ERICSSON

3GPP Bulk CM Import/Export File Format

Interwork Description

1/15519-CNA 403 2977 Uen AJ

Contents

1 ENM Interwork Description for 3GPP CM Import/Export

Abstract

The ENM 3GPP Bulk CM Import/Export feature allows you to do the following:

— Full export of configuration data for one, several, or all network elements.

— Filtered export of configuration data for one, several, or all network elements.

— Import of a set of modifications to one or more network elements in the Live configuration by uploading a file containing create, update, delete and action commands.

The data is exported and imported as 3GPP Bulk CM IRP compliant XML files. This document describes the format and content of those files.

This `interwork` description describes the following:

— The changes in import and export file content from OSS-RC to ENM.

— The structure of 3GPP Bulk CM import and export files.

— The structure of the data which may be contained in 3GPP Bulk CM import and export files.

— How Ericsson vendor specific data is modelled in a 3GPP compliant manner.

— Handling of special attribute types.

— Managed Object modifier order rules.

The relevant XML schemata and some example files are included in Resources on page 22.

Target Group

The intended target groups for this document are:

— Developers of management systems that use the 3GPP Bulk CM import/export feature to read and write ENM configuration data.

— Network Configuration Engineers.

— System Administrators.

Prerequisites

It is assumed that users of this document:

— Are familiar with creating and editing XML files.

— Are familiar with ENM.

— Have a working knowledge of Operation and Maintenance (O&M).

— Are familiar with Ericsson Networks and Network Elements.

— Are familiar with the concepts of Bulk Configuration and related XML format files.

— Have appropriate user privileges.

2 Changes in Import and Export File Content from OSS-RC to ENM

A number of changes exist between the import and export file content in OSS-RC, and the import and export file content in ENM.

Supported and Unsupported Models for Bulk CM in ENM

ENM is a model driven system. For Bulk CM, this means that ENM exposes the node models as they are, and it does not provide the model abstractions or modifications that OSS-RC provides. As a result, there are significant changes to the content of ENM import and export files compared to OSS-RC.

ENM supports the full node CM models. ENM does not support the following models: 3GPP UTRAN NRM IRP, 3GPP GERAN NRM IRP, or RANOS SubNetwork.

Migration Considerations

When migrating from OSS-RC to ENM, consider the following:

- Northbound systems that consume Bulk CM export files need to be modified to use ENM Bulk CM export files.
- Bulk CM Import files that are used with OSS-RC need to be modified to work with ENM.

OSS-RC to ENM Bulk Configuration Differences Report

The Excel spreadsheet OSS-RC to ENM Bulk Configuration Differences Report, provides the following information:

- A sheet for each node type listing attributes that are:
  - Stored locally in OSS-RC (‘local’ attributes)
  - Included in OSS-RC BULK-CM 3GPP export and import files
  - Excluded from or represented differently in the ENM BULK-CM 3GPP files

  The OSS-RC model can include 'local' attributes and 'local' MOs, but ENM node models cannot. Where alternative attributes exist for the 'local' attributes, they are included in the ENM BULK-CM 3GPP file, and the alternative is specified in the spreadsheet. There is a comment for each attribute explaining the exclusion or different representation, for example, some attributes are available in alternative MOs and other attributes are not supported.

- Additional sheets for the following:
  - RANOS Subnetwork

    Maps abstracted OSS-RC cell-model data to the node models exposed by ENM.

    The sheet describes both supported and unsupported MOs and their attributes from the RANOS subnetwork model. In OSS RC, the RANOS subnetwork model abstracts radio network data, instead of making node models fully transparent using Bulk CM import and export. In ENM, all node data is fully transparent, with no abstraction.

  - MO Name Mapping

    Provides mappings between OSS-RC external managed objects and ENM external managed objects, where vsData naming is used by the MOs in ENM but not in OSS-RC

  - netype enum Conversion Table

    Maps string values to associated integer values for neType and PlatformType enumerations.

3 File Structure

The 3GPP Bulk CM IRP [1] provides a mechanism for import and export of bulk quantities of configuration management data. It also specifies a file structure for data files containing the configuration management data to be imported or exported. CM Bulk import/export supports the basic XML file format and conventions for XML schema definition from the 3GPP Bulk CM IRP XML File Format Definition [2].

The 3GPP Generic NRM IRP data model [3] provides the network and node level context for all other supported configuration data. See Statement of Compliance [4] for the level of support.

The majority of the data which can be imported or exported is defined in Ericsson vendor specific data models. This is described in the Vendor Specific Schema Extension on page 10. Data which is persisted in the ENM data persistence service can be imported and exported.

Configuration File Format Overview

The XML file format is used for configuration files. XML schema files define the high level structure and content constraints that apply to an XML file that is compliant to the schema. Only "UTF-8" character encoding is supported. The schema structure and the format of configuration data files exchanged between an IRP Manager and an IRP Agent are specified in the Bulk CM IRP XML File Format Definition [2]. The overall file structure is defined in the configData.xsd schema.

Overall File Structure

The overall file structure defined in configData.xsd states that an import or export data file must contain the following:

— A `fileHeader` element.

— One or more `configData` elements.

— A `fileFooter` element.

XML elements `fileHeader` and `fileFooter` contain attributes which provide information regarding file format version, vendor and creation date and time. For further description of the attributes and values for these elements refer to [2]. For a full specification of the overall file structure, see the configData.xsd schema file.

Example 1 3GPP Bulk CM Export File

```
<fileHeader fileFormatVersion="32.615 V4.5" vendorName="Ericsson"/>
<configData dnPrefix="Undefined">
  <xn:MeContext id="LTE01ERBS00001">
    ...
```

```
  </xn:MeContext>
</configData>
<fileFooter dateTime="2008-09-11T15:27:07Z"/>
```

Namespaces

The bulkCmConfigDataFile XML element has all the XML attribute specifications that declare the XML namespaces used in the XML file.

The following XML namespaces may be used in configuration data XML files:

— The default XML namespace is associated with the configuration data files base XML schema configData.xsd.

— The XML namespace prefix xn is defined for the XML namespace associated with the NRM specific XML schema genericNrm.xsd from the Generic Network Resources IRP NRM.

  Managed Objects (MOs) like `SubNetwork`, `MeContext` and so on are defined in this schema. Examination of the definition of the MO's in the genericNrm.xsd schema outlines further relationships between these MOs and MOs defined in other 3GPP schemas.

— XML namespace prefixes starting with es, for example `es:productData`, are reserved for the XML namespace associated with all supported Ericsson vendor specific data models.

A configData XML element may contain all the persistent CM data for all of the network elements being managed by ENM, or any subset of it. For further details, refer to Bulk CM IRP XML file format .

Data Structure

The high level structure of the data contained in the configData element is defined by the Generic NRM IRP [3] . The main MO classes of relevance are:

SubNetwork: This may optionally be used to provide a sub-division, or grouping, of the network elements (ManagedElements) which make up the network being managed. It is also possible to have several layers of SubNetworks.

MeContext: This is required to provide unique naming for some types of ManagedElement which are pre-configured with the same managementElementId during manufacturing.

ManagedElement: This represents a network element being managed by ENM.

VsDataContainer:

All Ericsson vendor specific MO classes are defined as VsDataContainers. See section "Vendor Specific Data" for details.

The level of support for the Generic NRM IRP is specified in the Statement of Compliance [4].

Ordering of Entities in Export File:

By default, the order of nodes, MO instances, attributes, or `struct` members is not guaranteed. Northbound systems should not rely on ordering when parsing.

Note: It is possible to control the order of the MO instances to be ordered by ascending alphabetical order of FDNs, see ENM Configuration System Administration Guide [6]

Commands in Import Files

The structure of a 3GPP Bulk CM import file is the same as for an export file, with the addition that it is required to explicitly state which data changes are required to be performed on each managed object in the file. (This means that it is not possible to re-import an export file without updating it with both the required changes and modifier commands). A data change is specified using a modifier attribute. This may have one of the following values:

create

delete The managed object will be deleted.

update The atrribute(s) will be updated with the supplied value(s).

action The MO Action (with the supplied parameters(s) if required) will be invoked on the managed object.

The example below contains three modifier commands:

— The first modifier will update the `userLabel` attribute with the value 'The managed object will be created and its attributes will be assigned the supplied values.`MyLabel`'.

— The second modifier will delete the Msc managed object with `Id=4`.

— The third modifier will create a new Msc managed object with `Id=5` and the specified values of attributes `supportsGs`, `supportsSGs` and `mscVlrName`.

Example 2 Commands in Input Files

```
The managed object will be created and its attributes will be assigned the supplied<configData dnP →
refix="Undefined">
  <xn:SubNetwork id="NETSimW">
    <xn:MeContext id="SGSN-15B-V629">
```

```
      <xn:ManagedElement id="SGSN-15B-V629">
        <xn:VsDataContainer id="1" modifier="update">
          <xn:attributes>
            <xn:vsDataType>vsDataMmeFunction</xn:vsDataType>
            <xn:vsDataFormatVersion>EricssonSpecificAttributes</xn:vsDataFormatVersion>
            <es:vsDataMmeFunction>
              <es:userLabel>MyLabel</es:userLabel>
            </es:vsDataMmeFunction>
          </xn:attributes>
        </xn:VsDataContainer>
        <xn:VsDataContainer id="1">
          <xn:attributes>
            <xn:vsDataType>vsDataSgsnMme</xn:vsDataType>
            <xn:vsDataFormatVersion>EricssonSpecificAttributes</xn:vsDataFormatVersion>
            <es:vsDataSgsnMme/>
          </xn:attributes>
          <xn:VsDataContainer id="4" modifier="delete">
            <xn:attributes>
              <xn:vsDataType>vsDataMsc</xn:vsDataType>
              <xn:vsDataFormatVersion>EricssonSpecificAttributes</xn:vsDataFormatVersion>

              <es:vsDataMsc/>
            </xn:attributes>
          </xn:VsDataContainer>
          <xn:VsDataContainer id="5" modifier="create">
            <xn:attributes>
              <xn:vsDataType>vsDataMsc</xn:vsDataType>
              <xn:vsDataFormatVersion>EricssonSpecificAttributes</xn:vsDataFormatVersion>
              <es:vsDataMsc>
                <es:supportsGs>true</es:supportsGs>
                <es:supportsSGs>true</es:supportsSGs>
                <es:mscVlrName>5</es:mscVlrName>
              </es:vsDataMsc>
            </xn:attributes>
          </xn:VsDataContainer>
        </xn:VsDataContainer>
      </xn:ManagedElement>
    </xn:MeContext>
  </xn:SubNetwork>
</configData>
```

Constraints on Data in Import Files

The following constraints are applicable on managed object level:

— The managed object class exists in the model for the specified network element.

— The managed object class exists at the specified point in the model containment hierarchy.

— For create or delete operations, the managed object class does not have the property system created.

— For create operations, all attributes which are mandatory for the managed object class are specified.

— For update or delete operations, the managed object exists.

— Create operation is not supported for Network Element Managed Object.

The following constraints are applicable on attribute level:

— The attribute exists in the model (for the specified managed object class).

— The attribute is not read only.

— The attribute is not restricted.

— The attribute is of the correct type (for example integer or string).

— For numeric and boolean data types, the attribute value is within the allowed value range, or set of values.

— For strings, the attribute value is within the allowed length range.

— For enumerations, the attribute has a valid value from a predefined allowed set of string values.

— For multi-valued (list) attributes, the number of values is within the allowed cardinality range (for create or modify operations).

— For managed object references, the referenced managed object exists, or is contained earlier in the import file.

The following constraints are applicable on MO action level:

— The action exists in the model (for the specified managed object class).

— The action has a void return type.

4 Vendor Specific Schema Extension

> Most of the data in a 3GPP Bulk CM import or export file is vendor specific. The following sub-chapters describe how to represent Ericsson vendor specific data in a 3GPP compliant manner.

vsDataContainer

The 3GPP file format outlines the concept of vendor specific data in the Bulk CM IRP XML File Format [1]. There are two types of vendor specific data:

— Extension data to managed object classes defined in 3GPP schemata

— Vendor specific managed object classes that are not defined in 3GPP schemata

The MOC vsDataContainer is used to define vendor specific data and is defined as having the following attributes:

vsDataContainerId
: An attribute whose 'name+value' can be used as a relative distinguished name (RDN) when naming an instance of this object class. This RDN uniquely identifies the object instance within the scope of its containing (parent) object instance.

vsDataType
: Type of vendor specific data contained by this instance, for example cell specific parameters for power control or re-selection or a timer. The type itself is also vendor specific.

vsDataFormatVersion
: Name of the data format.

vsDataAbc
: This is a container of the attributes of an managed object with managed object class Abc.

Extending a Standard Managed Object

In the example below the managed object class MeContext, which is defined in the 3GPP Generic NRM IRP [3] , is extended with Ericsson vendor specific data:

— The 3GPP MeContext managed object has `id='LTE01ERBS00002'`.

— This Ericsson extensions are handled as follows:

  — A `VsDataContainer` object with the same id as the 3GPP MeContext managed object, `id=''LTE01ERBS00002''`.

- This contains an attributes element.
- The `vsDataType` attribute has value 'vsDataMeContext '.
- The `vsDataFormatVersion` has value 'EricssonSpecificAttributes'
- The `vsDataMeContext` element then contains the Ericsson additional (extension) attributes of the `MeContext` managed object. These are `MeContextId` (the id of the Ericsson extension object is the same as the id of the 3GPP object), `platformType` and `neType`.

Example 3 Extending a Standard Managed Object

```
<xn:MeContext id="LTE01ERBS00002">
  <xn:VsDataContainer id="LTE01ERBS00002">
    <xn:attributes>
      <xn:vsDataType>vsDataMeContext</xn:vsDataType>
      <xn:vsDataFormatVersion>EricssonSpecificAttributes</xn:vsDataFormatVersion →
>
      <es:vsDataMeContext>
        <es:MeContextId>LTE01ERBS00002</es:MeContextId>
        <es:platformType>CPP</es:platformType>
        <es:neType>ERBS</es:neType>
      </es:vsDataMeContext>
    </xn:attributes>
  </xn:VsDataContainer>
</xn:MeContext>
```

Representing an Ericsson Vendor Specific Managed Object

In the example below an Ericsson vendor specific managed object is handled as follows:

- In the Ericsson model the managed object class is `ENodeBFunction` and the `id` value is `'1'`.
- In the 3GPP export file this is mapped to:
  - A `VsDataContainer` object with `id='1'`
  - The `vsDataType` attribute has value `'vsDataENodeBFunction'`.
  - The `vsDataFormatVersion` has value `'EricssonSpecificAttributes'`.
  - The `vsDataENodeBFunction` element then contains the attributes of the `ENodeBFunction` managed object in the Ericsson model (a subset is shown below).

Example 4 Representing an Ericsson Vendor Specific Managed Object

```
<xn:VsDataContainer id="1">
  <xn:attributes>
    <xn:vsDataType>vsDataENodeBFunction</xn:vsDataType>
    <xn:vsDataFormatVersion>EricssonSpecificAttributes</xn:vsDataFormatVersion>
    <es:vsDataENodeBFunction>
      <es:timePhaseMaxDeviationMbms>50</es:timePhaseMaxDeviationMbms>
      <es:nnsfMode>RPLMN_IF_SAME_AS_SPLMN</es:nnsfMode>
      <es:timePhaseMaxDeviation>100</es:timePhaseMaxDeviation>
```

```
      <es:forcedSiTunnelingActive>false</es:forcedSiTunnelingActive>
      ...
    </es:vsDataENodeBFunction>
  </xn:attributes>
</xn:VsDataContainer>
```

References to vsDataContainers

— References to 3GPP defined managed objects must use the managed object class as the name component of the relative distinguished name, for example `"MeContext=LTE01ERBS00002"`.

— References to Ericsson vendor specific managed objects must use the managed object class, preprended by 'vsData', as the name component of the relative distinguished name, for example `"vsDataENodeBFunction=1"`.

— In the case where an Ericsson extension has been made to a 3GPP specified managed object it is only allowed to refer to the 3GPP specified managed object . If we assume that the MeContext object in Example 3 is contained in a `SubNetwork` with id "Dallas" the object may be referred to using the fully distinguished name `"SubNetwork=Dallas,MeContext=LTE01ERBS00002"`.

— If this `MeContext` contains a `ManagedElement` object (also defined by 3GPP) with id='1', which in turn contains an Ericsson vendor specific object `ENodeBFunction` with id='1' the `ENodeBFunction` object may be referred to using the fully distinguished name may be referred to using the fully distinguished name `"SubNetwork=Dallas,MeContext=LTE01ERBS00002,ManagedElement=1,vsDataENodeBFunction=1"`.

5 MO Actions

> This section describes the use of MO actions in the 3GPP Bulk CM schema and configuration file.

Action

As well as supporting the setting of MO attribute values, XML configuration files also support invoking certain MO actions during an import.

— Actions are operations that may be called on an MO to perform a specific configuration task, for example to add or remove an `Aal2` path to or from an existing list of `Aal2` paths.

— Bulk CM only supports actions that do not return values (void return type).

— Actions are modeled in the Bulk CM schema file in the same way as the other modifiers.

Example 5 MO Actions

```
<xn:VsDataContainer id="3" modifier="action">
```

Some actions take a parameter value or list of values. For example, `Aal2DistributionUnit.addPath` takes a list of aal2PathVccTpId values, while other actions do not; for example `vsDataMtp3bSlAnsi.activate`.

Those actions that do take parameters are modeled as structured attributes in the schema file.

The following example shows how a typical MO action is called:

The MO action `changeFrequency`, supported by `EUtranCellFDD`, is called to change the value of the `earfcndl` attribute in the `EUtranCellFDD` managed object, using the `earfcn` parameter of the `changeFrequency` action.

Example 6 MO Action changeFrequency

```
<xn:VsDataContainer id="3" modifier="action">
    <xn:attributes>
        <xn:vsDataType>vsDataEUtranCellFDD</xn:vsDataType>
        <xn:vsDataFormatVersion>EricssonSpecificAttributes.14.02</xn:vsDataFormatVersion>
        <es:vsDataEUtranCellFDD>
            <es:changeFrequency>
                <es:earfcn>17999</es:earfcn>
              </es:changeFrequency>
        </es:vsDataEUtranCellFDD>
    </xn:attributes>
</xn:VsDataContainer>
```

6 Breakpoint Support

This section describes the support for breakpoints in the 3GPP bulk import file.

6.1 Breakpoints

ENM supports commit breakpoints in the 3GPP XML bulk import file, to allow the control of what operations are committed per transaction. When commit breakpoints are not specified, the operations are grouped together into a transaction and committed to the node together, up to the system defined maximum number of operations.

When a commit `breakpoint` is specified for a given operation in the import file, the commit will be performed after that operation.The transaction will contain that operation and the previous operations for the node which have not yet been committed.

It is possible to split the committing of the operations into multiple smaller transactions. This is useful if there are dependencies between certain operations in the file, and some operations cannot be committed together with other operations.

The `breakpoint` element is added to the `genericNrm.xsd` schema, and is included as an element in the `vsDataContainer`. It is possible to optionally specify an ID and description for the `breakpoint`.

Example 7 vsDataContainer

```
<xn:VsDataContainer id="1" modifier="update">
<xn:breakpoint type="commit" id="BP_1" description="My Breakpoint"/>
<xn:attributes>
<xn:vsDataType>vsDataENodeBFunction</xn:vsDataType>
<xn:vsDataFormatVersion>EricssonSpecificAttributes.14.02</xn:vsDataFormatVersion →
>
<es:vsDataENodeBFunction>
<es:userLabel>myUserLabel4</es:userLabel>
</es:vsDataENodeBFunction>
</xn:attributes>
</xn:VsDataContainer>
```

7 Special Managed Object Handling

This section describes the usage and handling of special managed objects in the XML files.

Locally Scoped Managed Object

A locally-scoped Managed Object is a managed object whose type represents the actual hierarchy for that particular managed object. This naming pattern has to be maintained for successful import operations. The names for each level in the hierarchy are separated using '$$'. In the following example the `vsDataType` represents a static-route type that is scoped by IPv6.

Example 8 Locally Scoped Managed Object

```
<xn:vsDataType>vsDataipv6$$static-route</xn:vsDataType>
<xn:vsDataFormatVersion>EricssonSpecificAttributes</xn:vsDataFormatVersion>
<es:vsDatastatic-route>
<es:ipv6-prefix>171:59:9::/64</es:ipv6-prefix>
<es:distance/>
<es:permanent/>
<es:description/>
<es:tag/>
<es:next-hop-type>next-hop-address</es:next-hop-type>
<es:cost/>
<es:next-hop>171:45:9::5</es:next-hop>
</es:vsDatastatic-route>
```

Managed Object with Multiple Key Support

For some managed objects, their identifier is a combinations of values from a set of keys. This key had to be given as per the combinations defined in the models. The individual key values in the managed object identifier are separated using '%%'. In the following example the identifier is a combination of IPv6-prefix, next-hop-type and next-hop.

Example 9 Managed Object with Multiple Key Support

```
<xn:VsDataContainer id="171:59:9::/64%%next-hop-address%%171:45:9::5">
<xn:attributes>
<xn:vsDataType>vsDataipv6$$static-route</xn:vsDataType>
<xn:vsDataFormatVersion>EricssonSpecificAttributes</xn:vsDataFormatVersion>
<es:vsDatastatic-route>
<es:ipv6-prefix>171:59:9::/64</es:ipv6-prefix>
<es:distance/>
<es:permanent/>
```

```
<es:description/>
<es:tag/>
<es:next-hop-type>next-hop-address</es:next-hop-type>
<es:cost/>
<es:next-hop>171:45:9::5</es:next-hop>
</es:vsDatastatic-route>
</xn:attributes>
</xn:VsDataContainer
```

8 Special Attribute Handling

> This section describes the usage and handling of special attributes in the XML files.

List / Sequence / Multi-valued Attribute

Attributes that have multiple values of the same type are variously referred to as lists, sequences or multi-valued attributes. It is important to specify the multiple values in the required order. An example of the utranCellPosition attribute with two values follows:

Example 10

```
<es:vsDataUtranCell>
  <es:utranCellPosition>100</es:utranCellPosition>
  <es:utranCellPosition>200</es:utranCellPosition>
</es:vsDataUtranCell>
```

Struct

A `struct` is an attribute which contains multiple values that are individually named, and may have different data types. For example, the attribute `acBarringForCsfb` shown below has three `struct` members:

— `acBarringFactor` has an integer value.

— `acBarringForSpecialAC` has multiple values of type `boolean`.

— `acBarringTime` has an integer value.

Example 11

```
<es:acBarringForCsfb>
  <es:acBarringFactor>95</es:acBarringFactor>
  <es:acBarringForSpecialAC>true</es:acBarringForSpecialAC>
  <es:acBarringForSpecialAC>false</es:acBarringForSpecialAC>
  <es:acBarringForSpecialAC>false</es:acBarringForSpecialAC>
  <es:acBarringForSpecialAC>false</es:acBarringForSpecialAC>
  <es:acBarringForSpecialAC>true</es:acBarringForSpecialAC>
  <es:acBarringTime>64</es:acBarringTime>
</es:acBarringForCsfb>
```

For import, when working with a single `struct`, only the attribute members to be updated need be specified in the XML file.

Attributes not specified will not be updated. Supplying values for all attributes will update all `struct` members.

When updating member values of a sequence of `structs`, all attribute member values in the sequence must be specified. This is to remove any ambiguities in the operation.

Enumeration

An enumeration is an attribute with a value selected from a pre-defined set of allowed values. The allowed values are defined as strings but each string has a corresponding integer value. for example (IDLE=0, BUSY=1, LOCKED=2)

— For import and export an enumeration may be represented as a string value, or the corresponding integer value.

Choice

The "choice" statement defines a set of alternatives, only one of which can exist at any one time. The alternatives are defined in the models and the value in `xml` files must be one of the possible values for validation to pass.

Example 12

```
<xn:vsDataType>vsDataneighbor</xn:vsDataType>
    <xn:vsDataFormatVersion>EricssonSpecificAttributes</xn:vsDataFormatVersion>
    <es:vsDataneighbor>
        <es:asloop-in/>
        <es:bfd/>
        <es:accept-filter-prefix-list/>
        <es:as-override/>
        <es:remote-as>50009</es:remote-as>
        <es:peer-group/>
        <!-- Here the "neighbor-choice1" is a choice construct with possible val →
ues "external" and "internal" -->
    <es:neighbor-choice1>external</es:neighbor-choice1>
        <es:neighbor-choice>173:45:9::5</es:neighbor-choice>
        <es:advertisement-interval/>
        <es:description/>
        <es:next-hop-self/>
        <es:ebgp-multihop/>
        <es:shutdown/>
    </es:vsDataneighbor>
```

Union

A union represents a value that corresponds to one of its member types. When a string representing a union data type is validated, the string is validated against each member type, in the order they are specified in the "type" statement, until a match is found.

Example 13

```
<xn:VsDataContainer id="171:59:9::/64%%10.10.10.10%%171:45:9::5">
        <xn:attributes>
            <xn:vsDataType>vsDataipv6$$static-route</xn:vsDataType>
            <xn:vsDataFormatVersion>EricssonSpecificAttributes</xn:vsDataFormatV →
ersion>
```

```
            <es:vsDatastatic-route>
                <es:ipv6-prefix>171:59:9::/64</es:ipv6-prefix>
                <es:distance/>
                <es:permanent/>
                <es:description/>
                <es:tag/>
                <!-- Here "next-hop-type" is a union type -->
                <es:next-hop-type>10.10.10.10</es:next-hop-type>
                <es:cost/>
                <es:next-hop>171:45:9::5</es:next-hop>
            </es:vsDatastatic-route>
        </xn:attributes>
    </xn:VsDataContainer>
```

Passwords

Network Element attributes related to node passwords are removed from the export file. Examples of these attributes are: `normalUserPassword`, `rootUserPassword`, and `secureUserPassword`.

9 MO Modifier Order Rules for Import

> The rules for ordering managed object modifications in configuration data files to be imported are specified in the Bulk CM IRP Information Service.

The following rules must be followed when preparing a 3GPP Bulk CM Configuration File for Import:

- Modifier commands must be specified in the order in which they are expected to be imported.
- All delete managed object commands must precede any create managed object commands.
- It is recommended to put all child managed objects for deletion before deletion of a parent. In case a parent managed object is provided for deletion, without its child managed objects, then all of its child managed objects will be implicitly deleted.
- All parent managed objects must be created before creation of child managed objects.
- Any managed object referenced by another managed object in the same configuration file must be created before it can be referenced. For example, parents shall be created before children are created.
- All references to a managed object must be deleted before that managed object is deleted.

The specification is Bulk CM IRP Information Service [1].

10 3GPP Format Import Limitations

Currently support for modifiers on Generic NRM MO's is not present for 3GPP format.

To import `Transport Common Information Model` (TCIM) links, the import file must contain relevant TCIM link data under the `Network=1` datacontainer, and the involved interfaces must be configured on the nodes.

TCIM Model Description document details information on the managed objects, attributes and relationships. See *TCIM Model Description*, 3/15519-CNA 403 3141.

11 Resources

A set of files are available to help you to understand and perform Bulk CM Import and Export using 3GPP.

File Type	File Name	Definition	Discussed in Chapter
XSD	configData.xsd	Defines the overall structure of 3GPP import and export files.	File Structure on page 5
XSD	GenericNrm.xsd	Defines high-level Managed Objects such as `SubNetwork`, `MeContext`, and `ManagedElement`.	File Structure on page 5
XML	ActionChangeFrequency.xml	Provides an example of how an MO action can be called, in this case, `changeFrequency`.	MO Actions
XML	CreateEUtranCellFDD.xml	Provides an example of a create operation, in this case, the creation of `EuttranCellFDD`.	File Structure on page 5
XML	DeleteEUtranCellRelationFDD.xml	Provides an example of a delete operation, in this case, the deletion of `EUtranCellRelationFDD`.	File Structure on page 5
XML	UpdateChangeFrequencyEutranCellFDD.xml	Provides an example of an update operation, in this case, the update of `ChangeFrequencyEutranCellFDD`.	File Structure on page 5
Excel	OSS-RC to ENM Bulk Configuration Differences Report.xls	Provides details of attributes (organized by node type) that are stored locally in OSS-RC, included in the OSS-RC BULK-CM 3GPP export file, and excluded from or represented differently in the ENM BULK-CM 3GPP file. Gives reasons for exclusion of attributes. Also provides conversions and mappings. For more information, refer to the header page of the spreadsheet itself.	Changes in Import and Export File Content from OSS-RC to ENM on page 3

Reference List

Table 1

[1] *3GPP TS 32.612 V4.6.0 Bulk Configuration Management IRP: Information Service*, Not available in this library, refer to 3GPP documentation or www.3gpp.org.

[2] *3GPP TS 32.615 V4.5.0 Bulk Configuration Management IRP: XML File Format Definition*, Not available in this library, refer to 3GPP documentation or www.3gpp.org.

[3] *3GPP TS 32.622 V4.4.0 (2003-06) Generic network resources Reference Point (IRP): Network Resource Model*, Not available in this library, refer to 3GPP documentation or www.3gpp.org.

[4] *ENM Statement of Compliance for 3GPP Bulk CM IRP*, 1/17402-CNA 403 2977

[5] *ENM Glossary of Terms*, 1/0033-AOM 901 151

[6] *ENM Configuration System Administration Guide*, 1/1543-AOM 901 151-1