UML to YANG Tool User Guide

Contents

1 Running the UML-YANG Tool

The UML-YANG mapping tool translates a UML (Unified Modeling Language) model to a YANG model defined in RFC6020. The output YANG files can be conveniently used in REST style API definition.

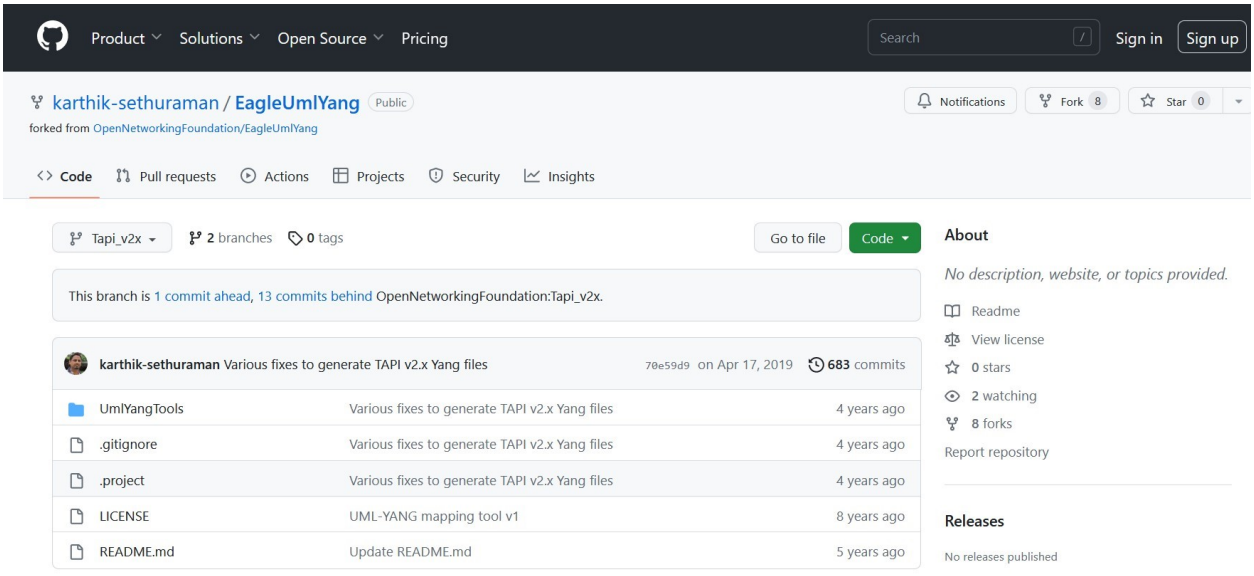
Running the UML-YANG mapping tool takes the following steps:

1. Download **NodeJS** from https://nodejs.org/en/, choose the latest version for your system.

2. Install NodeJS by double click the **node-v14.17.6-x64.msi** file. The default options are fine.

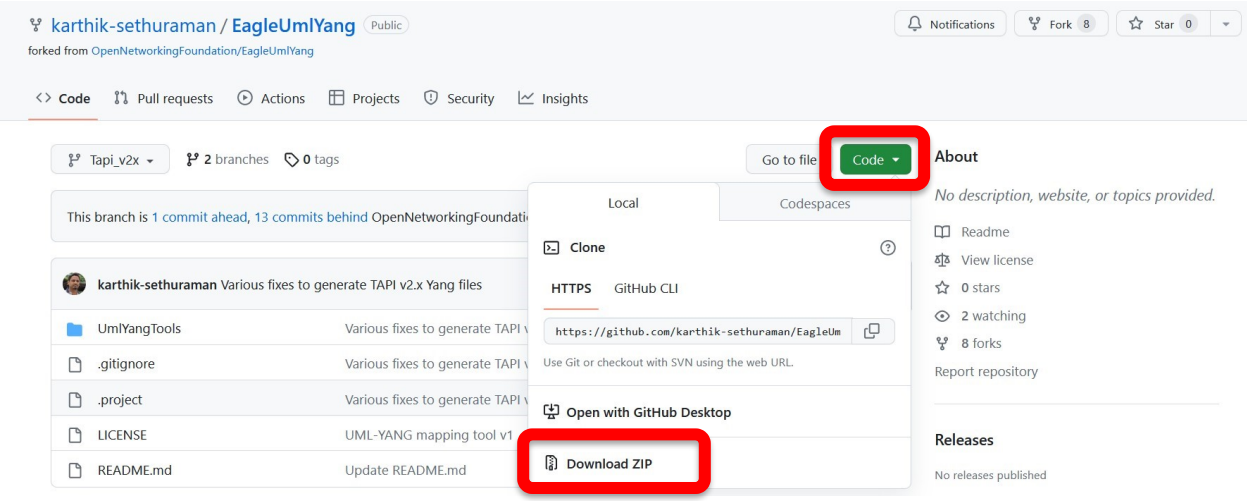
3. For the TAPI model this Eagle tool version is used:

   https://github.com/karthik-sethuraman/EagleUmlYang/tree/Tapi_v2x



4. Download the mapping tool zip in your local directory, for example:

   C:\Users*username*\EagleUmlYang-Tapi_v2x\UmlYangTools\xmi2yang



5. Open the terminal window in your system. If it's Windows, run "cmd" to open the command line window.

6. The .uml files should be copied from the directory of the Papyrus project, e.g.

   C:\Users*username*\ONF-TAPI-2.x\TAPI\UML

   to the **project** folder of the **xmi2yang** folder:

   C:\Users*username*\EagleUmlYang-Tapi_v2x\UmlYangTools\xmi2yang\project

   Note that if an ".uml" submodel depends on other ".uml" submodels, they need to be copied to the **project** folder as well. In general, it is better to copy all ".uml" submodels.

7. In the **project** folder, edit/create the **config.json** file, an example here:


```
{
 "tapi":{
   "namespace":"urn:onf:otcc:yang:",
   "organization":"ONF OTCC (Open Transport Configuration & Control) Project",
   "contact":"\r\n  Project  Web:  <https://wiki.opennetworking.org/display/OTCC/TAPI>\r\n  Project  List:  <mailto:transport-api@opennetworking.org>",
   "withSuffix":false,
   "revision": [
      {
                "date":"AutoGenerated",
                "description":"ONF Transport API - The TAPI YANG models included in this TAPI release (v2.1.1) are a *normative* part of the TAPI SDK.\r\n  - The YANG specifications have been generated from the corresponding UML model using the [ONF EAGLE UML2YANG mapping tool]\r\n  <https://github.com/OpenNetworkingFoundation/EagleUmlYang>\r\n  and further edited manually  to  comply  with  the  [ONF  IISOMI  UML2YANG  mapping  guidelines]\r\n <https://wiki.opennetworking.org/display/OIMT/IISOMI+Deliverables>\r\n  - Status of YANG model artifacts can be determined by referring to the corresponding UML artifacts.\r\n  As described in the UML models, some artifacts are considered *experimental*, and thus the corresponding YANG artifacts.\r\n  - The ONF TAPI release process does not guarantee backward compatibility of YANG models across major versions of TAPI releases.\r\n  The YANG model backward compatibility criteria are outlined in section 11 of <https://tools.ietf.org/html/rfc7950>.\r\n  YANG models included in this release are not backward compatible with previous TAPI releases.\r\n  - Changes included in this TAPI release are available in the Github repository",
                "reference":"ONF-TR-527, ONF-TR-512, ONF-TR-531, RFC 7950, RFC 6087 and ONF TAPI UML model\n"
               }
   ],
   "prefix":{
    "tapi-fm":"fm",
    "tapi-common":"com",
    "tapi-topology":"top",
    "tapi-connectivity":"con",
    "tapi-path-computation":"pat",
    "tapi-virtual-network":"vnw",
    "tapi-notification":"not",
    "tapi-oam":"oam",
    "tapi-digital-otn":"otn",
    "tapi-streaming":"str",
    "tapi-dsr":"dsr",
    "tapi-equipment":"eqp",
    "tapi-photonic-media":"pho",
    "tapi-eth":"eth"
   }
 }
}
```


8. It is possible to customize the names of the output ".yang" modules, e.g.:

   "date":"AutoGenerated"

   So obtaining:

   ***ModelModule*.uml --> *model-module*@AutoGenerated.yang**

9. The user can use the following command under the **xmi2yang** directory to run this tool.

   C:\Users*username*> cd C:\Users*username*\UML2YANG_TOOL\EagleUmlYang-Tapi_v2x\UmlYangTools\xmi2yang

   C:\Users*username*\UML2YANG_TOOL\EagleUmlYang-Tapi_v2x\UmlYangTools\xmi2yang> node main.js

10. Before running the xmi2yang tool, it is possible to check that the javascript dependencies are up-to-date:

```
C:\Users\username\UML2YANG_TOOL\EagleUmlYang-
Tapi_v2x\UmlYangTools\xmi2yang> npm install
```

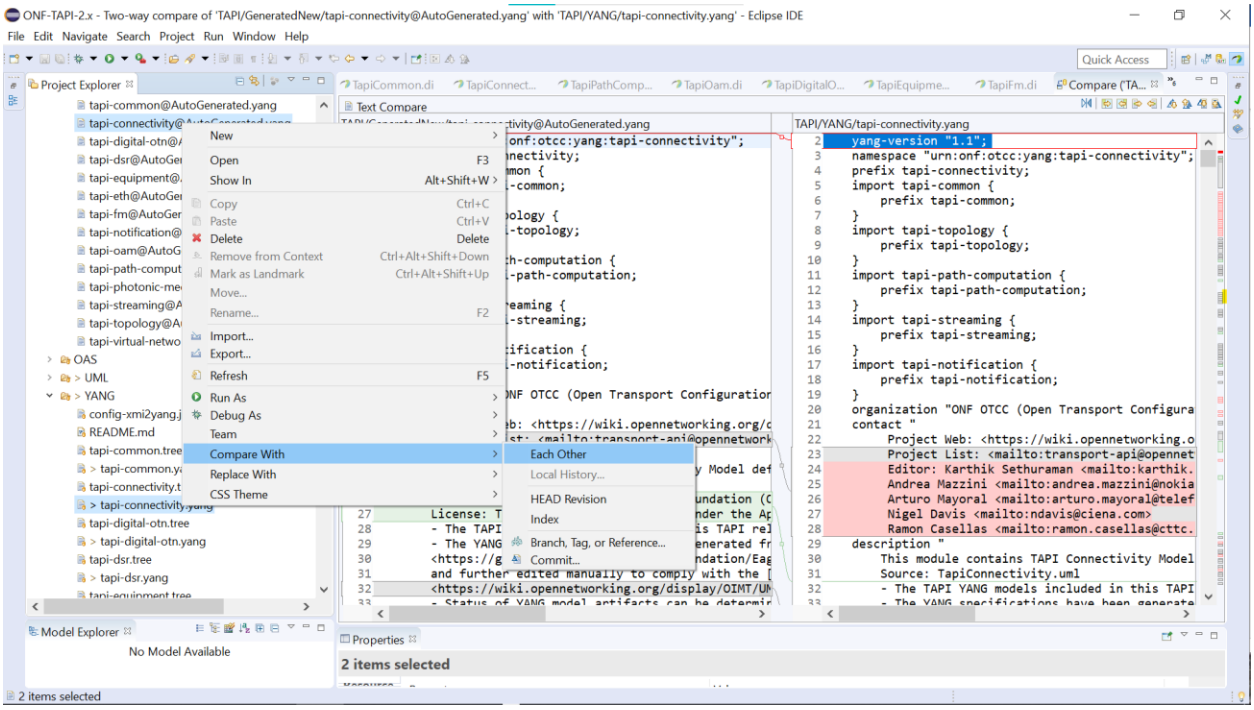
11. After running the tool, the output ".yang" files will be generated by the mapping tool in the **project** folder of the **xmi2yang** folder (i.e. together with the input .uml modules).

    - Lost the NodeJS installation? Run the **node-v14.17.6-x64.msi** with option *repair*.

12. Copy the ".yang" files in a folder for comparison with the previously edited ".yang" modules, e.g.

```
C:\Users\username\ONF-TAPI-2.x\TAPI\GeneratedNew
```

13. Compare the generated ".yang" with the existing, previously edited ".yang" modules, for example using the Eclipse/Papyrus compare function:



See the chapter 3 for the **necessary** manual editing.

14. Once the .yang modules are ready, then move to a LINUX shell, e.g. UBUNTU.

Copy from the YANG directory of the Papyrus project the .yang modules in a temporary directory, e.g. **linuxshare/yang**:

```
username@XYZ:/$ cd mnt/c/users/username/linuxshare/yang
username@XYZ:/$ dir
tapi-common.yang        tapi-equipment.yang     tapi-oam.yang               tapi-topology.yang
tapi-connectivity.yang  tapi-eth.yang           tapi-path-computation.yang  tapi-virtual-network.yang
tapi-digital-otn.yang   tapi-fm.yang            tapi-photonic-media.yang
tapi-dsr.yang           tapi-notification.yang  tapi-streaming.yang
```

Launch this script in the **linuxshare** directory for the generation of .tree modules:

```
username@XYZ:/$ cd ..
```

```
#! bash
echo "Generating TREE & SWAGGER files from YANG files"
for file in $(find ./YANG/ -name '*.yang'); do
  treefile=`echo ${file##*/} | sed 's/yang/tree/g'`;
  echo "Generating ${treefile} for ${file}"
  pyang --lint --strict -f tree -p YANG/ ${file} -o YANG/${treefile};
done
```

The .tree files are generated:

```
Generating tapi-common.tree for ./YANG/tapi-common.yang
Generating tapi-connectivity.tree for ./YANG/tapi-connectivity.yang
./YANG/tapi-connectivity.yang:388: error: the path for switch-control-uuid is config but refers to a
non-config leaf "uuid" defined at ./YANG/tapi-connectivity.yang:144 (at YANG/tapi-common.yang:177)
./YANG/tapi-connectivity.yang:398: error: the path for switch-local-id is config but refers to a non-
config leaf "local-id" defined at ./YANG/tapi-connectivity.yang:144 (at YANG/tapi-common.yang:202)
Generating tapi-digital-otn.tree for ./YANG/tapi-digital-otn.yang
Generating tapi-dsr.tree for ./YANG/tapi-dsr.yang
Generating tapi-equipment.tree for ./YANG/tapi-equipment.yang
Generating tapi-eth.tree for ./YANG/tapi-eth.yang
Generating tapi-fm.tree for ./YANG/tapi-fm.yang
Generating tapi-notification.tree for ./YANG/tapi-notification.yang
Generating tapi-oam.tree for ./YANG/tapi-oam.yang
Generating tapi-path-computation.tree for ./YANG/tapi-path-computation.yang
Generating tapi-photonic-media.tree for ./YANG/tapi-photonic-media.yang
Generating tapi-streaming.tree for ./YANG/tapi-streaming.yang
Generating tapi-topology.tree for ./YANG/tapi-topology.yang
Generating tapi-virtual-network.tree for ./YANG/tapi-virtual-network.yang
```

Then check the errors, if any.
Note that the:

```
error: the path for switch-control-uuid is config but refers to a non-config leaf "uuid"
```

is a warning which does not affect the correct generation of the YANG tree. It indicates a subtree with mismatching read/write access of the nodes. The error message doesn't show up, when the leafref is complemented with a **require-instance==false** statement. It is suggested to add the following comment: *"The require-instance statement is used as a workaround to enable write operation on a data node that refers to a read only list item".*

Note that if a class has one RW attribute, then the class id/key must be RW (**`config true`**), see section 7.21.1 of RFC 7950: *If a node has "config" set to "false", no node underneath it can have "config" set to "true".*

```
username@XYZ:/$ cd yang
username@XYZ:/$ dir
```

```
tapi-common.tree        tapi-dsr.tree        tapi-fm.tree            tapi-path-computation.tree  tapi-topology.tree
tapi-common.yang        tapi-dsr.yang        tapi-fm.yang            tapi-path-computation.yang  tapi-topology.yang
tapi-connectivity.tree  tapi-equipment.tree  tapi-notification.tree  tapi-photonic-media.tree    tapi-virtual-network.tree
tapi-connectivity.yang  tapi-equipment.yang  tapi-notification.yang  tapi-photonic-media.yang    tapi-virtual-network.yang
tapi-digital-otn.tree   tapi-eth.tree        tapi-oam.tree           tapi-streaming.tree
tapi-digital-otn.yang   tapi-eth.yang        tapi-oam.yang           tapi-streaming.yang
```

In addition, the command below generates a single ***`TapiTree.tree`*** module from all the input .yang modules:

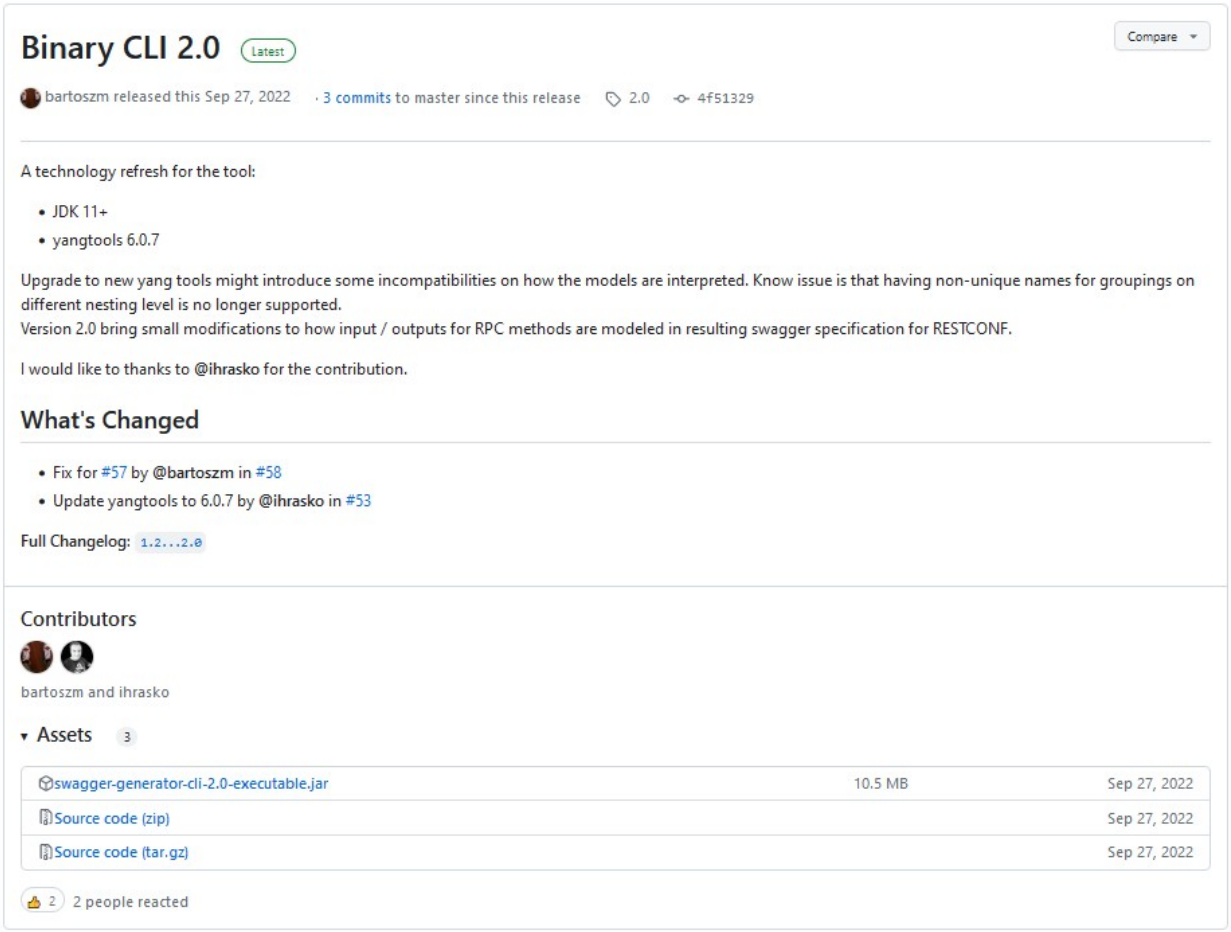
```
username@XYZ:/$ pyang -f tree YANG/*.yang -o YANG/TapiTree.tree
```

2 Running the YANG-OAS Tool

Running the YANG-OAS mapping tool takes the following steps:

1. Move to this repository:

   https://github.com/bartoszm/yang2swagger/releases/tag/2.0



Binary CLI 2.0

Latest

Compare

bartoszm released this Sep 27, 2022 · 3 commits to master since this release 2.0 4f51329

A technology refresh for the tool:

- JDK 11+
- yangtools 6.0.7

Upgrade to new yang tools might introduce some incompatibilities on how the models are interpreted. Know issue is that having non-unique names for groupings on different nesting level is no longer supported.
Version 2.0 bring small modifications to how input / outputs for RPC methods are modeled in resulting swagger specification for RESTCONF.

I would like to thanks to @ihrasko for the contribution.

What's Changed

- Fix for #57 by @bartoszm in #58
- Update yangtools to 6.0.7 by @ihrasko in #53

Full Changelog: 1.2...2.0

Contributors

bartoszm and ihrasko

Assets 3

swagger-generator-cli-2.0-executable.jar	10.5 MB	Sep 27, 2022
Source code (zip)		Sep 27, 2022
Source code (tar.gz)		Sep 27, 2022

2 2 people reacted

2. Download the swagger-generator-cli-2.0-executable.jar in a local directory, e.g. **YANG2OAS_TOOL**

3. Launch this script from the **linuxshare** directory for the .yaml generation (the script takes as input the .yang files in the **linuxshare/yang** directory)

```
#! bash
MODULES=(
[0]='tapi-common'
[1]='tapi-common tapi-notification'
[2]='tapi-common tapi-streaming'
[3]='tapi-common tapi-notification tapi-streaming tapi-topology'
[4]='tapi-common tapi-notification tapi-streaming tapi-topology tapi-path-computation tapi-
connectivity tapi-equipment'
[5]='tapi-common tapi-notification tapi-streaming tapi-topology tapi-virtual-network'
[6]='tapi-common tapi-notification tapi-streaming tapi-topology tapi-path-computation'
[7]='tapi-common tapi-notification tapi-streaming tapi-topology tapi-path-computation tapi-
connectivity'
[8]='tapi-common tapi-notification tapi-streaming tapi-topology tapi-path-computation tapi-
connectivity tapi-oam'
[9]='tapi-common tapi-notification tapi-streaming tapi-fm'
[10]='tapi-common  tapi-notification  tapi-streaming  tapi-topology  tapi-path-computation
tapi-connectivity tapi-oam tapi-fm tapi-dsr'
[11]='tapi-common  tapi-notification  tapi-streaming  tapi-topology  tapi-path-computation
tapi-connectivity tapi-oam tapi-fm tapi-dsr tapi-digital-otn'
[12]='tapi-common  tapi-notification  tapi-streaming  tapi-topology  tapi-path-computation
tapi-connectivity tapi-oam tapi-fm tapi-eth'
[13]='tapi-common  tapi-notification  tapi-streaming  tapi-topology  tapi-path-computation
tapi-connectivity tapi-oam tapi-fm tapi-photonic-media'
)

echo "Generating OpenAPI files from YANG files"
for elt in "${!MODULES[@]}"; do
  for module in ${MODULES[$elt]}; do
    yangfile=$(find ./YANG/ -name "${module}*.yang")
    tmpfile=`echo ${yangfile##*/} | sed 's/yang//g'`;
    oasfile=./OAS/${tmpfile}yaml
    cp ${yangfile} ./TMP/${tmpfile}yang
  done
  echo "Generating ${oasfile}"
  java -jar /mnt/c/Users/username/YANG2OAS_TOOL/swagger-generator-cli-2.0-executable.jar -
api-version   2.4.1   -use-namespaces   -yang-dir   ./TMP/   -output   ${oasfile}   &>
"./TMP/${tmpfile}log"
  sed -i '/originalRef/d' ${oasfile}
  sed -i '/responseSchema/{N;d;}' ${oasfile}
  rm ./TMP/*.yang
done
```

4. The logs of errors and warnings (.tmp files) are created in:

```
username@XYZ:/$ cd mnt/c/users/username/linuxshare/log
```

3 Editing the Generated Yang and Yaml Modules

1) Add the grouping *class*-ref, necessary for mapping the *relationships by name*. E.g. the **type leafref** statement is not correctly mapped by the tool, see items 2) and 3) below.

    a. Add the "grouping" of referenced *object:*

```
    grouping connection-end-point-ref {
        uses tapi-topology:node-edge-point-ref;
        leaf connection-end-point-uuid {
            type leafref {
                path '/tapi-common:context/tapi-topology:topology-context/tapi-
topology:topology/tapi-topology:node/tapi-topology:owned-node-edge-point/tapi-
connectivity:cep-list/tapi-connectivity:connection-end-point/tapi-
connectivity:uuid';
            }
            description "none";
        }
        description "none";
    }

    grouping node-edge-point-ref {
        uses node-ref;
        leaf node-edge-point-uuid {
            type leafref {
                path '/tapi-common:context/tapi-topology:topology-context/tapi-
topology:topology/tapi-topology:node/tapi-topology:owned-node-edge-point/tapi-
topology:uuid';
            }
            description "none";
        }
        description "none";
    }
```

```
    grouping node-ref {
        uses topology-ref;
        leaf node-uuid {
            type leafref {
                path '/tapi-common:context/tapi-topology:topology-context/tapi-
topology:topology/tapi-topology:node/tapi-topology:uuid';
            }
            description "none";
        }
        description "none";
    }

    grouping topology-ref {
        leaf topology-uuid {
            type leafref {
                path '/tapi-common:context/tapi-topology:topology-context/tapi-
topology:topology/tapi-topology:uuid';
            }
            description "none";
        }
        description "none";
    }
```

2) Manage leaf-list *with path* → **list**
List of *references by name* to objects in the tree, key is necessary to distinguish instances.
It is the translation of shared/lifecycleAggregate associations “to many”.

Intra-module (tapi-connectivity.yang), from:

```
    grouping connection {
        leaf-list connection-end-point {
            type leafref {
                path '/tapi-common:context/tapi-topology:topology/tapi-
topology:node/tapi-topology:owned-node-edge-point/tapi-connectivity:connection-
end-point/tapi-connectivity:uuid';
            }
            config false;
            min-elements 2;
            description "none";
        }
```

To:

```
  grouping connection {
        list connection-end-point {
            uses connection-end-point-ref;
            key 'topology-uuid node-uuid node-edge-point-uuid connection-end-
point-uuid';
            config false;
            min-elements 2;
            description "none";
     }
```

The path can be found in the tapi-connectivity.**tree**:

a. Find the object/grouping, e.g. connection
b. Find its “list/*” attribute, e.g. connection-end-point*

```
       +--ro connection* [uuid]
          +--ro connection-end-point* [topology-uuid node-uuid node-edge-point-
uuid connection-end-point-uuid]
```

Inter-module (tapi-connectivity.yang refers tapi-topology.yang), from:

```
          leaf-list client-node-edge-point {
              type leafref {
                  path '/tapi-common:context/tapi-topology:topology/tapi-
topology:node/tapi-topology:owned-node-edge-point/tapi-topology:uuid';
              }
              config false;
              description "none";
          }
```

To:

```
          list client-node-edge-point {
              uses tapi-topology:node-edge-point-ref;
              key 'topology-uuid node-uuid node-edge-point-uuid';
              config false;
              description "none";
          }
```

With **node-edge-point-ref** defined in tapi-topology.yang:

```
module tapi-topology {
…
     grouping node-edge-point-ref {
         uses node-ref;
         leaf node-edge-point-uuid {
             type leafref {
                 path  '/tapi-common:context/tapi-topology:topology-context/tapi-
topology:topology/tapi-topology:node/tapi-topology:owned-node-edge-point/tapi-
topology:uuid';
             }
             description "none";
         }
         description "none";
```

Note that the label can be different:

```
   leaf-list node-rule-group {
            type leafref {
                path              '/tapi-common:context/tapi-topology:topology/tapi-topology:node/tapi-
topology:node-rule-group/tapi-topology:uuid';
            }
            description "NodeRuleGroups may be nested such that finer grained rules may be applied.
                A nested rule group should have a subset of the NEPs of the superior rule group.";
        }

        list composed-rule-group {
            uses node-rule-group-ref;
            key 'topology-uuid node-uuid node-rule-group-uuid';
            description "none";
```

From tapi-topology.**tree**:

```
+--ro node-edge-point* [topology-uuid node-uuid node-edge-point-uuid]
|  +--ro topology-uuid           -> /tapi-common:context/tapi-topology:topology-context/topology/uuid
|  |  +--ro node-uuid               -> /tapi-common:context/tapi-topology:topology-context/topology/node/uuid
|  |  +--ro node-edge-point-uuid    -> /tapi-common:context/tapi-topology:topology-context/topology/node/owned-
node-edge-point/uuid
|  +--ro composed-rule-group* [topology-uuid node-uuid node-rule-group-uuid]
```

3) Manage leaf *with path with UUID* → **container**
Reference by name to one object in the tree, it is the translation of a **not composite** association "to one".

Intra-module (tapi-connectivity.yang), from:

```
        leaf coroute-inclusion {
            type leafref {
                path '/tapi-common:context/tapi-connectivity:connectivity-
service/tapi-connectivity:uuid';
            }
            description "none";
        }
```

To:

```
        container coroute-inclusion {
            uses connectivity-service-ref;
            description "none";
        }
```

Inter-module (tapi-connectivity.yang refers tapi-topology.yang), from:

```
        leaf parent-node-edge-point {
            type leafref {
                path '/tapi-common:context/tapi-topology:topology/tapi-
topology:node/tapi-topology:owned-node-edge-point/tapi-topology:uuid';
            }
            config false;
            description "none";
        }
```

To:

```
    container parent-node-edge-point {
            uses tapi-topology:node-edge-point-ref;
            config false;
            description "none";
        }
```

With `node-edge-point-ref` defined in tapi-topology.yang:

```
module tapi-topology {
…
    grouping node-edge-point-ref {
        uses node-ref;
        leaf node-edge-point-uuid {
            type leafref {
                path  '/tapi-common:context/tapi-topology:topology-context/tapi-
topology:topology/tapi-topology:node/tapi-topology:owned-node-edge-point/tapi-
topology:uuid';
            }
            description "none";
        }
        description "none";
```

Note 1: leaf and leaf-list can only use built-in types, typedef types or enumerations in their type substatement; i.e., not groupings. Complex data types with more than one item (e.g., name value pair) can only be defined using groupings. Groupings can only be used by grouping, container, and list statements.

Note 2: In case of types translated into groupings and with multiplicity *, the *part of object key* must be defined. Attributes with * cardinality are leaf-list if elementary types (string, integer) otherwise a *part of object key* must be defined.

Note 3: Keys are “invariant”, not “read-only”, because it is possible to modify them at creation time.

Note 4: Keys are not strictly necessary for *read only* lists. Without the key, the retrieval of a single list item is not possible.

4) Enumeration / identity, add the “base” type being extended.

From:

```
identity ODU_TYPE {
    description "none";
}
```

To:

```
identity ODU_TYPE {
    base tapi-common:LAYER_PROTOCOL_QUALIFIER;
    description "none";
}
```

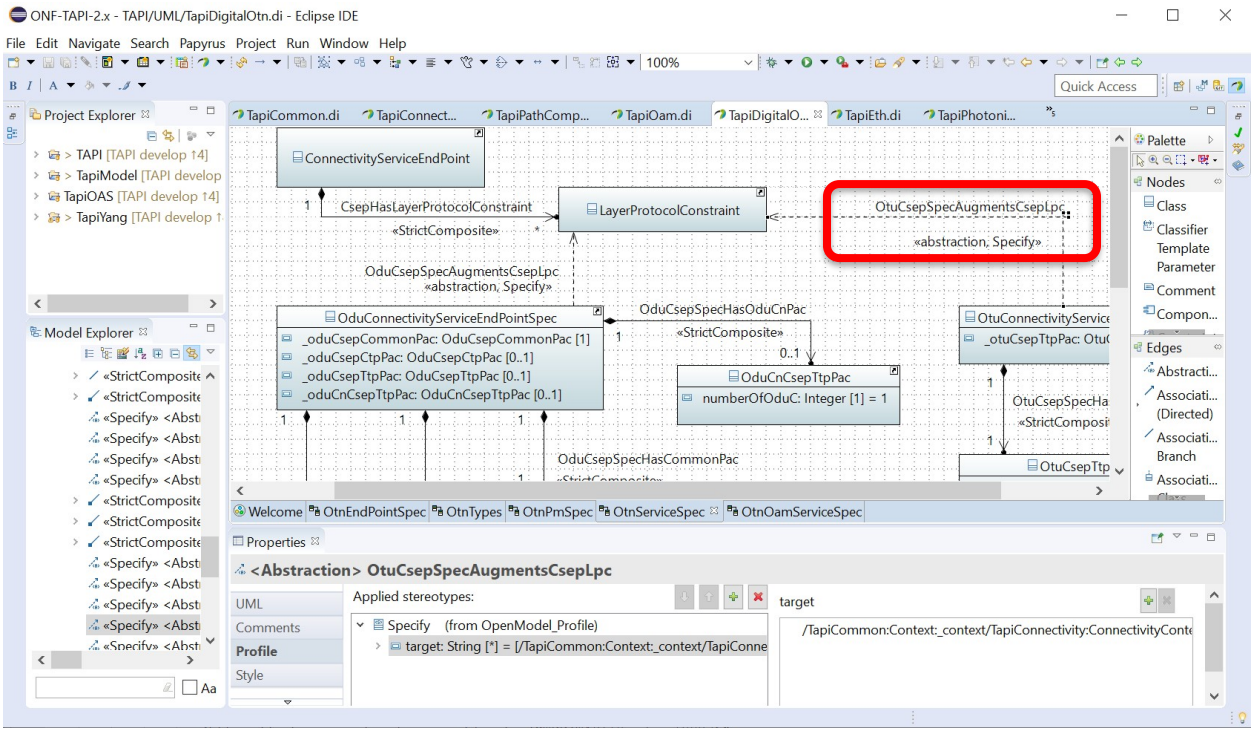
5) Remove “presence” statement, automatically added by the tool, from tapi-common.yang, tapi-context:

```
container context {
    uses tapi-context;
/*      presence "Root container for all TAPI interaction";*/
    description "none";
}
```

6) Check the YANG version, the tool does not manage it:

```
module tapi-topology {
    yang-version "1.1";
    namespace "urn:onf:otcc:yang:tapi-topology";
```

7) Manage the augmentations:



Specify stereotype of *OtuCsepSpecAugmentsCsepLpc* abstraction, the full text of the *target*:

```
/TapiCommon:Context:_context/TapiConnectivity:ConnectivityContext:_connectivityContext/TapiConnectivity:ConnectivityContext:_connectivityService/TapiConnectivity:ConnectivityService:_endPoint/TapiConnectivity:ConnectivityServiceEndPoint:_layerProtocolConstraint
```

Is translated into (note that the additional yellow part is always necessary in the UML, otherwise the YANG is not correctly generated):

```
augment "/tapi-common:context/tapi-connectivity:connectivity-context/tapi-connectivity:connectivity-service/tapi-connectivity:end-point/tapi-connectivity:layer-protocol-constraint" {
        container otu-connectivity-service-end-point-spec {
            uses otu-connectivity-service-end-point-spec;
            description "none";
        }
        description "none";
```

Other examples:

UML:

```
/TapiCommon:Context:_context/TapiTopology:TopologyContext:_topologyContext/TapiTopology:TopologyContext:_topology/TapiTopology:Topology:_node/TapiTopology:Node:_ownedNodeEdgePoint
```

YANG:

```
/tapi-common:context/tapi-topology:topology-context/tapi-
topology:topology/tapi-topology:node/tapi-topology:owned-node-edge-point
```

UML:

```
/TapiCommon:Context:_context/TapiTopology:TopologyContext:_topologyContext/Tapi
Topology:TopologyContext:_topology/TapiTopology:Topology:_node/TapiTopology:Nod
e:_ownedNodeEdgePoint/TapiConnectivity:CepList:_cepList/TapiConnectivity:CepLis
t:_connectionEndPoint
```

YANG:

```
/tapi-common:context/tapi-topology:topology-context/tapi-
topology:topology/tapi-topology:node/tapi-topology:owned-node-edge-point/tapi-
connectivity:cep-list/tapi-connectivity:connection-end-point
```

UML:

```
/TapiCommon:Context:_context/TapiOam:OamContext:_oamContext/TapiOam:OamContext:
_meg/TapiOam:Meg:_mep
```

YANG:

```
/tapi-common:context/tapi-oam:oam-context/tapi-oam:meg/tapi-oam:mep
```

8) If the augmenting object is imported from another module the tool does not work correctly:

UML:

```
/TapiCommon:Context:_context/TapiConnectivity:ConnectivityContext:_connectivity
Context/TapiConnectivity:ConnectivityContext:_connectivityService/TapiConnectiv
ity:ConnectivityService:_endPoint/TapiOam:ConnectivityOamServicePoint
```

YANG:

```
/tapi-common:context/tapi-connectivity:connectivity-context/tapi-
connectivity:connectivity-service/tapi-connectivity:end-point/tapi-
oam:undefined
```

To be edited as:

```
/tapi-common:context/tapi-connectivity:connectivity-context/tapi-
connectivity:connectivity-service/tapi-connectivity:end-point/tapi-
oam:connectivity-oam-service-point
```

In other cases the error is less visible, i.e. the tool simply copies the previous augment statement:

```
augment "/tapi-common:context/tapi-fm:fault-management-context/tapi-fm:active-condition" {
   container detected-condition {
       uses detected-condition;
          description "none";
    }
    description "none";
}
augment "/tapi-common:context" {
   container detected-condition {
       uses detected-condition;
          description "Augments the base TAPI Context with FM Context model.";
    }
    description "Augments the base TAPI Context with FM Context model.";
}
```

Corrected as:

```
augment "/tapi-common:context/tapi-fm:fault-management-context/tapi-fm:active-condition" {
   container detected-condition {
       uses detected-condition;
          description "none";
    }
    description "none";
}
augment "/tapi-common:context" {
   container fault-management-context {
       uses fault-management-context;
          description "Augments the base TAPI Context with FM Context model.";
    }
    description "Augments the base TAPI Context with FM Context model.";
}
```

9) Search “signed” and check it is INT64 rather than UINT64.

10) The UML “defaultValue” is translated into the "default" substatement, but sometimes the mapping is not performed, hence manual editing is necessary.

11) XOR Constraint to be translated into CHOICE.

12) OAS/YAML correction, Replace “/:” with “:”

Apply the following command in the OAS directory:

```
username@XYZ:/$ sed -i "s|/:|:|g" *.yaml
```

13) Correction in OAS/YAML tapi-connectivity module:

Before editing, Search:

/data/tapi-common:context/tapi-connectivity:connectivity-context:

```
    get:
      tags:
      - "tapi-connectivity"
      summary: "returns tapi.connectivity.ConnectivityContext"
      description: "Augments the base TAPI Context with ConnectivityService
information"
      parameters: []
      responses:
        200:
          description: "tapi.connectivity.ConnectivityContext"
          schema:
            $ref: "#/definitions/tapi.connectivity.ConnectivityContextWrapper"
        400:
          description: "Internal error"
    post:
      tags:
      - "tapi-connectivity"
      summary: "creates tapi.connectivity.ConnectivityContext"
      description: "Augments the base TAPI Context with ConnectivityService
information"
      parameters:
      - in: "body"
        name: "tapi.connectivity.ConnectivityContext.body-param"
        description: "tapi.connectivity.ConnectivityContext to be added to list"
        required: false
        schema:
          $ref: "#/definitions/tapi.connectivity.ConnectivityContextWrapper"
      responses:
        201:
          description: "Object created"
        400:
          description: "Internal error"
        409:
          description: "Object already exists"
```

After editing:

```
      /data/tapi-common:context/tapi-connectivity:connectivity-context:
          get:
            tags:
            - "tapi-connectivity"
            summary: "returns tapi.connectivity.ConnectivityContext"
            description: "Augments the base TAPI Context with ConnectivityService
      information"
            parameters: []
            responses:
              200:
                description: "tapi.connectivity.ConnectivityContext"
                schema:
                  $ref: "#/definitions/tapi.connectivity.ConnectivityContextWrapper"
              400:
```

```
          description: "Internal error"
    post:
      tags:
      - "tapi-connectivity"
      summary: "creates tapi.connectivity.ConnectivityContext"
      description: "Augments the base TAPI Context with ConnectivityService
information"
      parameters:
      - in: "body"
        name: "tapi.connectivity.ConnectivityContext.body-param"
        description: "tapi.connectivity.ConnectivityContext to be added to
list"
        required: false
        schema:
          $ref: "#/definitions/tapi.connectivity.ConnectivityContext"
      responses:
        201:
          description: "Object created"
        400:
          description: "Internal error"
        409:
          description: "Object already exists"
```

4 Other Recommendations

1. Do not change the project directory name, otherwise the style sheet profiles are lost.

2. Addressing:
   - Package-as-a-class, multiplicity "star": inherit LocalClass (which brings LocalId attribute)
   - Package-as-a-datatype, multiplicity "star": identify at least one attribute "partOfObjectKey“

3. The UML “comment” is translated into a ”description” substatement. Note that label of mentioned entities is not translated from UML lowerCamelCase to YANG hyphened notation.

4. The referenced version of the UML-YANG tool translates the *references by value* (aggregation type: **composite**) only if the **StrictComposite** stereotype of OpenModel_Profile is added to the profile of the association.

End of document