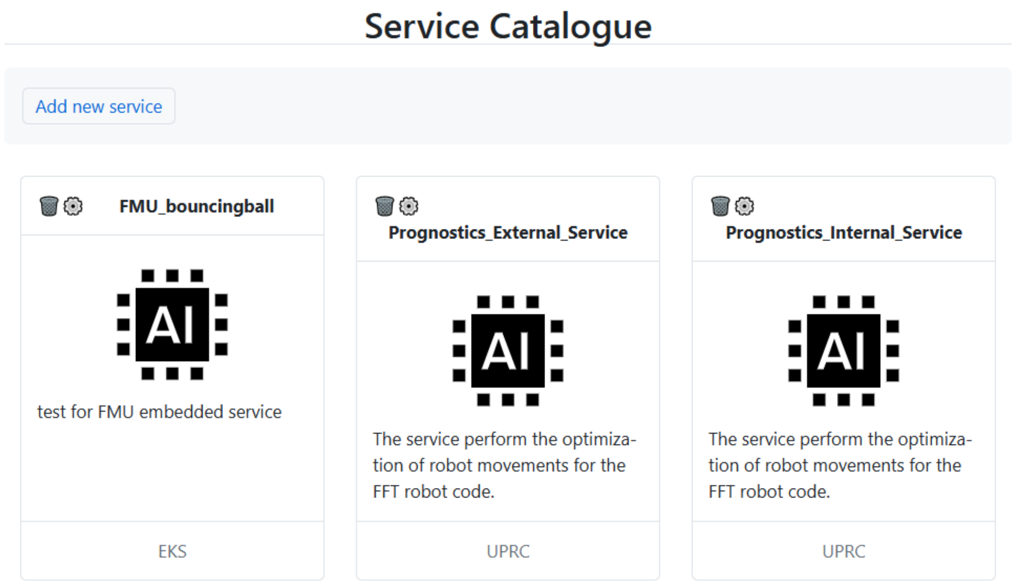
Service Catalogue User Manual

1. Concepts

The Service Catalogue is a component that allows to register services, list them, and expose their metadata. The services can be of four types as aligned in the MODAPTO System: FMU, External, Internal, Orchestrated. The FMU service is a package defined in FMU standard format that require a compliant FMU engine to be executed; the External service can represent any existing running service exposing a REST interface; the Internal service is used to register a service that must first be deployed and only then available over a REST interface; while the Orchestrated is a service exposed by the MODAPTO Orchestrator, combining different existing service invocations (only possible for Internal and External services).

2. User Interface

The main interface is composed of a simple view that show a button for registering a new service and a grid of registered services with their name, logo, description, and affiliation (Figure 1).



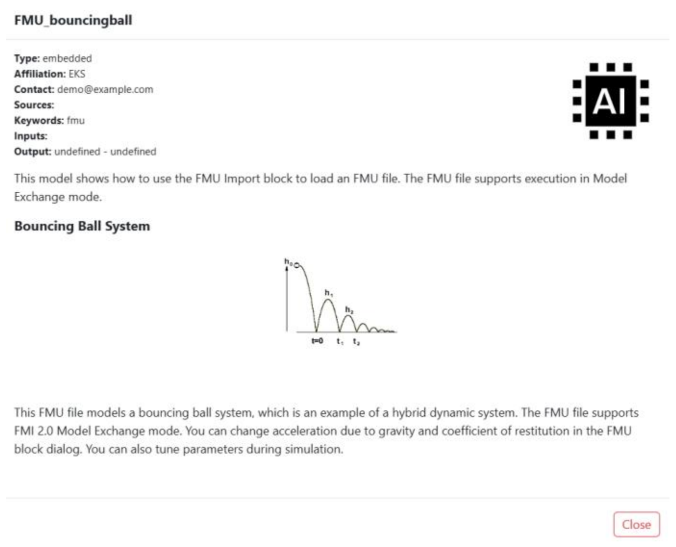
test for FMU embedded service

The service perform the optimization of robot movements for the FFT robot code.

The service perform the optimization of robot movements for the FFT robot code.

Figure 1 - Service Catalogue - Main UI

For each service is available a button to delete the service and a button to edit its details, while clicking over the service card will show an overview of the service details (Figure 2).


FMU_bouncingball

Type: embedded
Affiliation: EKS
Contact: demo@example.com
Sources:
Keywords: fmu
Inputs:
Output: undefined - undefined

This model shows how to use the FMU Import block to load an FMU file. The FMU file supports execution in Model Exchange mode.

Bouncing Ball System

This FMU file models a bouncing ball system, which is an example of a hybrid dynamic system. The FMU file supports FMI 2.0 Model Exchange mode. You can change acceleration due to gravity and coefficient of restitution in the FMU block dialog. You can also tune parameters during simulation.

Figure 2 - Service Catalogue - Service Details

The edit button of each service allows to access a form where all the details of the services can be modified. The form is exactly the same as the one for adding a new service, but in this case the files are filled with the values for the specific service.

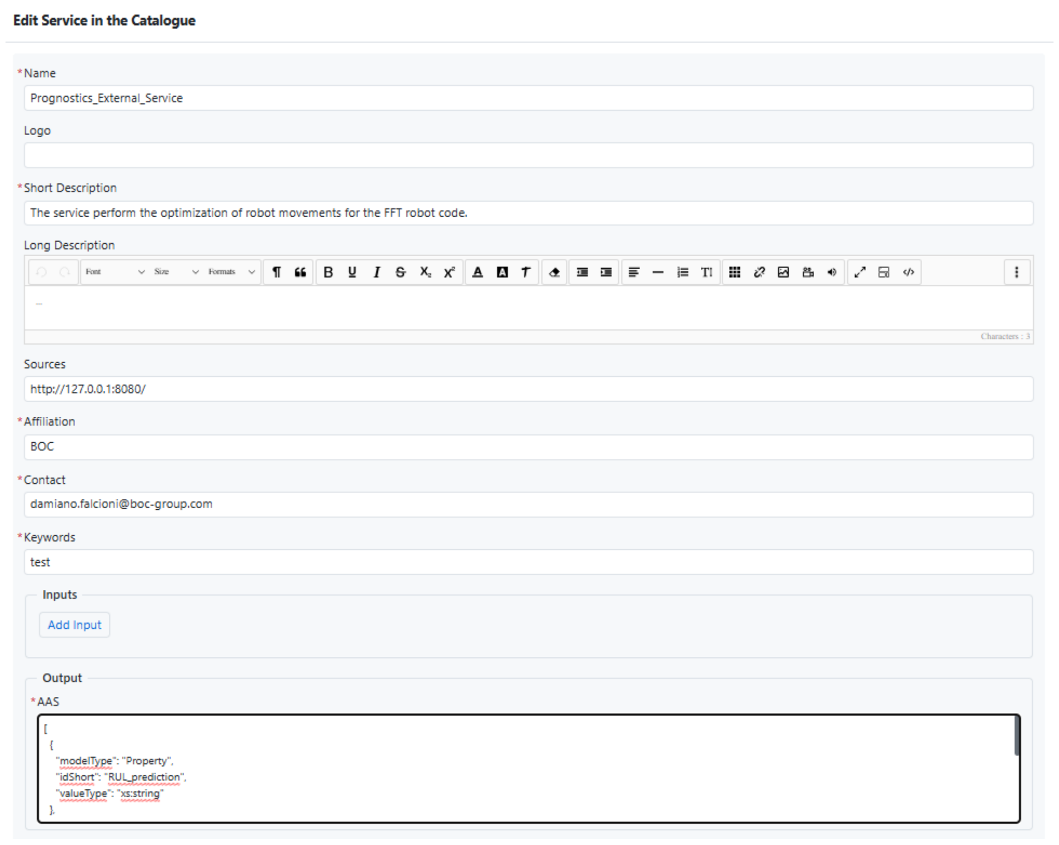
The form is divided into two part, a generic one and one specific for each type of service. In the generic part (Figure 3) the user can edit the service name, the logo (when available), the short description that appear in the service card on the grid, the long description as rich text that is visualized in the service details, the link to the sources of the service when available, the affiliated company with a contact point, a list of keywords for the service and the description of input and output format in standard AAS JSON. A sample is provided prefilled for the input and output fields and must be customized by the user for the specific service. The input in particular, when available, can be added clicking the button “Add Input”, that will add an additional text field (removable using the side X button) containing a JSON object of a single AAS SubmodelElement and is prefilled with:

```
{
  "modelType": "Property",
  "idShort": "id",
  "valueType": "xs:string",
  "description": ""
}
```

The output require instead a JSON array of SubmodelElement and is prefilled with:

```
[{
  "modelType": "Property",
  "idShort": "result",
  "valueType": "xs:string"
}]
```

The modelType indicate the type of the AAS SubmodelElement, the idShort is a unique name for the element, while the valueType is the AAS data type of the element value.



Edit Service in the Catalogue

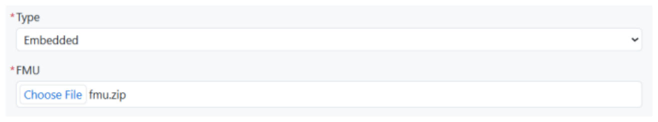
*Name	Prognostics_External_Service
Logo	
*Short Description	The service perform the optimization of robot movements for the FFT robot code.
Long Description	--
Sources	http://127.0.0.1:8080/
*Affiliation	BOC
*Contact	damiano.falcioni@boc-group.com
*Keywords	test

*AAS

```
[
  {
    "modelType": "Property",
    "idShort": "RUL_prediction",
    "valueType": "xs:string"
  },
```

Figure 3 - Service Catalogue - New/Edit Service Common Fields

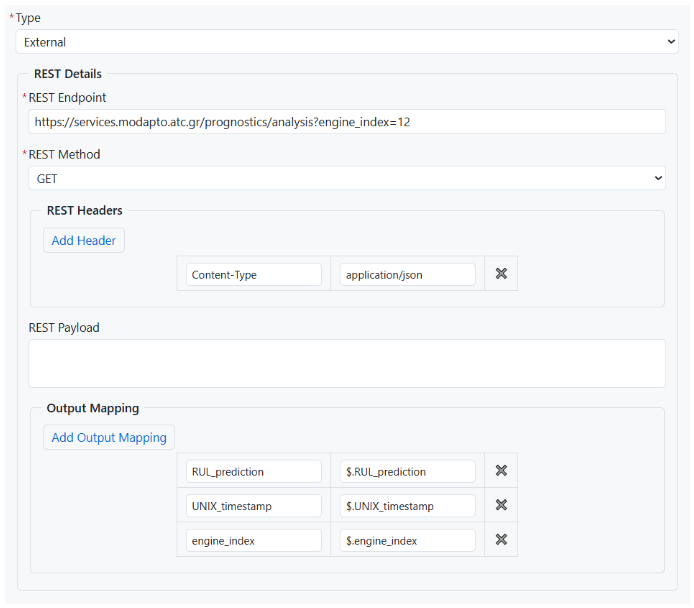
After selecting the type Embedded in the Type selection filed the form allows to upload a zip file containing the service code in standard FMU format (Figure 4).



*Type	Embedded
*FMU	Choose File fmu.zip

Figure 4 - Service Catalogue - New/Edit Service FMU Fields

When the Type selected is External (Figure 5) the form allows to add the REST details of the service and in particular: the Endpoint, the Method, any additional Headers, the Payload (when available), and optionally a mapping from the effective service output (expected as a JSON) to the previously defined output AAS definitions. Here the mapping expect the AAS idShort value as a Key and as value the JQ expression to extract the value from the service output (sample key: `result`; sample value: `$.data.status`). Any of the defined inputs can be used in all the REST fields (except the Output Mapping), simply by using the placeholder format `${_input_idShort_}` where `_input_idShort_` is the value of the idShort field of the defined AAS input.



*Type	External

REST Details

*REST Endpoint	https://services.modapto.atc.gr/prognostics/analysis?engine_index=12
*REST Method	GET

REST Headers

Content-Type	application/json

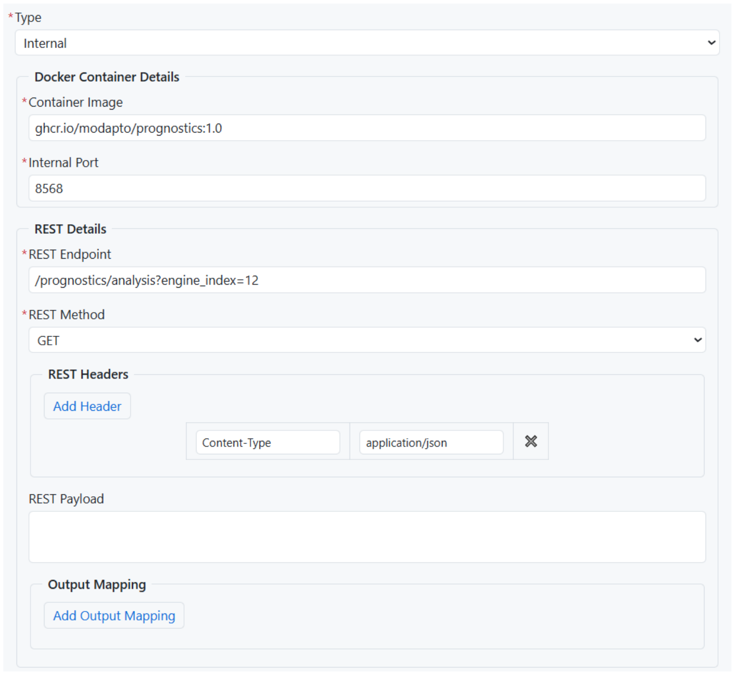
REST Payload

Output Mapping

RUL_prediction	$.RUL_prediction
UNIX_timestamp	$.UNIX_timestamp
engine_index	$.engine_index

Figure 5 - Service Catalogue - New/Edit Service External Fields

When the Type selected is Internal (Figure 6) the form allows to specify first the public repository of the Docker image of the service that must be deployed (sample: `ghcr.io/modapto/prognostics:1.0`), along with the port expected to be exposed on container run, and then the same REST details as in the External service, with the only difference that now the Endpoint is expected to be a relative path to the service, as the hostname part will be available only after the deployment phase.



*Type	Internal

Docker Container Details

*Container Image	ghcr.io/modapto/prognostics:1.0
*Internal Port	8568

REST Details

*REST Endpoint	/prognostics/analysis?engine_index=12
*REST Method	GET

REST Headers

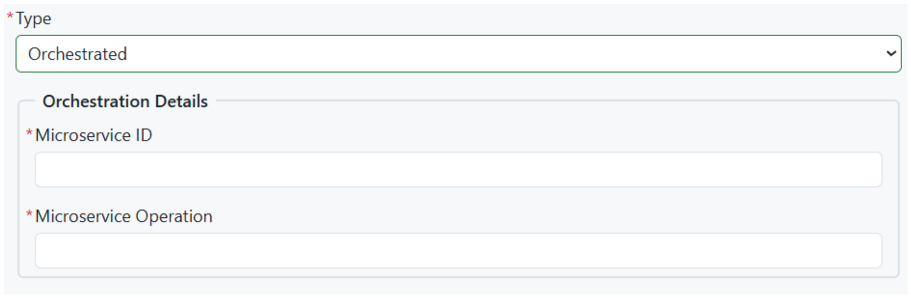
Content-Type	application/json

REST Payload

Output Mapping

Figure 6 - Service Catalogue - New/Edit Service Internal Fields

Finally, when the Type selected is Orchestrated (Figure 7) the form allows to specify the Microservice ID and Operation referencing the specific service instance in the Orchestrator. This information are visible in the Orchestrator using the service Test UI.


*Type

Orchestrated

Orchestration Details

*Microservice ID

*Microservice Operation

Figure 7 - Service Catalogue - New/Edit Service Orchestrated Fields

3. Deployment

As the Service Catalogue is a client only application and use the Orchestrator API as backend, it is deployed in the same Docker image of the Orchestrator, available at https://github.com/Modapto/orchestrator. After the deployment the Service Catalogue interface will be now available at http://127.0.0.1:8080/catalog/.