CTX-Nokia-NFMP User Guide

Contents

Versions

Document Revisions

The following revisions have been made to this document

Date	Revision	Notes
04/01/2020	0.1	First Draft
18/05/20	1.0	First Release

Module Versions

The following revisions have been made to this document

Date	Revision	Notes
04/01/2020	1.0	Creation of: • Nokia-NFMP-Send-Request

Preface

About this Manual

This document is a user guide for the CTX-Nokia-NFMP module.

Audience

The audience for this document is those wanting to understand how to use CTX-Nokia-NFMP module.

Related Material

Document
NSP NFM-P XML API Developer Guide (Release 17.9)

Abbreviations used in this Document

NSP Network Services Platform

NFM-P Network Functions Manager for Packet

Requirements

The CTX-Nokia-NFMP module requires the following:

- NFM-P application
- NFM-P user account with OSS Mgmt permission to access the XMLAPI. Please refer to chapter 6.3 of the **NSP NFM-P XML API Developer Guide (Release 17.9)**.

Integration

Integration with Third-Party Systems

Nokia NSP Network Functions Manager - Packet (NFM-P)

NFM-P is a network management system from Nokia NSP that is designed to manage proprietary devices as well as other third-party devices, which are called generic NEs.

This Cortex module utilises the XML API module from NFM-P to interface with it and perform the following tasks:

- Configure or retrieve NFM-P network management information
- Manipulate managed objects

For more information regarding the XML API architecture please refer to chapter 1.2 of the **NSP NFM-P XML API Developer Guide (Release 17.9)**.

This section of the user guide lays out the basic guidelines on how Cortex integrates with NFM-P using the XML API module was based on the **NSP NFM-P XML API Developer Guide (Release 17.9)**.

Cortex uses its out of the box SOAP interface to send SOAP requests to the NFM-P XML API module, which uses three types of SOAP messages: request, response and fault to handle the XML data. For information regarding the NFM-P XML message structure can be found in chapter 10 of the NSP NFM-P XML API Developer Guide.

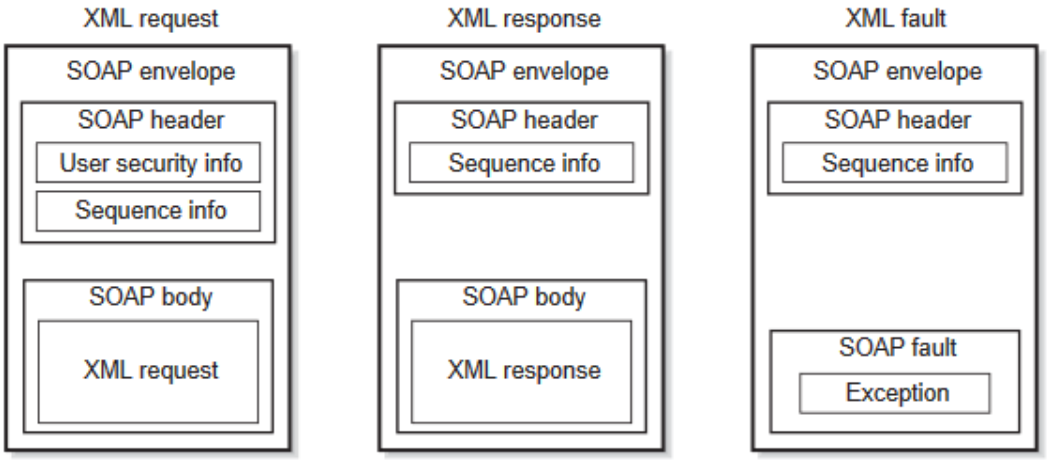

Figure 1 - NFM-P SOAP API message types[1]

[1] From chapter 10.1.2 of the NSP NFM-P XML API Developer Guide **(Release 17.9)**.

Building XML Requests

As seen in Figure 1, the XML request SOAP envelope has two main sections:

- SOAP Header, containing:
    - User security info
    - Sequence info
- SOAP Body, containing the XML request data

Below a simple example that can be used test the connectivity and access to NFM-P:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope
xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP:Header>
      <header xmlns="xmlapi_1.0">
         <security>
            <user>username</user>
            <password hashed="false">password</password>
         </security>
         <requestID>XML_API_client@n</requestID>
      </header>
   </SOAP:Header>
   <SOAP:Body>
      <ping xmlns="xmlapi_1.0" />
   </SOAP:Body>
</SOAP:Envelope>
```

The green highlighted XML part represents the user security info; the yellow highlighted XML part represents the sequence info; the turquoise highlighted XML part represents the xml request data.

The request data will be the dynamic part of the request which will allow Cortex to perform:

- Retrieval methods
- Configuration methods

Retrieval Methods

As described in chapter 14.3.1 of the NSP NFM-P XML API Developer Guide, there are 3 different methods to retrieve information from NFM-P: find, findToFile, findFaults. This section will focus on the find method, which is used to retrieve a set of objects that match a request criteria. The XML body of a find request has three main sections:

fullClassName

The package-qualified class name the retrieval is performed on. To find out what object classes exist in NFM-P refer to the XML API Reference help document which can be accessed from NFM-P documentation, see Figure 2 below for instructions.

```
<fullClassName>epipe.Epipe</fullClassName>
```


9.10 To display the XML API Reference

9.10.1 Steps

1 — Choose Help→User Documentation from the NFM-P main menu. A browser window opens to display the User Documentation Home page.

2 — Click on the XML API Reference link.

3 — Click on a package name in the upper center panel.

4 — To view the relationship and inheritance drawing of the package, click on the package name in the lower center panel.

5 — To view class hierarchy, relationship, property, and supported NE information, click on the class name in the lower center panel.

END OF STEPS

Figure 2 - XML API Reference2

Filter

A filter to be performed on the class, or child classes, properties. This can be used to limit the number of returned objects. Filters can be defined using several different operators, as specified in Figure 3. Below a simple example which will return the object with the objectFullName parameter being exactly equal to “EpipeName”:

```
<filter>
  <equal name="objectFullName" value="EpipeName" />
</filter>
```

It is also possible to use logical operators (<and> <or>) to build filters on multiple class properties, as an example:

```
<filter>
  <and>
    <equal name="objectFullName" value="EpipeName" />
    <wildcard name="description" value="%text%">
       <or>
          <equal name="administrativeState" value="down"/>
          <equal name="administrativeState" value="unknown"/>
       </or>
  <and>
</filter>
```

[2] From chapter 9.10 of the NSP NFM-P XML API Developer Guide **(Release 17.9)**.

More complex filters can be done if it is required to build a filter based on child classes properties[3]. Below is an example which filters Epipe.Epipe based on the objectFullName and that contain a child epipe.Site with siteId matching the wildcard criteria.

```
<filter>
  <equal name="objectFullName" value="EpipeName"/>
  <children>
    <filter class="epipe.Epipe" childClass="epipe.Site"
withChildrenOnly="true">
      <wildcard name="siteId" value="1.1.1.%" />
  </children>
</filter>
```

Filter element	Description	Example
equal	Return true if the value is the same as the specified value.	<equal name="localAS" value="20"/>
notEqual	Return true if the value is not the same as the specified value.	<notEqual name="localAS" value="20"/>
greater	Return true if the numeric value is greater than the specified value.	<greater name="localAS" value="20"/>
greaterOrEqual	Return true if the numeric value is greater than or equal to the specified value.	<greaterOrEqual name="localAS" value="20"/>
less	Return true if the numeric value is less than the specified value.	<less name="localAS" value="20"/>
lessOrEqual	Return true if the numeric value is less than or equal to the specified value.	<lessOrEqual name="localAS" value="20"/>
anyBit	Return true if an AND operation on the value bit and the specified value does not return 0. The filter is true if any of the "1" bits in the specified value are enabled in the property value. The function applies only to non-negative numeric types and bitmask types.	<anyBit name="parameter" value="3"/>
allBits	Return true if an AND operation on the value bit and the specified value returns the specified value. The filter is true if all of the "1" bits in the specified value are enabled in the property value. This applies only to non-negative numeric types and bitmask types.	<allBits name="parameter" value="5"/>
wildcard	Compares values using wildcards. The % character specifies 0 or more characters of any type, and the _ character specifies one character of any type.	<wildcard name="SiteId" value="1.%.%.%.%"/>
in	Return true if a numeric or string value equals one of the values in the specified comma-separated list.	<in name="parameter" value="[Edge_14, Aggregation_14, Core_14]"/>
subset	Return true if a string value is a subset of the specified string.	<subset class="service.Site" name="tier" value="123"/>
superset	Return true if a string value contains the specified string.	<superset class="service.Site" name="siteName" value="Core_"/>
notSuperset	Return true if a string value does not contain the specified string.	<notSuperset class="service.Site" name="siteName" value="Edge_"/>
between	Return true if the numeric value is between the first and second specified values.	<between name="preference" first="150" second="200" />
past	Return true if the time value is between (current time) and (current time - specified time interval).	<past time="300000" /> means find all items from the past 5 minutes (300000 ms)
empty	Match everything; this is the same as specifying nothing between the <filter> and </filter> tags.	<empty />

Figure 3 - Filter operators[4]

[3] For more information regarding Classes, including parent/child hierarchies please refer to chapter 9.5 of the of the NSP NFM-P XML API Developer Guide **(Release 17.9)**

[4] From chapter 5.6.5 of the NSP NFM-P XML API Developer Guide **(Release 17.9)**.

resultFilter

A filter that reduces the amount of object information retrieved. This useful as when retrieving a class all its properties, child classes and their properties are retrieved.

The <attribute>propertyName</attribute> tag specifies that the property is to be returned. When using multiple levels of nested resultFilter, a single empty attribute tag <attribute/> returns no attributes for that object level.

The <children> list of nested resultFilters</children> tag specifies that the given types of children be returned. The nested resultFilters can contain the class attributes specifying the package qualified class name of the child to be returned.

If the children tag is empty, no children are returned. If the children tag is not defined, then all children of this object are returned.

The class attribute on nested child resulFilters allows a class of ManageObject to be specified, which is the base class for all classes.[5]

Example of a resultFilter, which:

- Returns `description` and `objectFullName` from the parent class
- Only returns `epipe.Site` child class from the parent class
    - From this class only the `siteId` property is returned
    - From the class `epipe.Site`, only `vll.L2AccessInterface` child class is returned with only the below properties
        - `administrativeState`
        - `nodeName`
        - `portName`
    - No child classes are returned from `vll.L2AccessInterface` as the tag `<children />` is used

```
<resultFilter>
    <attribute>description</attribute>
    <attribute>objectFullName</attribute>
    <children>
    <resultFilter class="epipe.Site">
          <attribute>siteId</attribute>
          <children>
              <resultFilter class="vll.L2AccessInterface">
                  <attribute>administrativeState</attribute>
                  <attribute>nodeName</attribute>
                  <attribute>portName</attribute>
                  <children />
              </resultFilter>
              </children>
     </resultFilter>
     </children>
```

[5] From chapter 7.2.1 of the NSP NFM-P XML API Developer Guide **(Release 17.9)**


```
</resultFilter>
```

Complete XML request data example using Find method

The below XML request will retrieve all epipe.Epipe objects with the name “EpipeName”. The result set will be filtered down as explained in the section above.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope
xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP:Header>
      <header xmlns="xmlapi_1.0">
         <security>
            <user>username</user>
            <password hashed="false">password</password>
         </security>
         <requestID>XML_API_client@n</requestID>
      </header>
   </SOAP:Header>
   <SOAP:Body>
      <find xmlns="xmlapi_1.0">
         <fullClassName>epipe.Epipe</fullClassName>
         <filter>
            <equal name="objectFullName" value="EpipeName" />
         </filter>
         <resultFilter>
            <attribute>description</attribute>
            <attribute>objectFullName</attribute>
            <children>
               <resultFilter class="epipe.Site">
                  <attribute>siteId</attribute>
                  <children>
                     <resultFilter class="vll.L2AccessInterface">
                        <attribute>administrativeState</attribute>
                        <attribute>nodeName</attribute>
                        <attribute>portName</attribute>
                        <children />
                     </resultFilter>
                  </children>
               </resultFilter>
            </children>
         </resultFilter>
      </find>
   </SOAP:Body>
</SOAP:Envelope>
```

Configuration Methods

As described in chapter 16.2.1 of the NSP NFM-P XML API Developer Guide, there are 3 different methods to configure objects in NFM-P: create, modify and delete. This section will focus on the modify method, which is used to modify properties of an object that match request criteria.

The request would consist of the following:

1. Header containing security details and request ID, as previously discussed
2. Body containing two parts:
    a. Tags containing information about what to configure
    b. Configuration information about how it should be configured

In the following example, an L3AccessInterface is having its description updated.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope
xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP:Header>
      <header xmlns="xmlapi_1.0">
         <security>
            <user>username</user>
            <password hashed="false">password</password>
         </security>
         <requestID>XML_API_client@n</requestID>
      </header>
   </SOAP:Header>
   <SOAP:Body>
        <generic.GenericObject.configureInstance xmlns="xmlapi_1.0">
         <deployer>immediate</deployer>
         <synchronousDeploy>true</synchronousDeploy>
         <clearOnDeployFailure>true</clearOnDeployFailure>
         <distinguishedName>interfaceName</distinguishedName>
        <configInfo>
         <vprn.L3AccessInterface>
           <actionMask>
             <bit>modify</bit>
           </actionMask>
       <l3InterfaceDescription>NewDescription</l3InterfaceDescription>
           <children-Set/>
         </vprn.L3AccessInterface>
       </configInfo>
    </generic.GenericObject.configureInstance>
   </SOAP:Body>
</SOAP:Envelope>
```

The green highlighted tag is the parameter based on which we are searching for devices to configure, i.e. in the above example, devices with a distinguishedName of “InterfaceName” will be updated.

The yellow highlighted tag is the class of device to be configured, here an L3AccessInterface.

The turquoise highlighted tag is the action to be taken, here we wish to modify an existing device.

Finally, the red highlighted tag is the parameter to be updated, containing the new value of this parameter.

No child classes will be updated due to the `<children-Set/>` tag.

1 Nokia NFMP Subtasks

1.1 Nokia-NFMP-Send-Request

1.1.1 Overview

This subtask sends an xml request to a Nokia NFM-P XML API interface and returns the response both as raw xml and as a Cortex Structure.

1.1.2 Inputs

Input Variables	Type	Description
NNSR_i_Request	Text	REQUIRED, xml request to Nokia NFM-P
NNSR_i_Endpoint	Text	REQUIRED, the Nokia NFM-P Server endpoint. Example: http://server_name:port/xmlapi/invoke

1.1.3 Outputs

Output Variables	Type	Description
NNSR_o_Response-Str	Structure	The xml response converted to a Cortex structure.
NNSR_o_Response	Text	Raw response from Nokia NFM-P interface

