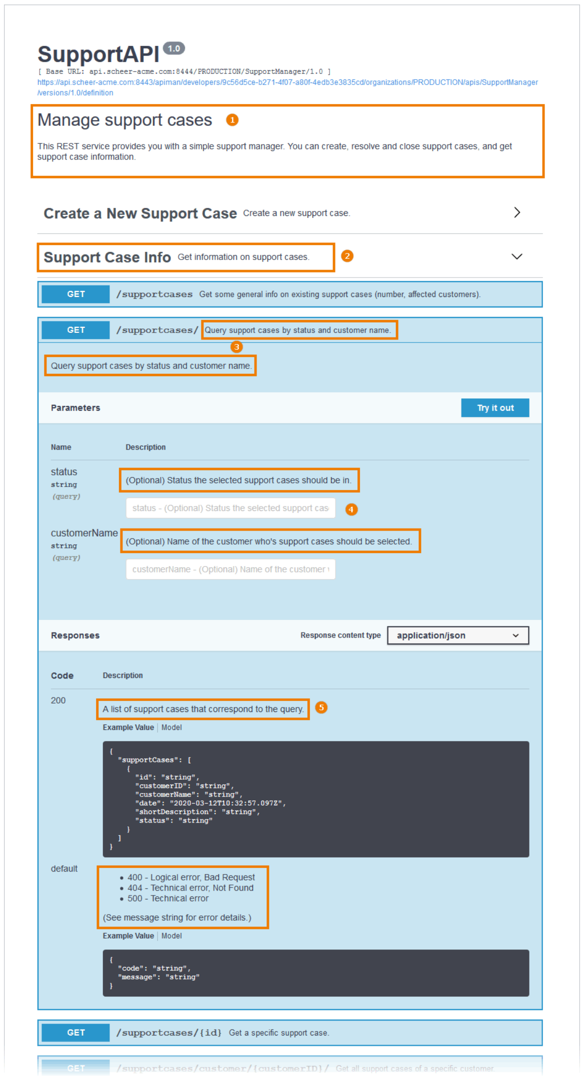
Documenting a REST Service

On this Page:

REST services can be tested with a REST Test Tool (see Testing REST Services), and if you are using API Management, your client's developers can access APIs via a Developer Portal (see Developer Access to APIs).
Especially in these cases it is important to provide an elaborated REST service documentation.

When compiling a REST service, an OpenAPI descriptor file is created that contains the service description. This file reflects the REST interface structure as implemented by you (see Defining a REST Service Interface).
The generated OpenAPI file also contains textual documentation. Tools supporting OpenAPI descriptor files (yaml format) can make the documentation visible:



SupportAPI 1.0

[Base URL: api.scheer-acme.com:8444/PRODUCTION/SupportManager/1.0]

https://api.scheer-acme.com:8443/apiman/developers/9c56d5ce-b271-4f07-a80f-4edb3e3835cd/organizations/PRODUCTION/apis/SupportManager/versions/1.0/definition

Manage support cases 1

This REST service provides you with a simple support manager. You can create, resolve and close support cases, and get support case information.

Create a New Support Case Create a new support case.

Support Case Info Get information on support cases. 2

GET /supportcases Get some general info on existing support cases (number, affected customers).

GET /supportcases/ Query support cases by status and customer name.

3

Query support cases by status and customer name.

Parameters

Name	Description
status string (query)	(Optional) Status the selected support cases should be in. status - (Optional) Status the selected support cas 4
customerName string (query)	(Optional) Name of the customer who's support cases should be selected. customerName - (Optional) Name of the customer w

Responses — Response content type: application/json

Code	Description
200	A list of support cases that correspond to the query. 5
default	• 400 - Logical error, Bad Request • 404 - Technical error, Not Found • 500 - Technical error (See message string for error details.)

Example Value | Model (200)

```
{
  "supportCases": [
    {
      "id": "string",
      "customerID": "string",
      "customerName": "string",
      "date": "2020-03-12T10:32:57.097Z",
      "shortDescription": "string",
      "status": "string"
    }
  ]
}
```

Example Value | Model (default)

```
{
  "code": "string",
  "message": "string"
}
```

GET /supportcases/{id} Get a specific support case.

GET /supportcases/customer/{customerID}/ Get all support cases of a specific customer.

The documentation from the model elements will be added to the **description** and **summary** tag of the service descriptor. We recommend to populate the following documentation in your REST service:

	Element	Description	OpenAPI File	xUML REST Service

1	**API Description**	Describe your API in general.	`info:`
			`  description: |-`
			`    ###Manage support cases.`
			`    This REST service provides`
			`you with a ...`
				<<E2ERESTPortType>>

2	**Operation Group**	Group operations with tags.	`tags:`
			`- description: Get information`
			`on support cases.`
			`  name: Support Case Info`
				<<RESTOperationTag>>

3	**Operation Description**	Provide a short description for each operation.	`get:`
			`  summary: Query support cases by status and ...`
			`  description: Query support cases by ...`
				<<REST>>

4	**Parameter Description**	Provide a short description for each parameter.	`parameters:`
			`- name: status`
			`  in: query`
			`  description: (_Optional_) Status the selected ...`
				<<RESTParameter>>

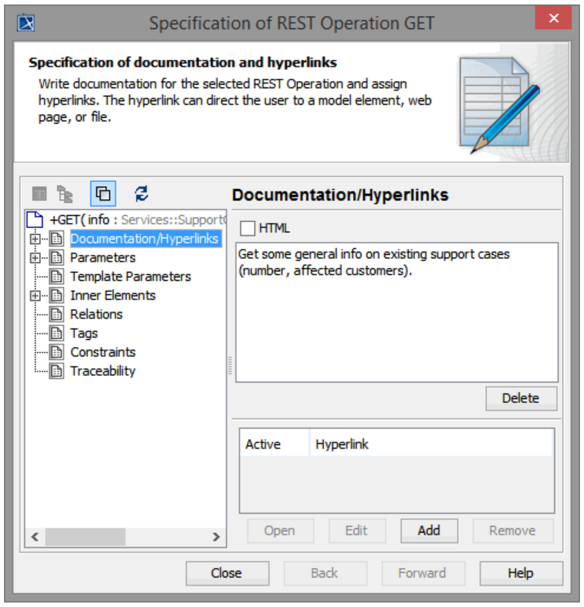
5	**Response Description**	Provide a short description for each response.	`responses:`
			`  200:`
			`    description: A list of support cases that ...`
				<<RESTError>>

Documenting REST Elements

You can add documentation to all REST elements listed in the table above:

- the REST port type
- REST operations
- REST parameters
- REST error classes

Documentation can be added in the specification dialog of each element:



Specification of REST Operation GET

Specification of documentation and hyperlinks

Write documentation for the selected REST Operation and assign hyperlinks. The hyperlink can direct the user to a model element, web page, or file.

Get some general info on existing support cases (number, affected customers).

While writing the documentation, you can use plain text or Git flavored markdown.

Do not use HTML mode in MagicDraw while adding documentation.

REST Response Definitions

Bridge 7.1.0 The xUML REST implementation provides error information via the HTTP body by an error class or a **Blob** (see REST Service Error Handling). All (default and specific) error classes (<<RESTError>>) are reflected in the OpenAPI file of the service as **responses**.

```
swagger: '2.0'
[...]
  /supportcases/{id}:
    delete:
      description: Close a specific support case.
      parameters:
      - in: path
        name: id
        required: true
        type: string
      responses:
        '200':
          description: A success message.
          schema:
            $ref: '#/definitions/ResolveMessage'
        '401':
          description: |-
            - 401 - Unauthorized
            - 404 - Technical error, Not Found

            (See message string and additionalInfo for error details.)
          schema:
            $ref: '#/definitions/RESTErrorPlus'
        default:
          description: |-
            - 400 - Logical error, Bad Request
            - 404 - Technical error, Not Found
            - 500 - Technical error

            (See message string for error details.)
          schema:
            $ref: '#/definitions/RESTError'
      summary: Close a specific support case.
      tags:
      - Transition Support Case
[...]
```

You can add documentation to the error and response classes to provide some documentation on the responses to the OpenAPI file. In the example OpenAPI file above you can find three responses for operation **delete**.

Status Code	Description	Response Class	Documentation
200	normal response	ResolveMessage	<<RESTParameter>> (message)
401	specific error response for authorization errors	RESTErrorPlus	<<RESTError>>
default	default error response	RESTError	<<RESTError>>

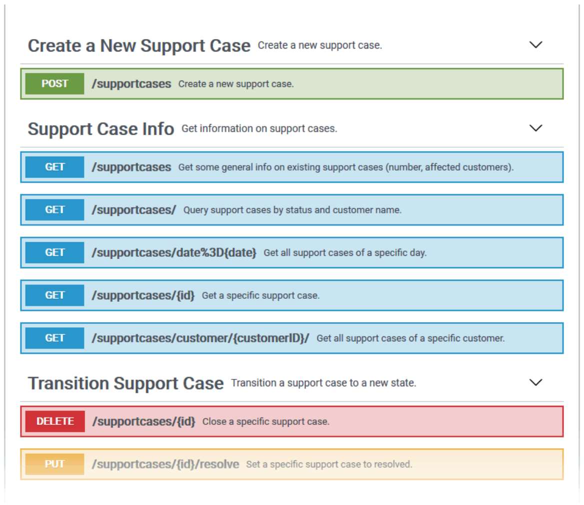
The E2E OpenAPI Importer will import these definitions to the calling xUML service as defined in the OpenAPI file.

Response definitions using patterns (like e.g. 40? or 4??) can not be generated to the OpenAPI file, so it is not recommended to use them. A response definition having pattern ??? will be generated as **default** response of the operation.

Grouping REST Operations

You can use <<RESTOperationTag>> to group your operations. The REST Test Tool will then display the operations in groups by tag. Operations without a tag assigned will appear under a group "default".

Figure: REST Operation Groups



Create a New Support Case Create a new support case.

POST /supportcases Create a new support case.

Support Case Info Get information on support cases.

GET /supportcases Get some general info on existing support cases (number, affected customers).

GET /supportcases/ Query support cases by status and customer name.

GET /supportcases/date%3D{date} Get all support cases of a specific day.

GET /supportcases/{id} Get a specific support case.

GET /supportcases/customer/{customerID}/ Get all support cases of a specific customer.

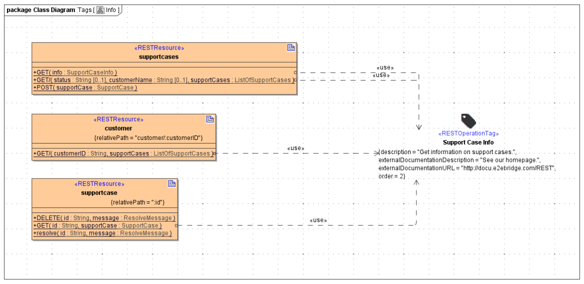
Transition Support Case Transition a support case to a new state.

DELETE /supportcases/{id} Close a specific support case.

PUT /supportcases/{id}/resolve Set a specific support case to resolved.

To tag a REST operation, create a new <<RESTOperationTag>> and draw a class diagram to connect it to operations via a <<use>> dependency.

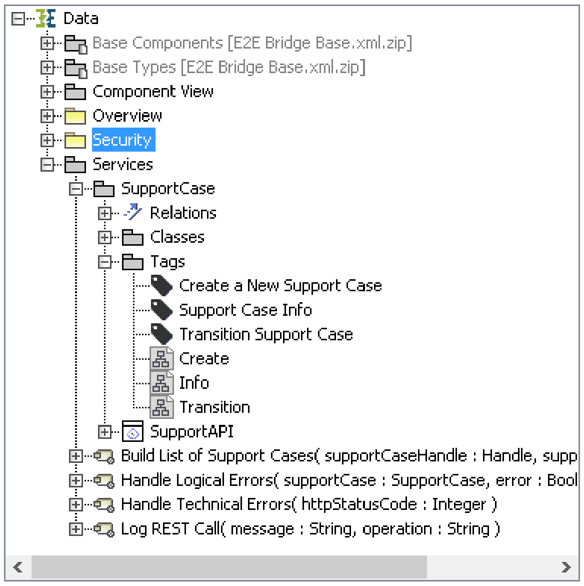
Figure: Tagging REST Operations


package Class Diagram Tags [Info]
«RESTResource»
supportcases
+GET(info : SupportCaseInfo)
+GET/(status : String [0..1], customerName : String [0..1], supportCases : ListOfSupportCases)
+POST(supportCase : SupportCase)
«use»
«use»
«RESTResource»
customer
{relativePath = "customer/:customerID"}
+GET/(customerID : String, supportCases : ListOfSupportCases)
«use»
«RESTOperationTag»
Support Case Info
{description = "Get information on support cases.",
externalDocumentationDescription = "See our homepage.",
externalDocumentationURL = "http://docu.e2ebridge.com/REST",
order = 2}
«RESTResource»
supportcase
{relativePath = ":id"}
+DELETE(id : String, message : ResolveMessage)
+GET(id : String, supportCase : SupportCase)
+resolve(id : String, message : ResolveMessage)
«use»

Artifact <<RESTOperationTag>> has the following tagged values:

Tagged Value	Description		Allowed Values
Name (name)	Defines the name of the tag. This name will be displayed in the Bridge REST Test Tool as a group heading.		any string
Description (description)	You can add a short description of the tag that will be displayed in the Bridge REST Test Tool together with the heading.		any string
External Documentation Description (externalDocumentationDescription)	You can add a short description of the documentation.	These field values will be generated to the OpenAPI descriptor, but are not displayed on the Test UI at the moment.	any string
External Documentation URL (externalDocumentationURL)	Defines a documentation URL for this tag group.		a valid URL
Order (order)	Defines the order in which the tag groups will be displayed on the screen. Tag groups with empty order will be displayed last.		any number

For a better overview, we recommend to put all tag definitions in a separate package and to create a separate class diagram for each tag:


Data
Base Components [E2E Bridge Base.xml.zip]
Base Types [E2E Bridge Base.xml.zip]
Component View
Overview
Security
Services
SupportCase
Relations
Classes
Tags
Create a New Support Case
Support Case Info
Transition Support Case
Create
Info
Transition
SupportAPI
Build List of Support Cases(supportCaseHandle : Handle, supp
Handle Logical Errors(supportCase : SupportCase, error : Bool
Handle Technical Errors(httpStatusCode : Integer)
Log REST Call(message : String, operation : String)