REST Client

This topic explains how to configure and apply the REST Client tool, which sends messages to RESTful services. Messages can be sent with HTTP GET, POST, OPTIONS, HEAD, PUT, DELETE, TRACE, or custom methods. You can use REST Client tools in SOAtest and Virtualize.

Sections include:

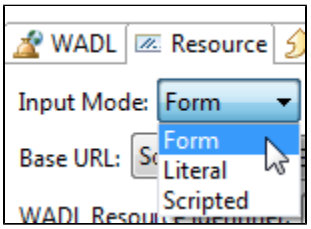
REST Client Migration (from 9.7.x and earlier)

The REST Client tool was redesigned in version 9.8 to facilitate the configuration process and provide additional flexibility.

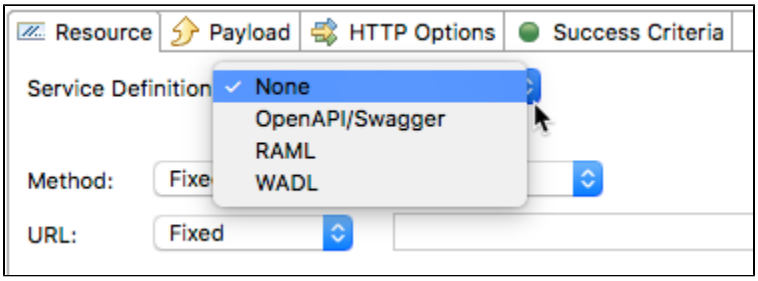
REST Client tools created in versions 9.7.x and earlier will be automatically updated to use the new format when you open and save the associated .tst or .pvn

in the current version of the product.

During the automated migration process, settings from legacy input modes are mapped to the new input modes. The legacy REST Client had Form, Literal, and Scripted input modes.



The current REST Client has constrained (OpenAPI/Swagger, RAML, WADL) and unconstrained service definition modes.



Migration is performed as follows:

- If the legacy tool used Literal mode, the new tool will use the unconstrained mode and the URL field will be populated from the existing Literal settings.
- If the legacy tool used Scripted mode, the new tool will use the unconstrained mode and the URL field will be set to Scripted. The existing script will be applied.
- If the legacy tool used Form Input for Payload and had the content-type set to application/x-www-form-urlencoded or multipart/form-data, the new tool's Payload tab will be set to Table view.
- If the legacy tool used Form Mode and was constrained to a WADL (a WADL URL was set and constrain was enabled), the new tool will use WADL mode and retain the existing WADL URL.
- If the legacy tool used Form Mode but was not constrained to a WADL (or had no WADL URL), the new tool will use the unconstrained mode. The unconstrained mode settings will be defined based on settings from the legacy Form view—*unless the Base URL is scripted or Query is completely scripted.*

Exact mappings are not available in the following cases:

- If the legacy tool was not constrained to a WADL and used the Scripted option for the Base URL, the unconstrained mode is used and the scripted settings are not transferred.
- If the legacy tool was not constrained to a WADL and used the Scripted option for the Query tab in Form input mode, the unconstrained mode is used and the scripted settings are not transferred.
- If the legacy tool used a WADL, was not constrained to that WADL, and used the Auto setting in any parameter (path, query, or payload), the Auto setting will not be transferred.

When a tool with these unsupported mappings is migrated, details will be printed to the console.

When a legacy tool with unsupported mappings is executed, the tool will fail with a Quality Task outlining the mapping issues.

Resource Tab

In the **Resource** tab, you specify the resource to which you want to send messages. The default service definition mode (unconstrained) allows you to specify a URL and query for accessing the service. If a service definition (OpenAPI/Swagger, RAML, WADL) is available, you can change to one of the constrained configuration modes.

Most of the available controls will vary depending on what you have set for the Service Definition field:

- **None:** Described in Unconstrained Configuration.
- **OpenAPI/Swagger:** Described in OpenAPI/Swagger Configuration.
- **RAML:** Described in RAML Configuration.
- **WADL:** Described in WADL Configuration.

Parameters can be configured in the Path and Query tabs, as described in Parameter Configuration.

Unconstrained Configuration

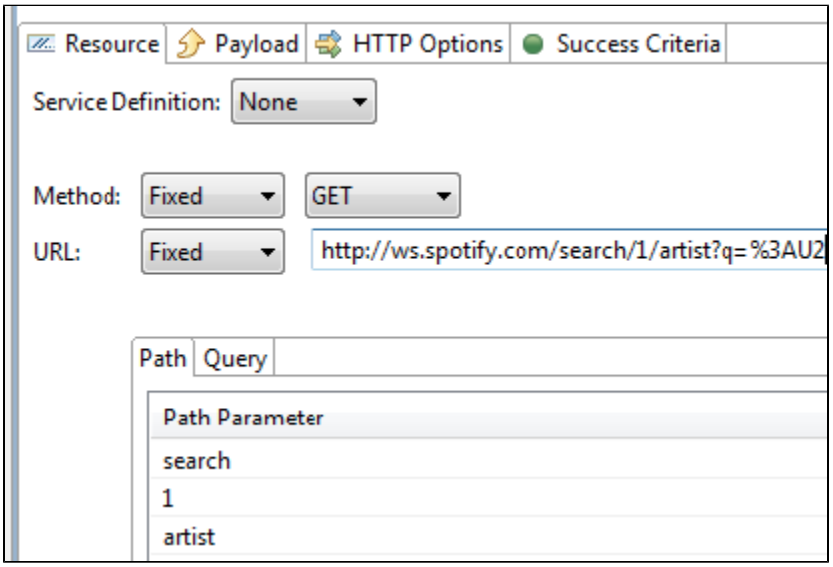
Unconstrained mode (the mode enabled when **Service Definition** is set to **None**) lets you specify the URL as a literal string, which can be either a fixed value, parameterized value, or scripted value. In addition, the method to invoke can also be specified as a fixed value, parameterized value, or scripted value. For details about

For details about parameterizing values, see the following chapters:

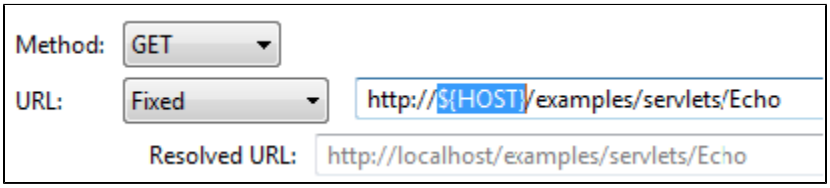
- Parameterizing Tests with Data Sources, Variables, or Values from Other Tests
- Parameterizing Tools with Data Source Values, Variables, and Extracted Values

For details about scripting values, see Extensibility and Scripting Basics.

With fixed values, you can access data source values using `${var_name}` syntax. You can also use the environment variables that you have specified. For details about environments, see Configuring Testing in Different Environments or Configuring Virtualize Environments.



If the URL uses environment variables, the **Resolved URL** field will display how they resolve into an actual URL.

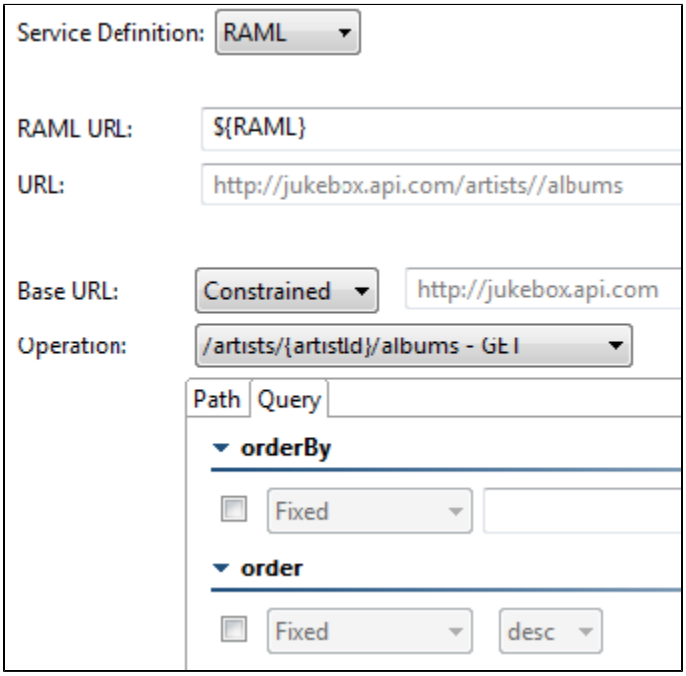


Parameters can be configured in the Path and Query tabs, as described in Parameter Configuration. Any changes made in the URL will automatically be propagated to the Path/Query table. Also, the URL will automatically be updated to reflect any changes made to the Path/Query table.

RAML Configuration

RAML configuration mode lets you specify the various URL components as follows:

- **RAML URL:** A specific RAML URL, or a variable referencing a RAML URL. The value entered here will be used to populate the Base URL and Operation controls.
- **URL:** A preview of the complete URL for accessing the service; it is constructed from the Base URL, Operation, and the values entered in the Path / Query tabs. Note that this field shows the full constructed URL, but does not resolve variables. See **Resolved URL** to view resolved variables.
- **Resolved URL:** The resolved RAML URL. If you are configuring the tool using environment variables, this will show you how they resolve into an actual URL. For example, assuming that HOST and PORT are environment variables, and `var_name` is a data source column, a URL of `http://${HOST}:${PORT}/some/url/${var_name}` might result in a Resolved URL of `http://localhost:8080/some/url/${var_name}`
- **Base URL:** Specifies the protocol to use (HTTP or HTTPS), host, port, and path.
- **Operation:** When a RAML definition is configured, you can select an operation and HTTP Method from this box. The Form Input views for the Path/Query table as well as the Payload tab will update depending on the selected operation and HTTP Method.



You can set the **Base URL** field to use constrained (from the specified RAML URL), fixed (modifiable), or scripted inputs.

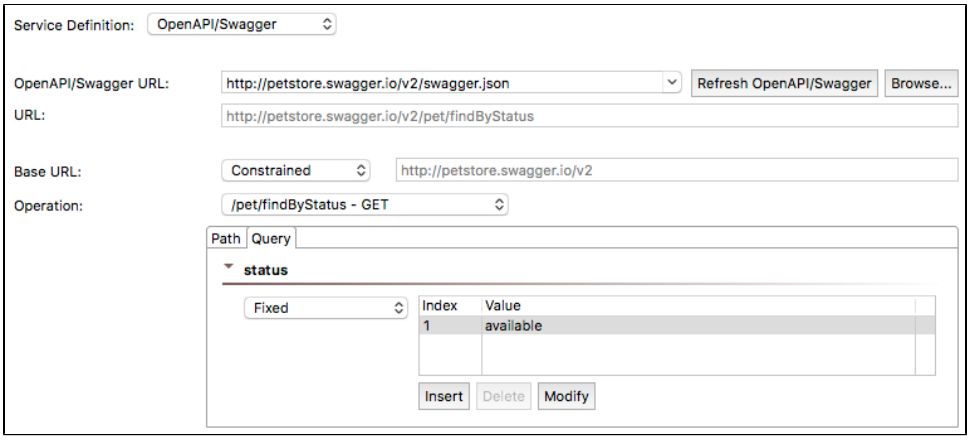
- For details about scripting values, see Extensibility and Scripting Basics.
- With fixed values, you can access data source values using `${var_name}` syntax. You can also use the environment variables that you have specified. For details about environments, see Configuring Testing in Different Environments or Configuring Virtualize Environments.

Parameters can be configured in the Path and Query tabs, as described in Parameter Configuration. The URL will automatically be updated to reflect any changes made to the Path/Query table.

OpenAPI/Swagger Configuration

OpenAPI/Swagger configuration mode lets you specify the various URL components as follows:

- **OpenAPI/Swagger URL:** A specific OpenAPI/Swagger URL or a variable referencing a OpenAPI/Swagger URL. The value entered here will be used to populate the Base URL and Operation controls.
- **URL:** A preview of the complete URL for accessing the service; it is constructed from the Base URL, Operation, and the values entered in the Path / Query tabs. Note that this field shows the full constructed URL, but does not resolve variables. See **Resolved URL** to view resolved variables.
- **Resolved URL:** The resolved OpenAPI/Swagger URL. If you are configuring the tool using environment variables, this will show you how they resolve into an actual URL. For example, assuming that HOST and PORT are environment variables, and `var_name` is a data source column, a URL of `http://${HOST}:${PORT}/some/url/${var_name}` might result in a Resolved URL of `http://localhost:8080/some/url/${var_name}`
- **Base URL:** Specifies the protocol to use (HTTP or HTTPS), host, port, and path.
- **Operation:** When a OpenAPI/Swagger definition is configured, you can select an operation and HTTP Method from this box. The Form Input views for the Path/Query table as well as the Payload tab will update depending on the selected operation and HTTP Method.



You can set the **Base URL** field to use constrained (from the specified OpenAPI/Swagger URL), fixed (modifiable), or scripted inputs.

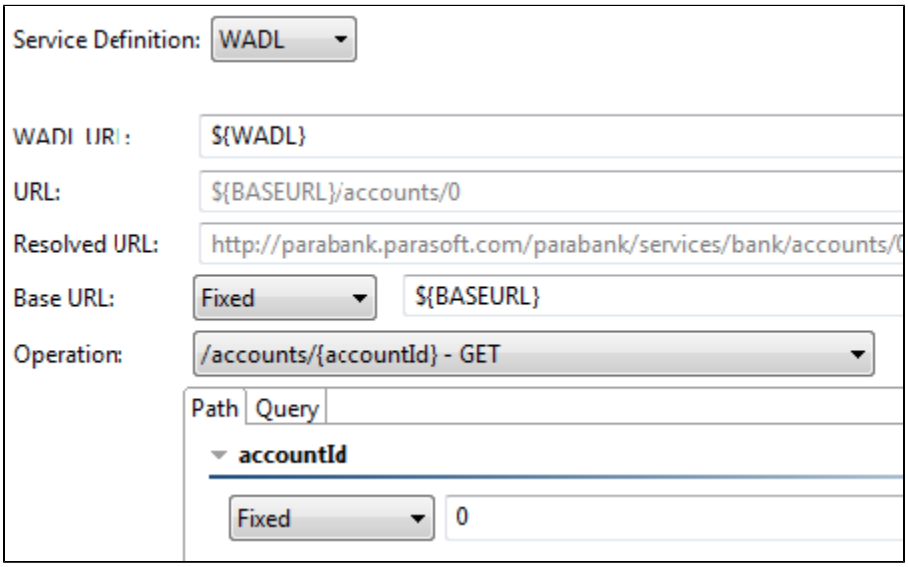
- For details about scripting values, see Extensibility and Scripting Basics.
- With fixed values, you can access data source values using `${var_name}` syntax. You can also use the environment variables that you have specified. For details about environments, see Configuring Testing in Different Environments or Configuring Virtualize Environments.

Parameters can be configured in the Path and Query tabs, as described in Parameter Configuration. The URL will automatically be updated to reflect any changes made to the Path/Query table.

WADL Configuration

WADL configuration mode lets you specify the various URL components as follows:

- **WADL URL:** A specific WADL URL, or a variable referencing a WADL URL. The value entered here will be used to populate the Base URL and Operation controls.
- **URL:** A preview of the complete URL for accessing the service; it is constructed from the Base URL, Operation, and the values entered in the Path / Query tabs. Note that this field shows the full constructed URL, but does not resolve variables. See **Resolved URL** to view resolved variables.
- **Resolved URL:** The resolved WADL URL. If you are configuring the tool using environment variables, this will show you how they resolve into an actual URL. For example, assuming that HOST and PORT are environment variables, and `var_name` is a data source column, a URL of `http://${HOST}:${PORT}/some/url/${var_name}` might result in a Resolved URL of `http://localhost:8080/some/url/${var_name}`
- **Base URL:** Specifies the protocol to use (HTTP or HTTPS), host, port, and path.
- **Operation:** When a WADL is configured, you can select an operation and HTTP Method from this box. The Form Input views for the Path/Query table as well as the Payload tab will update depending on the selected operation and HTTP Method.



You can set the **Base URL** field to use constrained (from the specified OpenAPI/Swagger URL), fixed (modifiable), or scripted inputs.

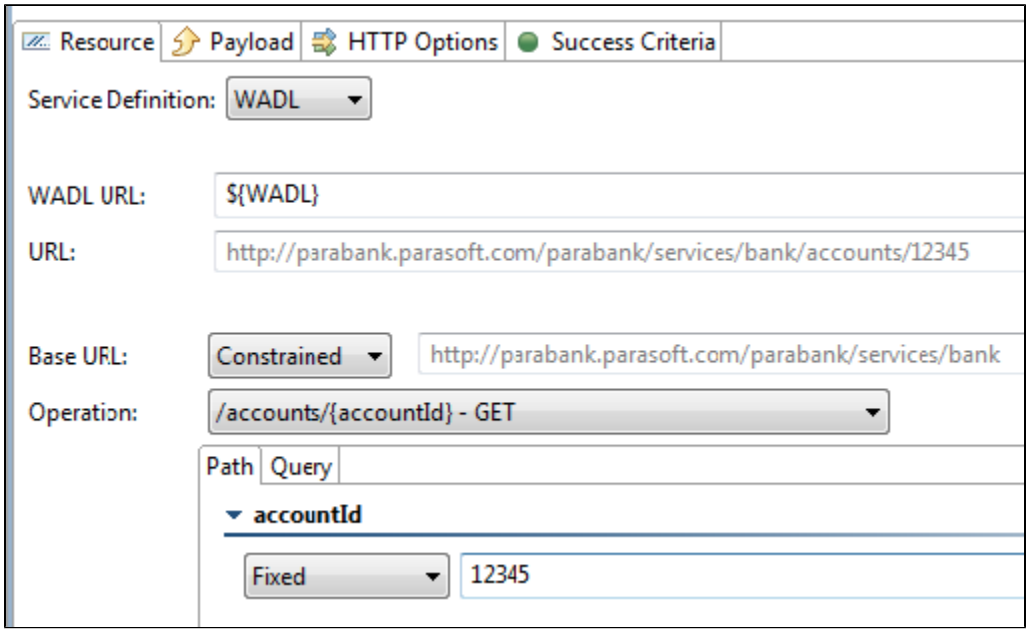
- For details about scripting values, see Extensibility and Scripting Basics.
- With fixed values, you can access data source values using `${var_name}` syntax. You can also use the environment variables that you have specified. For details about environments, see Configuring Testing in Different Environments or Configuring Virtualize Environments.

Parameters can be configured in the Path and Query tabs, as described in Parameter Configuration. The URL will automatically be updated to reflect any changes made to the Path/Query table.

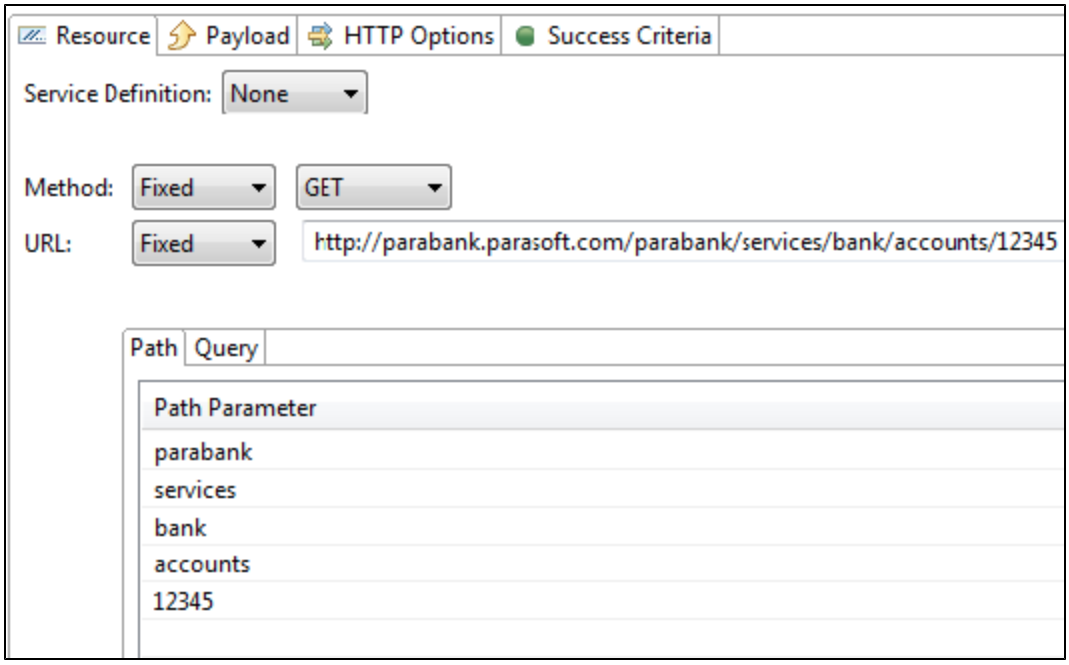
Switching Between Views

If you switch from a constrained mode (**Service Definition** set to **RAML**, **OpenAPI/Swagger**, or **WADL**) to unconstrained mode (**Service Definition** set to **None**), you will have the option of automatically populating the unconstrained view with values from the constrained view.

For example, assume you had the following in WADL view:



If you chose to populate the unconstrained view from the WADL one, you would get the following:



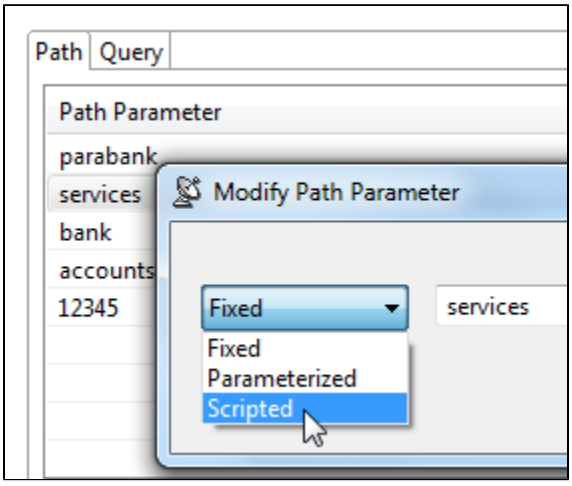
Parameter Configuration

In all modes (constrained and unconstrained), parameters can be configured in the Path and Query tabs.

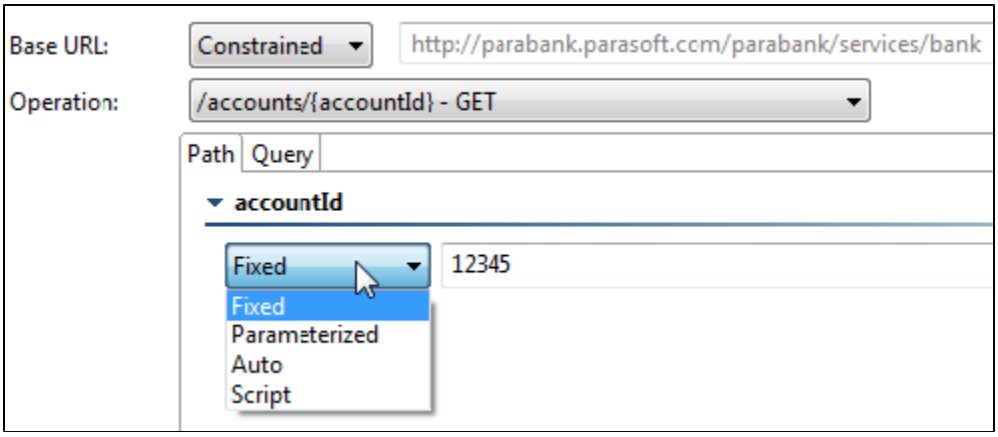
Template Parameters

The **Paths** tab lets you configure template parameters for the currently-selected operation. For example, a path of "/parabank/services/bank/accounts/{accountId}" has a single path parameter: "accountId".

In the unconstrained mode, parameters can be fixed, parameterized, or scripted.



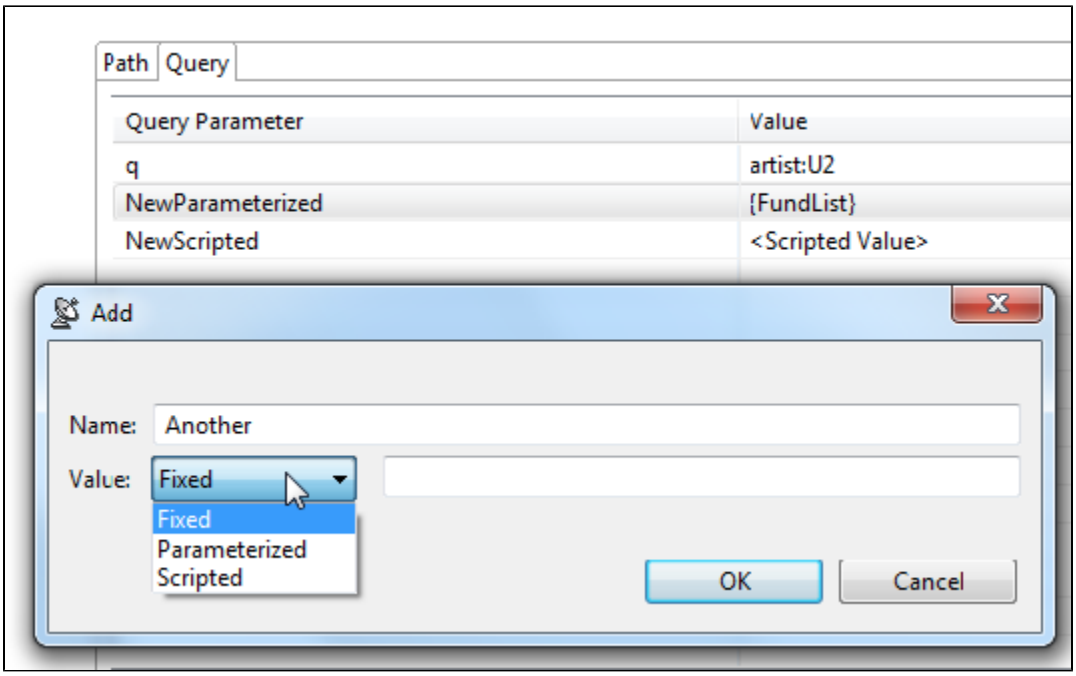
In the constrained modes, parameters can be set to fixed, parameterized, automatically-generated, or scripted values.



Query Parameters

The **Query** tab lets you configure the URL query parameters for the currently-selected operation. You can add fixed, parameterized, or scripted values.

In unconstrained mode, parameters can be fixed, parameterized, or scripted.

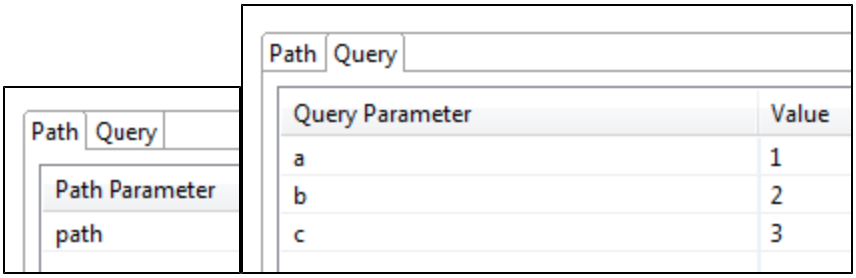


In the constrained modes, parameters can be set to fixed, parameterized, automatically-generated, or scripted values.

Encoding Note

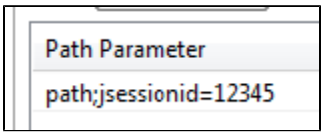
URL query parameters are formatted according to the "application/x-www-form-urlencoded" content type. Space characters are replaced with '+'. Non alpha numeric characters are replaced with a percent sign followed by two hexadecimal digits representing the character code. Names and values are separated by '=' and name-value pairs are separated by '&'.

If you want to use a different format, query parameters can also be specified directly at the end of the tool's endpoint URL (instead of in the Query Parameters section). For example, the following could also be specified as `http://host:8080/path?a=1&b=2&c=3`



Matrix Parameters

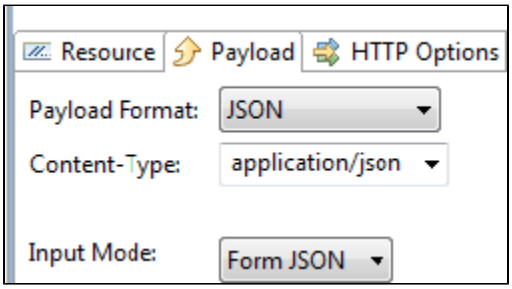
Matrix parameters can be configured by appending them directly to the end of the last path segment in the resource URL. For example, the following parameter could also be specified as `http://host:8080/path;jsessionid=12345`



Payload Tab

If the method you are using sends data (e.g., PUT, POST, DELETE), the **Payload** tab allows you to specify the payload for the message that will be sent.

Before you specify the payload itself, ensure that the appropriate payload format and media type are selected (in the **Payload Format** and **Content-Type** boxes). If the service definition is set to RAML, OpenAPI/Swagger, or WADL in the Resource tab, the payload format and content type here will be constrained to that URL and the selected resources.



These two settings control the Content-Type HTTP header and determine the available input modes. For example, the Form JSON view will only be available for JSON media types; the Form XML view will only be available for XML media types. If the media type is "application/x-www-form-urlencoded" and the service definition defines representation parameters for the currently-selected operation, then a Form Input view will be available to configure the parameters. This is similar to the Form Input used on the Path and Query tabs.

When specifying the payload, you can select input modes from the **Input Mode** drop-down list. The REST Client tool shares **Input Mode** options with other client tools. For more information on these shared options, see Message Tool and Responder Options.

For XML or JSON associated with a schema, the applicable form view will be automatically populated according to the values in the definition and editing will be restricted to ensure that the message complies with the specified schema. For instance, you will not be able to insert, delete, rename, copy, or paste tree nodes.

HTTP Options Tab

The HTTP options allow you to determine which protocol (HTTP 1.0 or 1.1) is used to send the request, as well as various options related to the protocol (security, headers, cookies, etc.).

Select the appropriate protocol from the **Transport** drop-down list, then configure its properties, which are described in the following sections:

SOAtest

- HTTP 1.0
- HTTP 1.1

Virtualize

- HTTP 1.0
- HTTP 1.1

Success Criteria Tab

The following options are available in the **Success Criteria** tab of the REST Client tool:

- **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 (milliseconds):** Specifies the length of delay (in milliseconds) after which your FTP, telnet, or HTTP requests should time 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 if you want the tool to fail on the specified timeout.
    - **Pass the test only if a timeout occurred:** Select this option to have the tool pass if the specified timeout occurs (i.e. the test does not finish execution within the specified time).

Related Tutorials

The following tutorial lesson demonstrates how to use this tool:

- Testing RESTful Services