Model manipulation

On this page

Teamwork Cloud operates at the EMF level, while MagicDraw operates at the UML level, which is on top of EMF. REST API, which is the Teamwork Cloud API, also operates at the EMF level.

Therefore, Teamwork Cloud is not affected by UML-specific implementation, and all derived properties may not be saved in the Teamwork Cloud database. The best-known derived property is *owner*. This is the result when there is no such "*setOwner"* in REST API.

Even though all non-derived properties are saved in the Teamwork Cloud database, they are saved as raw EMF data. The user is required to have knowledge about UML models to manipulate data. For example, an applied stereotype is saved as a hierarchy of *InstanceSpecification*, *Slot*, and *ValueSpecification*.

This section describes how to traverse into a UML model and create some elements in it.

MagicDraw project

In Teamwork Cloud terminology, a MagicDraw project is referred to as a *resource*. Although REST API can be used to create a Teamwork Cloud resource, it is only a bare project MagicDraw cannot read. A MagicDraw project is composed of many MagicDraw-specific meta-data such as project options, primary project, and used projects configuration. The best way to create a MagicDraw project is to create it by MagicDraw and submit it to the Teamwork Cloud server.

MagicDraw primary project

To traverse through a UML model, the primary project must be identified. The model data starts at the revision level. Issuing GET to */revisions/{revisionId}* shows the first-level object in the revision. UUIDs of the first-level object are listed in **rootObjectIDs**.

```
{
        "commitType": "NORMAL",
    "branchID": "../..",
    "resourceID": "../../..",
    "@base": "http://127.0.0.1:8111/osmc/resources/4615e8fa-81e5-40e0-a51b-8496a48caf18/revisions/5/elements",
    "author": "Administrator",
    "@type":
        [
            "RDFSource",
            "kerml:Revision"
    ],
    "pickedRevision": -1,
    "description": "Branch \"xx\" created",
    "@context": "http://127.0.0.1:8111/osmc/schemas/revision",
    "directParent": 3,
    "dependencies": [],
    "rootObjectIDs":
        [
              "429f969a-5c81-45f4-94af-8cf983f22950",
                "ec6060a3-f3d9-482b-93a3-32af9e19202c",
                "ca9a0235-f0f7-46b7-a142-e79a67c2d00d",
                "f7c3ae92-af44-4dab-8163-a199ca05c006",
                "ba3d0700-1062-4baf-a1df-a55a4f31ce54",
                "0f14cd2d-2fd0-4523-950c-627d59e1a43d",
              "7cd22dea-aaf8-4e08-bd67-5bd975c3f06a",
             "af1042fa-8b1b-4cf2-bb7d-98dd1b881da3",
                "6d24e5e7-cdff-4e9e-85b8-28b3088f85b6",
                "243020e5-da6c-4896-b32a-fcba0e93ac8d",
                "f7449238-5cd1-41eb-9025-040210b02d93",
              "4d2459a1-49dc-4eb7-aa82-9bbb4a76b038",
                "b242613d-957e-4aec-9333-e5938f50b2ab",
                "9b953064-e422-4391-b7d9-43a2d4f14a32",
                "fc997cfd-23c5-4d0b-9953-06667dcde0dd",
                "29d9416b-ead5-4a9d-b530-b23de836f1b8",
                "7af3f24b-2da9-4b31-94b3-a87f15747296"
        ],
        "createdDate": "1533051367",
        "ID": "",
    "artifacts": "artifacts"
},
```


Among these entries, the first element with the **esiproject:EsiProject** type specifies the primary project named **main decomposition project**. Other **esiproject:EsiProject** objects are the models (used projects) in a MagicDraw project. The main decomposition project is element **429f969a**. You need to load element **429f969a** and all their children. An element is represented in the JSONLD format. The attributes of the elements are in **kerml:esiData**. There will be an element named **UML Model**. The below code fragment shows the attributes of this element.


```
"kerml:esiData":
{
        "name": "UML Model",
    "namespace": "com.nomagic.magicdraw.uml_model",
        "project":
        {
                "@id": "429f969a-5c81-45f4-94af-8cf983f22950"
        },
        "internalVersion": "1",
        "version": "17.0",
        "sections":
        [
                {
                        "@id": "b3e68e8e-2253-4726-8d05-d8f74fd0ba5a"
        }
        ]
},
```


Take the first **esiproject:EsiDataSection** from the feature's sections list, which is **b3e68e8e**. The excerpt of the **b3e68e8e** data is shown below.

```
{
        "kerml:name": "model",
        "@base": "http://127.0.0.1:8111/osmc/resources/4615e8fa-81e5-40e0-a51b-8496a48caf18/revisions/5
/elements",
        "kerml:nsURI": "http://www.nomagic.com/ns/magicdraw/esiproject/1.0",
        "@type": "esiproject:EsiDataSection",
    "kerml:owner":
        {
                "@id": "429f969a-5c81-45f4-94af-8cf983f22950"
    },
    "kerml:revision": "http://127.0.0.1:8111/osmc/resources/4615e8fa-81e5-40e0-a51b-8496a48caf18/revisions/5",
    "@context":
        {
                "esiproject:EsiDataSection": "http://127.0.0.1:8111/osmc/schema/esiproject/2014345
/EsiDataSection",
                "kerml": "http://127.0.0.1:8111/osmc/schema/kerml/20140325"
    },
    "kerml:ownedElement": [],
    "kerml:modifiedTime": "20180731223607ICT",
    "kerml:esiData":
        {
                "featuredBy":
                {
                        "@id": "c9256728-4617-4097-8e9e-dc63e2823bf8"
                },
                "rootElements":
                [
                        "ca9a0235-f0f7-46b7-a142-e79a67c2d00d"
                ],
                "name": "model",
              "project":
                {
                        "@id": "429f969a-5c81-45f4-94af-8cf983f22950"
              },
                "properties": []
        },
        "kerml:resource": "http://127.0.0.1:8111/osmc/resources/4615e8fa-81e5-40e0-a51b-8496a48caf18",
    "kerml:esiID": "b3e68e8e-2253-4726-8d05-d8f74fd0ba5a",
        "@id": "#b3e68e8e-2253-4726-8d05-d8f74fd0ba5a"
}
```


The first ID in the **rootElements** list of the data section will be the ID of the UML model root element in the resource. The root element should always be of the **uml:Package** type or derived from it.

Traversing to a MagicDraw model

Once a root project is retrieved, the whole hierarchy can be retrieved layer by layer. An element can be retrieved from */elements/{elementId}*. Child UUIDs of the resulting element are in **kerml:ownedElement**.

Multiple elements can be retrieved by POST to */elements*. In this case, you need to put UUID in the **uuid.txt** file to load elements.

```
4eeb2e6d-9cbc-4f59-9d74-69263e52ba54,520bc74d-b5b4-40d5-87cb-9d0b8aba5d2e,7a057ae7-6281-462d-9dcd-de9931633f5c,
a6212656-5ad1-4de6-a47f-db656c2c25fd
```

The command to load those elements is:

```
curl -v -H "Content-Type: text/plain" -X POST -s --cookie cookie.txt --insecure -d @uuid.txt --insecure
"https://server:8111/osmc/resources/${projectId}/elements"
```

In most cases, loading multiple elements at once is substantially faster than loading them one by one.

Creating elements

To use REST API to create elements, you are required to have knowledge about UML models, especially how to set the owner. A combination of parent-child type yields differences in the owner attribute. For example, if you want to create a class under a class, you need to set the **UMLClass** attribute in the child class. For a class under a package, you need to set **owningPackage** in the child class. The easiest way to know which attribute is needed is to try creating it in MagicDraw and get it in REST API.

There are two URLs to create element(s), */resources/{resourceId}/elements* and */resources/{resourceId}/elements{elementId}*. Please note that the *element Id* in the latter form is not the parent. An Ecore is needed to create an element. It can be specified in the following ways:

- Specified from the *elementId* in */resources/{resourceId}/elements/{elementId}*. This form is used in most cases, except from creating the first element in the project. The new element data type will be in the same namespace of this specified element.

```
{
        "@type": "uml:Class",
        "kerml:nsURI": "http://www.nomagic.com/magicdraw/UML/2.5.1.1",
        "kerml:esiData":
        {
                "owningPackage":
                {
                        "@id": "757be712-f397-404d-a5ff-b97567eb240f"
        },
        "name": "c3"
    }
}
```

- Specified from **kerml:nsURI**.
- Specifying the whole Ecore in **kerml:ecore**. This mode is rarely used.

```
{
        "@type": "ikml:Container",
        "kerml:esiData":
        {
                "name": "txdhdhdhd",
                "uri": "http://www.chula.ac.th",
    },
        "kerml:ecore": "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\r\n<ecore:EPackage xmi:version=\"2.0
\" xmlns:xmi=\"http://www.omg.org/XMI\"
        xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"\r\n
        xmlns:ecore=\"http://www.eclipse.org/emf/2002/Ecore\"
        name=\"ikml\" nsURI=\"http://www.nomagic.com/ikml/1.0\"
        nsPrefix=\"ikml\">\r\n
        <eClassifiers xsi:type=\"ecore:EClass\" name=\"Element\" abstract=\"true\"><\/eClassifiers>\r\n<\
/ecore:EPackage>\r\n"
}
```

The following script creates an element:

```
cat file.txt | curl -v -H "Content-Type: application/ld+json" -X POST -s --cookie cookie.txt --insecure -d @-
https://server:8111/osmc/resources/${projectId}/elements
```

Multiple elements can be created at once by specifying the query parameter *batch=true* in the URL. The script is shown below:

```
cat file.txt | curl -v -H "Content-Type: application/ld+json" -X POST -s --cookie cookie.txt --insecure -d @-
"https://server:8111/osmc/resources/${projectId}/elements/${elementId}?batch=true"
```

Applied stereotypes

Applied stereotypes stored in the **appliedStereotype** attribute are the arrays containing reference objects to stereotype objects. The code below shows an example of an applied stereotype.

```
{
    "@id": "#bdeffbc9-daae-407c-8663-d3b1a0ed6077",
        "kerml:esiID": "bdeffbc9-daae-407c-8663-d3b1a0ed6077",
        "@type": "uml:Class",
        "kerml:esiData": {
                "ID": "_2021x_2_3360135_1630482378802_586147_17",
                "mdExtensions": [],
                "_representationText": null,
                "taggedValue": [
                        {
                                "@id": "edaa1dd8-583d-4a8b-8974-3111aef8d47b"
            }
                ],
                "appliedStereotype": [
                        {
                                "@id": "8dbdc804-5eed-424d-9812-28e183a8dfdc"
            }
                ]
        }
}
```

To apply the stereotype, set PATCH to the element you are working on, as shown below.

```
{
        "@context": {
                "@base": "http://127.0.0.1:8111/osmc/workspaces/59068af0-a7ae-4663-
b972-3c58a2fde23f/resources/b0585059-c4e3-4d82-90d9-c5386045bb2e/branches/
5dfb978f-0984-47ea-aaec-bdbb5a945687/elements/",
                "kerml": "http://127.0.0.1:8111/osmc/schema/kerml/20140325",
                "uml:Class": "http://127.0.0.1:8111/osmc/schema/uml/2014345/Class"
        },
        "kerml:esiData": {
                "appliedStereotype": [
                        {
                                "@id": "8dbdc804-5eed-424d-9812-28e183a8dfdc"
            }
                ]
    }
}
```

The UUID **8dbdc804** is an ID of the applied stereotype whose content is applied via the following command where the UUID 8df9f306 is an ID of the element to which the stereotype is applied:

```
$ curl -v -H "Content-Type: application/ld+json" -X POST -s -c ..\get\cookie.txt -b
..\get\cookie.txt --insecure -d @- https://127.0.0.1:8111/osmc/resources/2663059e-8ef5-4f13-97b6-864a0353d2d0
/elements/8df9f306-c4dc-41fb-ac9f-4a87fe71610a
```

Tagged values

Tagged values stored in the **taggedValue** attribute are the arrays containing reference objects to tagged value objects. The code below shows an example of an applied stereotype and tagged values.

```
{
        "@id": "#bdeffbc9-daae-407c-8663-d3b1a0ed6077",
        "kerml:esiID": "bdeffbc9-daae-407c-8663-d3b1a0ed6077",
        "@type": "uml:Class",
        "kerml:esiData": {
                "ID": "_2021x_2_3360135_1630482378802_586147_17",
        "mdExtensions": [],
                "_representationText": null,
                "taggedValue": [
                        {
                                "@id": "d9388ec6-ddca-4c18-9a5c-409d8649731f"
            },
            {
                                "@id": "edaa1dd8-583d-4a8b-8974-3111aef8d47b"
            }
                ],
                "appliedStereotype": [
                        {
                                "@id": "8dbdc804-5eed-424d-9812-28e183a8dfdc"
            }
        ]
    }
}
```

To add a tagged value, apply the stereotype to the target element first.

> **Note**
> The model is incorrect if tagged values do not have their corresponding stereotypes.

To apply the tagged value, use the POST request with the following code:

```
{
        "@type": "uml:StringTaggedValue",
        "kerml:esiData": {
                "tagDefinition": {
                        "@id": "657a3511-d012-4134-af5d-411a6df78745"
        },
                "taggedValueOwner": {
                        "@id": "515073dd-fe25-4327-bfc2-6466375a50f7"
        },
                "value": [
                        "somevaluefromrest"
        ]
    }
}
```

The command will be:

```
curl -v -H "Content-Type: application/ld+json" -X POST -s -c ..\get\cookie.txt -b ..\get\cookie.txt
--insecure -d @- https://127.0.0.1:8111/osmc/resources/2663059e-8ef5-4f13-97b6-864a0353d2d0/elements/8df9f306-
c4dc-41fb-ac9f-4a87fe71610a
```

Related pages

- REST APIs
- General convention
- Authentication
- MagicDraw-specific extensions
- Developer Guide