Testing REST Services

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

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



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).

tatus				Preferences		
Running	Start S	top Delete	Kill Export	A Tracing and Mockups Enabled	•	
				Dump Context on Error	•	
ervice l	nstances			Bridge Server Log Level	Info	~
Activation	Label	Stereotype	Port WSDL	Transaction Log Level	None	~
1	RESTService:RESTService	E2ERESTService	19020	Transaction Log Rotation Interval	DAILY	~
~	RESTService:RESTService	E2ESOAPService	59020 WSDL	Automatic Startup		
				Automatic Restart		
est Por	ts			Owner	admin	~
Service	Port		Test	Timer Enabled	n/a	
RESTService	supportA	рі	Test	Scheduler Enabled	n/a	
						Anni

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

Manage s	upport cases.				
This REST s	service provides you with a simple support manager. You can create, resolve and clo	se support cases, ar	id get supp	ort case inform	ation.
 Please ; Addition 	provide a valid API token (e2e) to access all methods. nally provide valid user credentials (e2e/e2e) to access DELETE or PUT.				
Create	a New Support Case : Create a new support case.		Show/Hide	List Operations	Expand Operation
POST	/supportcases			Crea	te a new support case
Suppo	rt Case Info : Get information on support cases.		Show/Nide	List Operations	Expand Operation
Suppo	rt Case Info : Get information on support cases.	Get some general info	Show/hide	List Operations	Expand Operation
Suppo GET GET	rt Case Info : Get information on support cases. /www.	Get some general info	Show/hide on existing su Query su	Ust Operations pport cases (numbe	Expand Operation 1, effected customers; 5 and customer name
Suppo GET GET GET	rt Case Info : Get information on support cases. /www. /www.www. /www.ase/M	Get some general info	Show/Mide on existing su Query su	List Operations opport cases (numbe opport cases by statu Get s	Expand Operation r, affected customers s and customer name specific support case
Suppo GET GET GET	rt Case Info : Get information on support cases. /wportawa /wportawa/ /wportawa//with /wportawa/wither/juntemet0/	Get some general info	Show/Hide on existing su Query su G	List Operations pport cases (numbe pport cases by statu Get a et all support cases	Expand Operation r, affected customers; s and customer name specific support case of a specific custome
Suppo GET GET GET Transit	rt Case Info : Get information on support cases. /www. /www. /www. /www. /www. /www. /www. /www. /www. /www. /www. /www. /www. /www. /www. /www. //ww	Set some general into	Show/Mide on existing su Query su G Show/Mide	Ust Operations pport cases (numbe opport cases by stata Get a et all support cases Ust Operations	Expand Operation r, affected customers; s and customer name specific support case of a specific custome Expand Operation
Suppo GET GET GET Transit	rt Case Info : Get information on support cases. //www. //www. //www. //www. //www. //www. //www. //www. //www. //www. //www. //www. //www. //www. //www.	Get some general into	Show/Vide on existing su Query su G Show/Vide	List Operations pport cases (numbe pport cases by statu Get a et all support cases List Operations Close a	Expand Operation r, affected customers; s and customer name specific support case of a specific customer Expand Operation specific support case

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:

A					
GET /sup	portcases/{id}			Get	a specific support case
Implementation N Get a specific suppo Response Class (St Model Example Va	otes rt case. atus 200) Iue				6
{ "id": "strin "customerNam "date": "201 "shortDescri "status": "at " Besponse Content Ty Parameters Parameters	r", "string", "-string", -:l:l:effection", string", string" string" (ppe[appkatbor/json ∨] Whee	Description	Parameter Type	Data Type	
id	(required)		path	string	
Response Message HTTP Status Code	85 Reason	Response Model			Headers
default Try it out!	409 - Lagical error, if ed Repuest 404 - Technical error (Net Found 509 - Technical error (Gee message string for error details.)	<pre>Model Example Value { "code": "string", "message": "string" }</pre>			

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).

GET /su	pportcases/(id)			Ge	et a specific support case
Implementation I	Notes				
Get a specific supp	ort case.				
Response Class (S	Status 200)				0
Model Example V	/alue				
{ "id": "strii" "customerDo" "customerDo" "date": "280 "shortbear" "status": "/ " Response Content" Parameters Parameter id	ng", ": stratag", ": "stratag", "Stratag", Statuto "stratag", Statuto "stratag", stratag" https://statuto.com/statuto.com/ https://statuto.com/ wate (EST-1	Description	Parameter Type path	Data Type string	
Response Messag	<u>Res</u>				
HTTP Status Code	Reason	Response Model			Headers
default	 400 - Logical error, Bad Request 	Model Example Value			
	404 - Technical error, Not Found				
	 500 - Technical error 	"code": "string",			
	(See message string for error details.)	"message": "string")			
Try it out!					

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.

POST /su	pportcases			Create a new support case.
Implementation Create a new supp or Reproposation of the support	Notes orf case. Spatian-2024			
Parameters	Value	Description	Parameter Type	Data Type
supportCase	("d": "string", 	Provide esstonentD instanenthme shortDescription All other values will be generated automatically.	body	Mode Earliet Value ("Lift": "netring", "Continents": "Lift", "Netring", "Description: "Lift", "Secretary, "Lift", "Secretary, "Lift", "Secretary, "Lift", "Netring"
HTTP Status Code	Reason	Response Model		Headers
default Try it out!	400 - Logical error, Rad Request 404 - Technical error, Not Pound 500 - Technical error (See message string for error details.)	<pre>Model Example Value { code": "String", "message": "string" }</pre>		

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

POST /sup	pportcases			Create a new support case.
Implementation N Create a new suppo Resnapse Filese/S	éolos ort cese. Hefylallala-manana ana anna an anna an anna an anna an an			
Parameters				
Parameter	Value	Description	Parameter Type	Data Type
supportCase	("customerID": "4711", "customerITENT": "Alass Unitd", "bactioectiption": "Order Belay") Parameter content type [appRatter/json ~]	Provide • customentilo • customentilame • shortDescription All other values will be generated automatically.	body	Model Example Value ("taft: "string", "ostonectro", "string", "ostonectro", "string", "shertbescription", "string", "shertbescription", "string", "string" }
Response Messag	203			
HTTP Status Code	Reason	Response Model		Headers
default	400 - Logical error, Biol Request 404 - Technical error, Not Found 500 - Technical error (See message soring for error details.)	Model Example Value ("code": "string", "message": "string")		
Try it out				

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

<u>4</u>
wi 3 400° -header Schener-Dyne sepilation/jon* -baader 'Accept application/jon* -baader 'Authorization: Bail IIIDab/20-4 -baade - scheme UP: -uity : heater - scheme UP: -uity : heater - scheme UP: -uity : Scheme UP: -uity : heater - scheme UP: -uity : Scheme UP: -uity : heater - scheme UP: -uity : Scheme UP: -uity : heater - scheme UP: -uity : Scheme UP: -uity : heater - scheme UP: -uity : Scheme UP: -uity : heater - scheme UP: -uity : Scheme UP: -uity : heater - scheme UP: -uity : Scheme UP: -uity : heater - scheme UP: -uity : Scheme UP: -uity : heater - scheme UP: -uity : Scheme UP: -uity : heater - scheme UP: -uity : scheme UP: -uity : heater - scheme UP: -uity : scheme UP: -uity : heater - scheme UP: -uity : scheme UP: -uity : heater - scheme UP: -uity : scheme UP: -uity : heater - scheme UP: -uity : scheme UP: -uity : heater - scheme UP: -uity : scheme UP: -uity : heater - scheme UP: -uity : scheme UP: -uity : heater - scheme UP: -uity : scheme
quest URL
ttp://localhost:19022/support/supportcases
iponse Body
<pre>("idt" "000000044534600000110000006ccs448704", "catter/Dir "4711", "catter/idt", "Alama Unit", "catter/idta-file/idta-Siz0.02747", "bottloscription", "Once Delay", "tatter/idta-file/idta-Delay", "bottloscription", "00000000000000000000000000000000000</pre>
iponse Code
01
sponse Headers
"server" 'ESL-B-lgs/285.6.235464', "praget" 'No-cable, "colocility" 'No-cable, "colocility" 'Server' 'spailcalled'son', "colocil: Server' 'Server', "Server', "Server'

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

POST /sup	pportcases			Create a new support case.
Implementation N Create a new suppo	Notes ort case.			
Response Class (S	رور د در	وي المح الحد في والج التان الإخرى وفي ورودو وو وال	, , , , , , , , , , , , , , , , , , ,	
	an a			
Parameters				
Parameter	Value	Description	Parameter Type	Data Type
supportcase	{ "customerHame": "Wishes Unitd", "shortDescription": "Order Delay" }	customerID customerName shortDescription All other values will be	66 0 9	<pre>Model Example Value { "id": "string", "outdowr20": "string", "outdowr20": "string", "outdowr30e": "string", "det": "sol: 11:301:11:3042", </pre>
	Parameter content type: application/json ~	generated automatically.		"shortDescription": "string", "status": "string")
Response Messag	Person	Removare Model		Maarlarr
default	NEBSUII	Nesponse model	-	neauers
	 400 - Logical error, Bad Request 	Model Example value		
	 404 - lecrinical error, Not Found 500 - Technical error 	C		
	· Sov - recrimical error	"code": "string", "message": "string"		
	(See message string for error details.)	2		
Try it out	Hide Basenne header, 'Content-Type: application/json'head header, 'Withes Unite', ' head	der 'Accept: mpplication/jso	n'header 'Autho	rization: Basic ZT310mUyZQ=='header
Request LIRI				
Request ORL				
http://localho	ost:19022/support/supprtcases			
Response Body {	100°, "Bad Request: customerID is missing."			
Response Code				
400				
Response Header	3			
<pre>{ "server": "E "pragma": "e "cache-contr "content-typ "content-ler }</pre>	ElE-Bridge/2016.6.235446d", no-cathe", no[1] "no-cathe", neEl" "Boplication/json", gth": "64"			

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).