D2.1 SPECIFICATION OF INTEROPERABLE APIS FOR THE PLANNING TOOL

VERSION 1.0

Markus Mirz

25 June 2020
# TABLE OF CONTENT

1. **MOTIVATION** .................................................................................................................. 7
2. **API SPECIFICATION AND TOOLBOX DEVELOPMENT** ................................................. 8
3. **ANM4L USE CASES – POWER SYSTEM PLANNING** ...................................................... 10
   3.1 Outage Planning and Contingency Analysis .................................................................... 10
   3.2 Reinforcement and Reinvestment Planning .................................................................. 10
   3.3 Time Series and Stochastic Power flow ...................................................................... 10
4. **OVERVIEW ANM4L PLANNING API** .......................................................................... 12
5. **CONCLUSIONS AND OUTLOOK** .................................................................................. 15
6. **APPENDIX – API SPECIFICATION (OPENAPI 3.0)** ....................................................... 17
Disclaimer

The content and views expressed in this material are those of the authors and do not necessarily reflect the views or opinion of the ERA-Net SES initiative. Any reference given does not necessarily imply the endorsement by ERA-Net SES.

About ERA-Net Smart Energy Systems

ERA-Net Smart Energy Systems (ERA-Net SES) is a transnational joint programming platform of 30 national and regional funding partners for initiating co-creation and promoting energy system innovation. The network of owners and managers of national and regional public funding programs along the innovation chain provides a sustainable and service oriented joint programming platform to finance projects in thematic areas like Smart Power Grids, Regional and Local Energy Systems, Heating and Cooling Networks, Digital Energy and Smart Services, etc.

Co-creating with partners that help to understand the needs of relevant stakeholders, we team up with intermediaries to provide an innovation eco-system supporting consortia for research, innovation, technical development, piloting and demonstration activities. These co-operations pave the way towards implementation in real-life environments and market introduction.

Beyond that, ERA-Net SES provides a Knowledge Community, involving key demo projects and experts from all over Europe, to facilitate learning between projects and programs from the local level up to the European level.

www.eranet-smartenergysystems.eu
INTRODUCTION TO ANM4L

The ANM4L (Active network management for all) project, anm4l.eu, will develop solutions to enable integration of renewables with the agility required from developments in demand and production.

Alternatives to traditional network expansion are needed to ensure sustainable development of the power grids. New technologies, methods, and markets are emerging to provide increased flexibility in consumption, generation, and power transfer capacity.

ANM4L aims at demonstrating innovative active network management (ANM) solutions to increase integration of renewable energy sources (RES) in electricity distribution systems.

ANM solutions will consider management of active and reactive power to avoid overload situations, maintain voltages within limits, minimize the need of RES curtailment, and enable further RES uptake even above the theoretical design limit of the electricity network.

Core research and development activities include development of:

- Active network management methods for local energy systems.
- Business models to provide decision support for market players.
- An integrated toolbox to support the planning and operation of the distribution system.

The toolbox, methods and business models for ANM will be demonstrated in Sweden and Hungary. The project will also prepare solutions and recommendations for replication in other local and regional energy systems.

The ANM4L project is an international cooperation with a consortium consisting of partners in Sweden, Germany and Hungary:

- RISE Research Institutes of Sweden (coordinator)
- Municipality of Borgholm
- Lumenaza GmbH
- Lund University
- RWTH Aachen University
- E.ON Energidistribution AB
- E.ON Észak-dunántúli Áramhálózati Zrt.
- E.ON Solutions GmbH
This deliverable is part of Work package 2 (WP2): *ICT solutions for ANM planning and operation*.

The main objective of WP2 is to develop an interoperable and replicable ICT toolbox for planning and operation using ANM methods.

This document provides an overview of the planning tool and API specification. The first section discusses the need for an interoperable API, which complements common data formats such as the IEC 61970 CIM. Section 2 explains how the API is specified in a human- and machine-readable format that supports the implementation of the planning toolbox itself. Use cases of the API are briefly described in Section 3 to motivate the extent and structure of the API. The structure is depicted in Section 4 and the API itself can be found in the Appendix.
1 MOTIVATION

Interoperability is becoming increasingly important, especially vertically between TSO and DSO. This is highlighted by the TSO-DSO Data Management Report¹, which describes the need for a deeper vertical integration of TSOs and DSOs.

Furthermore, the focus should be on services rather than platforms or technology stacks. The advantage of this approach is that the same service can be provided by different platforms. Hence, a deeper integration does not necessarily require the use of the same platform. Focusing on the services enables a faster transition than trying to move all partners to a specific platform.

To provide access to a service, the user only requires a specification of its Application Programming Interface (API). The source code implementing the service is not required. Figure 1.1 visualizes the role of an API with an example. Here, a waiter in a restaurant represents the API that enables the interaction between a customer and a restaurant’s kitchen.

![Figure 1.1: An API has a similar function as a waiter in a restaurant.](image)

Standardized services for grid planning facilitate not only the data exchange but also the access to functions on this data by other grid operators. In some cases, it might be sufficient for the TSO to run functions on the DSOs grid data and receive the results instead of retrieving the DSO grid data and run the analysis. Besides, standardized interfaces support flexibility in the IT system of grid operators, which allows them to replace certain parts of their infrastructure without affecting other parts.

With regard to the data format for data exchanged between grid operators, standardization has already progressed significantly. While IEC61970 CIM XML is becoming the standard format to exchange grid data for planning purposes, a standard API for planning services is not defined yet. Similar efforts to define services can be observed in other EU H2020 projects, such as SOGNO and PlatOne. However, these projects focus on grid operation rather than planning.

2 API SPECIFICATION AND TOOLBOX DEVELOPMENT

As stated above, it is preferable to specify the API of services rather than the technology stack implementing the API. A popular type of interface, which is programming language agnostic, is called REST API. OpenAPI is a project to standardize the specification of REST APIs. The advantage of OpenAPI is that many tools exist to either present the API in a human readable format or generate source code to implement the API in a specified programming language.

The idea is that the OpenAPI specification abstracts the usage of the planning tool from its actual implementation. Therefore, it becomes possible to replace parts of the planning tool without affecting the IT system interacting with the planning tool.

The general structure of the ANM4L planning tool, which implements the functions defined by the API, is depicted in Figure 2.1.

![Figure 2.1: Modular planning tool based on IEC 61970 CIM input data and an open API.](image)

The main components are

- a grid data processing tool to import, export and modify CIM grid data,
- a simulation core for power flow and possibly dynamic simulation and
- algorithms that include, for example, the Active Network Management methods (ANM) developed within the frame of the ANM4L project.

These blocks are extended by an interface to commercial tool, which may serve two purposes:

- provide functionality that is not available in the open source software blocks and
- represent a reference solution that can be compared to the open source blocks in order to evaluate their correctness.
All of these blocks are encapsulated by the REST API, which will be the only interface between the user and the ANM4L toolbox. Open source projects that could be integrated with the toolbox include two projects developed by RWTH: CIMpy² for grid data processing and DPsim³ as simulation core.

² https://github.com/cim-iec/cimpy
³ https://github.com/dpsim-simulator/dpsim

D2.1 | Specification of interoperable APIs for the planning tool
3 ANM4L USE CASES – POWER SYSTEM PLANNING

The following sections describe the current use cases of power system software for planning purposes. Generally, the objective is to prevent overvoltage and overloading of equipment, such as power lines and transformers. For this reason, it is necessary to compare the expected power flow to the current and voltage ratings of the equipment.

3.1 Outage Planning and Contingency Analysis

Equipment outages can be planned or unplanned events. The evaluation of the network state after unforeseen single equipment failure is often referred to as contingency analysis. This means that many different single contingency scenarios have to be simulated depending on the equipment that fails. The N-1 criterion means that the system is operated within its limits considering the failure of one component. In addition to the failure of equipment, planned outages due to maintenance and replacement can be considered in the network state analysis.

The analysis has to take into account on-load tap-changers, shunt capacitor and reactor switching and dynamic equipment ratings, e.g. for overhead lines. To drive the system closer to its limits and avoid reinvestments, overhead line ratings can be considered dynamic depending on line loading, wind speed and outside temperature. These three parameters determine the line sag, which has to be contained within certain limits.

3.2 Reinforcement and Reinvestment Planning

The reinforcement and reinvestment planning is similar to the outage planning in that it is based on power flow considering various single contingency scenarios. However, the grid model is updated with respect to the future scenario, either replacing equipment or extending the model, e.g. for wind farms or solar panels.

When placing new equipment, it is necessary to optimize certain control parameters of generation units and transformers. The setting of the transformer’s tap changer should be optimized to obtain a good voltage along the section connected to the secondary side of the transformer. In addition, the power factor set point of generation units can be adjusted to improve the voltage level and a capacitor bank might be considered to provide reactive power to the grid.

Power curtailment should be minimized to utilize the full capacity of generation units as much as possible. Besides, power losses may be minimized depending on the relation between investment costs and costs due to the losses. Replacing old equipment, such as lines and transformers, or the provision of reactive power by capacitor banks can be measures to reduce power losses.

3.3 Time Series and Stochastic Power flow

Often, power flow studies for planning purposes consider only maximum and minimum load and maybe maximum and minimum generation scenarios. A more detailed analysis can be conducted with time series or stochastic methods.

Time series studies are particularly interesting for battery storage applications and demand side management. For the latter, stochastic models could be applied to consider the fluctuating availability.
In case of connection of intermittent generation, time series and stochastic models are important to quantify the amount of expected curtailment if the available capacity in the connection point is temporarily lower than the capacity of the connected generation units. When connecting load, time series data and stochastic models could be interesting to identify load profiles and diversity factors.

Furthermore, time series and stochastic power flow studies can be used to determine the frequency of exceptional operating conditions to allow for a more detailed evaluation of the lifetime of equipment.
4 OVERVIEW ANM4L PLANNING API

The REST API specification developed in the frame of task T2.1 is included in the Appendix and available as open source\(^4\). This section gives an overview of the REST API specification. The main sections and resources available from the API are:

- Network models
- Static Analysis
- Scenarios, Events and Time Series Data
- Time Domain Simulations
- Control Algorithms

The *network models* section has functions to import, export and manipulate imported network data. The network data is formatted in CIM IEC61970 and the CIM files are included in a file archive for import and export. Furthermore, it should be possible to receive an image of the network imported into the tool as demonstrated in Figure 4.2. To modify the *network model*, the API provides operations to create and change elements of the network and their attributes, such as the impedance of a transmission line.

Stochastic parameters associated to an attribute defined in the CIM grid data, should be provided in separate files, which are included in the same archive as the CIM data. These additional files could be named after the unique CIM identifiers of the components which they are related to.

---

\(^4\) https://git.rwth-aachen.de/acs/public/deliverables/anm4l/api
Network models can be directly used in static analysis operations, which do not operate in the time domain. These are, for example, powerflow and outage analysis for single and double contingency cases based on fixed or stochastic parameters. Stochastic parameters could be for example the active power level for consumers or producers. Analysis or simulation of network models in combination with time series data is not part of the static analysis section.

Scenarios group a network model with time series data, e.g. power profiles, and events defined in the time domain, e.g. a network reconfiguration at a predetermined time. Time domain simulations and control algorithms can be executed on these scenarios. Time series data is not managed through the ANM4L API directly. Instead, the resources managed by the ANM4L API keep track of time series data by their labels, which uniquely identify a time series in a time series database management system. The time series data that is input to the control algorithm maybe stored in advance, generated by a simulation or measured.
in a real network. For example, the ANM4L voltage control algorithm will be either conducted based on simulation results in the planning phase or based on measurements during operation.
5 CONCLUSIONS AND OUTLOOK

The API specification presented in this deliverable allows the user to decouple two fundamental parts of the grid planning workflow: the domain specific but scenario independent tools for power system simulation / automation and the specific planning scenarios, which are often unique for each user and use case. The idea is to decrease the dependency of the grid planning workflow on specific tools and emphasize algorithms, models in common data formats rather than vendor specific implementations. This will increase comparability and interoperability.

The vision is to develop the API specification as a living document throughout the project life time and beyond. The next iteration of the planning API is foreseen during the implementation of the operational tool for ANM4L to be able to exploit synergies between the planning and operational use cases.
This document was created as part of the ERA-Net Smart Energy Systems project 91341, funded from the European Union’s Horizon 2020 research and innovation programme under grant agreement no. 775970 (RegSys).
APPENDIX – API SPECIFICATION (OPENAPI 3.0)
ANM4L API

All rights reserved
http://apache.org/licenses/LICENSE-2.0.html

Access

1. APIKey KeyParamName:X-API-KEY KeyInQuery:false KeyInHeader:true

Methods

[ Jump to Models ]

Table of Contents

Control

- POST /control/{id}/input
- POST /control
- DELETE /control/{id}
- GET /control/{id}
- GET /control/{id}/output
- GET /control
- PUT /control/{id}

NetworkModels

- POST /models/{modelid}/elements
- POST /models
- DELETE /models/{modelid}/elements/{id}
- DELETE /models/{id}
- GET /models/export/{id}
- GET /models/{modelid}/elements/{id}
- GET /models/{modelid}/elements
- GET /models/{id}
- GET /models/image/{id}
- GET /models
- POST /models/import/{id}
- PUT /models/{modelid}/elements/{id}
- PUT /models/{id}

Scenarios

- POST /scenarios/{scenarioid}/events
- POST /scenarios
- POST /scenarios/{scenarioid}/timeseries
- DELETE /scenarios/{scenarioid}/events/{id}
- DELETE /scenarios/{scenarioid}
- DELETE /scenarios/{scenarioid}/timeseries/{id}
- GET /scenarios/{scenarioid}/events/{id}
- GET /scenarios/{id}
- GET /scenarios
- GET /scenarios/{scenarioid}/timeseries
- GET /scenarios/{scenarioid}/events
- PUT /scenarios/{scenarioid}/events/{id}
- PUT /scenarios/{id}
- PUT /scenarios/{scenarioid}/timeseries/{id}
**StaticAnalysis**

- POST /analysis/outages
- POST /analysis/powerflows
- DELETE /analysis/{id}
- GET /analysis/{id}
- GET /analysis/{analysisid}/elements/{id}
- GET /analysis/{analysisid}/elements
- GET /analysis/outages
- GET /analysis/powerflows

**TimeDomainSimulation**

- POST /simulation
- DELETE /simulations/{id}
- GET /simulations/{id}
- GET /simulation/{simid}/elements/{id}
- GET /simulation

**Users**

- POST /users
- DELETE /users/{id}
- GET /users/{id}
- GET /users
- PUT /users/{id}

**Control**

- POST /control/{id}/input

Add new measurements to be processed by the control ([addControlInput](#))

**Path parameters**

<table>
<thead>
<tr>
<th>id (required)</th>
</tr>
</thead>
</table>

*Path Parameter* — Controller id

**Consumes**

This API call consumes the following media types via the Content-Type request header:

- application/json

**Request body**

- body [map](#) (required)

*Body Parameter* — New control input

**Return type**

[Control](#)

**Example data**

Content-Type: application/json

```
{
    "name" : "name",
    "id" : 0
}
```
Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses
200
OK Control
default
unexpected error Error

POST /control
Create new control instance (addControls)

Consumes
This API call consumes the following media types via the Content-Type request header:

- application/json

Request body

body NewControl (required)
Body Parameter — Controller to be added

Return type
Control

Example data
Content-Type: application/json

```json
{
    "name": "name",
    "id": 0
}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses
200
OK Control
default
unexpected error Error

DELETE /control/{id}
Delete a controller (deleteControl)

Path parameters

- id (required)
Path Parameter — Controller id
Return type
Control

Example data
Content-Type: application/json

```json
{
  "name": "name",
  "id": 0
}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses
200
OK Control
default
unexpected error Error

GET /control/{id}

Get one control instance (getControl)

Path parameters

- id (required)
  
  `Path Parameter` — Control id

Return type
array[Control]

Example data
Content-Type: application/json

```json
[ {
  "name": "name",
  "id": 0
}, {
  "name": "name",
  "id": 0
}
]
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses
200
OK
default
unexpected error Error
GET /control/{id}/output

Get latest control set points (getControlOutput)

**Path parameters**

- **id** (required)
  
  *Path Parameter* — Controller id

**Return type**

map[String, null]

**Example data**

Content-Type: application/json

```json
{
  "key" : ""
}
```

**Produces**

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

**Responses**

- 200
- OK
- default
- unexpected error Error

---

GET /control

Get all control instances (getControls)

A control instance computes output values based on time series input data and a network model. A control instance might include states as part of the control algorithm.

**Return type**

array[Control]

**Example data**

Content-Type: application/json

```json
[ {
  "name" : "name",
  "id" : 0
}, {
  "name" : "name",
  "id" : 0
} ]
```

**Produces**

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

**Responses**

- 200
PUT /control/{id}

Update control instance (updateControl)

**Path parameters**

id (required)

*Path Parameter* — Controller id

**Consumes**

This API call consumes the following media types via the Content-Type request header:

- application/json

**Request body**

body **NewControl** (required)

*Body Parameter* — Controller attributes to be updated

**Return type**

**Control**

**Example data**

Content-Type: application/json

```
{
  "name": "name",
  "id": 0
}
```

**Produces**

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

**Responses**

200

OK **Control**

default

unexpected error **Error**

---

**NetworkModels**

POST /models/{modelid}/elements

Add element to model (addElement)

**Path parameters**

modelid (required)

*Path Parameter* — Model id
POST /models

Add a network model (addModel)

Consumes
This API call consumes the following media types via the Content-Type request header:

- application/json

Request body

- body NewModelElement (required)

Body Parameter – Element to be added to model

Return type

ModelElement

Example data

Content-Type: application/json

```json
{
    "name" : "name",
    "id" : 0,
    "type" : "type",
    "uuid" : "uuid"
}
```

Produce

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

OK ModelElement

default

unexpected error Error

Up

POST /models

Add a network model (addModel)

Consumes
This API call consumes the following media types via the Content-Type request header:

- application/json

Request body

- body NewModel (required)

Body Parameter – Network model to be added

Return type

Model

Example data

Content-Type: application/json

```json
{
    "name" : "name",
    "id" : 0
}
```
**Produce**
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

**Responses**
200
OK **Model**
default
unexpected error **Error**

DELETE /models/{modelid}/elements/{id}

Delete element of model (**deleteElement**)

**Path parameters**
- modelid (required)
  - Path Parameter — model id
- id (required)
  - Path Parameter — element id

**Return type**
**ModelElement**

**Example data**
Content-Type: application/json

```json
{
  "name": "name",
  "id": 0,
  "type": "type",
  "uuid": "uuid"
}
```

**Produce**
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

**Responses**
200
OK **ModelElement**
default
unexpected error **Error**

DELETE /models/{id}

Delete a network model (**deleteModel**)

**Path parameters**
- id (required)
  - Path Parameter — Model id

**Return type**
Example data
Content-Type: application/json

```json
{
   "name": "name",
   "id": 0
}
```

Producers
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses
200
OK Model
default unexpected error Error

GET /models/export/{id}

Export model to file (exportModel)
Returns an archive containing the grid data in CIM formatted files and profile files that might have been imported previously.

Path parameters
- id (required)
  Path Parameter — Model id

Return type
byte[]

Example data
Content-Type: application/json

```
...
```

Producers
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/octet-stream
- text/uri-list
- application/json

Responses
200
OK byte[]
default unexpected error Error

GET /models/{modelid}/elements/{id}

Get element of model (getElement)
Path parameters

- `modelid` (required)
  *Path Parameter* — Model id

- `id` (required)
  *Path Parameter* — Element id

Return type

*ModelElementAttributes*

Example data

Content-Type: application/json

```json
{
  "name": "name",
  "attributes": {
    "key": "",
  },
  "id": 0,
  "type": "type",
  "uuid": "uuid"
}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses

200 OK *ModelElementAttributes*

default
unexpected error Error

GET `/models/{modelid}/elements`

Get all elements of a model (*getElements*)

Path parameters

- `modelid` (required)
  *Path Parameter* — Model id

Return type

Array*ModelElement*

Example data

Content-Type: application/json

```json
[  
  {
    "name": "name",
    "id": 0,
    "type": "type",
    "uuid": "uuid"
  },
  {
    "name": "name",
    "id": 0,
    "type": "type",
```
GET /models/{id}

Get a network model (getModel)

Path parameters
id (required)
  Path Parameter — Model id

Return type
Model

Example data
Content-Type: application/json

```json
{
  "name" : "name",
  "id" : 0
}
```

GET /models/image/{id}

Render and return image of network model (getModelImage)

Returns an SVG image of the network based on CIM information.

Path parameters
id (required)
  Path Parameter — Model id

Return type
**Example data**

Content-Type: application/json

```
...
```

**Produces**

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/octet-stream
- text/uri-list
- application/json

**Responses**

200 OK `byte[]`
default

unexpected error **Error**

---

GET `/models`

Get all network models (getModels)

**Return type**

array[Model]

**Example data**

Content-Type: application/json

```json
[
  {
    "name": "name",
    "id": 0
  },
  {
    "name": "name",
    "id": 0
  }
]
```

**Produces**

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

**Responses**

200 OK
default

unexpected error **Error**

---

POST `/models/import/{id}`

Import model from file (importModel)

The input file should be an archive containing the grid data in the CIM format. Optionally, profiles or stochastic parameters can be given as additional files, where file and column name should correspond to the CIM component uuid and attribute name.

**Path parameters**
id (required)
Path Parameter — Model id

Consumes
This API call consumes the following media types via the Content-Type request header:

- application/octet-stream
- text/uri-list

Request body
body Object (required)
Body Parameter — Files defining the model

Return type
Model

Example data
Content-Type: application/json

```json
{
    "name": "name",
    "id": 0
}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header:

- application/json

Responses
200
OK Model
default
unexpected error Error

PUT /models/{modelid}/elements/{id}

Update element of model (updateElement)

Path parameters
- modelid (required)
  Path Parameter — model id
- id (required)
  Path Parameter — element id

Consumes
This API call consumes the following media types via the Content-Type request header:

- application/json

Request body
body ModelElementUpdate (required)
Body Parameter — Model Element attributes to be updated
Return type
ModelElement

Example data
Content-Type: application/json

```json
{
  "name" : "name",
  "id" : 0,
  "type" : "type",
  "uuid" : "uuid"
}
```

Produce
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Response
200 OK ModelElement
default unexpected error Error

PUT /models/{id}

Update a network model (updateModel)

Path parameters
- id (required)
  Path Parameter — Model id

Consumes
This API call consumes the following media types via the Content-Type request header:

- application/json

Request body
- body ModelUpdate (required)
  Body Parameter — Network model to be updated

Return type
Model

Example data
Content-Type: application/json

```json
{
  "name" : "name",
  "id" : 0
}
```

Produce
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json
Scenarios

POST /scenarios/{scenarioId}/events

Add an event (addEvent)
An event defines the change of an attribute at a certain point in time.

Path parameters

scenarioId (required)
Path Parameter — Scenario id

Consumes
This API call consumes the following media types via the Content-Type request header:

- application/json

Request body

body NewEvent (required)
Body Parameter — Event to be added

Return type

Event

Example data
Content-Type: application/json

```json
{
   "name": "name",
   "id": 0
}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header:

- application/json

Responses

200
OK Event
default
unexpected error Error

POST /scenarios

Add a scenario (addScenario)
Add a new scenario, which holds input data necessary for simulation and control, such as time intervals, events, time series data and the network model.
Consumes
This API call consumes the following media types via the Content-Type request header:

- application/json

Request body

- body **NewScenario** (required)
  
  *Body Parameter* — scenario to be added

Return type

**Scenario**

Example data

Content-Type: application/json

```json
{
  "name" : "name",
  "id" : 0
}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200 OK **Scenario**

default

unexpected error **Error**

```
POST /scenarios/{scenarioid}/timeseries
```

Add a time series to this scenario (**addTimeseries**)

Path parameters

- scenarioid (required)

  *Path Parameter* — Scenario id

Consumes

This API call consumes the following media types via the Content-Type request header:

- application/json

Request body

- body **NewTimeseries** (required)

  *Body Parameter* — Time series to be added

Return type

**Timeseries**

Example data

Content-Type: application/json

```json
{
  "name" : "name",
```
```
"attributes": "attributes",
"id": 0,
"elementuuid": "elementuuid",
"labels": ["labels", "labels"]
```

**Produce**
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

**Responses**

200  
OK  **Timeseries**  
default  
unexpected error  **Error**

DELETE /scenarios/{scenarioid}/events/{id}

Delete an event of a specific scenario (deleteEvent)

**Path parameters**

- scenarioid (required)  
  **Path Parameter** — Scenario id  
- id (required)  
  **Path Parameter** — Event id

**Return type**

**Event**

**Example data**

Content-Type: application/json

```
{
  "name": "name",
  "id": 0
}
```

**Produce**
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

**Responses**

200  
OK  **Event**  
default  
unexpected error  **Error**

DELETE /scenarios/{id}

Delete a scenario (deleteScenario)

**Path parameters**

- id (required)
DELETE /scenarios/{scenarioid}/timeseries/{id}

Delete a Timeseries of a specific scenario (deleteTimeseries)

Path parameters

- scenarioid (required)
  
- id (required)

Return type

Timeseries

Example data

Content-Type: application/json

```
{
  "name" : "name",
  "attributes" : "attributes",
  "id" : 0,
  "elementuuid" : "elementuuid",
  "labels" : [ "labels", "labels" ]
}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200
OK Timeseries

default
unexpected error Error
GET /scenarios/{scenarioid}/events/{id}

Get an event of a specific scenario (getEvent)

Path parameters

- scenarioid (required)
  - Path Parameter — Scenario id
- id (required)
  - Path Parameter — Event id

Return type

array[Event]

Example data

Content-Type: application/json

```
[ {
   "name" : "name",
   "id" : 0
 },
 {  
   "name" : "name",
   "id" : 0
 }
]
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200
OK
default
unexpected error Error

GET /scenarios/{id}

Get individual scenario (getScenario)

Path parameters

- id (required)
  - Path Parameter — scenario id

Return type

array[Scenario]

Example data

Content-Type: application/json

```
[ {
   "name" : "name",
```
"id" : 0
}, {
"name" : "name",
"id" : 0
}

Produce
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses
200
OK
default
unexpected error Error

GET /scenarios

Get all scenarios (getScenarios)

Return type
array[Scenario]

Example data
Content-Type: application/json

```
[ {
   "name" : "name",
   "id" : 0
}, {
   "name" : "name",
   "id" : 0
}
]
```

Produce
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses
200
List of scenarios
default
unexpected error Error

GET /scenarios/{scenarioid}/timeseries/{id}

Get a time series of a specific scenario (getSingleTimeseries)

Path parameters

- scenarioid (required)
  Path Parameter — Scenario id
- id (required)
  Path Parameter — Timeseries id
Example data
Content-Type: application/json

```
[  
  "name" : "name",
  "id" : 0  
],  
[  
  "name" : "name",
  "id" : 0
  
}

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses
200
OK
default
unexpected error Error

GET /scenarios/{scenaroid}/timeseries

Get all time series associated to scenario (getTimeseries)

Path parameters

scenaroid (required)
Path Parameter — Scenario id

Example data
Content-Type: application/json

```
[  
  "name" : "name",
  "attributes" : "attributes",
  "id" : 0,
  "elementuuid" : "elementuuid",
  "labels" : [ "labels", "labels" ]

},
{
  "name" : "name",
  "attributes" : "attributes",
  "id" : 0,
  "elementuuid" : "elementuuid",
  "labels" : [ "labels", "labels" ]
}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json
GET /scenarios/{scenarioId}/events

Get all events (scenariosScenarioidEventsGet)

Path parameters

scenarioId (required)
Path Parameter — Scenario id

Return type
array[Event]

Example data
Content-Type: application/json

```
[{
  "name": "name",
  "id": 0,
}, {
  "name": "name",
  "id": 0
}
]
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header:

- application/json

Responses
200
OK
default
unexpected error Error

PUT /scenarios/{scenarioId}/events/{id}

Update an event of a specific scenario (updateEvent)

Path parameters

scenarioId (required)
Path Parameter — Scenario id

id (required)
Path Parameter — Event id

Consumes
This API call consumes the following media types via the Content-Type request header:

- application/json

Request body
body **NewEvent** (required)

*Body Parameter* — Event attributes to be updated

Return type

**Event**

**Example data**

Content-Type: application/json

```json
{
    "name": "name",
    "id": 0
}
```

**Producers**

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header:

- application/json

**Responses**

200 OK **Event**
default unexpected error **Error**

**PUT /scenarios/{id}**

Update scenario (**updateScenario**)

**Path parameters**

- **id** (required)
  
  *Path Parameter* — Model id

**Consumes**

This API call consumes the following media types via the Content-Type request header:

- application/json

**Request body**

body **NewScenario** (required)

*Body Parameter* — Scenario attributes to be updated

**Return type**

**Scenario**

**Example data**

Content-Type: application/json

```json
{
    "name": "name",
    "id": 0
}
```

**Producers**

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.
PUT /scenarios/{scenarioid}/timeseries/{id}

Update a time series of a specific scenario (updateTimeseries)

Path parameters
- scenarioid (required)
  Path Parameter — Scenario id
- id (required)
  Path Parameter — Timeseries id

Consumes
This API call consumes the following media types via the Content-Type request header:
- application/json

Request body
- body NewTimeseries (required)
  Body Parameter — Timeseries to be updated

Return type
Timeseries

Example data
Content-Type: application/json

```
{
  "name" : "name",
  "attributes" : "attributes",
  "id" : 0,
  "elementuuid" : "elementuuid",
  "labels" : [ "labels", "labels" ]
}
```

Producers
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses
200
OK Timeseries
default
unexpected error Error
POST /analysis/outages

Run outage analysis to check N-1 and N-2 criterion (**addOutages**)

Runs several power flows on the specified model, disabling one component at a time. The components to be disabled are listed as parameter. Static power set points or stochastic data can be used as input.

**Consumes**
This API call consumes the following media types via the Content-Type request header:

- application/json

**Request body**

body **NewOutage** (required)

**Body Parameter** — Powerflow analysis to be added

**Return type**

**Analysis**

**Example data**

Content-Type: application/json

```
{
   "modelid" : 6,
   "name" : "name",
   "id" : 0,
   "type" : "type",
   "status" : "status"
}
```

**Produces**

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

**Responses**

200
OK **Analysis**
default
unexpected error **Error**

POST /analysis/powerflows

Run powerflow on specified model (**addPowerflow**)

Run powerflow without grid modifications or events. Static power set points or stochastic data can be used as input. Stochastic parameters could be indicated in files separate from the CIM grid data, where file and column name should correspond to the CIM component uuid and attribute name.

**Consumes**
This API call consumes the following media types via the Content-Type request header:

- application/json

**Request body**

body **NewPowerflow** (required)

**Body Parameter** — Powerflow analysis to be added
DELETE /analysis/{id}

Delete specific analysis including results (deleteOutage)

Path parameters

id (required)

Path Parameter — Model id

Return type
Analysis

Example data
Content-Type: application/json

```
{
  "modelid": 6,
  "name": "name",
  "id": 0,
  "type": "type",
  "status": "status"
}
```

Produce
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses
200
OK Analysis
default
unexpected error Error
GET /analysis/{id}

Get specific analysis status and results (getAnalysis)

Path parameters

id (required)
Path Parameter — Model id

Return type
inline_response_200_1

Example data
Content-Type: application/json

{
    "status": "status"
}

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json
- application/octet-stream
- text/uri-list

Responses
200 OK inline_response_200_1
default unexpected error Error

GET /analysis/{analysisid}/elements/{id}

Get results for element of model (getAnalysisElement)

Path parameters

analysisid (required)
Path Parameter — Analysis id

id (required)
Path Parameter — Element id

Query parameters

attribute (optional)
Query Parameter —

Return type
ModelElementAttributes

Example data
Content-Type: application/json

{
    "name": "name",
    "attributes": {

"key": "",
"id": 0,
"type": "type",
"uuid": "uuid"
}

Produce
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses
200
OK ModelElementAttributes
default
unexpected error Error

GET /analysis/{analysisid}/elements

Get results for element type of model (getAnalysisElementType)

Path parameters

- analysisid (required)
  Path Parameter — Analysis id

- id (required)
  Path Parameter — Element id

Query parameters

- type (optional)
  Query Parameter —

- attribute (optional)
  Query Parameter —

Return type
array[ModelElementAttributes]

Example data
Content-Type: application/json

[ {
  "name": "name",
  "attributes": {
    "key": ""
  },
  "id": 0,
  "type": "type",
  "uuid": "uuid"
},
{ "name": "name",
  "attributes": {
    "key": ""
  },
  "id": 0,
  "type": "type",
]


```json
{
    "uuid" : "uuid"
}
```

**Produce**
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

**Responses**
200
OK
default
unexpected error Error

GET /analysis/outages

Get all outage analyses (getOutages)

Return all outage analyses whether they are completed already or not.

**Return type**
array[Analysis]

**Example data**
Content-Type: application/json

```
[ {
    "modelid" : 6,
    "name" : "name",
    "id" : 0,
    "type" : "type",
    "status" : "status"
}, {
    "modelid" : 6,
    "name" : "name",
    "id" : 0,
    "type" : "type",
    "status" : "status"
} ]
```

**Produce**
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

**Responses**
200
OK
default
unexpected error Error

GET /analysis/powerflows

Get all powerflows (getPowerflows)

Return all powerflows whether they are completed already or not.

**Return type**
Example data
Content-Type: application/json

[{
   "modelid": 6,
   "name": "name",
   "id": 0,
   "type": "type",
   "status": "status"
}, {
   "modelid": 6,
   "name": "name",
   "id": 0,
   "type": "type",
   "status": "status"
}]

Producers
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header:

- application/json

Responses
200
OK
default
unexpected error Error

TimeDomainSimulation

POST /simulation

Run a simulation (addSimulation)

Consumes
This API call consumes the following media types via the Content-Type request header:

- application/json

Request body

- body NewSimulation (required)
- Body Parameter — Simulation to be added

Return type
Simulation

Example data
Content-Type: application/json

{
   "name": "name",
   "id": 0
}

Producers
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses
200
OK Simulation
default
unexpected error Error

DELETE /simulations/{id}

Delete a simulation including results (deleteSimulation)

Path parameters
id (required)
Path Parameter — Simulation id

Return type
Simulation

Example data
Content-Type: application/json

```json
{
  "name" : "name",
  "id" : 0
}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses
200
OK Simulation
default
unexpected error Error

GET /simulations/{id}

Get a simulation status and results (getSimulation)

Path parameters
id (required)
Path Parameter — Simulation id

Return type
Simulation

Example data
Content-Type: application/json
GET /simulation/{simid}/elements/{id}

Get simulation results for element of model (getSimulationElement)

Path parameters

simid (required)
* Path Parameter — Simulation id

id (required)
* Path Parameter — Element id

Query parameters

attribute (optional)
* Query Parameter —

Return type

ModelElementAttributes

Example data
Content-Type: application/json

```json
{
    "name": "name",
    "attributes": {
        "key": "",
        "id": 0,
        "type": "type",
        "uuid": "uuid"
    }
}
```

Produce

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

* application/json

Responses

200
OK ModelElementAttributes
default

unexpected error Error
GET /simulation

Get all simulations (getSimulations)

Return type
array[Simulation]

Example data
Content-Type: application/json

```
[ 
   { 
      "name" : "name",
      "id" : 0
   }, 
   { 
      "name" : "name",
      "id" : 0
   }
]
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses
200
OK
default
unexpected error Error

Users

POST /users

Add a user (addUser)

Consumes
This API call consumes the following media types via the Content-Type request header:

- application/json

Request body

body NewUser (required)

Body Parameter – User to be added

Return type
User

Example data
Content-Type: application/json

```
{
   "role" : "role",
   "mail" : "mail",
   "name" : "name",
```
DELETE /users/{id}

Delete user (deleteUser)

Path parameters

id (required)

Path Parameter — User id

Return type

User

Example data

Content-Type: application/json

```json
{
  "role": "role",
  "mail": "mail",
  "name": "name",
  "active": true,
  "id": 0
}
```

GET /users/{id}

Get individual user (getUser)

Path parameters

id (required)

Path Parameter — User id
Return type
User

Example data
Content-Type: application/json

```json
{
  "role": "role",
  "mail": "mail",
  "name": "name",
  "active": true,
  "id": 0
}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses
200
OK User
default
unexpected error Error

GET /users

Get all users (getUsers)

Return type
inline_response_200

Example data
Content-Type: application/json

```json
{
  "users": [ {
    "role": "role",
    "mail": "mail",
    "name": "name",
    "active": true,
    "id": 0
  }, {
    "role": "role",
    "mail": "mail",
    "name": "name",
    "active": true,
    "id": 0
  } ]
}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses
PUT /users/{id}

Update user (updateUser)

Path parameters
id (required)
*Path Parameter* — User id

Consumes
This API call consumes the following media types via the Content-Type request header:

- application/json

Request body
body *UserUpdate* (required)
*Body Parameter* — User to be updated. Anything except for the id can be changed, role can only be changed by admin.

Return type
*User*

Example data
Content-Type: application/json

```json
{
    "role": "role",
    "mail": "mail",
    "name": "name",
    "active": true,
    "id": 0
}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses
200
OK *User*
default
unexpected error *Error*

Models

[ Jump to Methods ]

Table of Contents

1. Analysis
2. Control
Analysis

id (optional)

**Integer**

name (optional)

**String** Name of analysis case

type (optional)

**String** Type of analysis, e.g. powerflow

modelid (optional)

**Integer** Model to consider for analysis

status (optional)

**String** Indicate if analysis is completed or not

Control

id (optional)

**Integer**

name (optional)

**String** Name of controller instance

Error

code

**Integer** format: int32

message

**String**

Event
id (optional)

**Integer**

name (optional)

**String** Event name

**Model**

id (optional)

**Integer**

name (optional)

**String** Name of simulation model

**ModelElement**

id (optional)

**Integer**

uuid (optional)

**String** CIM uuid of model element

name (optional)

**String** Name of model element

type (optional)

**String** CIM type of model element

**ModelElementAttributes**

id (optional)

**Integer**

uuid (optional)

**String** CIM uuid of model element

name (optional)

**String** Name of model element

type (optional)

**String** CIM type of model element

attributes (optional)

**map[String, null]** Element attributes, e.g. strings and numbers

**ModelElementUpdate**

name (optional)

**String** Name of model element

type (optional)

**String** CIM type of model element

param (optional)

**map[String, null]** Element attributes, e.g. strings and numbers

**ModelUpdate**

name (optional)

**String**
NewControl

- name: String
- method: String
  - param (optional): String
    map[String, null] attribute map, e.g. strings and numbers

NewEvent

- name: String
- time (optional): BigDecimal
- param (optional): String
  map[String, null] attribute map, e.g. strings and numbers

NewModel

- name: String

NewModelElement

- name: String
  Name of model element
- type: String
  CIM type of model element
- param (optional): String
  map[String, null] Element attributes, e.g. strings and numbers

NewOutage

- name: String
- modelid: Integer
  Model to consider for the powerflow calculation
- nm1List: String
  List of N-1 components
- nm2List (optional):
  List of N-2 components
- param (optional):
  map[String, null] attribute map, e.g. strings and numbers

NewPowerflow

- name: String
modelid
*Integer* Model to consider for the powerflow calculation

param (optional)
*map[String, null]* attribute map, e.g. strings and numbers to define solver settings etc.

NewScenario

name
*String*

model (optional)
*Integer* Name of model associated to this scenario

param (optional)
*map[String, null]* attribute map, e.g. strings and numbers

NewSimulation

name
*String*

starttime (optional)
*BigDecimal*

endtime (optional)
*BigDecimal*

timestep (optional)
*BigDecimal*

solver (optional)
*String*

param (optional)
*String*

param (optional)
*map[String, null]* attribute map, e.g. strings and numbers

NewTimeseries

elementuuid (optional)
*String* CIM uuid of model element associated to time series

attributes (optional)
*String* (element) attribute associated to time series

name
*String*

labels (optional)
*array[String]* time series labels and tags

NewUser

name
*String*

password
*String*

mail
*String*

role
String

Scenario

id (optional)
Integer

name (optional)
String

Simulation

id (optional)
Integer

name (optional)
String Name of simulation case

Timeseries

id (optional)
Integer

elementuuid (optional)
String CIM uuid of model element associated to time series

attributes (optional)
String element attribute associated to time series

name (optional)
String

labels (optional)
array[String] time series labels and tags

User

active (optional)
Boolean Indicating status of user (false means user is inactive and should not be able to login)

id (optional)
Integer

mail (optional)
String Mail of user

role (optional)
String Role of user

name (optional)
String Username of user

UserUpdate

active (optional)
String

mail (optional)
String

old_password (optional)
String
password (optional)
String
role (optional)
String
name (optional)
String

inline_response_200

users (optional)
array[User]

inline_response_200_1

status (optional)
String Return status if not completed