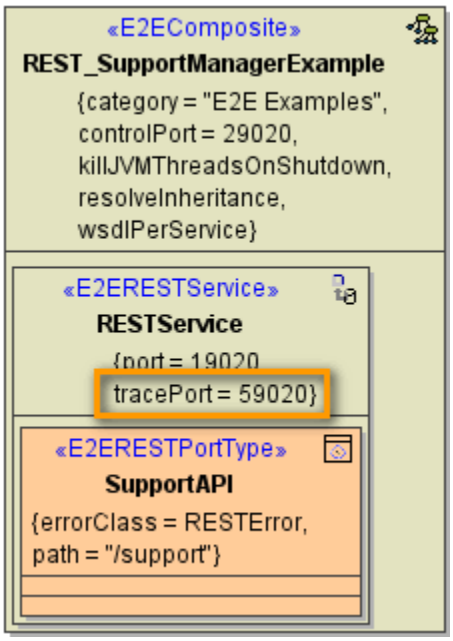
Testing REST Services

To test REST services, you can use the embedded SOAP Test Tool, the Analyzer, and the Bridge REST Test Tool.

On this Page:

Testing the REST Methods via Their SOAP Port

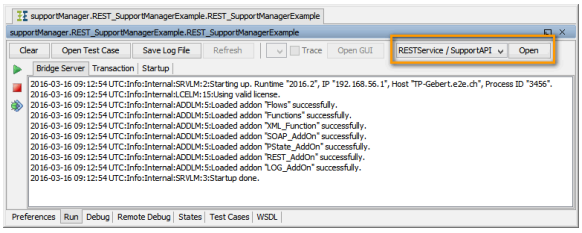
The Builder provides SOAP trace ports for each REST service.



Testing the REST Methods with the Embedded SOAP Test Tool

Run the REST service in the Embedded Runtime. You can test the implementation of the REST methods with the SOAP Test tool as described on Working with the Test Cases View pp. You also can use the Interactive Debugger to track down errors. How to use the Interactive Debugger is described on The Interactive xUML Debugger pp.

Additionally, you can access the Bridge REST Test Tool via the Builder. If the xUML service is running on the embedded Runtime, switch to **Run View**, select the REST port type from the list and click **Open**.



For a detailed description of the Bridge REST Test Tool refer to section Testing the REST Interface with the Bridge REST Test Tool further below.

Testing the REST Methods with the Analyzer

Opening the REST Builder project with the Analyzer, you can use the SOAP ports to test the implementation of the REST methods with the **Trace Analyzer** and the **Regression Test Tool** just the way you are used to (see also Analyzer User's Guide).

The REST example comes with predefined test cases and regression tests. You can just open them with the Analyzer and try them out.

Related Pages:

SOAP Test Tool

- Working with the Test Cases View
- The Interactive xUML Debugger

Analyzer

- Analyzer User's Guide

Bridge REST Test Tool

- OpenAPI 2.0 Specification
- REST Interface Documentation
- Optional Parameters
- Implementing REST Methods
- Error Handling

Example File (Builder project Add-ons/REST):

<your example path>\Add-ons\REST\uml\supportManager.xml

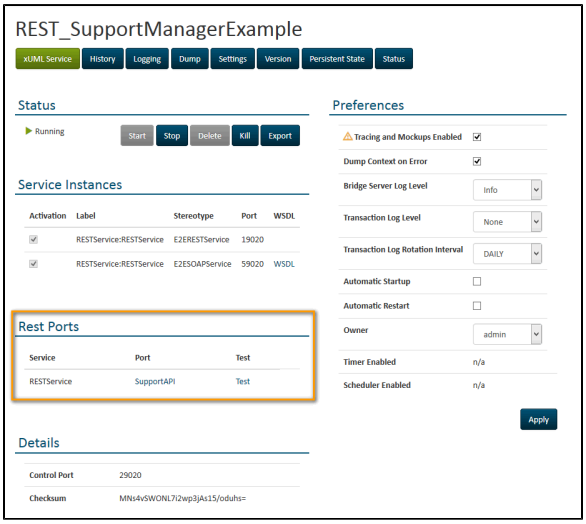
<your example path>\Add-ons\REST\uml\supportManager_auth.xml

Hint: There is a service setting regarding the support case id: **Generate unique support case ID on server (true) or client (false)**. Set it to false so the id given from the test cases will be used.
If you leave this setting to true, all tests will be red due to divergent ids.

Testing the REST Interface with the Bridge REST Test Tool

The Bridge as of above mentioned version provides an OpenAPI 2.0 Specification for documentation and testing purposes. Via a link on the xUML service page, you can access a REST service documentation page where you can inspect the service interface and make HTTP calls on the service. The information displayed on that page comes from the service descriptor associated to the service.

On the Bridge, go to the xUML service preferences page (see xUML Service Details for more information on how to do this).

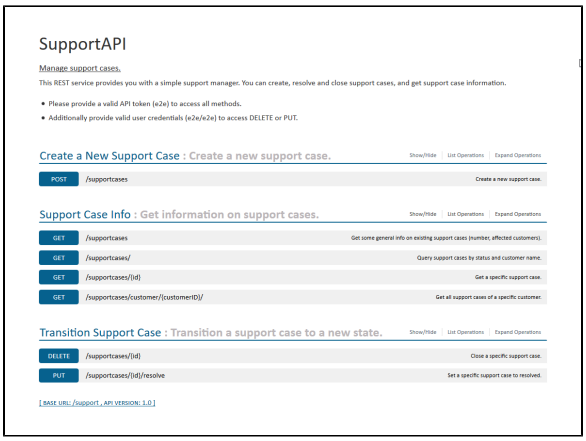


In section **Rest Ports**, you find a list of all REST service components and their port types. You can do the following here:

- Click the port type (e.g. **SupportAPI** in the figure above) to access the REST service documentation page.
- Click the **Test** link to access the same documentation page with additional test capabilities.

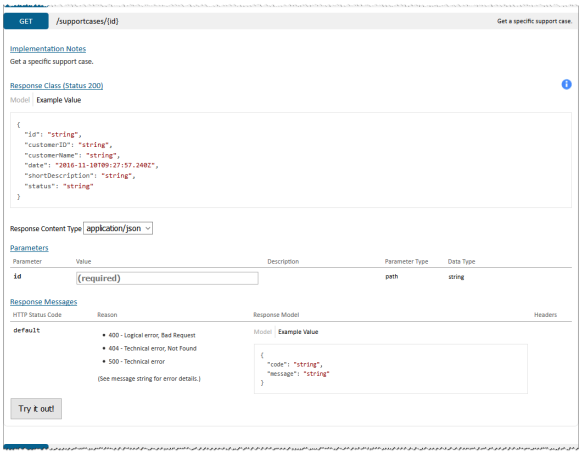
Clicking these links, you will get the following UI:

Figure: REST Service Documentation and Test



The REST service documentation shows the REST port type (**SupportAPI** in this case) and lists all REST methods the REST interface offers, and the resources they apply to. If provided, additional textual documentation coming from the documentation tags of model elements is displayed (see also REST Interface Documentation).

By clicking the method button, the method expands and you will see a documentation of the REST method:



The documentation shows

- some implementation notes (coming from the documentation tag of the REST method)
- the response class
- parameters
- response messages

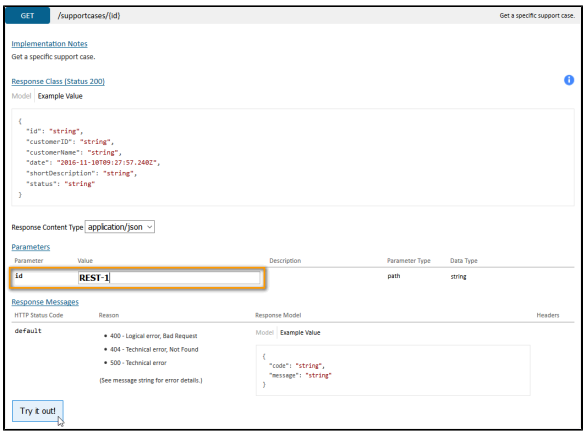
If you entered the documentation page by clicking **Test**, you can make a HTTP call on the method by clicking **Try it out!**.

Calling the Service

Expand the method you want to test by clicking the method button and provide parameter values in the **P arameters** section.

Providing Path or Query Parameters

For each path parameter of the REST method you get a field to enter a value. The field displays whether the parameter is mandatory or not. Path parameters are always mandatory and must be provided. Query or header parameters are mandatory as per default, but this setting can be changed using the multiplicity tagged value of the parameter (see Optional Parameters).

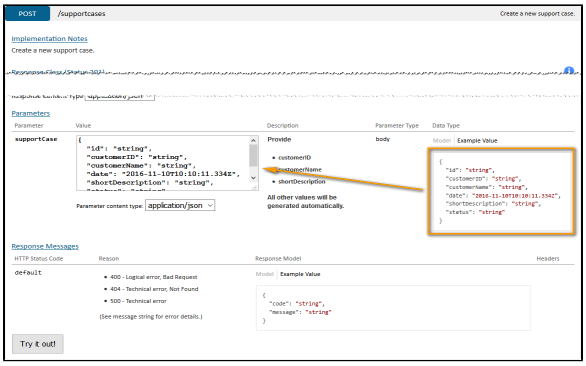


Now, click **Try it out!**.

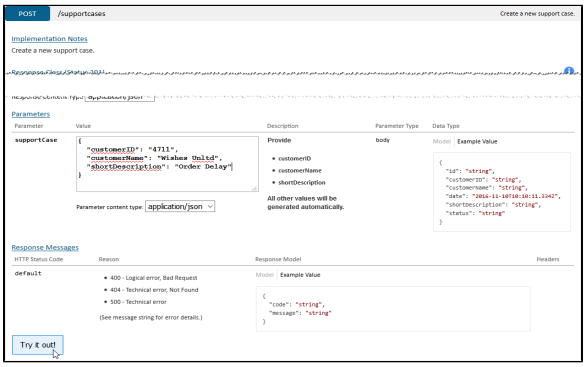
Providing Body Parameters

Parameters that are supplied via the request body have to be provided in the selected content type. As per default, this is JSON, but you can change the content type to XML. For more details on how the content type headers are implemented to the Bridge, refer to Calling REST Services.

To get a template to fill in for the parameter value, just click the **Model Schema** link in column **Data Type**.



Fill in all necessary or mandatory values. Delete lines you do not need.



Now, click **Try it out!**.

Reading the Response

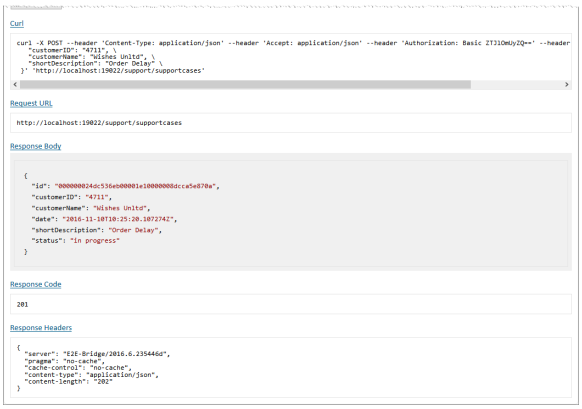
After having made the HTTP call by clicking **Try it out!**, the method's section is expanded by a response section. It shows

- a cURL call illustrating the HTTP call (this can easily be copied for own purposes)
- the request URL
- the response body in the selected response content type
- the HTTP response of the request
- the HTTP response headers

Example: A Successful Request

Find below an example of a successful POST request:

Figure: Sucessful POST Request

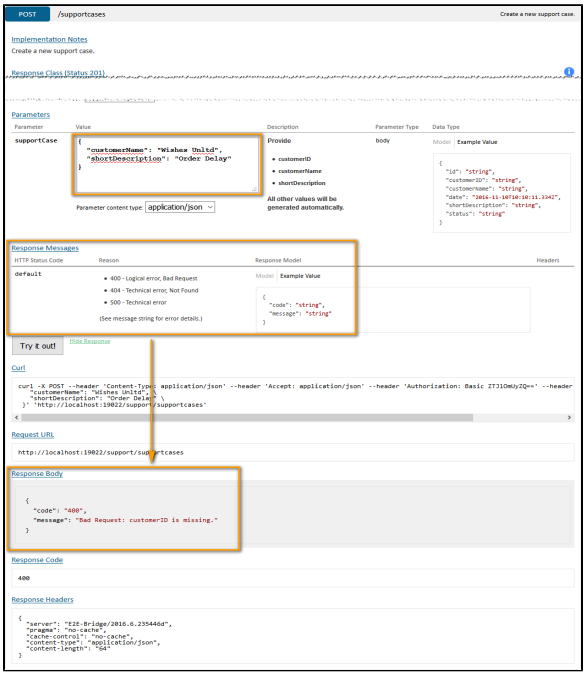


The response body contains the data of the newly created support request.

Example: An Erroneous Request

Try to post a support case without a customer ID.

Figure: Erroneous POST Request



The request comes back with HTTP response code 400. The response body does not contain a support case object but an error object stating "Bad request: customerID is missing.". For more information on implementing REST error handling see Error Handling.

The structure of the error object is documented on the REST service documentation page in section **Resp onse Messages**, line default. The error code documentation displayed in column **Reason** comes from the documentation tag of the REST error class (see also further above).