Manually Providing the REST Interface

It may be that the import of the YAML file to a connector fails or the REST service provides no description file at all. In this case, you can provide the needed definitions manually.
You need:

- a REST alias
  The REST alias specifies where the REST service is located and more connection options. Refer to the REST Adapter Reference for more details.
- a REST interface
  The REST interface describes how resources of this endpoint can be accessed. On REST Import Rules you can find an overview on how OpenAPI entities are mapped to UML elements.

On this Page:

Related Pages:

- Alias
  - Service Panel > Aliases
  - REST Adapter Reference
- Interface
  - REST Import Rules
  - Creating an OpenAPI Connector
  - Modeling Data Structures

REST Alias

Add a RESTAlias to your service in the **Service** panel. How to add an alias is explained on Service Panel > Aliases. The REST adapter configuration options are explained on REST Adapter Reference.

REST Interface

You need to create the types, resources, operations and parameters that are used by the REST service manually in the **Implementation** folder. We recommend to store the information in a packet structure that resembles the structure that would be created when importing to a connector.

Element	Description	Example
Main Package	Create a main package to store the API definitions. As a name, assign the name of the API as provided by the API description	**SupportAPI**
Types Folder	Create a folder to store the API types (see API Types below).	**Types**
Interface	Create a REST interface as described below.	**API**
Resources	Create a REST resources as described below.	**supportcases**

API Types

Collect all parameter and type information from the REST API documentation and create classes and their relations accordingly. If you followed the recommendations above, you should store them to the folder **Types**. Refer to Modeling Data Structures for more information on how to create packages, classes, and so on.
You do not need to depict the complete interface - it is sufficient to create the needed classes for the resources you want to access and the properties you want to deal with.

API Interface

Collect all needed information on the REST resources you want to access from the documentation of the REST service. You need to create all service elements that are listed as mandatory in the table below.

Element	Stereotype	Mandatory	Reference Link	Additional Attributes
REST interface package	**RESTInterface**	✓	RESTInterface	
REST resource classes	**RESTResource**	✓	RESTResource	• Relative Path • Is Verbatim Path
REST operations	**REST**	✓	REST	• HTTP Method • Relative Path • Is Verbatim Path • Blob Body Content Type • Reject Other Response Types • Accepted Request Content Types • Reject Other Request Content Types

REST parameters	**RESTParameter**	(✓)	RESTParameter	• External Name • In

Create classes, operations and parameters in the correct structure and apply the necessary stereotypes. Set the additional attributes that are related to the stereotype as needed.

1. Create a package <your REST interface> (e.g. **API**) under package <your API name> (e.g. **SupportAPI**) and assign stereotype **RESTInterface**.
2. In this package, create REST resources (classes with stereotype **RESTResource**) and their operations (stereotype **REST**) and parameters (stereotype **RESTParameter**) according to the REST service documentation.

   If the REST operation path contains fix segments (like e.g. `date=`) that should not be URL encoded before accessing the resource, set tagged value **isVerbatimPath** to **true**. Refer to REST Adapter Reference for more details on this.

   If a REST parameter has an ugly name that would have to be escaped, use tagged value **externalName**. Refer to REST Adapter Reference for more details.