updated api doc closes #28
Jakob Taraben authored 3 days ago

`c1f8f564`

apidoc-rest.md 14.6 KB

Lynks REST API

This is the documentation for the usage of the Lynks REST API version 2.2.0.

1 Comments

- all ids have to follow the standard uuid conventions
- all unknown JSON keys given into the data of PUT or POST commends will be ignored
- 3rd-level objects (DataResource, Relatum) need the parent id as an additional JSON key when executing POST
- All additional data representing objects from the lynks model have to follow the corresponing json schema as explained in `Lynks/readme.md` otherwise the PUT or POST method is rejected with an error
- All responses containing detailed object descriptions contain the json object following the defined schema
- the general convention the api is following connects key words as following:
  - GET for requesting data,
  - POST adding data,
  - PUT for overwriting data,
  - PATCH for partly updating data and
  - DELETE for removing data

2 Multimodel

2.1 GET

Returns the JSON object of the requested multi model except the detailed snapshot, model and link lists. Snapshots and models are represented by a string array of their ids instead.

```
GET /:id
```

Parameters

Name	Type	Description

Name	Type	Description
id	String	The multi model id

Examples

CURL example:

```
curl -X GET http://localhost:8080/974345f3-0000-40da-9c10-4f241e0caf9d/
```

Response

Success-Response (example):


```
{
  "models": [
    "974345f3-1111-40da-9c10-4f241e0caf9d",
    "974345f3-2222-40da-9c10-4f241e0caf9d",
    "974345f3-3333-40da-9c10-4f241e0caf9d",
    "974345f3-4444-40da-9c10-4f241e0caf9d",
    "974345f3-7777-40da-9c10-4f241e0caf9d"
  ],
  "linkModels": [
    "5c6bfbef-2b59-48c8-9271-40dc4bfe1364"
  ],
  "snapshots": [
    "snap2",
    "snap1"
  ],
  "metaData": {
    "entries": [
      {
        "key": "ContainerType",
        "value": "Inspection"
      }
    ]
  },
  "transformation": {
    "matrix": [
      [
        1.0,
        0.0,
        0.0,
        0.0
      ],
      [
        0.0,
        1.0,
        0.0,
        0.0
      ],
      [
        0.0,
        0.0,
        1.0,
        0.0
      ],
      [
        0.0,
        0.0,
        0.0,
        1.0
      ]
    ]
  },
  "id": "974345f3-0000-40da-9c10-4f241e0caf9d",
  "topics": [
    "Point cloud data",
```

```
    "UAS",
    "Images",
    "Finite element model",
    "Continuous sensor data",
    "Industry Foundation Classes",
    "BIM"
  ]
}
```

Error Response

Error-Response (example):

```
   <html>
<head>
<meta http-equiv="Content-Type" content="text/html;charset=utf-8"/>
<title>Error 404 Not Found</title>
</head>
<body><h2>HTTP ERROR 404</h2>
<p>Problem accessing /974345f3-0000-40da-9c10-4f241e0caf8a/. Reason:
<pre>    Not Found</pre></p><hr><a href="http://eclipse.org/jetty">Powered by Jetty:// 9.4.3.v20170317</a><

</body>
</html>
```

2.2 POST

This function is not available for multi models in the current state.

2.3 PUT

This function is not available for multi models in the current state.

3 Model

3.1 GET

Get the list of all model ids contained in the multi model, link model or snapshot as json list or a detailed description of a model data object as json map.

Model id list

```
GET /:multimodel/models/:related
```

Parameters

Name	Type	Description
multimodel	String	The multimodel id
related	String	Optional parameter representing a snapshot or link model id

Examples

Get all model ids from the multi model:

```
   curl -X GET http://localhost:8080/974345f3-0000-40da-9c10-4f241e0caf9d/models
```

To get all model ids from a snapshot with the id *24b40b06-7015-11e9-a923-1681be663d3e*:

```
   curl -X GET
http://localhost:8080/974345f3-0000-40da-9c10-4f241e0caf9d/models/24b40b06-7015-11e9-a923-1681be663d3e
```

Response

```
[
  "974345f3-1111-40da-9c10-4f241e0caf9d",
  "974345f3-2222-40da-9c10-4f241e0caf9d",
  "974345f3-3333-40da-9c10-4f241e0caf9d",
  "974345f3-4444-40da-9c10-4f241e0caf9d",
  "974345f3-7777-40da-9c10-4f241e0caf9d"
]
```


Model object

```
GET /:multimodel/models/:id
```

Parameters

Name	Type	Description
multimodel	String	The multimodel id
id	String	The model id as defined in the multimodel

Examples

```
   curl -X GET
http://localhost:8080/974345f3-0000-40da-9c10-4f241e0caf9d/models/974345f3-3333-40da-9c10-4f241e0caf9d
```

Response


```
   {
  "resources": [
    "974345f3-3334-40da-9c10-4f241e0caf9d",
    "974345f3-3335-40da-9c10-4f241e0caf9d"
  ],
  "topics": [
    "Finite element model"
  ],
  "id": "974345f3-3333-40da-9c10-4f241e0caf9d",
  "type": "Finite element model",
  "metaData": {
    "entries": [
      {
        "key": "fileFormat",
        "value": ".dat"
      },
      {
        "key": "software",
        "value": "Sofistik"
      }
    ]
  },
  "transformation": {
    "matrix": [
      [
        -0.870893,
        -0.491321,
        -0.012193,
        356.291534
      ],
      [
        -0.491272,
        0.870979,
        -0.006878,
        1118.672119
      ],
      [
        0.013999,
```

```
        0.0,
        -0.999902,
        201.69783
      ],
      [
        0.0,
        0.0,
        0.0,
        1.0
      ]
    ]
  }
}
```

3.2 POST

Creates a new model in the multi model using the additional data to create the model object. The id defined in the model has to apply the requirements and should not be already assigned to another model. The additional data is checked against the `modeldata.json` schema to ensure that the object is formed well. If the post was successful the response is `null`. If some errors occured, the response is a json map informing about the error under the key `"error"`.

```
POST /:multimodel/models
```

Parameters

Name	Type	Description
multimodel	String	The multimodel id

Examples

```
   curl -X POST -H "Content-Type: application/json" -d '{"topics": [ "Industry Foundation Classes", "BIM"
 ], "id": "974345f3-7777-40da-9c10-4f241e0caf9d", "type": "IFC model", "metaData": { "entries": [ { "key"
: "software", "value": "Solibri" } ] }, "label": { "map" : { "default" : "architecture model"   } } }'
http://localhost:8080/974345f3-0000-40da-9c10-4f241e0caf9d/models
```

Response

```
        null
```

Error response

```
        { "error" : "error description message" }
```

3.3 PUT

Overwrites an existing model in the multi model using the additional data. The id defined in the model has to be already assigned to an existing model, otherwise the put is rejected. The additional data is checked against the `modeldata.json` schema to ensure that the object is formed well. If the post was successful the response is `null`. If some errors occured, the response is a json map informing about the error under the key `"error"`.

```
PUT /:multimodel/models
```

The response behaviour and additional data structure is analogous to the POST.

4 Resource

4.1 GET

Get the list of all resource ids contained in the multi model, link model or snapshot as json list or a detailed description of a data resource object as json map.

Resource id list

```
GET /:multimodel/resources/:related
```

Parameters

Name	Type	Description
multimodel	String	The multimodel id
related	String	Optional parameter representing a snapshot or link model id

Examples

Get all resource ids from the multi model:

```
   curl -X GET http://localhost:8080/974345f3-0000-40da-9c10-4f241e0caf9d/resources
```

To get all resource ids from a snapshot with the id *24b40b06-7015-11e9-a923-1681be663d3e*:

```
   curl -X GET
http://localhost:8080/974345f3-0000-40da-9c10-4f241e0caf9d/resources/24b40b06-7015-11e9-a923-1681be663d3e
```

Response

```
[
  "974345f3-1112-40da-9c10-4f241e0caf9d",
  "974345f3-2223-40da-9c10-4f241e0caf9d",
  "974345f3-3334-40da-9c10-4f241e0caf9d",
  "974345f3-3335-40da-9c10-4f241e0caf9d",
  "974345f3-4445-40da-9c10-4f241e0caf9d",
  "974345f3-4446-40da-9c10-4f241e0caf9d",
  "974345f3-4447-40da-9c10-4f241e0caf9d"
]
```

Resource object

```
GET /:multimodel/resources/:id
```

Parameters

Name	Type	Description
multimodel	String	The multimodel id
id	String	The resource id as defined in the multimodel

Examples

```
   curl -X GET
http://localhost:8080/974345f3-0000-40da-9c10-4f241e0caf9d/resources/974345f3-3333-40da-9c10-4f241e0caf9d
```

Response

```
{
  "id": "974345f3-3334-40da-9c10-4f241e0caf9d",
  "location": "scherkonde.dat",
  "formatType": "Sofistik DAT file",
  "dataOf": "974345f3-3333-40da-9c10-4f241e0caf9d",
  "current": true
}
```

4.2 POST

Creates a new data resource in the multi model using the additional data to create the resource object. The id defined in the resource

has to apply the requirements and should not be already assigned to another resource. The additional data is checked against the `resource.json` schema to ensure that the object is formed well. Additionally, the json object of the resource has to hold an attribute `"model_id"` with the model id of an existing model to assign the resource. If the post was successful the response is `null`. If some errors occured, the response is a json map informing about the error under the key `"error"`.

```
POST /:multimodel/resources
```

Parameters

Name	Type	Description
multimodel	String	The multimodel id

Examples

```
   curl -X POST -H "Content-Type: application/json" -d '{ "id": "974345f3-7778-40da-9c10-4f241e0caf9d", "
location": "any_buidling.ifc",  "formatType": "Industry Foundation Classes", "dataOf": "974345f3-7777-40d
a-9c10-4f241e0caf9d", "model_id": "974345f3-7777-40da-9c10-4f241e0caf9d" }'
http://localhost:8080/974345f3-0000-40da-9c10-4f241e0caf9d/resources
```

Response

```
        null
```

Error response

```
        { "error" : "error description message" }
```

4.3 PUT

Creates a new data resource in the multi model using the additional data to create the resource object. The id defined in the resource has to apply the requirements and should not be already assigned to another resource. The additional data is checked against the `resource.json` schema to ensure that the object is formed well. Additionally, the json object of the resource has to hold an attribute `"model_id"` with the model id of an existing model to assign the resource. If the post was successful the response is `null`. If some errors occured, the response is a json map informing about the error under the key `"error"`.

```
PUT /:multimodel/resources
```

The response behaviour and additional data structure is analogous to the POST.

5 Snapshot

5.1 GET

Get the list of all snapshot ids contained in the multi model.

Snapshot id list

```
GET /:multimodel/snapshots/
```

Parameters

Name	Type	Description
multimodel	String	The multimodel id

Examples

Get all snapshot ids from the multi model:

```
   curl -X GET http://localhost:8080/974345f3-0000-40da-9c10-4f241e0caf9d/snapshots
```

Response

```
[
  "6d866f32-7096-11e9-a923-1681be663d3e",
  "754820c6-7096-11e9-a923-1681be663d3e",
]
```

Snapshot object

```
GET /:multimodel/snapshots/:id
```

Parameters

Name	Type	Description
multimodel	String	The multimodel id
id	String	The snapshot id as defined in the multimodel

Examples

```
   curl -X GET
http://localhost:8080/974345f3-0000-40da-9c10-4f241e0caf9d/snapshots/754820c6-7096-11e9-a923-1681be663d3e
```

Response

```
{
  "exclude": [
    "974345f3-3334-40da-9c10-4f241e0caf9d"
  ],
  "topics": [
    "Continuous sensor data",
    "Finite element model"
  ],
  "timestamp": 1548979201,
  "id": "snap1"
}
```

5.2 POST

Creates a new snapshot in the multi model using the additional data to create the resource object. The id defined in the snapshot has to apply the requirements and should not be already assigned to another snapshot. The additional data is checked against the `snapshot.json` schema to ensure that the object is formed well. If the post was successful the response is `null`. If some errors occured, the response is a json map informing about the error under the key `"error"`.

```
POST /:multimodel/snapshots
```

Parameters

Name	Type	Description
multimodel	String	The multimodel id

Examples

```
   curl -X POST -H "Content-Type: application/json" -d '{ "exclude": [ "974345f3-3334-40da-9c10-4f241e0ca
f9d" ], "topics": [ "Continuous sensor data", "Finite element model" ], "timestamp": 1548979201, "id": "7
54820c6-7096-11e9-a923-1681be663d3e" }'
http://localhost:8080/974345f3-0000-40da-9c10-4f241e0caf9d/snapshots
```

Response

```
        null
```

Error response

```
        { "error" : "error description message" }
```

5.3 PUT

Creates a new snapshot in the multi model using the additional data to create the resource object. The id defined in the snapshot has to apply the requirements and should not be already assigned to another snapshot. The additional data is checked against the `snapshot.json` schema to ensure that the object is formed well. If the post was successful the response is `null`. If some errors occured, the response is a json map informing about the error under the key `"error"`.

```
PUT /:multimodel/snapshots
```

The response behaviour and additional data structure is analogous to the POST.