custom responder correlation is based on the return value of a custom method written using Java or scripting languages.

Note that:

- The method must take 0 or 1 arguments.
- The optional argument is of type `com.parasoft.api.ScriptingContext`.

- The method can access the transport headers, URL parameters, and message contents of the incoming message using the ScriptingContext object.

More information can be found at **Parasoft> Help> Extensibility API> CorrelationScriptingHookConstants**.

Regular Expression

For example, if you want to use a plain text message based on a regular expression, you might use:

```
from com.parasoft.api import CorrelationScriptingHookConstants
from java.lang import *
def match(context):
  strMessage = context.get(CorrelationScriptingHookConstants.MESSAGE_STR)
  return String(strMessage).matches("myRegularExpression")
```

XML Element

If you want to use an element in an XML message, you might use:

```
from com.parasoft.api import CorrelationScriptingHookConstants
from org.w3c.dom import *
from javax.xml.parsers import *
from javax.xml.xpath import *
from java.io import *
from java.lang import *
def match(context):
  xmlDocument = context.get(CorrelationScriptingHookConstants.MESSAGE_DOM)
  if xmlDocument != None:
    xPathFactory = XPathFactory.newInstance();
    xpath = xPathFactory.newXPath()
    expression = xpath.compile("//*[local-name(.)='someElement' and namespace-uri(.)='someNamespace'][1]
/text()")
    elementValue = expression.evaluate(xmlDocument)
    return String(elementValue).matches("myRegularExpression")
  return 0
```

HTTP Method

If you want to correlate on an HTTP method, you might use:

```
from com.parasoft.api import *

def correlateHTTPMethod(context):
    method = context.get(CorrelationScriptingHookConstants.REQUEST_METHOD)
    return "GET" == method
```

Attachment Content

To access attachments in a custom responder correlation, use the key `CorrelationScriptingHookConstants.ATTACHMENTS` to get a list of all attachments. The attachment interface that is returned has a single method, `getContents`, that returns an object based on the mime type of the attachment:

- For text/plain, a string is returned.
- For text/xml, a javax.xml.transform.stream.StreamSource.StreamSource is returned.
- For all other types a java.io.InputStream is returned.

For example, if you wanted to correlate on an attachment (here, a string with the word "Attachment"), you might use:

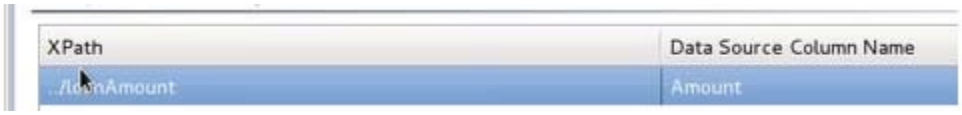
```
from com.parasoft.api import CorrelationScriptingHookConstants
from org.python.core.util import FileUtil
from org.python.core.util import StringUtil

def correlateOperation(context):
    # List<com.parasoft.api.ICorrelationAttachment>
    attachments = context.get(CorrelationScriptingHookConstants.ATTACHMENTS)
    # java.io.InputStream
    instream = attachments.get(0).getContents()
    bytes = FileUtil.readBytes(instream)
    return "Attachment" == StringUtil.fromBytes(bytes)
```

Data Source Correlation Tab

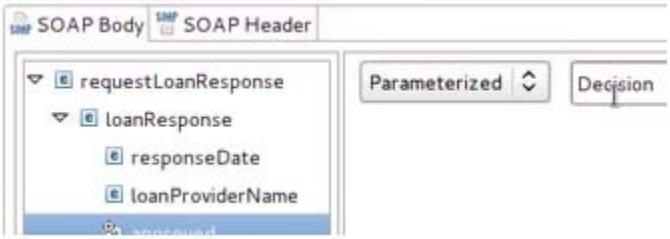
The **Data Source Correlation** tab allows you to specify which data source row values to use in virtual asset responses. Based on the settings here, Virtualize performs a data lookup in the data source to find a row of data. This row is then used to populate the response (as defined in the Response tab) with parameterized values from the data source.

For example, the following configures a correlation between the loanAmount value in incoming messages and the Amount column within the ApprovalLists data source:

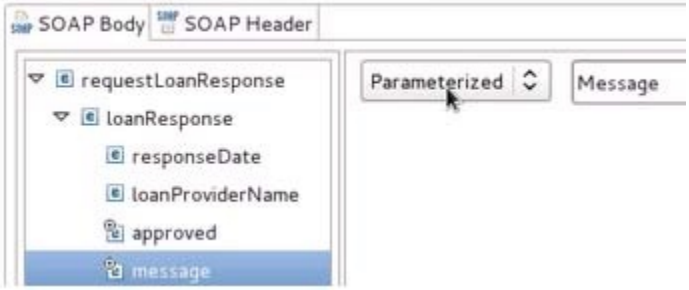


For each incoming request, the loanAmount value will be matched to one of the rows in the Amount column. The response will then be parameterized with values from other columns for that same row.

With the following settings in the Response tab, the approved value will be parameterized with values from the corresponding value in the Decision data source column...



and the message value will be parameterized with values from the corresponding value in the Message data source column.



Benefits

One way to configure virtual assets with multiple responses is to manually configure the Message Responder tool's **Multiple Responses** mode to send different response messages based on the nature of the incoming message.

Another way to configure virtual assets to dynamically respond with the desired message is to use data sources. You can easily fill out data source tables (Excel, etc.) where each row contains values in the incoming message (typically, just the leaf node values) that you want the virtual asset to respond to, then another column that specifies how you want the virtual asset to respond when the specified conditions are met (see Using Existing Data Sources or Rapidly Creating Data Sources for Responses for details). After that, you can configure the mapping between the request and response message values and the columns within the data source.

This allows for request/response use cases to be configured easily in one easy-to-edit table (data source), it allows them to be managed there to scale further with more and more messages, and it also provides more flexibility with response messages (since Form Input allows you to have some values fixed, others parameterized, some automatic or even scripted).

This data source correlation is very flexible. You can configure the virtual asset to respond to incoming request values with rules and logic beyond exact match/correlation. For example, you could configure a virtual asset to evaluate the last 4 digits of a credit card number, numerical values less or greater than a given value, or other patterns and expressions.

The value matching supports the wild cards * and ?. * matches zero or more characters; ? matches a single character.

For example, if you want an incoming value named "title keyword" to match a certain row as long as it contains the word Linux, then you can have the data source value *"[like *Linux]"*

Configuration Procedure

To parameterize virtual asset responses from a data source:

1. Ensure that you have a data source configured in Virtualize and available for the Message Responder.
2. Click the **Add** button in the appropriate part of the **Data Source Correlation** tab, provide an XPath, URL parameter, URL path and/or criteria expression and select the appropriate column name. When the deployed virtual asset is exercised, Virtualize will extract the specified values from the request, then search the named column for a match that corresponds to the extracted value. If a match is found, the data in the corresponding data source row will be used to populate the response
    - If you specify the same column name multiple times (e.g., in URL Parameters and URL Paths), only one value will be set; the previous value(s) will be overwritten.
    - If a WSDL was provided, the XPath can be generated using the **Edit** button. That dialog validates the XPath expression and the column name in real time.
    - Virtualize validates the XPath syntax while you type/edit these XPath expressions—if you choose to customize them instead of using the visual **Edit** option.
    - Wildcards can be used in the data source column.
    - Expressions (e.g., `[> "55"], [like "alpha?beta*"], [like "*b*" or like "*a*" and like "*z*"]`) can be used as described in Criteria Expressions for Matching Values with the Message Responder.
    - See Data Source Correlation Options below for details on the available configuration options.
3. Parameterize the Response area of the Message Responder by referencing other data source columns within that same data source (i.e., Form Input, Form XML, etc.). You can add several XPaths or URL parameters to column name mappings.

Data Source Correlation Options

The following configuration options are available for data source correlations.

Failover Options

When data source correlation is enabled, Virtualize uses the criteria specified in this tab to try to match values in the incoming message with data source values. If no data source row matches the specified incoming message value, this is considered to be a *data source correlation failure.*

When **Continue searching for a matching responder if data source correlation fails** is enabled (the default setting) and data source correlation fails for this responder, Virtualize will continue searching the responder suite for a responder with matching responder correlation. An error will be reported only if:

- No responders meet the responder correlation criteria, or
- Another responder in the responder suite matches the responder correlation, but its data source correlation fails and its data source correlation failover option is disabled.

If **Continue searching for a matching responder if data source correlation fails** is disabled and the data source correlation for this responder fails, an error will be reported immediately and Virtualize will stop searching for matching responders.

When an error is returned to the requester, an error event is also logged into the Event Details view (if monitoring of the virtual asset is enabled as described in Gaining Visibility into Server Events).

Request Body

This area allows you to specify one or more XPaths to run on the incoming messages in order to extract one or more values. The extracted values will be matched with values from the mapped data source columns.

Request HTTP URL Parameters

This area applies to assets that are accessed over HTTP/HTTPS. Specify one or more URL parameters to look for in the request URL. The parameter values will be matched with values from the mapped data source columns.

If multiple columns are used for the correlation, the values for each column row must all be either decoded or encoded. If all values are encoded, they must all be encoded in the same way—using either the plus sign or “%20” for spaces, but not both.

For example, the following is allowed:

Borrower 1	Borrower 2
John Smith	Jane Doe
John+Smith	Jane+Doe
John%20Smith	Jane%20Doe

In this case, Virtualize will be able to correlate the incoming request with the query string Borrower1=John+Smith&Borrower2=Jane+Doe or any other variations in URL-encoding.

The following is NOT allowed:

Borrower 1	Borrower 2
John Smith	Jane+Doe
John+Smith	Jane%20Doe
John%20Smith	Jane Doe

Matching on Absent/Empty Fields and Parameters

Data source correlations can correlate on the absence of a field in the request. For example, assume you have a responder that’s configured with data source correlation on a text value of an XML element on the Request Body—but that element is optional. You can use the same data source to populate the response with parameterized values when the field is absent. To do this, you would set the value of the corresponding field in the data source row as the empty string. Because of the property of XPath function, the empty string would match the case when the XML element is absent and when the XML element is empty. This is also the case for data source correlation for URL Paths and ISO 8583 Messages: to match the absence of the field in the request, you can use the empty string as the value for the corresponding field in the data source row.

URL parameter data correlation works somewhat differently (in order to handle cases where the parameter value is the empty string and where the parameter is altogether absent):

- To match the case where the URL parameter is present but its value is the empty string, use the empty string as the value for the corresponding field in the data source row.
- To match the case where the parameter is absent in the request URL, right-click the corresponding field in the data source row (in the data repository editor) and choose **Set field to exclude**.

Request URL Paths

This area applies to assets that are accessed over HTTP/HTTPS. You can configure data source value mappings based on path segment indexes.

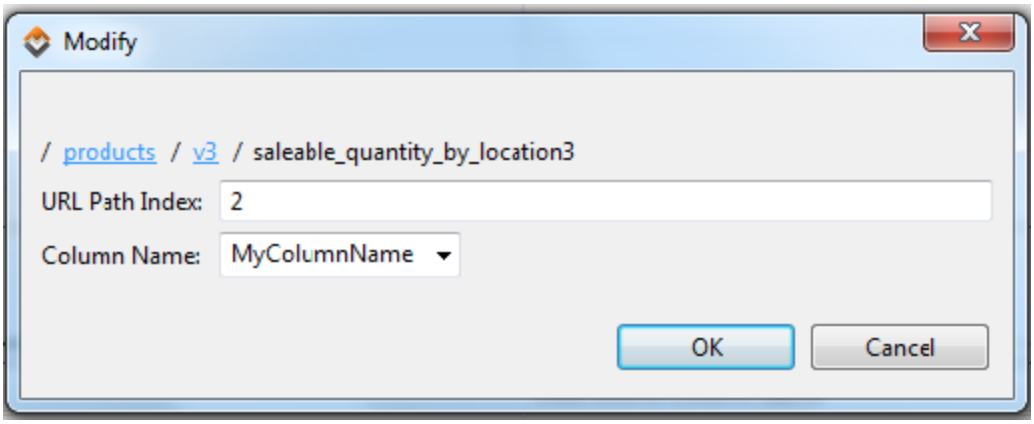
For example, assume that:

- The asset is deployed on the path `http://myvirtserver:9080/MyAsset/MyPath`
- Incoming requests have the pattern `http://myvirtserver:9080/MyAsset/MyPath/[somedynamicvalue]/B/C`

If you want to take the value under the `[somedynamicvalue]` segment and match it with a data source column, then specify 2 for the URL Path Index. The counting is inclusive of the path segments that are used in the asset deployment path and start from zero.

If you want to match a dynamic value from the URL `http://myvirtserver:9080/MyAsset/MyPath/A/[somedynamicvalue]/C`, then use the path index value 3.

To automatically enter the appropriate path index, select the appropriate hyperlink at the top of the dialog.



If multiple columns are used for the correlation, the values for each column row must all be either decoded or encoded. If all values are encoded, they must all be encoded in the same way—using either the plus sign or "%20" for spaces, but not both.

For example, the following is allowed:

Borrower 1	Borrower 2
John Smith	Jane Doe
John+Smith	Jane+Doe
John%20Smith	Jane%20Doe

In this case, Virtualize will be able to correlate the incoming request with the query string Borrower1=John+Smith&Borrower2=Jane+Doe or any other variations in URL-encoding.

The following is NOT allowed:

Borrower 1	Borrower 2
John Smith	Jane+Doe
John+Smith	Jane%20Doe
John%20Smith	Jane Doe

Request Headers

Provide the headers for the request values you want to extract and match, then map each to a data source column. The extracted values will be matched with values from the mapped data source columns.

Request ISO 8583 Message

This area applies only to matching values for ISO 8583 messages. Provide the field IDs for the values you want to extract and match, then map each to a data source column. Only messages that contain these ISO 8583 fields and have matching values will correlate.

These options are designed specifically for SOAP, Plain XML, and Literal Message Responders.

Processing Details

When a Message Responder is active, the incoming messages are evaluated through these criteria. The values are then matched against the corresponding data source values (each with its respective column) to find a row that matches the values.

A data source row is considered a match if all the values specified in the responder's data source correlation list matched in that row.

If a matching row is found, then that row will be used for any parameterized values in the Message Responder's response message. This way, the virtual asset can respond with the desired message values based on values in the incoming messages.

For more details on processing, see Criteria Expressions for Matching Values with the Message Responder.

Attachment Tab

The **Attachment** tab allows you to send either Binary or XML HTTP attachments to the response message.

When you are using MTOM, you do NOT need to add anything to the **Attachment** tab. The MTOM option will cause it to automatically create the MIME_boundaries with the optimized (non-encoded) SOAP Envelope xsd:base64Binary content. The **Attachment** tab applies only to MIME/DIME options.

To send an attachment, perform the following from the **Attachment** tab:

1. Click the **Add** button. An **XML Attachment** entry displays in the Attachment table.
2. Double-click the **XML Attachment** entry. An **Edit Attachments** dialog displays.
3. In the Edit Attachments dialog, select either **XML** or **Binary** from the **Mode** drop-down menu.
4. Configure the attachment.
    - The following option is available for **XML** Mode:
        - **Views:** Select the desired view from the drop-down menu and configure accordingly.
    - The following options are available for **Binary** Mode:
        - **Base 64 Encoding:** Enables Base 64 Encoding to encode the binary value.
        - **Data Source Column:** Select to send values from a data source column.
        - **File:** Select to send values from a file. Choose the desired file by clicking the **Browse** button. Check the **Persist as Relative Path** option if you want the path to this file to be saved as a path that is relative to the current configuration file.
        - **Text:** Select to send a text value.
        - **Content type:** Specify the content type. Click the **Edit Headers** button if you want to add, modify, or delete attachment headers.

Options Tab

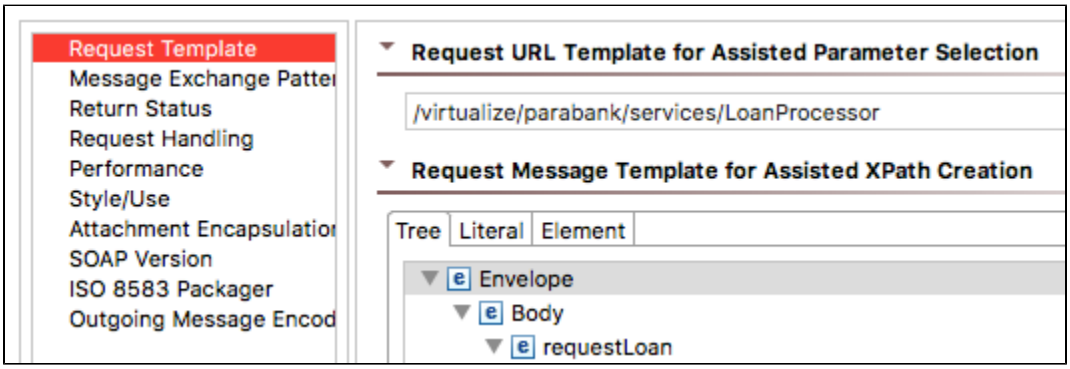
The **Options** tab allows you to configure how the message is processed. The following options are available:

Request Template

Request URL Template for Assisted Parameter Selection

Allows you to enter a URL that is typical or representative of the URLs that the application under test would serve (and the responder should simulate). If the responder is created through the parameterized traffic wizard (described in Creating Parameterized Message Responders from Traffic), this field will be populated with the URL from one of the requests in the traffic corresponding to this responder.

The value specified here will be used to configure URL Paths and URL Parameters correlation settings in both the Responder Correlation and Data Source Correlation tabs.



Request Message Template for Assisted XPath Creation

Allows you to specify an XML-format template that will be used to automatically populate the expected XML response when XPath parameters (e.g. in the message request XPath dialogs for data source correlation and responder correlation or in the multiple response XPath dialog).

- When a Message Responder is created from traffic logs, the Request Message Template is generated automatically. The largest request message identified in the Traffic Log is used for this purpose.
- When a Message Responder is NOT created from traffic logs, the template will be empty. In this case, you can manually modify the Request Message Template through the Tree or Literal views (e.g., by copying in a sample request message that correlates with the response message you're configuring).
- The completed Request Message Template is used to populate the Tree/Literal/Element views in the Edit XPath Function dialogs available from the Message Responder. The XML Data Bank and XML Transformer output will also use the template if it is attached to the Message Responder as a Request output.

Message Exchange Pattern

Select Solicit Response or Notification Only.

- **Solicit Response** is the default option. It indicates that a response message body is returned.
- **Notification Only** indicates that the request message is merely acknowledged. In the case of HTTP, the response is just an HTTP header with no body content. In the case of JMS or WebSphere MQ, the incoming request message is picked up from the request queue, but no response message is generated into a response.

Return Status (Only applicable to HTTP)

Allows you to specify how the message is returned—for instance, to emulate working or faulty services. To use the default value of 200 OK, select the **Use Default Return Status** check box. If this option is not selected, the following options are available:

- **Return Status:** Enter the custom return status value. If a data source is available, you can also parameterize this value.
- **Return Message:** Enter the custom return message. If a data source is available, you can also parameterize this value.

Request Handling

These options let you determine whether (and how) incoming are modified before correlations are processed.

Select **Allow incoming request tools to modify the message before applying data source or multiple response correlations** if you want to chain a tool that alters the request message before the data source or multiple response correlation criteria execute on that message. However, if you are chaining a tool that might alter the request—but do not want such alterations to impact the message that is passed to the data source and multiple response correlation criteria—then leave that option unselected. This is important when the XML must be altered to make it suitable for correlation—or if the message is not XML at all and requires some transformation before the virtual asset sends the original caller a suitable response for the request.

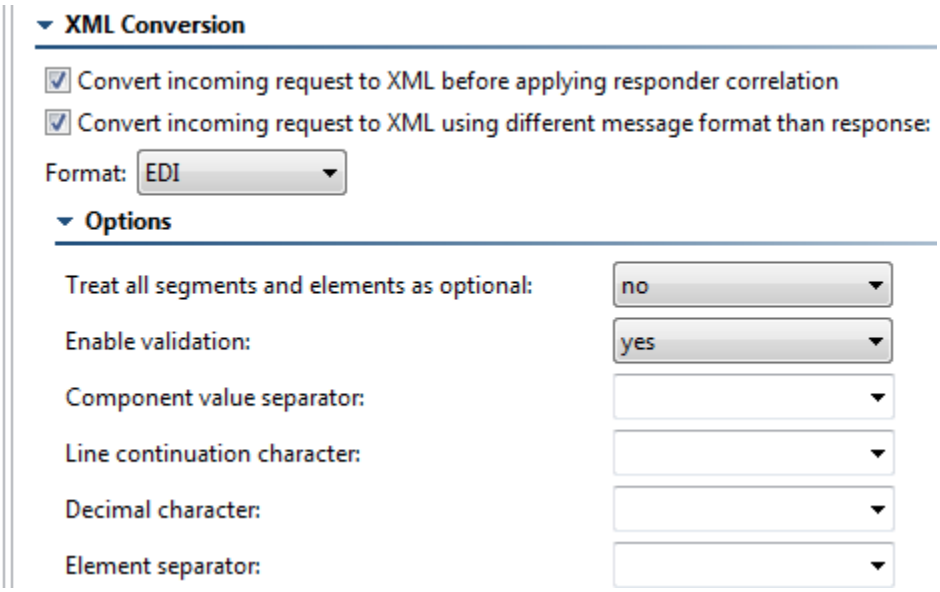
Select **Skip incoming request tools if data sources correlations fail** if you want to prevent chained request tools from being executed if the incoming message is not succesfully correlated to data source columns.

XML Conversion options are also available for message formats that are converted to XML:

- **Convert incoming request to XML before applying responder correlation** ensures that XML conversion occurs before correlations are run. If this is enabled, correlations will be based on the converted XML.

> **Impact on Message Modelling**
>
> With **Convert incoming request to XML before applying responder correlation** enabled, you can paste sample native-format traffic into Literal tabs (in the Request Message Template as well as the XPath areas of the Responder Correlation and Data Source Correlation controls) and the Tree/Element tabs will render that message using the structure designated for that message format. For example, if you paste a JSON message into a Literal tab within a JSON responder, then open the Tree tab, the message will be rendered in Form JSON structure.
>
> If you want to paste a message that is in a different format than the response, be sure to enable the **Convert incoming request to XML using different message format than response** option and specify the desired message format.

- **Convert incoming request to XML using different message format than response** lets you configure the Message Responder to receive messages in a different format than the one it is configured to respond with. For example, if you wanted to receive an EDI message and respond with a Fixed Length message, you would create a Fixed Length Message Responder, then configure it to convert incoming requests as EDI.



Performance

Allows you to set the following options related to the execution time of the Message Responder:

- **Performance group:** If you're using performance profiles, you can set or change this responder's performance group using this control. For details on performance profiles, see Working with Performance Profiles.
- **Think time:** Enter the amount of time (in milliseconds) for the message delay you want to emulate (e.g., to emulate a slow service). This time will be added to a) the execution time, which is calculated from the time that the server finishes receiving the request to the time that the sending of the response begins and b) any additional time specified via performance profiles, which are described in Working with Performance Profiles. If a data source is available, you may also parameterize this value.

Style/Use

These options allow you to select the body style and encoding of the message:

- **Body Style:** Enable either **document** or **rpc**.
- **Use:** Enable either **encoded** or **literal**.
- **Encoding Style URI:** (Automatically completed if available) Lists the encoding style URI used to send requests.
- **Target Object URI:** Specifies the target object URI.

Attachment Encapsulation Format

Specifies whether to use the **Default** or **Custom** encapsulation format. The **Default** option specifies whatever is chosen as the Attachment Encapsulation Format in the Preferences panel. The **Custom** option allows you to choose **MIME**, **DIME**, **MTOM Always**, or **MTOM Optional**. For details, see Working with Attachments.

SOAP Version

Specifies whether SOAP 1.1 or 1.2 is used.

ISO 8583 Packager

Allows you to configure packager files for ISO 8583. See Configure an ISO 8583 Message Packager for details. *This option is designed specifically for SOAP and Plain XML Message Responders*

Outgoing Message Encoding

These options enable you to specify a character encoding for the outgoing message that's different than the request message encoding.

Enable the **Use different encoding from request** option to activate the Encoding options. When the Encoding options are active, you can choose **Custom** from the drop-down menu and choose an encoding option from the the drop-down menu. You can also choose **Default** to use the encoding set in the responder suite.

Conversion Options Tab

The **Options** tab allows you to configure how the message is processed. These options are available only for message formats that are converted to XML, and the available options vary from format to format. For details on a specific tool's conversion options, see the relevant topic:

- Fixed Length Message Responder
- EDI Message Responder
- Custom Message Responder
- Lines Message Responders
- CSV Message Responders
- Plain Text Message Responders

Using Existing Data Sources or Rapidly Creating Data Sources for Responses

Specifying response values in a data source is a very efficient way to add a significant volume of request/response pairs.

Using Existing Data Sources

If you already have a data source that specifies values for request parameters and the desired corresponding response parameters, you can use those values as follows:

1. Add the data source (see Understanding How Virtualize Uses Data Sources).
2. Configure the Message Responder's data source mapping with the appropriate request columns (as described in Data Source Correlation Tab).

Automatically Generating a Data Source for the Response XML

If you do not already have such a data source but want a fast way to specify multiple request/response sets:

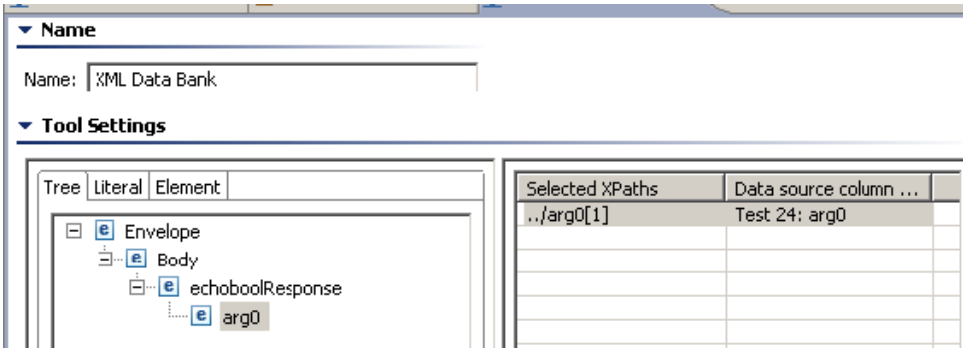
1. Use the procedure described in Generating a Data Source Template for Populating Message Elements to create a CSV file from the Form Input view of the Message Responder tool. A data source will be generated and added to the responder suite. This generated data source will contain columns for responses. The Message Responder tool's Form Input view will be parameterized automatically.
2. Add a new data source column (e.g., using Excel, OpenOffice or a similar spreadsheet application) for each request parameter that should be used to determine the response. Wildcards can be used.
3. Configure the Message Responder's data source mapping with the new request columns you added. See Data Source Correlation Tab for details.
4. Add new rows to the data source (e.g., using Excel, OpenOffice or a similar spreadsheet application) in order to specify values for request parameters and the desired corresponding response parameters.

Scripting Values Based on Incoming Requests

You can script response values based on the incoming request. This allows for more complex logic for your virtual assets. In addition, the Message Responder allows you to access data source values through the script. Access to these values is similar to how you would access them through the Extension Tool.

To use scripted logic within the Message Responder tool:

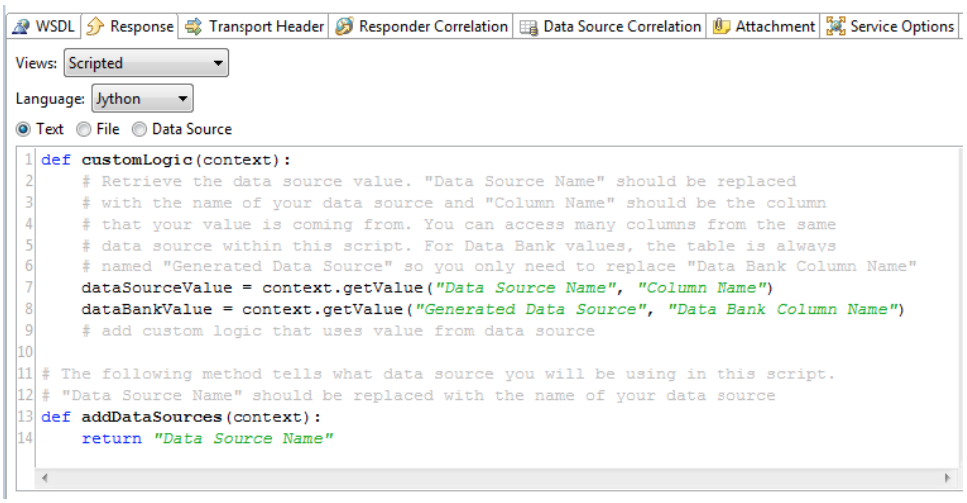
1. Enter a WSDL in the **WSDL URL** field of the Message Responder tool.
2. Right-click the **Message Responder** node and choose **Add Output**. The **Add Output** wizard displays.
3. From the Add Output wizard, select **Incoming Request** from the left pane and **XML Data Bank** from the right pane and click **Finish**.
4. Double-click the **Incoming Request> XML Data Bank** node beneath the Message Responder node. The XML Data Bank configuration panel displays in the right GUI panel.
5. In the right GUI panel, add the XPath of the value you want to access in your script.



6. Double-click the **Message Responder** node. The Message Responder configuration panel displays in the right GUI panel.
7. In the **Response** tab, choose the **Scripted** view.
8. Specify your logic. The following is the basic template for accessing data source and data bank values:

```
def customLogic(context):
    # Retrieve the data source value. "Data Source Name" should be replaced
    # with the name of your data source and "Column Name" should be the column
    # that your value is coming from. You can access many columns from the same
    # data source within this script. For Data Bank values, the table is always
    # named "Generated Data Source" so you only need to replace "Data Bank Column Name"
    dataSourceValue = context.getValue("Data Source Name", "Column Name")
    dataBankValue = context.getValue("Generated Data Source", "Data Bank Column Name")
    # add custom logic that uses value from data source

# The following method tells what data source you will be using in this script.
# "Data Source Name" should be replaced with the name of your data source
def addDataSources(context):
    return "Data Source Name"
```



9. Select the correct method from the **Method** combo box. The method you select should be your entry point. In the above example, the method would be **customLogic()**.

Adding Attachment Handlers to the Message Responder

If desired, you can add an Attachment Handler to the Message Responder to manage all of the MIME attachments that are received. To add an Attachment Handler to a Message Responder:

1. Select the Message Responder node and click the **Add responder or output** button. The **Add Output** wizard displays.
2. From the Add Output wizard, select **Incoming Attachment** from the left pane and **Attachment Handler** from the right pane and click **Finish**.
3. Double-click the **AttachmentHandler** node, then configure the tool in the tool configuration panel.
4. (Optional) Select the **Attachment Handler** node beneath the Message Responder node and click the **Add Responder or Output** button. The **Add Output** wizard displays from which you can add a Write File tool to write the attachment out as a binary file.

Forwarding Messages to Another Endpoint

The Message Forward tool can be attached to a Message Responder tool in order to forward messages to another endpoint (e.g., the actual resource). The response returned from the endpoint will be used as the response returned by the Message Responder.

For details, see Message Forward.

Updating Virtual Database Values

The SQL Responder Action tool enables you to have a Message Responder update values in a virtual database that is represented by a .CSV-driven SQL Responder.

For details, see SQL Responder Action.

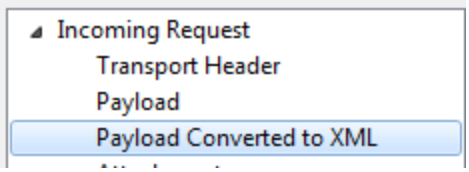
Chaining Tools to the Responder's Incoming Request or Outgoing Response

You can chain tools to a Message Responder's incoming request or outgoing response as follows:

1. Right-click the Virtual Asset Explorer node to which you want to add this tool.
2. Choose **Add Output**.
3. Select the desired output type on the left and the desired tool on the right.

Payload Converted to XML Option

If you are using a custom responder or EDI / CSV Fixed Length Responder, you will have the option of attaching a tool the payload converted to XML (incoming request) or they payload modeled as XML (outgoing response). This is in addition to the standard payload options, which add the tool to the payload in its native format.



Execution Order

Chained tools will be executed in the following order:

1. Incoming Request
    a. Transport Header
    b. Payload
    c. Payload Converted to XML
    d. Attachment
2. Outgoing Response
    a. Transport Header
    b. Payload Modeled as XML
    c. Payload
3. Both > Traffic Object
4. Forward > Traffic

Examples

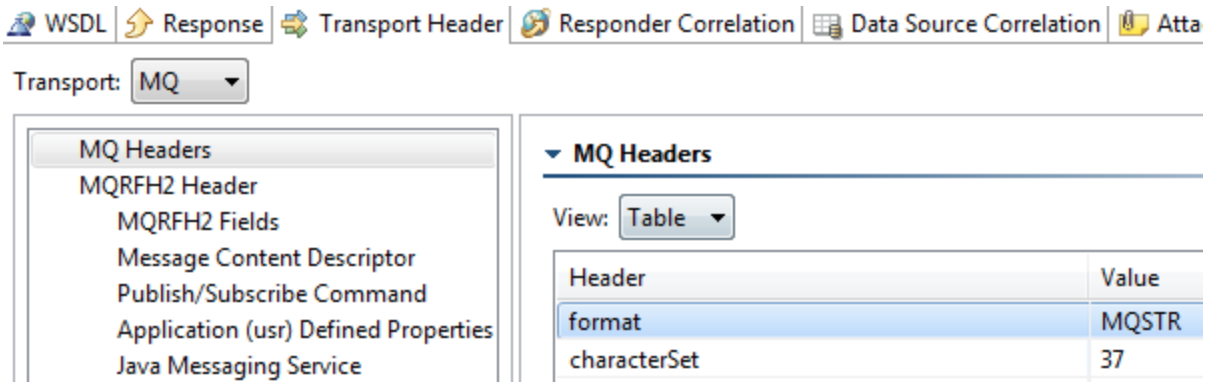
For example, you might want to chain tools for the following purposes:

To achieve this...	Do this....
A responder triggers a database update action For example: update /remote/add a row to a relational database or run a SELECT query	Chain a DB Tool to the Incoming Request of a Message Responder. You may also chain an XML Data Bank to the XML output of the DB tool and reference these data bank values to in the Responder output in order to have them populate the response with values obtained from a relational database.
Database data constructs a response message manually For example: iterate over returned records to generate a desired XML response format	Chain the DB tool to the outgoing response of a Message Responder and attach an XML Transformer, an XSLT Tool or an Extension tool to it. The output of these tools will replace the content configured in the Message Responder editor's Response tab. This provides flexibility to construct the message manually. If you want to use only certain select values form the database, chain the DB Tool to the Incoming Request as described above.
A responder invokes another service	Chain a SOAP Client or Messaging Client to the Incoming Request of the Message Responder. Chaining a SOAP or Messaging Client to the Outgoing Response of a Message Responder is not supported.
A responder invokes another service and uses some of its output to populate response parameters	Chain a SOAP/Messaging Client to the Incoming Request of the Message Responder and attach a Data Bank to the response of the client. These data bank values can be referenced within the Message Responder Form Input or Form XML views to construct a dynamic response message. If you need to invoke that service with parameters extracted from the request received by the responder, then chain an XML Data Bank to the Incoming Request of the Responder, then chain a SOAP/Messaging Client to the Incoming Request of the responder and parameterize its request with the data bank values you've extracted from the request. Note that order matters here: the data bank needs to be placed before the client in order for the client to be able to use its values.

MQ Character Set Handling

Virtualize sets the outgoing (response) MQ message characterSet field to the same value as the incoming (request) MQ message.

When sending responses with character data such as XML, CSV, fixed-length, or plain text, the "format", the MQ header type must be set to the value of the MQFMT_STRING constant, which is "MQSTR". This format header should be configured in the Message Responder's **Transport Header** tab (in the MQ Headers table, add an entry with `format` set to `MQSTR`).



The response message will be encoded based on the MQ "characterSet" header field. By default, Virtualize automatically sets the characterSet field on the outgoing (response) MQ message to the same value as the incoming (request) MQ message. However, this character set can also be explicitly configured by adding a "characterSet" header in the Message Responder's **Transport Header** tab.

The value can be one of the supported coded character set number (CCSID) values:

- 850 - ASCII
- 819 - ISO standard ASCII
- 37 - American EBCDIC
- 1200 - Unicode
- 1208 - UTF-8