RESTful HTTP Service

Bridge 6.0.55.2Runtime 2016.3Builder 6.0.17.2This page describes how to build RESTful HTTP service with the E2E Bridge. For more information on the concepts of REST refer to the Wikipedia pages of Representational State Transfer.

As of Bridge version 6.0.55.2 implementation of native REST services is available and implementing RESTful HTTP is deprecated. Please refer to REST Service for information on the new approach.

Example File (Builder project Basic Modeling/Frontend):

	<your example path>\Basic Modeling\Frontend\uml\simpleRESTSupportManager.xml

On this Page:

Related Pages:

- Plain HTTP Service
- Representational State Transfer
- REST Service

Specifying the Resource Path

Each <<E2EHTTPPortType>> contains the following tagged values:

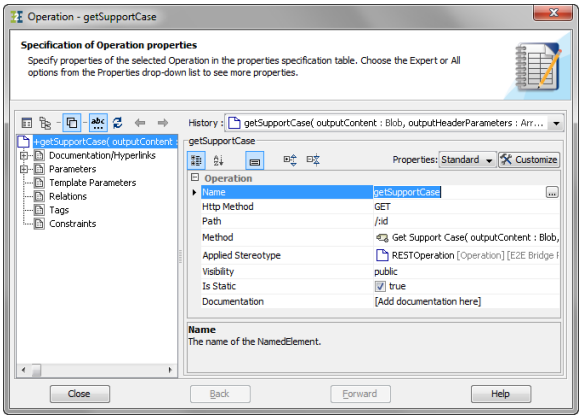
Tagged Value	Description
path	The tagged value path is used to define the service URL: `url = http://<hostname>:<port>/<path>`

In the example, the path is **/supportcases**. That is, the first part of the URL becomes **http://localhost:12007/supportcases**. (Whereas we assumed that the service runs on localhost.)

Be aware that the modeler is responsible to choose a value being unique within its xUML service.

The rest of the service URL derives from the <<RESTOperation>> within the <<E2EHTTPPortType>>.

Figure: <<RESTOperation>> getSupportCase



On the REST operation, you can specify a **path** parameter to match the REST resource path with that specific operation.

Tagged Value	Description	Values
path	Contains a path pattern. The overall path the REST operation is given by the path of the HTTP port plus this path. The path pattern may consist of literals (such as /test) and parameters (such as /:a).	Example: /test/:a/:b

In the example, the complete path to a specific support case is **http://localhost:12007/supportcases/<id of the support case>** and that link will call the operation **getSupportCase**. (Whereas we assumed that the service runs on localhost.)

Accessing Path Parameters in Activity Diagrams

As mentioned in HTTP Service, the path parameters are going into the REST operation via a parameter **pathParameters**, which is of type **Map**. You can get the parameter values using the `getMapValue()` operation, e.g.

```
set id = pathParameters.getMapValue("id");
```

Accessing the Service via cURL Calls

You can test a RESTful HTTP service via cURL calls. The example project **Basic Modeling/Frontend** contains test cURL calls as batch files in folder **cURL_test_calls**. You only have to provide the host the example service is running on. Just make sure, that you have access to a valid cURL installation from your command shell. cURL is also delivered with the E2E Bridge and you can find a valid installation in your MagicDraw installation folder in **plugins\ch.e2e.builder.plugin.magicdraw\bin**.

POST Data

For example, the batch script **create_support_case.bat localhost -v** will invoke the following cURL call:

```
curl -X POST --upload "post_data/post_support_request_12345.json" -G
"http://localhost:12007/supportcases" -v
```

The cURL `POST` needs a JSON input file containing the `POST` data. Option **`-v`** makes cURL logging more information on the HTTP input and output parameters.

This request results in the following answer:

```
* About to connect() to localhost port 12007 (#0)
*   Trying 127.0.0.1...
* connected
* Connected to localhost (127.0.0.1) port 12007 (#0)
> POST /supportcases HTTP/1.1
> User-Agent: curl/7.27.0
> Host: localhost:12007
> Accept: */*
> Content-Length: 121
> Expect: 100-continue
>
< HTTP/1.1 100 CONTINUE
* We are completely uploaded and fine
* HTTP 1.0, assume close after body
< HTTP/1.0 201 Created
< Server: E2E-Bridge/2013.2
< Pragma: no-cache
< Location: http://localhost:12007/supportcases
/00000002e99f3e0600000b1400001238960a4878
< Content-Length: 312
< Cache-control: no-cache
<
{"self":"http:\/\/localhost:12007\/supportcases\
/00000002e99f3e0600000b1400001238960a4878",
 "id":"00000002e99f3e0600000b1400001238960a4878",
 "customerID":"12345",
 "customerName":"Wishes unltd",
 "date":"2013-10-24T08:40:29.446264Z",
 "shortDescription":"This is a test support case.",
 "status":"in progress"}* Closing connection #0
```

GET Data

For example, the batch script **get_support_case.bat localhost 00000002e99f3e0600000b1400001238960a4878 -v** will retrieve the newly created support case as follows:

```
curl -X GET -G "http://localhost:12007/supportcases
/00000002e99f3e0600000b1400001238960a4878" -v
```

Option **-v** makes cURL logging more information on the HTTP input and output parameters.

This request results in the following answer:

```
* About to connect() to localhost port 12007 (#0)
*   Trying 127.0.0.1...
* connected
* Connected to localhost (127.0.0.1) port 12007 (#0)
> GET /supportcases/00000002e99f3e0600000b1400001238960a4878 HTTP/1.1
> User-Agent: curl/7.27.0
> Host: localhost:12007
> Accept: */*
>
* HTTP 1.0, assume close after body
< HTTP/1.0 200 OK
< Server: E2E-Bridge/2013.2
< Pragma: no-cache
< Content-Length: 312
< Cache-control: no-cache
<
{"self":"http:\/\/localhost:12007\/supportcases\
/00000002e99f3e0600000b1400001238960a4878",
 "id":"00000002e99f3e0600000b1400001238960a4878",
 "customerID":"12345",
 "customerName":"Wishes unltd",
 "date":"2013-10-24T08:40:29.446264Z",
 "shortDescription":"This is a test support case.",
 "status":"in progress"}* Closing connection #0
```

PUT Data

For example, the batch script **resolve_support_case.bat localhost 00000002e99f3e0600000b1400001238960a4878 -v** will update the corresponding support case as follows:

```
curl -X PUT -G "http://localhost:12007/supportcases
/00000002e99f3e0600000b1400001238960a4878/resolve" -v
```

Option **-v** makes cURL logging more information on the HTTP input and output parameters.

This request results in the following answer:

```
* About to connect() to localhost port 12007 (#0)
*   Trying 127.0.0.1...
* connected
* Connected to localhost (127.0.0.1) port 12007 (#0)
> PUT /supportcases/00000002e99f3e0600000b1400001238960a4878/resolve HTTP
/1.1
> User-Agent: curl/7.27.0
> Host: localhost:12007
> Accept: */*
>
* HTTP 1.0, assume close after body
< HTTP/1.0 200 OK
< Server: E2E-Bridge/2013.2
< Pragma: no-cache
< Content-Length: 25
< Cache-control: no-cache
<
Support case is resolved.* Closing connection #0
```