ERICSSON

CM Events External REST Northbound Interface

Interwork Description

1/155 19-CNA 403 3055 Uen AA

Contents

1 CM Events Northbound Interface

This document describes the external RESTful web service interface (HTTP) provided by the Configuration Management (CM) Events Northbound Interface (NBI) application.

The CM Events NBI is a feature deployed in the ENM domain management for external systems. These systems include Network Resource Management System, and access to CM event data using a machine-to-machine interface.

The Dynamic CM NBI feature offers a set of interfaces for external machine systems to perform CM-related operations towards nodes managed by ENM. This document describes the Dynamic CM NBI interface for CM Events NBI.

Functionality Overview

One of the main functions of ENM is to perform mediation of CM data, both northbound and southbound. In addition to the human consumers of this mediated data, there are machine-based consumers such as external software systems (for example, NMS). Most external systems need to maintain an up-to-date representation/copy of the mediated CM data.

The CM Events NBI application provides an interface for external systems to receive events for CM changes. These changes occur on nodes managed by ENM, and any other CM data that ENM retains, regarding those nodes (for example, Topology Administration & Mediation Configuration settings).

It is possible for external systems to access change data for events that have already been received and processed by ENM. These systems can retrieve all events since last contact.

Prerequisites

— The license for the Dynamic CM NBI value pack must be installed.

— To support secure communication from the external northbound system, a certificate must be installed in the northbound system.

Glossary of Terms

See CM Events NBI References on page 34 for a description of the terms used in this document.

2 CM Events NBI Overview

Basic REST Concepts

All services are accessible through the standard HTTP interface. The interface is designed following RESTful principles. These principles include:

- Client-server communication is stateless.

  This information must be stored/managed on the client side, and must be resent with every request.

- REST resources can be addressed using URIs.
- Standard HTTP methods (GET / POST / PUT / DELETE) can be used to manipulate these resources unless otherwise indicated.
- Error reporting is through standard HTTP response codes and messages.
- Expected input parameter encoding is:
  - Standard GET query strings attached to the base URI in GET request.

    Example: `?param1=value1&param2=value2`

  - Standard POST form encoding in POST requests with MIME type application/hal+json.
  - Standard RESPONSE form encoding in GET responses with MIME type application/hal+json
- Responses are encoded in JavaScript Object Notation (JSON) format. See *RFC 4627* in CM Events NBI References on page 34 for additional information.

For additional information about the standards, see to: *RFC 2616*, *RFC 3986*, and *RFC 4627* in CM Events NBI References on page 34.

Base URL

The RESTful services are accessible from https://<customer-domain>/config-mgmt.

Note: The exact root context is subject to change.

Hypermedia (HAL)

HAL is included in this interface: a "_links" keyword is available where it is possible to specify links.

Note: In accordance to the HAL specifications, a "self" link is mandatory.

In the context of an endpoint related to a resource, HAL describes other resources with an association to it.

In the context of root endpoints, HAL describes the endpoints for the supported services (for example: `/bulk` points to import and export services) without a specific reference to a resource.

Basic Domain Concepts

The CM Events NBI Application makes CM events available that occur on network elements. It also makes available any other CM data that ENM retains regarding those network elements.

A Network Element represents a managed element in the network which is mediated by ENM.

Configuration management mediation refers to the handling of notifications of configuration changes.

CM events are generated when there are CM changes to MOs on network elements or in the ENM topology model.

The following network element managed object CM event operations are supported by the CM Events NBI application;

— Create MO.

— Delete MO.

— Update MO (Attribute Value Change, AVC).

The following topology managed object CM event operations are supported by the CM Events NBI application;

— Network Element creation and deletion

— Changes to supervision

— Changes related to connectivity

Note: Only changes due to CM event notifications received from the Network Elements are supported. Network Element Configuration Management changes introduced into ENM using a synchronization operation are not supported. However, events reporting the node synchronization status are supported.

Filtering and Batching

The External System requests a batch of CM Events. The request can include a reference to a saved filter or an explicit set of filter criteria (same ones supported by saved filter).

The request can also include a maximum number of events to return.

3 CM Events Service Description

REST Endpoint Overview

REST Endpoints	HTTP Methods Supported		
/config-mgmt/event	GET		
/config-mgmt/event/filters	GET	POST	
/config-mgmt/event/filters/{filterId}	GET		DELETE
/config-mgmt/event/events	GET		

3.1 /config-mgmt/event

> Top context for event service

GET	Returns the list of available operations.

3.2 /config-mgmt/event/filters

Table 1

GET	Returns the collection of filters
POST	Creates a filter and returns the filter object with its unique ID. If an ID is provided, it is used, otherwise the ID is auto-generated. A filter has a 'filterClauses' array. A filterClause is a JSON object with 'attrName', 'operator', and one of either 'attrValue' or 'attrValues'. The attribute '`attrName`' is one of the attributes from the event payload, for example, 'operationType', 'moClass'. (See a sample event payload in GET events). The value for the attribute 'operator' is one of ['eq', 'neq', 'lt', 'gt', 'lte', 'gte', 'contains', 'not_contains']. The attribute '`attrValue`' is the desired value of 'attrName' to query. The attribute 'attrValues' allows the entry of multiple possible matching values for the 'attrName' to query. In such case, the OR Boolean operator is applied. This is only compatible with the 'eq' and 'contains' operators. A maximum of 25 possible values can be applied. Only one of 'attrValue' or 'attrValues' can be specified.

3.2.1 Get Response Attributes

The response attributes meaning are clear by name. The response only contains the minimal amount of information needed to identify a filter.

Some additional clarification:

Response Properties	
filters	A collection of filters.

Filter Properties in Response	
<id=<string>>	Unique ID for the filter.
<filterName=<string>>	Optional name of the filter.
<filterDescription=<string>>	Mandatory description of the filter.

3.2.2 Filter Resource

Filter Resource Properties	Description
<id=<string>>	Unique filter ID. The filter ID is optional, and if not provided, will be system generated. If an ID is provided, it can only contain the characters [a-z, A-Z, 0-9] or the underscore character. The ID is not present in POST requests.
<filterName=<string>>	Name of the filter. This is: — Mandatory if an ID is not provided. — Optional if an ID is provided.
<filterDescription=<string>>	Mandatory description of the filter.
<filterClauses=<string>>	The filter structure can contain one or more conditions represented as a filter clause, in an array. Each condition consists of an attrName, operator, and one of attrValue or attrValues. **attrValue** The value of the filterable attribute. It can be expressed by a string or a number. If expressed as a number the numerical operator is applied. The attrValue can also be expressed by a list of values. In such case, the OR boolean operator is applied. **attrValues** A list of filterable attribute values with the Boolean OR operator applied between values. **attrName** The name of the filterable attribute **operator** The applicable operator — eq - means equal operator — neq - means not equal operator — lt - means less than operator — gt - means greater than operator — lte - means less than or equal operator — gte - means greater than or equal operator — contains - means that the filter matches if attrValue is contained as a substring within the named attribute — not_contains - means that the filter only matches if attrValue is not contained as a substring within the named attribute If multiple filter clauses are specified, then the boolean operator used between the filter clauses is 'AND'. Only events matching all the filters are returned.

3.3 /config-mgmt/event/filters/{filterId}

GET	Retrieves the filter specified by its ID.
DELETE	Deletes the filter specified by its ID.

3.3.1 Accepted GET Query Parameters on Filters Resource Collection

Query Parameters	
filterId=< String > (for example, filterId=TZUF2WP7EW6LM)	A `filterId` from a previously saved filter. If this parameter is used, then the 'filterClauses' query string parameter cannot be used.

3.3.2 Accepted DELETE Query Parameters on Filters Resource Collection

Query Parameters	
filterId=< String > (for example, filterId= TZUF2WP7EW6LM)	A `filterId` from a previously saved filter.

3.4 /config-mgmt/event/events

GET	Retrieves CM events. This returns the CM events up to the limit specified, using the optional filtering and sorting. If no limit is specified, then a system constraint limit of `50000` is applied. If the limit query string attribute is set, and the query returns a higher number of events, only a subset of events are returned.

3.4.1 Accepted GET Query Parameters on Events Resource Collection

All parameters are optional. When no parameters are given, all CM event data up to the maximum number of `50000` events is returned.

Multiple parameters can be specified in a query, but each parameter can be specified only once (combination of different parameters means a logical "and" relationship).

Query Parameters	Description
<limit =< Integer >> (for example, limit=500)	The maximum number of events that the server sends. Valid values are 1 - 50000. If a limit outside that range is specified, the limit is ignored.
<filterIds=[<string>,...]> (for example, filterIds=["TZUF2WP7EW6LM"])	An array of unique filterIds from previously saved filters. 'filterIds' and 'filterClauses' are mutually exclusive. If more than one filter is specified in the array, events matching any of the specified filters are returned.
<filterClauses=[<string>,...]> (for example, [{'attrName':'user', 'operator':'contains', 'attrValue':'admin'},{'attrName':'moClass', 'operator':'eq', 'attrValues':['EUtranCellFDD', 'UtranCell']}])	An array of filter clauses. This parameter can be used in the case where the query string parameter 'filterId' is not provided. 'filterId' and 'filterClauses' are mutually exclusive. If multiple filter clauses are specified, then the boolean operator applied between the clauses is 'AND'. Only events matching all the filters are returned. This query parameter must be URL encoded.
<orderBy=<[attribute order]>> (for example, moclass asc)	Sort by an event payload attribute ('operationType', 'targetName', 'moClass', 'moFdn'). The sorting order can be specified using 'asc' for

Query Parameters	Description
	ascending order or 'desc' for descending order. Defaults to 'asc'. If 'orderBy' is not specified, then eventRecordTimestamp is used. This query parameter must be URL encoded.
<enumRepresentation=<string>>	The returned representation of 'newAttributeValues' of type ENUM can be specified as either integer or string. If a value other than string or integer is entered, all ENUMs shall be returned in string format.

3.4.2 Event Resource

Event Resource Properties	
events	A collection of events.
<status=<[PARTIAL | COMPLETE]>>	If no limit or an invalid limit was specified in the GET request, this parameter indicates: — if the response contains all the events matching the query, — or if there are possibly more events available. If a valid limit is specified in that GET request, the status parameter always returns COMPLETE.

Event Properties	Description
<id=<string>>	unique system generated event ID
<operationType=<string>>	Indicates the operation type. Can be CREATE, UPDATE, or DELETE.

<eventDetectionTimestamp=<string>>	The time in UTC timezone when the event occurred in the network. For nodes based on CPP platform, the timestamp represents the time that ENM received the notification of the event.
	For Nodes based on the Ericsson Common Information Model (ECIM), the timestamp represents the time of the event as recorded by the node.
	This parameter can be used to determine the order of events in the nodes to millisecond precision.
	`dateTime` in the XML Schema specification:
	`YYYY-MM-DDThh:mm:ss.SSSZ`
	— `YYYY` is the year.
	— `MM` is the month.
	— `DD` is the day of the month.
	— `hh` is the hour of the day as on a 24-hour clock.
	— `mm` is minutes.
	— `ss` is seconds.
	— SSS is milliseconds
	— `Z` is a literal 'Z' character indicating that this string representation of the date is in UTC
	Note:
	that no time zone can be specified; the String representations of dates are always expressed in Coordinated Universal Time (UTC). Here is an example value:
	`1972-05-20T17:33:18.231Z`
	Note:
	When specifying a timestamp as part of a filter clause you can also include fractional seconds if you wish, Here are example values with milliseconds included:
	— `1972-05-20T17:33:18.772Z`
	— `1972-05-20T17:33:18.77Z`

Event Properties	Description
	— 1972-05-20T17:33:18.7Z
<eventRecordTimestamp=<string>>	The format used is a restricted form of the canonical representationThe time in UTC format when the event was stored by ENM. CM events can be stored by ENM out of the sequence that they occurred in the network. As such, for any given event detected by ENM, a later event could be made available, over the interface, for consumption, before the earlier event. For this reason, the eventRecordTimestamp must be used in the GET request to ensure that no events are missed by the NBI client. The format is that same as the timestamp in eventDetectionTimestamp.
<targetName=<string>>	The name of the node that generated the event.
<moClass=<string>>	The Managed Object Class that generated the event.
<moFDN=<string>>	The FDN of the Managed Object Instance that generated the event
<modelVersion=<string>>	The model version of the Managed Object that generated the event.
<modelNamespace=<string>>	The model namespace of the Managed Object that generated the event.
<newAttributeValues=[<string>,...]> (for example, "newAttributeValues": [{"CmFunctionId": "1"}, {"failedSyncsCount": "0"}])	An array of new attribute values represented as structured name value pairs in JSON. The array contains one or more pairs for UPDATE or CREATE operationTypes. The DELETE operationType for events on MO Classes from ENM topology (for example, NetworkElement and its descendents) include attribute values for the attributes at the time of MO deletion. The DELETE operationType, for events on all other MO Classes, the newAttributeValues are not included, and an empty array is returned.

3.5 Response Codes

A minimum set of response codes follow. Additional details can be found in the response body.

Response Codes	Description
HTTP 200 OK	The request has succeeded. The response returns information on the message-body.
HTTP 201 Created	The request has been fulfilled and resulted in a new resource being created.
HTTP 204 No Content	The request has succeeded. The response does not need to return a message-body.
HTTP 400 Bad Request	The request could not be understood by the server as a result of malformed syntax. Information message about the specific error is also returned. Examples: Missing mandatory properties, conflicting names.
HTTP 401 Unauthorized	The user does not have the necessary authorization to access the resource. Contact the system administrator.
HTTP 403 Forbidden	The system does not have the necessary license to access the resource. Contact the system administrator.
HTTP 404 Not Found	The server has not found anything matching the Request-URI.
HTTP 5XX	This represents any response code 500-599. The request has failed. The response code indicates problem on the service side.

4 CM Events Working with the NBI

> Describes, with examples, how to use the CM Events NBI REST feature. The examples provided demonstrate how the client of CM Events NBI must use the interface.

Authorization

A resource is defined for the CM Events NBI:

— cm-events-nbi

Three capabilities are available for use with custom roles:

1. read - Users can perform HTTPS GET operation.
2. create - Users can perform HTTPS POST operation to create persistent filters.
3. delete - Users can perform HTTPS DELETE operation to delete persistent filters.

Two predefined roles with the mentioned capabilities are available:

1. `CM_EVENTSNBI_Operator`
2. `CM_EVENTSNBI_Administrator`

The `CM_EVENTSNBI_Administrator` role can perform HTTPS POST, DELETE, and GET operations. The `CM_EVENTSNBI_Operator` role can only perform HTTPS GET.

4.1 Authentication

Prerequisites

— A valid Username and Password, with a CM Events Rest NBI role or capability to access the ENM System.

— A fully deployed ENM System.

Authentication via NBI

Establish a session with the ENM system. See *Export ENM PKI Root CA Certificate* section in *ENM Public Key Infrastructure System Administrator Guide*, 2/1543-AOM 901 151-3 for details, and to *Establish a User Session over REST*, and *Curl examples for Identity and Access Management* sections in the *ENM*

Identity and Access Management Programmer's Guide, *19817-CNA 403 3016*, for details.

The session is required to call the CM Events NBI operations.

4.2 Usage Scenarios

> Describes some examples of the cm events NBI interface. In these examples, the curl command is used to send the REST requests.

Before the requests can be submitted, you must be authorized and a cookie file received. This cookie file must be provided for the subsequent requests.

For security purposes a file containing the user credentials should be created using an editor. The contents of the file should be in the following format:

```
&IDToken1=<username>&IDToken2=<password>
```

In the following `<customer-domain>` must be replaced by the hostname of the ENM instance.

The `<user-credentials-filename>` must be replaced by the user credentials file name created, and include the file path with the file name, if the file is not in the current directory.

Use the following to request authorization and get the cookie:

```
curl --insecure --request POST --data-ascii @<user-credentials-filename> --cooki →
e-jar cookie.txt 'https://<customer-domain>/login'
```

4.2.1 Basic Query

> These examples contain basic queries which do not use filters.

4.2.1.1 Get Root Context

> List the supported REST endpoints for this root context.

Required user role: `CM_EVENTSNBI_Operator` or `CM_EVENTSNBI_Administrator`

Request:

```
curl -4 -G -s --insecure --request GET --cookie cookie.txt
'https://<customer-domain>/config-mgmt/event'
```

Response:

```
{
  "_links": {
    "self": {
      "href": "/config-mgmt/event"
    },
    "filters": {
      "href": "/config-mgmt/event/filters/"
    },
    "events": {
      "href": "/config-mgmt/event/events/"
    }
  }
}
```


4.2.1.2 Get One Event and Order by Descending Detection Time

> Retrieve all events for all nodes. Limit the number of events returned and sort by the event detection time from newest to oldest.

Required user role: `CM_EVENTSNBI_Operator` or `CM_EVENTSNBI_Administrator`

Request:

```
curl --insecure -s -G --data-urlencode 'limit=1' --data-urlencode
'orderBy=eventDetectionTimestamp desc' --cookie cookie.txt
'https://<customer-domain>/config-mgmt/event/events'
```

Response:


```
{
    "_links": {
        "self": {
            "href": "/config-mgmt/event/events/"
        }
    },
    "events": [
        {
             "eventDetectionTimestamp": "2018-03-22T05:51:11.826Z",
            "eventRecordTimestamp": "2018-03-22T05:51:21.706Z",
            "id": "66f723c0-d5e6-4dba-ab5a-bfd79cb95245",
            "moClass": "CmNodeHeartbeatSupervision",
            "moFDN": "NetworkElement=K3C128801,CmNodeHeartbeatSupervision=1",
            "modelNamespace": "OSS_NE_CM_DEF",
            "modelVersion": "1.0.1",
            "newAttributeValues": [
                {
                    "active": "false"
                }
            ],
            "operationType": "UPDATE",
            "targetName": "K3C128801"
        }
        }
    ],
    "status": "COMPLETE"
}
```

4.2.2 Non-Persistent Filter Query

> These examples contain queries to return the events using a non-persistent filter.

4.2.2.1 Get UPDATE Events

> Retrieve all UPDATE events.

Required user role: `CM_EVENTSNBI_Operator` or `CM_EVENTSNBI_Administrator`

Request:

```
curl --insecure -s -G --cookie cookie.txt --data-urlencode
'filterClauses=[{"attrName":"operationType","operator":"eq","attr
Value":"UPDATE"}]' 'https://<customer-domain>/config-mgmt/event/
events'
```

Response:


```
{
    "_links": {
        "self": {
            "href": "/config-mgmt/event/events/"
        }
    },
    "events": [
    {
      "id": "d6533fb7-ac0e-42a6-a689-fbe1683c5051",
      "operationType": "UPDATE",
      "moClass": "CmNodeHeartbeatSupervision",
      "eventRecordTimestamp": "2017-05-02T06:39:25.133Z",
      "targetName": "LTE02ERBS00001",
      "moFDN": "NetworkElement=LTE02ERBS00001,CmNodeHeartbeatSupervision=1",
      "modelNamespace": "OSS_NE_CM_DEF",
      "modelVersion": "1.0.1"
      "eventDetectionTimestamp": "2017-05-02T06:39:16.158Z",
      "newAttributeValues": [
        {
          "active": "true"
        }
      ]
    }
    ],
    "status": "COMPLETE"
}
```


4.2.2.2 Get All Events for a Specific Network Element

> Retrieve all events for a specific node.

Required user role: `CM_EVENTSNBI_Operator` or `CM_EVENTSNBI_Administrator`

In the following command replace <node-name> with the specific node name required.

Request:

```
curl --insecure -s -G --cookie cookie.txt --data-urlencode
'filterClauses=[{"attrName":"targetName","operator":"eq","attrVal
ue":"<node-name>"}]' 'https://<customer-domain>/config-mgmt/
event/events'
```

Response:


```
{
    "_links": {
        "self": {
            "href": "/config-mgmt/event/events/"
        }
    },
    "events": [
        {
            "id": "44bc5d70-de5e-498b-9afb-5832324e6c6f",
            "operationType": "UPDATE",
            "moClass": "CmFunction",
            "eventRecordTimestamp": "2016-02-26T23:12:27.943Z",
            "targetName": "ieatnetsimv5048-21_M-MGw-C1193-V6lim38",
            "moFDN": "NetworkElement=ieatnetsimv5048-21_M-MGw-C1193-V6lim38,CmFu →
nction=1",
            "modelNamespace": "OSS_NE_CM_DEF",
            "modelVersion": "1.0.1"
            "eventDetectionTimestamp": "2016-02-26T23:12:16.372Z",
            "newAttributeValues": [
                {
                    "syncStatus": "UNSYNCHRONIZED"
                }
            ]
        },
        {
            "id": "f66232f8-e1c3-4d57-95a0-f3925914697f",
              "operationType": "UPDATE",
            "moClass": "CmFunction",
            "eventRecordTimestamp": "2016-02-26T23:12:27.943Z",
            "targetName": "ieatnetsimv5048-21_M-MGw-C1193-V6lim38",
            "moFDN": "NetworkElement=ieatnetsimv5048-21_M-MGw-C1193-V6lim38,CmFu →
nction=1",
            "modelNamespace": "OSS_NE_CM_DEF",
            "modelVersion": "1.0.1"
            "eventDetectionTimestamp": "2016-02-26T23:12:16.378Z",
            "newAttributeValues": [
                {
                    "lostSynchronization": "Fri Feb 26 23:12:16 GMT 2016"
                }
            ]
        },
        {
            "id": "3ecab607-6a9a-46c7-a2c9-5d29ebb6d0e6",
            "operationType": "UPDATE",
               "moClass": "CmFunction",
            "eventRecordTimestamp": "2016-02-26T23:15:27.944Z",
            "targetName": "ieatnetsimv5048-21_M-MGw-C1193-V6lim38",
            "modelNamespace": "OSS_NE_CM_DEF",
            "modelVersion": "1.0.1"
            "moFDN": "NetworkElement=ieatnetsimv5048-21_M-MGw-C1193-V6lim38,CmFu →
nction=1",
            "modelNamespace": "OSS_NE_CM_DEF",
            "modelVersion": "1.0.1"
            "eventDetectionTimestamp": "2016-02-26T23:15:07.066Z",
            "newAttributeValues": [
                {
                    "syncStatus": "PENDING"
                }
            ]
        },
        {
            "id": "052785eb-cbe7-4469-9ae6-444c48d36cb9",
            "operationType": "UPDATE",
               "moClass": "CmFunction",
            "eventRecordTimestamp": "2016-02-26T23:15:27.944Z",
```

```
            "targetName": "ieatnetsimv5048-21_M-MGw-C1193-V6lim38",
            "moFDN": "NetworkElement=ieatnetsimv5048-21_M-MGw-C1193-V6lim38,CmFu
nction=1",
            "modelNamespace": "OSS_NE_CM_DEF",
            "modelVersion": "1.0.1"
            "eventDetectionTimestamp": "2016-02-26T23:15:14.355Z",
            "newAttributeValues": [
                {
                    "syncStatus": "DELTA"
                }
            ]
        },
        {
            "id": "ae7d1268-e744-44cb-8911-9fc1fbc408d1",
            "operationType": "UPDATE",
            "moClass": "CmFunction",
            "eventRecordTimestamp": "2016-02-26T23:15:27.944Z",
            "targetName": "ieatnetsimv5048-21_M-MGw-C1193-V6lim38",
            "moFDN": "NetworkElement=ieatnetsimv5048-21_M-MGw-C1193-V6lim38,CmFu
nction=1",
            "modelNamespace": "OSS_NE_CM_DEF",
            "modelVersion": "1.0.1"
            "eventDetectionTimestamp": "2016-02-26T23:15:14.465Z",
            "newAttributeValues": [
                {
                    "lostSynchronization": ""
                },
                {
                    "syncStatus": "SYNCHRONIZED"
                }
            ]
        }
    ],
    "status": "COMPLETE"
}
```


4.2.2.3 Get All Events for a Specific Managed Object Classes

> Retrieve all events for specific managed object classes.

Required user role: `CM_EVENTSNBI_Operator` or `CM_EVENTSNBI_Administrator`

Request:

```
curl --insecure -s -G --cookie cookie.txt --data-urlencode
'filterClauses=[{"attrName":"moClass","operator":"eq","attrValues
":["EUtranCellFDD","CmFunction"]}]' 'https://<customer-domain>/
config-mgmt/event/events'
```

Response:


```
{
    {
    "_links": {
        "self": {
            "href": "/config-mgmt/event/events/"
        }
    },
    "events": [
        {
            "id": "a37d2a01-b33f-4fa2-b8de-ea79cc2913e3",
            "operationType": "UPDATE",
            "moClass": "EUtranCellFDD",
            "eventRecordTimestamp": "2016-02-28T12:04:03.897Z",
            "targetName": "ieatnetsimv5048-16_LTE64ERBS00041",
            "moFDN": "SubNetwork=ERBS-SUBNW-1,MeContext=ieatnetsimv5048-16_LTE64
ERBS00041,ManagedElement=1,ENodeBFunction=1,EUtranCellFDD=LTE64ERBS00041-1",
```

```
            "modelNamespace": "OSS_NE_CM_DEF",
            "modelVersion": "1.0.1"
            "eventDetectionTimestamp": "2016-02-28T12:03:11.073Z",
            "newAttributeValues": [
                {
                    "administrativeState": "1"
                },
                {
                    "operationalState": "1"
                }
            ]
        },
        {
            "id": "86c5735a-625e-4fdd-8cd9-527adc180926",
            "operationType": "UPDATE",
            "moClass": "EUtranCellFDD",
            "targetName": "ieatnetsimv5048-16_LTE64ERBS00041",
            "eventRecordTimestamp": "2016-02-28T12:04:03.897Z",
            "moFDN": "SubNetwork=ERBS-SUBNW-1,MeContext=ieatnetsimv5048-16_LTE64 →
ERBS00041,ManagedElement=1,ENodeBFunction=1,EUtranCellFDD=LTE64ERBS00041-2",
            "modelNamespace": "OSS_NE_CM_DEF",
            "modelVersion": "1.0.1"
            "eventDetectionTimestamp": "2016-02-28T12:03:11.764Z",
            "newAttributeValues": [
                {
                    "administrativeState": "1"
                },
                {
                    "operationalState": "1"
                }
            ]
        },
        {
            "id": "4438fb06-20e4-42fd-a46a-34867527e6f3",
            "operationType": "UPDATE",
            "moClass": "CmFunction",
            "eventRecordTimestamp": "2016-02-28T12:04:03.897Z",
            "targetName": "ieatnetsimv5048-16_LTE64ERBS00033",
            "moFDN": "SubNetwork=ERBS-SUBNW-1,NetworkElement=ieatnetsimv5048-16_ →
LTE64ERBS00033,CmFunction=1,
            "modelNamespace": "OSS_NE_CM_DEF",
            "modelVersion": "1.0.1"
            "eventDetectionTimestamp": "2016-02-28T12:03:11.798Z",
            "newAttributeValues": [
                {
                    "syncStatus":"TOPOLOGY"
                }
            ]
        },
        {
            "id": "225c9482-2ff6-4b6f-ad29-29e88ed01144",
            "operationType": "UPDATE",
            "moClass": "EUtranCellFDD",
            "eventRecordTimestamp": "2016-02-28T12:04:03.897Z",
            "targetName": "ieatnetsimv5048-16_LTE64ERBS00041",
            "moFDN": "SubNetwork=ERBS-SUBNW-1,MeContext=ieatnetsimv5048-16_LTE64 →
ERBS00041,ManagedElement=1,ENodeBFunction=1,EUtranCellFDD=LTE64ERBS00041-4",
            "modelNamespace": "OSS_NE_CM_DEF",
            "modelVersion": "1.0.1"
            "eventDetectionTimestamp": "2016-02-28T12:03:11.833Z",
            "newAttributeValues": [
                {
                    "administrativeState": "1"
                },
                {
                    "operationalState": "1"
                }
            ]
        },
        {
            "id": "dc2c185d-b8d8-4461-aa5d-e2c64e3f0ca6",
            "operationType": "UPDATE",
            "moClass": "EUtranCellFDD",
            "eventRecordTimestamp": "2016-02-28T12:15:03.899Z",
                "targetName": "ieatnetsimv5048-16_LTE64ERBS00042",
            "moFDN": "SubNetwork=ERBS-SUBNW-1,MeContext=ieatnetsimv5048-16_LTE64 →
ERBS00042,ManagedElement=1,ENodeBFunction=1,EUtranCellFDD=LTE64ERBS00042-11",
            "modelNamespace": "OSS_NE_CM_DEF",
            "modelVersion": "1.0.1"
            "eventDetectionTimestamp": "2016-02-28T12:10:00.658Z",
```

```
            "newAttributeValues": [
                {
                    "administrativeState": "1"
                },
                {
                    "operationalState": "1"
                }
            ]
        }
    ],
    "status": "COMPLETE"
}
}
```


4.2.2.4 Get All Events for All Managed Objects under a Specific Managed Object since a Specific Detection Time but Exclude One Managed Object

> Retrieve all events for all managed objects under ENodeBFunction, except for events on Managed Objects of type PmEventService or its descendents. Only return events detected after a specific time.

Required user role: `CM_EVENTSNBI_Operator` or `CM_EVENTSNBI_Administrator`

Request:

```
curl --insecure -s -G --cookie cookie.txt --data-urlencode
'filterClauses=[{"attrName":"eventDetectionTimestamp","operator":
"gt","attrValue":"2016-02-28T12:10:00.920Z"},
{"attrName":"moFDN","operator":"contains","attrValue":",ENodeBFun
ction="},
{"attrName":"moFDN","operator":"not_contains","attrValue":",PmEve
ntService="}]' 'https://<customer-domain>/config-mgmt/event/
events'
```

Response:


```
{
    "_links": {
        "self": {
            "href": "/config-mgmt/event/events/"
        }
    },
    "events": [
        {
            "id": "ceeea9b3-08b9-41b9-9747-752fbd11035e",
            "operationType": "UPDATE",
            "moClass": "EUtranCellFDD",
            "eventRecordTimestamp": "2016-02-28T12:16:03.897Z",
            "targetName": "ieatnetsimv5048-16_LTE64ERBS00042",
            "moFDN": "SubNetwork=ERBS-SUBNW-1,MeContext=ieatnetsimv5048-16_LTE64 →
ERBS00042,ManagedElement=1,ENodeBFunction=1,EUtranCellFDD=LTE64ERBS00042-8",
            "modelNamespace": "OSS_NE_CM_DEF",
            "modelVersion": "1.0.1"
            "eventDetectionTimestamp": "2016-02-28T12:10:00.932Z",
            "newAttributeValues": [
                {
                    "administrativeState": "1"
                },
                {
                    "operationalState": "1"
                }
            ]
        },
```

```
        {
            "id": "b8c1e7ef-110a-476c-97ec-ad19019c8227",
            "operationType": "UPDATE",
            "moClass": "EUtranCellFDD",
            "eventRecordTimestamp": "2016-02-28T12:16:18.893Z",
            "targetName": "ieatnetsimv5048-16_LTE64ERBS00042",
            "moFDN": "SubNetwork=ERBS-SUBNW-1,MeContext=ieatnetsimv5048-16_LTE64 →
ERBS00042,ManagedElement=1,ENodeBFunction=1,EUtranCellFDD=LTE64ERBS00042-9",
            "modelNamespace": "OSS_NE_CM_DEF",
            "modelVersion": "1.0.1"
            "eventDetectionTimestamp": "2016-02-28T12:10:00.968Z",
            "newAttributeValues": [
                {
                    "administrativeState": "1"
                },
                {
                    "operationalState": "1"
                }
            ]
        }
    ],
    "status": "COMPLETE"
}
```


4.2.2.5 Get All Events that Contain ossModelIdentity in Either Key or Value of newAttributeValues

If the attrName is 'newAttributeValues' from the event payload, It brings all the events which matches either key or value of newAttributeValues.

Required user role:

CM_EVENTSNBI_Operator

or

CM_EVENTSNBI_Administrator

Request:

```
curl --insecure -s -G --data-urlencode
'filterClauses=[{"attrName":"newAttributeValues","operator":"eq",
"attrValue":"ossModelIdentity"}]' --cookie cookie.txt 'https://
<customer-domain>/config-mgmt/event/events'
```

Response:


```
{
   "_links":{
      "self":{
         "href":"/config-mgmt/event/events/"
      }
   },
   "events":[
      {
         "id":"531a4921-9d8f-4c74-a0c2-3efc10361cb7",
```

```
      "eventDetectionTimestamp":"2018-12-24T04:40:25.482Z",
      "eventRecordTimestamp":"2018-12-24T04:40:30.869Z",
      "targetName":"LTE01dg2ERBS00002",
      "moClass":"NetworkElement",
      "modelNamespace":"OSS_NE_DEF",
      "modelVersion":"2.0.0",
      "moFDN":"NetworkElement=LTE01dg2ERBS00002",
      "operationType":"UPDATE",
      "newAttributeValues":[
         {
            "ossModelIdentity":"18.Q2-R43A23"
         }
      ]
   },
   {
      "id":"48e585b0-b4ab-4567-b0e1-d210b7c6a186",
      "eventDetectionTimestamp":"2018-12-24T05:11:27.239Z",
      "eventRecordTimestamp":"2018-12-24T05:11:40.794Z",
      "targetName":"LTE02ERBS00001",
      "moClass":"NetworkElement",
      "modelNamespace":"OSS_NE_DEF",
      "modelVersion":"2.0.0",
      "moFDN":"NetworkElement=LTE02ERBS00001",
      "operationType":"DELETE",
      "newAttributeValues":[
         {
            "neProductVersion":[
               {
                  "identity":"CXP102051/27",
                  "revision":"R28J01"
               }
            ]
         },
         {
            "failedSoftwareSyncsCount":"0"
         },
         {
            "technologyDomain":[
               "EPS"
            ]
```

```
            },
            {
               "utcOffset":"+00:00"
            },
            {
               "release":"J.2.300"
            },
            {
               "softwareSyncStatus":"AS_IS"
            },
            {
               "platformType":"CPP"
            },
            {
               "ossModelIdentity":"18.Q4-J.2.300"
            },
            {
               "networkElementId":"LTE02ERBS00001"
            },
            {
               "nodeModelIdentity":"18.Q4-J.2.300"
            },
            {
               "ossPrefix":"MeContext=LTE02ERBS00001"
            },
            {
               "lastSuccessfulSoftwareSync":"Mon Dec 24 04:38:40 GMT 2018"
            },
            {
               "neType":"ERBS"
            },
            {
               "managementState":"NORMAL"
            }
         ]
      },
      {
         "id":"110b0c31-55ac-450d-951b-c4472a638620",
         "eventDetectionTimestamp":"2018-12-24T05:13:37.836Z",
         "eventRecordTimestamp":"2018-12-24T05:13:50.806Z",
```

```
      "targetName":"K3C131301",
      "moClass":"NetworkElement",
      "modelNamespace":"OSS_NE_DEF",
      "modelVersion":"2.0.0",
      "moFDN":"NetworkElement=K3C131301",
      "operationType":"CREATE",
      "newAttributeValues":[
         {
            "failedSoftwareSyncsCount":"0"
         },
         {
            "networkElementId":"K3C131301"
         },
         {
            "softwareSyncStatus":"UNINITIALIZED"
         },
         {
            "platformType":"CPP"
         },
         {
            "ossPrefix":"MeContext=K3C131301"
         },
         {
            "ossModelIdentity":"16A-C.1.203"
         },
         {
            "neType":"MGW"
         },
         {
            "managementState":"NORMAL"
         }
      ]
   },
   {
      "id":"351ce111-1051-49fa-acb0-6d47fb4e58f4",
      "eventDetectionTimestamp":"2018-12-24T05:17:50.149Z",
      "eventRecordTimestamp":"2018-12-24T05:18:00.812Z",
      "targetName":"LTE02ERBS00011",
      "moClass":"NetworkElement",
      "modelNamespace":"OSS_NE_DEF",
```

```
      "modelVersion":"2.0.0",
      "moFDN":"NetworkElement=LTE02ERBS00011",
      "operationType":"DELETE",
      "newAttributeValues":[
         {
            "failedSoftwareSyncsCount":"0"
         },
         {
            "networkElementId":"LTE02ERBS00011"
         },
         {
            "softwareSyncStatus":"UNINITIALIZED"
         },
         {
            "platformType":"CPP"
         },
         {
            "ossPrefix":"MeContext=LTE02ERBS00011"
         },
         {
            "ossModelIdentity":"17A-H.1.160"
         },
         {
            "neType":"ERBS"
         },
         {
            "managementState":"NORMAL"
         }
      ]
   },
   {
      "id":"51ed60af-9aba-4908-8fb4-38258e407b82",
      "eventDetectionTimestamp":"2018-12-24T05:25:41.395Z",
      "eventRecordTimestamp":"2018-12-24T05:25:50.934Z",
      "targetName":"LTE02ERBS00001",
      "moClass":"NetworkElement",
      "modelNamespace":"OSS_NE_DEF",
      "modelVersion":"2.0.0",
      "moFDN":"NetworkElement=LTE02ERBS00001",
      "operationType":"UPDATE",
```

```
         "newAttributeValues":[
            {
               "ossModelIdentity":"18.Q4-J.2.300"
            }
         ]
      }
   ],
   "status":"COMPLETE"
}
```


4.2.2.6 Get All Events that Contain a Specific Value for ossModelIdentity in newAttributeValues

To filter only those events that contains a specific value for `ossModelIdentity` in `newAttributeValues`. The following example gets all the events that contain `17A-H.1.160` for `ossModelIdentity` in `newAttributeValues`.

Required user role:

CM_EVENTSNBI_Operator

or

CM_EVENTSNBI_Administrator

Request:

```
curl --insecure -s -G --data-urlencode
'filterClauses=[{"attrName":"newAttributeValues","operator":"eq",
"attrValue":"ossModelIdentity"},
{"attrName":"newAttributeValues","operator": "eq","attrValue":
"17A-H.1.160"}]' --cookie cookie.txt 'https://<customer-domain>/
config-mgmt/event/events'
```

Response:


```
{
   "_links":{
      "self":{
         "href":"/config-mgmt/event/events/"
      }
   },
   "events":[
      {
         "id":"351ce111-1051-49fa-acb0-6d47fb4e58f4",
```

```
      "eventDetectionTimestamp":"2018-12-24T05:17:50.149Z",
      "eventRecordTimestamp":"2018-12-24T05:18:00.812Z",
      "targetName":"LTE02ERBS00011",
      "moClass":"NetworkElement",
      "modelNamespace":"OSS_NE_DEF",
      "modelVersion":"2.0.0",
      "moFDN":"NetworkElement=LTE02ERBS00011",
      "operationType":"DELETE",
      "newAttributeValues":[
         {
            "failedSoftwareSyncsCount":"0"
         },
         {
            "networkElementId":"LTE02ERBS00011"
         },
         {
            "softwareSyncStatus":"UNINITIALIZED"
         },
         {
            "platformType":"CPP"
         },
         {
            "ossPrefix":"MeContext=LTE02ERBS00011"
         },
         {
            "ossModelIdentity":"17A-H.1.160"
         },
         {
            "neType":"ERBS"
         },
         {
            "managementState":"NORMAL"
         }
      ]
   },
   {
      "id":"3e24a8fa-098b-4d0a-a5b5-05f11a6999b6",
      "eventDetectionTimestamp":"2018-12-24T09:28:02.503Z",
      "eventRecordTimestamp":"2018-12-24T09:28:13.366Z",
      "targetName":"LTE02ERBS00036",
```

```
      "moClass":"NetworkElement",
      "modelNamespace":"OSS_NE_DEF",
      "modelVersion":"2.0.0",
      "moFDN":"NetworkElement=LTE02ERBS00036",
      "operationType":"CREATE",
      "newAttributeValues":[
         {
            "failedSoftwareSyncsCount":"0"
         },
         {
            "networkElementId":"LTE02ERBS00036"
         },
         {
            "softwareSyncStatus":"UNINITIALIZED"
         },
         {
            "platformType":"CPP"
         },
         {
            "ossPrefix":"MeContext=LTE02ERBS00036"
         },
         {
            "ossModelIdentity":"17A-H.1.160"
         },
         {
            "neType":"ERBS"
         },
         {
            "managementState":"NORMAL"
         }
      ]
   },
   {
      "id":"beb0150e-cd26-4e06-9aad-82e8047930ec",
      "eventDetectionTimestamp":"2018-12-24T09:28:02.941Z",
      "eventRecordTimestamp":"2018-12-24T09:28:13.366Z",
      "targetName":"LTE02ERBS00011",
      "moClass":"NetworkElement",
      "modelNamespace":"OSS_NE_DEF",
      "modelVersion":"2.0.0",
```

```
      "moFDN":"NetworkElement=LTE02ERBS00011",
      "operationType":"CREATE",
      "newAttributeValues":[
         {
            "failedSoftwareSyncsCount":"0"
         },
         {
            "networkElementId":"LTE02ERBS00011"
         },
         {
            "softwareSyncStatus":"UNINITIALIZED"
         },
         {
            "platformType":"CPP"
         },
         {
            "ossPrefix":"MeContext=LTE02ERBS00011"
         },
         {
            "ossModelIdentity":"17A-H.1.160"
         },
         {
            "neType":"ERBS"
         },
         {
            "managementState":"NORMAL"
         }
      ]
   },
   {
      "id":"676f4a39-4e40-4e2f-891f-d35e71139bc7",
      "eventDetectionTimestamp":"2018-12-24T09:32:11.986Z",
      "eventRecordTimestamp":"2018-12-24T09:32:23.339Z",
      "targetName":"LTE02ERBS00036",
      "moClass":"NetworkElement",
      "modelNamespace":"OSS_NE_DEF",
      "modelVersion":"2.0.0",
      "moFDN":"NetworkElement=LTE02ERBS00036",
      "operationType":"DELETE",
      "newAttributeValues":[
```

```
            {
               "failedSoftwareSyncsCount":"0"
            },
            {
               "networkElementId":"LTE02ERBS00036"
            },
            {
               "softwareSyncStatus":"UNINITIALIZED"
            },
            {
               "platformType":"CPP"
            },
            {
               "ossPrefix":"MeContext=LTE02ERBS00036"
            },
            {
               "ossModelIdentity":"17A-H.1.160"
            },
            {
               "neType":"ERBS"
            },
            {
               "managementState":"NORMAL"
            }
         ]
      }
   ],
   "status":"COMPLETE"
}
```


4.2.3 Persistent Filter Query

> These examples include requests for persistent filter management and use.

4.2.3.1 Create Persistent filter to get EUtranCellFDD UPDATE Events

> Create a filter to retrieve UPDATE events for a specific MO Class.

Required user role:

CM_EVENTSNBI_Administrator

Create a filter file as follows:

```
echo '{"filterId":"EUtranCellFDD_filter", "filterName":
"EUtranCellFDD UPDATE Events", "filterDescription": "Get all
UPDATE Events for EUtranCellFDD MOs", "filterClauses" :
[{"attrName":"moClass","operator":"eq","attrValue":"EUtranCellFDD
"},
{"attrName":"operationType","operator":"eq","attrValue":"UPDATE"}
]}' > /tmp/EUtranCellFDD_filter_LTE01.json
```

Request:

```
curl --insecure -H "Content-Type: application/json" -X POST --
cookie cookie.txt --data-binary "@/tmp/
EUtranCellFDD_filter_LTE01.json" 'https://<customer-domain>/
config-mgmt/event/filters'
```

Response:

```
{
    "_links": {
        "self": {
            "href": "/config-mgmt/event/filters/EUtranCellFDD_filter"
        }
    },
    "filterClauses": [

        {
            "attrName": "moClass",
            "attrValue": "EUtranCellFDD",
            "operator": "eq"
        },
        {
            "attrName": "operationType",
            "attrValue": "UPDATE",
            "operator": "eq"
        },
    ],
    "filterDescription": "Get all UPDATE Events for EUtranCellFDD MOs",
    "filterId": "EUtranCellFDD_filter",
    "filterName": "EUtranCellFDD UPDATE Events"
}
```

4.2.3.2 Use a Persistent filter

> Use the filter created above to retrieve UPDATE events for EUtranCellFDD MOs.

Required user role:

CM_EVENTSNBI_Operator or CM_EVENTSNBI_Administrator

Request:

```
curl --insecure -G -s --cookie cookie.txt --data-urlencode
'filterIds=["EUtranCellFDD_filter"]' --request GET 'https://
<customer-domain>/config-mgmt/event/events'
```

Response:


```
{
  "_links": {
    "self": {
      "href": "/config-mgmt/event/events/"
    }
  },
  "events": [
    {
      "id": "786fbf59-d0cc-4f9d-bf40-762135ce19d4",
      "operationType": "UPDATE",
      "moClass": "EUtranCellFDD",
      "eventRecordTimestamp": "2017-05-05T09:08:36.628Z",
      "targetName": "LTE04dg2ERBS00001",
      "moFDN": "ManagedElement=LTE08dg2ERBS00039,ENodeBFunction=1,EUtranCellFDD=
2",
      "modelNamespace": "OSS_NE_CM_DEF",
      "modelVersion": "1.0.1"
      "eventDetectionTimestamp": "2017-05-05T09:08:27.987Z",
      "newAttributeValues": [
        {
          "userLabel": "highCellPower"
        }
      ]
    },
    {
      "id": "ffaa88f4-4ac0-4ccb-86e0-74e6aa766e76",
      "operationType": "UPDATE",
      "moClass": "EUtranCellFDD",
      "eventRecordTimestamp": "2017-05-05T09:19:56.608Z",
      "targetName": "LTE04dg2ERBS00037",
      "moFDN": "NetworkElement=LTE04dg2ERBS00037,ENodeBFunction=1,EUtranCellFDD=
4",
      "modelNamespace": "OSS_NE_CM_DEF",
      "modelVersion": "1.0.1"
      "eventDetectionTimestamp": "2017-05-05T09:19:49.050Z",
      "newAttributeValues": [
        {
          "userLabel": "lowCellPower"
        }
      ]
    }
  ],
  "status": "COMPLETE"
}
```


4.2.3.3 List all Persistent filters

> List all the persistent filters. The system responds with the list of persistent filters, including the ID, name and description of each.

Required user role:

`CM_EVENTSNBI_Operator` or `CM_EVENTSNBI_Administrator`

Request:

```
 curl --insecure -G -s --cookie cookie.txt 'https://<customer-
domain>/config-mgmt/event/filters'
```

Response:


```
 {
  "_links": {
    "self": {
      "href": "/config-mgmt/event/filters/"
    }
  },
  "filters": [
    {
      "filterId": "QMLCYLQMG5MTM",
      "filterDescription": "Get all UPDATE Events for EUtranCellFDD MOs",
      "filterName": "EUtranCellRelation UPDATE Events",
      "_links": {
        "_self": {
          "_href": "/event/filters/QMLCYLQMG5MTM"
        }
      }
    },
    {
      "filterId": "UQ57EAY7DZ6LG",
      "filterDescription": "Get UPDATE Events for ManagedElement MO",
      "filterName": "ManagedElement_update_events",
      "_links": {
        "_self": {
          "_href": "/event/filters/UQ57EAY7DZ6LG"
        }
      }
    }
  ]
}
```


4.2.3.4 Retrieve specified Persistent filter details

> Retrieve all the details for a specific persistent filter, including the filter clauses.

Required user role:

`CM_EVENTSNBI_Operator` or `CM_EVENTSNBI_Administrator`

Request:

```
curl --insecure -s -G --cookie cookie.txt 'https://<customer-
domain>/config-mgmt/event/filters/EUtranCellFDD_filter'
```

Response:


```
{
  "_links": {
    "self": {
      "href": "/config-mgmt/event/filters/QMLCYLQMG5MTM"
    }
  },
  "filterClauses": [
    {
      "attrName": "moClass",
      "operator": "eq",
      "attrValue": "EUtranCellFDD"
    },
    {
      "attrName": "operationType",
      "operator": "eq",
      "attrValues": ["UPDATE","CREATE"]
    }
```

```
  ],
  "filterDescription": "Get all UPDATE Events for EUtranCellFDD MOs",
  "filterId": "QMLCYLQMG5MTM",
  "filterName": "EUtranCellFDD UPDATE Events"
}
```


4.2.3.5 Delete a Persistent filter

Required user role:

`CM_EVENTSNBI_Administrator`

Request:

```
curl --insecure --cookie cookie.txt -X DELETE 'https://
<customer-domain>/config-mgmt/event/filters/EUtranCellFDD_filter'
```

Response:

HTTP status code 200
If the operation succeeds.

HTTP status code 204
If there was nothing to delete.

4.3 Algorithm to Keep a Network Management System (NMS) Synchronized with the Network Configuration

4.3.1 Main Flow

The CM events NBI can be used to keep the database in the NMS up to date, based on events from the network, once an initial network configuration baseline has been read.

The following steps give a guideline on how to initially populate the NMS, and to keep it up to date with changes in the network.

The initial node scope for the steps below should be the full network, or a subset of the network that the NMS is managing, this is referred to as full scope in the steps below.

Steps

1. CM Export for required node scope.

   In order to get a baseline of the current configuration of the network, run a bulk CM export of the CM data for the required nodes.

   See CM Bulk Import Export and CM REST Northbound Interface for details.

2. Populate NMS database with contents of Bulk Export.

   Use the exported file to populate a NMS database. When this is completed, the NMS has a baseline of the network configuration at the time that the bulk export was started.

   Changes that occurred in the nodes' CM data after the export started may or may not be present in the export file depending on the time each node was exported.

   Therefore is it necessary to query from the start time of the export to capture all CM changes in the network.

   Note: The results of the query may include some configuration events that are already represented in the export file. For example, there may be a CREATE event for a Managed Object which already exists within the export file, or a DELETE event for a Managed Object which does not exist in the export file.

3. Retrieve events for configuration changes that occurred since a specified time.

   Note: The *specified time* (`eventDetectionTimestamp`) to use in the query is the `timestamp` of the start of the export job..

   The query must include an `orderBy eventRecordedTimestamp` ascending, and with an `eventRecordedTimestamp` greater than or equal to the specified time. If the required node scope is not the full scope, then the required list of nodes must be specified in the query.

   For example of a Full Network Query see Get UPDATE Events on page 13.

   For example of a Node Scoped Query see to Get All Events for a Specific Network Element on page 13.

4. Process the retrieved events to introduce the configuration changes into the NMS database.

   a. Update the NMS database with the event details as required.

      Note: The results of the query may include some configuration events that are already introduced into the NMS database. This is expected, and must be considered when updating the NMS database. For example, there may be a CREATE event for a Managed Object which already exists in the NMS database or a DELETE event for a Managed Object which does not exist in the NMS database.

   b. Identify if any of the nodes have been synchronized using a full sync. Add each of those nodes to a list of synched nodes. See Node Sync Handling on page 33 for details.

5. Check if NMS is up to date with latest network configuration.

The status field of the query indicates if there are more events available.

a. If the value of the `status` field in the response to the query is 'PARTIAL', there are more events to fetch. Set the time specified in the next query as the time of the last event from the query result and go to Step 3.

b. If the value of the `status` field in the response to the query is 'COMPLETE' and it was a full scoped query, record the `eventRecordedTimestamp` of the last event in the query result as the `lastFullScopeQueryTime`.

c. If there are any nodes in the list of synched nodes from Step 4 (b) (nodes synched using full sync), go to Step 1 , using the list of synched nodes as the node scope and also clear the list of synched nodes.

d. Sleep for a period of time that suits your use case. The sleep period must be a minimum of 1 minute.

e. Set the time specified in the query as `lastFullScopeQueryTime` (stored in Step 5 (b)), use the full node scope and go to Step 3.

4.3.2 Node Sync Handling

There are two types of CM synchronization (sync) in ENM:

— Full sync

— Delta sync

CM Events includes events due to CM changes detected for both these sync types.

An `Initial sync` is similar to a `Full sync`, but is specific to where a node is added to ENM for the first time, and CM supervision is turned on. CM Events does not include events for an `Initial sync`.

To get the configuration data of the node after the `Initial sync`, the NMS must perform a bulk CM Export of the node. This applies in the following scenarios:

— When node is added for the first time:

  The NMS must use CM Events NBI to listen for a `NetworkElement` create event to trigger the Export.

— When a MIM Switch happens because of software upgrade on the node:

  The NMS must use CM Events NBI to listen for an update on the `OssModelIdentity` attribute on the `NetworkElement` to trigger the Export.

5 CM Events NBI References

Reference	Description	Document Number
ENM Glossary of Terms	Glossary of ENM terminology	1/0033-AOM 901 151
RFC 2616	Hypertext Transfer Protocol — HTTP/1.1	N/A
RFC 3986	(plus errata) - the current generic URI syntax specification	N/A
RFC4627	The application/json Media Type for JavaScript Object Notation (JSON)	N/A