// Concept: Test Data Generation for Telco entities

Information history

Date	Changed sections	Description	Author
18.05.2022	-	Creation of document	Sarah Hägele

Wird das Dokument aus dem FNT System entnommen, ist es ein ungelenktes Dokument.

Inhalt

1 General Preconditions

- The following database settings should be set to enable full functionality of the Reconciliation Engine.

Setting name	Setting value	Set by Reconciliation Engine installation	Description
TCO_ALLOW_NE_DEVICE_REMOVE	TRUE	N	Allows deletion of Device in case its linked to NE
CLI_LOGICAL_PORT_LOCK_SUB_PORTS	FALSE	N	Logical ports lock subport
TCO_NE_DEVICE_LINK_MANDATORY	FALSE	N	NE can exist without Device link
TCO_ECS_SERVICE_OCCUPIES_ENDPORT		N	End customer services occupies Endports
TCO_NE_DEFAULT_MASTER_ZONE	[Autonomous system]	N	Default zone with which NE will be linked
TCO_SERVICE_OCCUPIES_ENDPORT		N	Services occupies logical ports
TCO_ALLOW_DEVICE_MOVE_IN_REALISED_STATE		N	Allow moving devices in realized state

Before the execution of any tests, these settings should be checked. If one of them is not set, it should be automatically set via a database update statement. Since the setting only effectively becomes active after a server restart, test execution should be aborted, and an error should be returned to the caller that lists the settings that have been altered. Furthermore, the caller should be notified that a manual server restart is necessary to activate the settings.

This can be achieved by executing the check in either a separate test in the Precondition-Suite or as a general precondition in the Login-Method that is executed before every Test Suite thanks to the ***@BeforeAll***-Tag.

2 Network Element

2.1 Attributes

The following is a list of attributes that should be generated for every Network Element object.

Attribute name	Mandatory	Get from CIF Entity Config-uration	NMS Attribute	Notes
name	Y	Y	nmsId	Max. length: 40 Characters
sourceId	N	Y	nmsInternalId	
sourceSystem	N (Y for NMS)	Y	nmsSourceSys-tem	
type	N	Y	commandType	
id	N	N	-	Must be unique
addMasterZone	N	N	commandZoneId	
remark	N	Y	remark	Can be used for UPDATE

notice

Getting the attributes

A “**Y**” in the column “**Get from CIF Entity Configuration**” means that the attribute is marked as “**synchro-nizeToCommand = Y**” in the CIF Entity/Attribute Configuration. These attributes can be queried via the BGE and – usually – can be formatted to be used as attribute names for NMS (field “**attrName**”) and Com-mand (field “**commandAttribName**”) objects.

An example (non-NMS) Network Element in JSON-form could look like this:

```
{
  "id": "TEST_NE_1234",
  "sourceSystem": "CIF_AUTOMATED_TEST",
  "name": "TEST_NE_DELTACASE",
  "sourceId": "TEST_NE_DELTACASE",
  "type": "Antenna",
  "remark": "Test Network Element for delt case CASE",
  "addMasterZone": {
    "linkedElid": "ELID"
  }
}
```

2.2 Data Generation

To generate test data for a Network Element object, the following procedure can be executed:

setupTestData()

- Query CIFEntityConfiguration with Query-Restriction *entityName = NetworkElement*
- Get Elid from the response-object
- Query CIFAttributeConfiguration with the Entity Elid
- Call *generateTestData*-method with the CIFAttributeConfiguration-List

generateTestData()

- Query Data Dictionary *SDDTCO_NE_TYPE*
- Collect NE Types from the Query Response
- Query Zone data (Room-Elid & commandZoneId)
- Create base records

createNmsBaseRecord()

- Filter attributes from CIFAttributeConfiguration for attributes with *synchronizeToCommand = Y*, the exception being *commandZoneId*, which doesn't have this attribute set yet should be present in the base record anyways. ***Important***: The attribute *sourceSystem* must be renamed to *nmsSourceSystem* for the base record; its name differs between the CIFAttributeConfiguration and actual NMS objects.

createCommandBaseRecord()

- CIFAttributeConfiguration attributes should once again be filtered, but this time the attribute *id* should be placed into the base record instead of *commandZoneId*. The attribute names can be mapped from the values in the field *commandAttribName*.
- Create test object of entity Network Element and place it in Command to check if any attributes are still missing from the Command base record. If the creation of the object returns an error code that is not 0, the missing attribute should be identified via the method *DataGeneratorUtils.findMissingAttribute.*
- Delete object from Command again after successful creation

generateTestData()

- Loop over Delta Cases
- Within one Delta Case, loop over the different record types (nms, command, planningCreate, planningDelete)
- Generate a record from the NMS or Command base record via the *generateValue* method, if it is the first record to be created for the Delta Case, or generate it as a similar record from a previously generated reference record

setupTestData()

- Sort the record into separate lists for NMS objects, Command objects, objects to be created in planning mode and objects to be deleted in planning mode

createTestObjects()

- Create Command objects from the corresponding list
- Create NMS objects

3 Logical Port

3.1 Preconditions

For a logical port to be created, the following preconditions need to be fulfilled:

- Type 2: A device (e.g. Chassis) is must be created that a logical port can be added to
- Type 3: A Network Element must be created that a logical port can be added to

3.2 Attributes

PTOLP Type 2

Attribute name	Mandatory	Get from CIF Entity Configuration	NMS Attribute	Notes
portNameOrg	N	Y	logicalPortNmsName	
bandwidth	N	Y	bandwidth	Can be used for UPDATE
ptolp	N	N (isn't marked as synchronize)	commandPtype	Needed for NMS objects only – set to 2
portType	Y (N for NMS)	Y	commandType	Can be used for UPDATE
-	N	N	groupingId	
id	N (Y for NMS)	Y	nmsId	NE/DeviceNmsId-portName Can only be set for NMS objects; for Command objects it is generated in Command!
sourceId	N	Y	nmsInternalId	
overloadWarning	Y (N for NMS)	Y	overloadWarning	
signalRate	N	Y	signalRate	Can be used for UPDATE
sourceSystem	N (Y for NMS)	Y	nmsSourceSystem	
createLinkDeviceAll	Y	N	-	Suboperation for link to a device (needs Elid of that device)
-	N	N	deviceNmsId	Create link to device for NMS object

PTOLP Type 3

Attribute name	Mandatory	Get from CIF Entity Configuration	NMS Attribute	Notes

portNameOrg	N	Y	logicalPortNms-Name	
bandwidth	N	Y	bandwidth	Can be used for UPDATE
ptolp	N	N (isn't marked as synchronize)	commandPtype	Needed for NMS objects only – set to 3
portType	Y (N for NMS)	Y	commandType	Can be used for UPDATE
-	N	N	groupingId	
id	N (Y for NMS)	Y	nmsId	NE/DeviceNmsId-portName
sourceId	N	Y	nmsInternalId	
overloadWarning	Y (N for NMS)	Y	overloadWarning	
signalRate	N	Y	signalRate	Can be used for UPDATE
sourceSystem	N (Y for NMS)	Y	nmsSourceSys-tem	
createLinkNet-workElement	Y	N	-	Suboperation for link to a NE (needs Elid of that NE)
-	N	N	neNmsId	Create link to NE for NMS object

An example (non-NMS) Logical Port with PTOLP Type 2 in JSON form could look like this:

```
{
  "bandwidth": 100000,
  "overloadWarning": "NO_PORT_MONITORING",
  "portNameOrg": "1",
  "portType": "100 Gbit/s",
  "sourceSystem": "CIF_AUTOMATED_TEST",
  "sourceId": "TEST_LOGICAL_PORT_DELTACASE",
  "signalRate": null,
  "createLinkDeviceAll": {
    "linkedElid": "ELID"
  }
}
```

notice

The PTOLP Type for Command objects is chosen by using the right REST-Endpoint to create the object.

3.3 Data Generation

To generate test data for a Logical Port object, the following procedure can be executed:

setupTestData()

- Query CIFEntityConfiguration with Query-Restriction *entityName = LogicalPort*
- Get Elid from the response-object
- Query CIFAttributeConfiguration with the Entity Elid
- Call *generateTestData*-method with the CIFAttributeConfiguration-List

generateTestData()

- Query Port Types and Bandwidths from database table STCSPC_LOGICAL_PORT
- Query Zone data (Room-Elid & commandZoneId)
- Create NE to place Logical Ports onto
- Create base records

createNmsBaseRecord()

- Filter attributes from CIFAttributeConfiguration for attributes with *synchronizeToCommand = Y* (some exceptions apply, which can be seen in the table in section 3.2). ***Important***: The attribute *sourceSystem* must be renamed to *nmsSourceSystem* for the base record; its name differs between the CIFAttributeConfiguration and actual NMS objects.

createCommandBaseRecord()

- CIFAttributeConfiguration attributes should once again be filtered
- Create test object of entity Logical Port and place it in Command to check if any attributes are still missing from the Command base record. If the creation of the object returns an error code that is not 0, the missing attribute should be identified via the method *DataGeneratorUtils.findMissingAttribute.*
- Delete object from Command again after successful creation

generateTestData()

- Loop over Delta Cases
- Within one Delta Case, loop over the different record types (nms, command, planningCreate, planningDelete)
- Generate a record from the NMS or Command base record via the *generateValue* method, if it is the first record to be created for the Delta Case, or generate it as a similar record from a previously generated reference record

setupTestData()

- Sort the record into separate lists for NMS objects, Command objects, objects to be created in planning mode and objects to be deleted in planning mode

createTestObjects()

- Create Command objects from the corresponding list
- Create NMS objects

4 Bearer

4.1 Preconditions

- Two devices or NEs
- Logical Ports on each device or NE

4.2 Attributes

Attribute name	Mandatory	Get from CIF Entity Configuration	NMS Attribute	Notes
sourceId	Y	Y	nmsId	
id	N	N	-	Must be unique
-	N	N	groupingId	
remark	N	Y	remark	Can be used for UPDATE
sourceSystem	N (Y for NMS)	Y	nmsSourceSystem	
unibi	N	Y	unibi	
visibleId	N	Y	visibleId	Custom attribute for < Command 13.0!
createLinkLogicalPortStart	Y (N for NMS)	N	-	Suboperation for link to start Logical Port
-	N	N	logicalPortNmsIdA	sourceId of start Logical Port for NMS object
createLinkLogicalPortEnd	Y (N for NMS)	N	-	Suboperation for link to end Logical Port
-	N	N	logicalPortNmsIdB	sourceId of end Logical Port for NMS object
createLinkServiceTypeDefinition	Y (N for NMS)	N	-	Suboperation for link to Service Type Definition
-	N	N	commandServiceType	Service type for NMS objects

An example (non-NMS) Bearer in JSON-Form could look like this:

```
{
  "id": "TEST_BEARER_1234",
  "sourceSystem": "CIF_AUTOMATED_TEST",
  "visibleId": "TEST_BEARER_DELTACASE",
  "sourceID": "TEST_BEARER_DELTACASE",
  "unibi": "B",
  "remark": "Test Bearer for delta case DELTACASE",
  "createLinkLogicalPortEnd": {
```

```
        "linkedElid": "ELID"
      },
      "createLinkLogicalPortStart": {
        "linkedElid": "ELID"
      },
      "createLinkServiceTypeDefinition": {
        "linkedElid": "ELID"
      }
    }
```

4.3 Data Generation

To generate test data for a Bearer object, the following procedure can be executed:

setupTestData()

- Query CIFEntityConfiguration with Query-Restriction *entityName = Bearer*
- Get Elid from the response-object
- Query CIFAttributeConfiguration with the Entity Elid
- Call *generateTestData*-method with the CIFAttributeConfiguration-List

generateTestData()

- Query Service Type Definitions with restrictions *serviceCategory = BEARER* and *transmissionTechnology = PACKET_DATA* and choose one
- Query Zone data (Room-Elid & commandZoneId)
- Create at least two NEs
- Create Logical Ports to add to the NEs (**Important**: Bandwidth must be equal or greater than that of the chosen Service Type!)
- Create base records

createNmsBaseRecord()

- Filter attributes from CIFAttributeConfiguration for attributes with *synchronizeToCommand = Y* (some exceptions apply, which can be seen in the table in section 3.2). ***Important***: The attribute *sourceSystem* must be renamed to *nmsSourceSystem* for the base record; its name differs between the CIFAttributeConfiguration and actual NMS objects.

createCommandBaseRecord()

- CIFAttributeConfiguration attributes should once again be filtered
- Create test object of entity Bearer and place it in Command to check if any attributes are still missing from the Command base record. If the creation of the object returns an error code that is not 0, the missing attribute should be identified via the method *DataGeneratorUtils.findMissingAttribute.*
- Delete object from Command again after successful creation

generateTestData()

- Loop over Delta Cases
- Within one Delta Case, loop over the different record types (nms, command, planningCreate, planningDelete)
- Generate a record from the NMS or Command base record via the *generateValue* method, if it is the first record to be created for the Delta Case, or generate it as a similar record from a previously generated reference record

setupTestData()

- Sort the record into separate lists for NMS objects, Command objects, objects to be created in planning mode and objects to be deleted in planning mode

createTestObjects()

- Create Command objects from the corresponding list
- Create NMS objects

5 Services

5.1 Path

5.1.1 Preconditions

- Two NEs that Logical Ports can be added to
- Logical Ports on each NE
- A Bearer connecting the Logical Ports

5.1.2 Attributes

Unrouted Path

Attribute name	Mandatory	Get from CIF Entity Configuration	NMS Attribute	Notes
sourceId	Y	Y	nmsId	
id	N	N	-	Must be unique
-	N	N	groupingId	
sourceSystem	N (Y for NMS)	Y	nmsSourceSystem	
unibi	N	Y	unibi	
visibleId	N	Y	visibleId	Custom attribute for < Command 13.0!
createLinkLogicalPortStart	Y (N for NMS)	N	-	Suboperation for link to start Logical Port
-	N	N	logicalPortNmsIdStart	sourceId of start Logical Port for NMS object
createLinkLogicalPortEnd	Y (N for NMS)	N	-	Suboperation for link to end Logical Port
-	N	N	logicalPortNmsIdEnd	sourceId of end Logical Port for NMS object
createLinkServiceTypeDefinition	Y (N for NMS)	N	-	Suboperation for link to Service Type Definition
-	N	N	commandType	Type for NMS objects

Path

Attribute name	Mandatory	Get from CIF Entity Configuration	NMS Attribute	Notes
sourceId	Y	Y	nmsId	
id	N	N	-	Must be unique
-	N	N	groupingId	
sourceSystem	N (Y for NMS)	Y	nmsSourceSystem	

unibi	N	Y	unibi	
visibleId	N	Y	visibleId	Custom attribute for < Command 13.0!
-	N	N	neNmsIdStart	sourceId of NE the bearer is starting on
-	N	N	neNmsIdEnd	sourceId of NE the bearer is ending on
create-LinkService-TypeDefinition	Y (N for NMS)	N	-	Suboperation for link to Service Type Definition
-	N	N	commandType	Type for NMS objects
addRoute	Y (N for NMS)	N	-	Direction: AB, Elid of Bearer, sequenceNo 1 for first Bearer in the route

An example (non-NMS) Path in JSON-form could look like this:

```
{
  "id": "TEST_PATH_1234",
  "sourceSystem": "CIF_AUTOMATED_TEST",
  "unibi": "B",
  "visibleId": "TEST_PATH_DELTACASE",
  "sourceId": "TEST_PATH_DELTACASE",
  "addRoute": [
    {
      "direction": "AB",
      "linkedElid": "BEARER_ELID",
      "sequenceNo": 1
    }
  ],
  "createLinkServiceTypeDefinition": {
    "linkedElid": "ELID"
  }
}
```

5.1.3 Data Generation

To generate test data for a Path/Unrouted Path object, the following procedure can be executed:

setupTestData()

- Query CIFEntityConfiguration with Query-Restriction *entityName = Path (or UnroutedPath)*
- Get Elid from the response-object
- Query CIFAttributeConfiguration with the Entity Elid
- Call *generateTestData*-method with the CIFAttributeConfiguration-List

generateTestData()

- Query Service Type Definitions with restrictions *serviceCategory = PATH* and *transmissionTechnology = PACKET_DATA* and choose one
- Query Zone data (Room-Elid & commandZoneId)
- Create at least two NEs
- Create Logical Ports to add to the NEs (**Important**: Bandwidth must be equal or greater than that of the chosen Service Type!)
- Create a compatible Bearer to connect the two NEs
- Create base records

createNmsBaseRecord()

- Filter attributes from CIFAttributeConfiguration for attributes with *synchronizeToCommand = Y* (some exceptions apply, which can be seen in the table in section 3.2). ***Important***: The attribute *sourceSystem* must be renamed to *nmsSourceSystem* for the base record; its name differs between the CIFAttributeConfiguration and actual NMS objects.

createCommandBaseRecord()

- CIFAttributeConfiguration attributes should once again be filtered
- Create test object of entity Path/Unrouted Path and place it in Command to check if any attributes are still missing from the Command base record. If the creation of the object returns an error code that is not 0, the missing attribute should be identified via the method *DataGeneratorUtils.findMissingAttribute.*
- Delete object from Command again after successful creation

generateTestData()

- Loop over Delta Cases
- Within one Delta Case, loop over the different record types (nms, command, planningCreate, planningDelete)
- Generate a record from the NMS or Command base record via the *generateValue* method, if it is the first record to be created for the Delta Case, or generate it as a similar record from a previously generated reference record

setupTestData()

- Sort the record into separate lists for NMS objects, Command objects, objects to be created in planning mode and objects to be deleted in planning mode

createTestObjects()

- Create Command objects from the corresponding list
- Create NMS objects

5.2 Unrouted Multipoint

5.2.1 Preconditions

- At least one NE that a Logical Port can be added to
- Logical Port Type 3 on each NE

important

The Logical Ports must not be occupied by a Bearer or else they won't be recognized as available.

5.2.2 Attributes

Attribute name	Mandatory	Get from CIF Entity Configuration	NMS Attribute	Notes
sourceId	Y	Y	nmsId	
id	N	N	-	Must be unique
-	N	N	groupingId	
sourceSystem	N (Y for NMS)	Y	nmsSourceSystem	
unibi	N	Y	unibi	
visibleId	N	Y	visibleId	Custom attribute for < Command 13.0!
createLinkService-TypeDefinition	Y	N	-	Suboperation for link to Service Type Definition
-	N	N	commandType	Type for NMS objects
createLinkLogicalPort	Y	N	-	Suboperation for link to End Point/Logical Port

important

Link between NMS Unrouted Multipoint and Logical Port

An NMS Unrouted Multipoint needs a link to a Logical Port. For this, the nmsId of the Logical Port should be used (NE/DeviceNmsId-portName; e.g. TEST_NE-1).

An example (non-NMS) Unrouted Multipoint in JSON-form could look like this:

```
{
  "id": "TEST_UNROUTED_MULTIPOINT_123",
  "sourceId": "TEST_UNROUTED_MULTIPOINT",
  "sourceSystem": "CIF_AUTOMATED_TEST",
  "visibleId": "TEST_UNROUTED_MULTIPOINT",
  "createLinkLogicalPort": [
    {
```

```
        "linkedElid": "ELID"
      }
    ],
    "createLinkServiceTypeDefinition": {
      "linkedElid": "ELID"
    }
  }
```

And a link between an NMS Unrouted Multipoint and a Logical Port could look like this:

```
  {
    "logicalPortNmsId": "TEST_NE_MTP_5-1",
    "nmsSourceSystem": "CIF_AUTOMATED_TEST",
    "linkNmsId": "TEST_LINK_MULTIPOINT_LOGPORT",
    "serviceNmsId": "TEST_NMS_UNROUTED_MULTIPOINT_1"
  }
```

5.2.3 Data Generation

To generate test data for a Unrouted Multipoint object, the following procedure can be executed:

setupTestData()

- Query CIFEntityConfiguration with Query-Restriction *entityName* = *Unrouted Multipoint*
- Get Elid from the response-object
- Query CIFAttributeConfiguration with the Entity Elid
- Call *generateTestData*-method with the CIFAttributeConfiguration-List

generateTestData()

- Query Service Type Definitions with restrictions *serviceCategory* = *MULTIPOINT* and *transmissionTechnology* = *PACKET_DATA* and choose one
- Query Zone data (Room-Elid & commandZoneId)
- Create at least two NEs
- Create Logical Ports to add to the NEs (**Important**: Bandwidth must be equal or greater than that of the chosen Service Type!)
- **Important**: Do NOT create a bearer like for the Path Services; the Unrouted Multipoint can only be placed on unoccupied Logical Ports!
- Create base records

createNmsBaseRecord()

- Filter attributes from CIFAttributeConfiguration for attributes with *synchronizeToCommand* = *Y* (some exceptions apply, which can be seen in the table in section 3.2). ***Important***: The attribute *sourceSystem* must be renamed to *nmsSourceSystem* for the base record; its name differs between the CIFAttributeConfiguration and actual NMS objects.

createCommandBaseRecord()

- CIFAttributeConfiguration attributes should once again be filtered
- Create test object of entity Unrouted Multipoint and place it in Command to check if any attributes are still missing from the Command base record. If the creation of the object returns an error code that is not 0, the missing attribute should be identified via the method *DataGeneratorUtils.findMissingAttribute.*
- Delete object from Command again after successful creation

generateTestData()

- Loop over Delta Cases
- Within one Delta Case, loop over the different record types (nms, command, planningCreate, planningDelete)
- Generate a record from the NMS or Command base record via the *generateValue* method, if it is the first record to be created for the Delta Case, or generate it as a similar record from a previously generated reference record

setupTestData()

- Sort the record into separate lists for NMS objects, Command objects, objects to be created in planning mode and objects to be deleted in planning mode

createTestObjects()

- Create Command objects from the corresponding list
- Create NMS objects


FNT
// when transparency matters.