REST Import Rules

This page explains the **REST Import Rules** in Bridge context. If you were looking for the same information regarding the PAS Designer, refer to OpenAPI Import Rules in the Designer guide.

The xUML REST Importer imports OpenAPI 2.0 Specification service descriptors encoded in YAML (Swagger) and creates UML model elements corresponding to the definitions.
Additionally, the importer can generate classes and activity diagrams enabling the modeler to execute the imported services immediately.

Find here how OpenAPI entities are mapped to UML model elements.

Basic Building Blocks of a REST Service

A OpenAPI document is simply a set of nested definitions. The grammar is as follows:

```
basePath: /support
consumes:
- application/json
- text/xml
definitions:
  SupportCase:
    properties:
      id:
        type: string
      customerID:
        type: string
      customerName:
        type: string
      date:
        format: date-time
        type: string
      shortDescription:
        type: string
      status:
        type: string
  RESTError:
    properties:
      message:
        type: string
  SupportCaseInfo:
    properties:
      supportCaseCount:
        type: integer
      customers:
        items:
          $ref: '#/definitions/String'
        type: array
  ListOfSupportCases:
    properties:
      supportCases:
        items:
          $ref: '#/definitions/SupportCase'
        type: array
  ResolveMessage:
    properties:
      message:
        type: string
info:
  description: |-
    ###Manage support cases.

    This REST service provides you with a simple support manager. You can
create, resolve and close support cases, and get support case information.
    - Please provide a valid API token to access all methods.
    - Additionally provide valid user credentials to access DELETE or PUT.
  title: SupportAPI
```

On this Page:

Related Pages:

- REST Parameter Object
- REST Schema Object

- REST Adapter
- Importing OpenAPI Files (REST)

```
  version: 1.0
paths:
  /supportcases:
    get:
      description: Get some general info on existing support cases
(number, affected customers).
      responses:
        '200':
          description: ''
          schema:
            $ref: '#/definitions/SupportCaseInfo'
        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: Get some general info on existing support cases (number,
affected customers).
      tags:
      - Support Case Info
    post:
      description: Create a new support case.
      parameters:
      - in: body
        name: supportCase
        required: true
        schema:
          $ref: '#/definitions/SupportCase'
      responses:
        '201':
          description: ''
          schema:
            $ref: '#/definitions/SupportCase'
        default:
          [...]
          schema:
            $ref: '#/definitions/RESTError'
      summary: Create a new support case.
      tags:
      - Create a New Support Case
  /supportcases/:
    [...]
  /supportcases/{id}:
    [...]
  /supportcases/{id}/resolve:
    [...]
  /supportcases/customer/{customerID}/:
    [...]
produces:
- application/json
- text/xml
security:
- basic: []
- API-Key: []
securityDefinitions:
  basic:
    description: Authenticate using HTTP Basic Authentication
    type: basic
  API-Key:
    description: Authenticate using pre-acquired API key
    in: header
    name: API-Key
    type: apiKey
swagger: '2.0'
tags:
- description: Create a new support case.
  name: Create a New Support Case
```

```
- description: Get information on support cases.
  name: Support Case Info
- description: Transition a support case to a new state.
  name: Transition Support Case
```

Services are defined using the following elements:

Element				Supported by Importer	Description	More Information at ...
`swagger`				✓	Specifies the Swagger specification version being used. The importer only supports **Swagger 2.0**.	
`info`				✓	Provides metadata about the API.	
	`title`			✓	The title is used as name for the <<RESTInterface>>.	
	`description`			✓	The description is used as documentation for the <<RESTInterface>>.	
	`version`			✓	The version is only set in the <<RESTPortType>> if a new model is created with the import.	
`host`				✓	The host (name or IP) serving the API. This must be the host only and does not include the scheme nor sub-paths. It may include a port, though. **host** and **port** are set in the <<RESTAlias>> template.	
`basePath`				✓	The base path on which the API is served. **basePath** is relative to the host. If it is not included, the API is served directly under **host**. The **basePath** is set in the <<RESTAlias>> template.	
`schemes`				✓	A list of transfer protocols of the API. The first scheme is set as **protocol** in the <<RESTAlias>> template. The REST adapter only supports HTTP and HTTPS.	
`consumes`				✗	A list of MIME types the APIs can consume. The REST adapter only parses JSON and XML.	
`produces`				✗	A list of MIME types the APIs can produce. The REST adapter only parses JSON and XML .	
`paths`				✓	The available paths and operations for the API. <<RESTResource>> classes are created to reproduce each paths structure.	
	`$ref`			✗	Allows for an external definition of this path item.	
	`methods`			✓	The http methods defined for this path. A <<REST>> operation is created for each methods.	
		`tags`		✓	A list of tags for API documentation control. An Usage is created from the operation to the corresponding <<RESTOperationTag>> for each tags.	
		`summary`		✓	A short summary of what the operation does. If the description if empty, the summary is used as documentation for the operation.	

		description	✓	A verbose explanation of the operation behavior. The description is used as documentation for the operation.	
		externalDocs	✗	Additional external documentation for this operation.	
		operationId	✗	Unique string used to identify the operation.	
		consumes	✓	A list of MIME types the operation can consume. As the REST adapter only support JSON and XML if consumes is defined and none of these are in the list the parameters are ignored.	
		produces	✓	A list of MIME types the operation can produce. As the REST adapter only support JSON and XML if produces is defined and none of these are in the list the responses are ignored.	
		parameters	✓	A list of parameters of parameter reference that are applicable for this operation. A <<RESTParameter>> is created for each parameter object.	Parameter Object
		responses	✓	The list of possible responses as they are returned from executing this operation. An output parameter is created for the default response status code (201 for POST, 200 for the others). For other response status codes a <<RESTResponseDefinition>> usage is created from the operation to the corresponding class. If the status code is < 400 or default, the <<RESTResponse>> stereotype is added to the class. If the status code is >= 400 or default the <<RESTError>> stereotype is added to the class.	
			description	✓	A short description of the response. The description is used as a documentation for the parameter or the usage.	
			schema	✓	A definition or definition reference of the response structure. The adapter does not support primitives as response therefore a responses of these types are ignored.	Schema Object
			headers	✗	A list of headers that are sent with the response.	
			examples	✗	An example of the response message.	
		schemes	✗	The transfer protocol for the operation.	
		deprecated	✗	Declares this operation to be deprecated.	
		security	✗	A declaration of which security schemes are applied to this operation.	
	parameters	✗	A list of parameters that are applicable to all the operations described under this path.	
definitions	✓	A list to hold data types produced and consumed by operations. A class is created for each schema object.	Schema Object
parameters	✓	A list to hold parameters that can be used across operations.	Parameter Object
responses	✗	An list to hold responses that can be used across operations.	
securityDefinitions	✗	Security scheme definitions that can be used across the specification.	
security	✗	A declaration of which security schemes are applied for the API as a whole.	

`tags`		✓	A list of tags used by the specification with additional metadata. A <<RESTOperationTag>> class is created for each tag definition.	
	`name`	✓	The name of the tag is used as name for the class.	
	`description`	✓	A short description for the tag. The description is used as documentation for the operation.	
	`externalDocs`	✓	Additional external documentation for this tag. The external documentation is set to the tagged values of the <<RESTOperationTag>>.	
`externalDocs`		✗	Additional external documentation.	

When importing an OpenAPI description, the importer will generate a package structure from the OpenAPI definitions. The `definitions` section corresponds to the **Types** package, the `paths` section corresponds to the **Services** package within the imported service.