WebNMS

ReST API User Guide

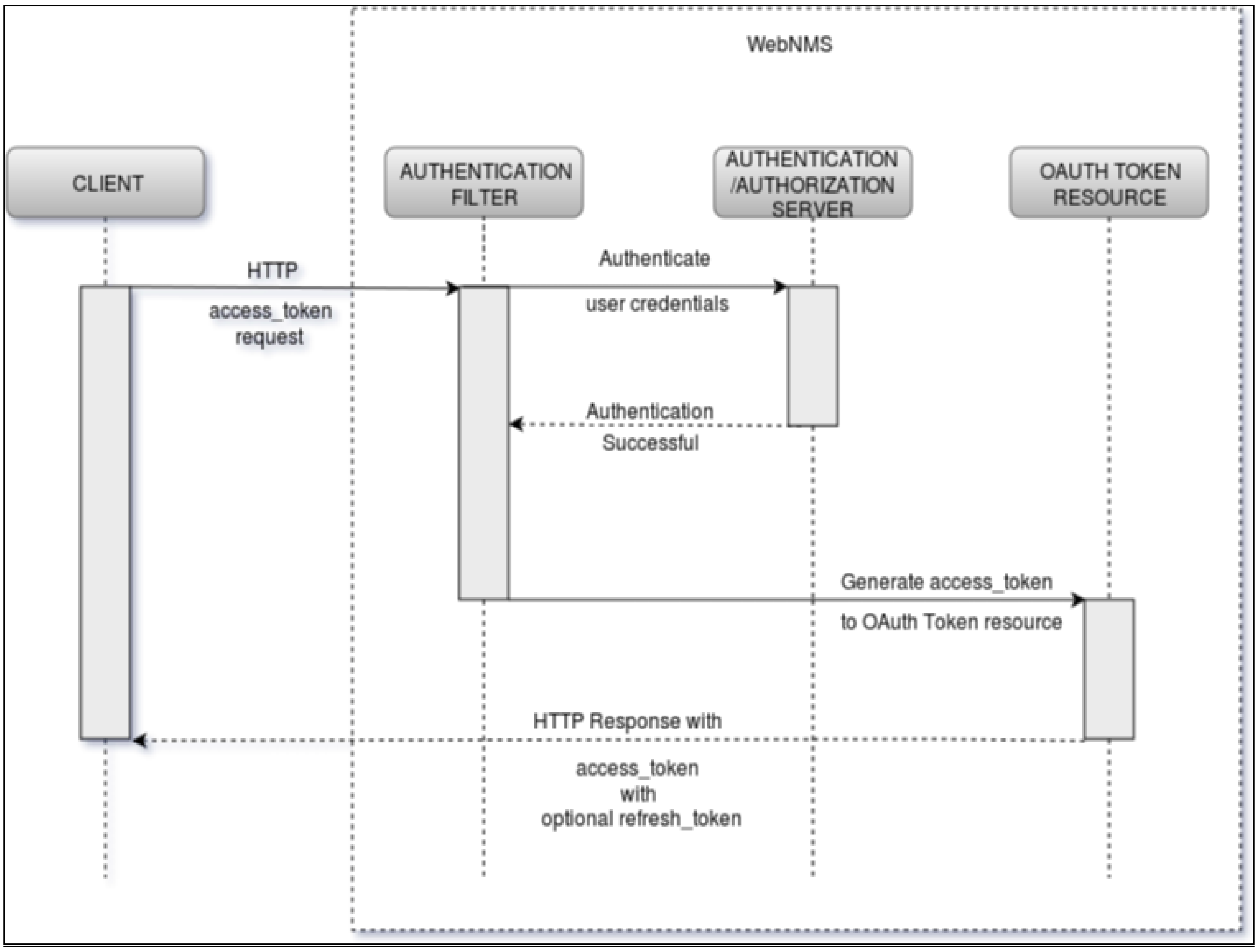
Table of Contents

WebNMS ReST Requests Types

1. access_token request

URL:

Method	URL
POST	https://<hostname>:<port>*/rest/authentication/oauth/access_token*



Sequence Diagram

1.1 Headers for the access_token request

Authorization Header

Headers:

Name	Type	Value
Authorization	Basic + Base64 encoded string of <userName>+”:”+<password>	*Basic cm9vdDpwdWJsaWM=*
WebNMS (Optional)	String	data_type="JSON",version="5.2"

Basic: (Mandatory):

In WebNMS, the “Client Credentials Grant” type described in The OAuth 2.0 Authorization Framework in rfc6749, is used. In this case the client credentials are encrypted using Base64 encryption mechanism.

Base64 encrypt the string <UserName>+”:”+<Password>, and provide it as the input for the Autorization header, with the String “Basic” as shown below,

Basic cm9vdDpwdWJsaWM=

Parameters:(application/x-www-form-urlencoded)

The following parameter uses the "application/x-www-form-urlencoded" format with a character encoding of UTF-8 in the HTTP request entity-body.

Type	Parameters	Values	Sample
POST	grant_type	<String>	client_credentials

1.2 access_token response

When data_type is JSON:

```
{
  "NmsRESTResult": {
    "message": "5.2",
    "timeStamp": "19 Feb, 2016 5:29:18 AM",
    "status": "SUCCESS",
    "token_validity": 1456012758330,
    "expires_in": 1800,
    "token_type": "bearer",
    "refresh_token": "fd5043b1c5514193a69c3234c2ba01b2",
    "access_token": "dcd16149bd98464d96e061d4774759b2"
  }
}
```

When data_type is XML:

```
<?xml version="1.0" ?>
<NmsRESTResult>
    <status>SUCCESS</status>
    <message>Access token generated successfully</message>
    <timeStamp>19 Feb, 2016 5:39:23 AM</timeStamp>
    <access_token>46a20eb41e7e44059fa4947fca5c48a8</access_token>
    <refresh_token>8a66eaabb4214cd6a8c18c9c6bb54055</refresh_token>
    <token_type>bearer</token_type>
    <expires_in>1240</expires_in>
    <token_validity>1456012802158</token_validity>
</NmsRESTResult>
```

1.3 Elements of a WebNMS ReST access_token response:

NmsRESTResult	– The root element
message	– A short message that tells about the response.

In case of access_token request the result object contains the following:

- expires_in – This is the time period in seconds, till which the access_token will be valid.

Expiry time for access token can be set using the parameter, **REST_ACCESS_TOKEN_TIMEOUT**, in the serverparameters.conf file present under <WebNMS_HOME>/conf directory.

If -1 is set, then the access_token never expires.

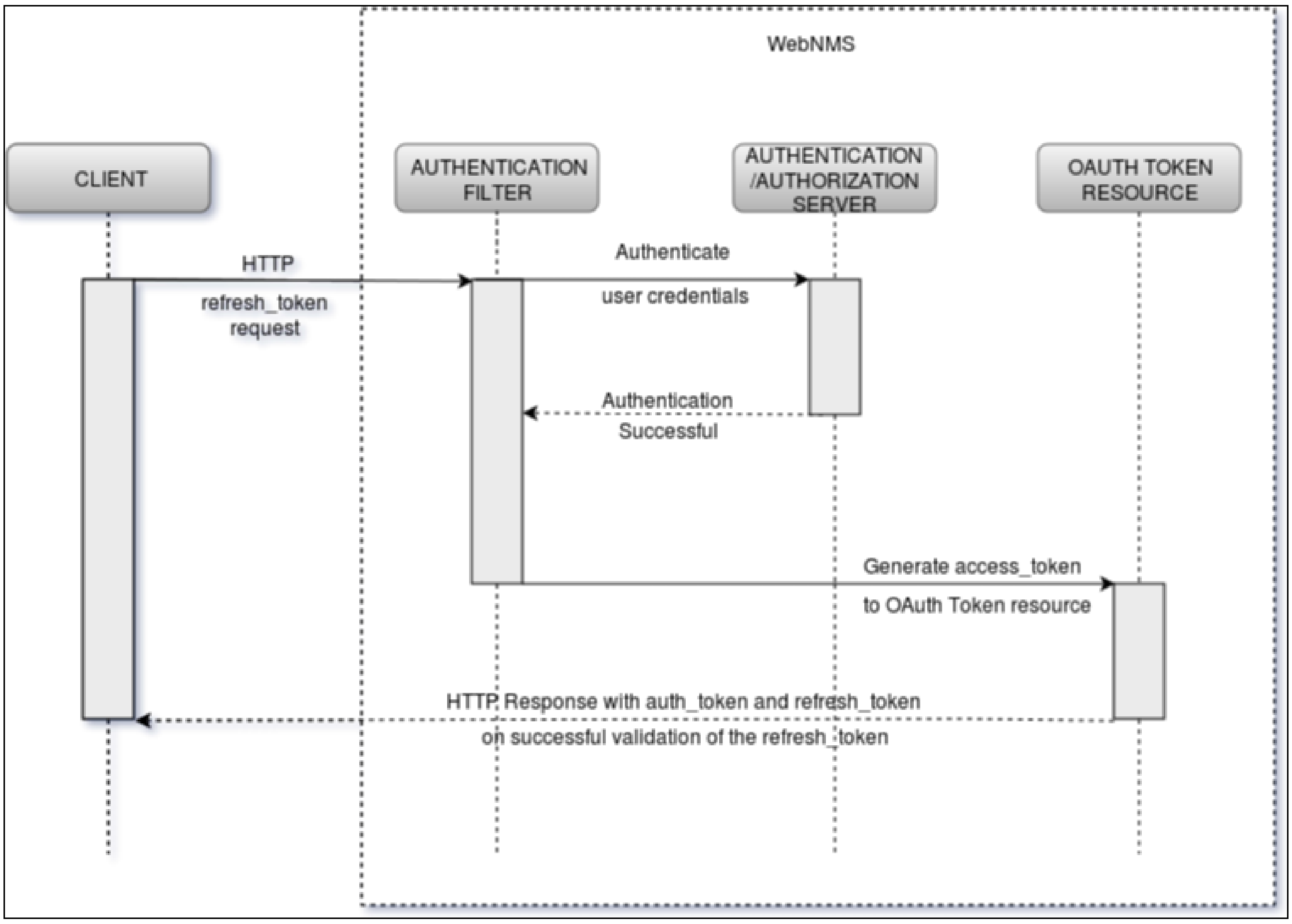
If no value is set or a value < 600 or invalid value is set, then default value of 1800 is used.

- refresh_token – The refresh_token which has to be used to refresh an expired access_token.
- token_type – The token type used in WebNMS. Only the “bearer” type is supported.
- access_token – The access_token which is used for authentication when a resource request is made.
- token_validity – The time till which the Auth Token is Valid.
- timeStamp – The time at which the ReST request is processed at the server side.
- status – The status of the ReST request.

2. refresh_token request:

URL:

Method	URL
POST	*http://<server-host>:<server-port>/rest/authentication/oauth/refresh_access_token*


WebNMS
CLIENT
AUTHENTICATION FILTER
AUTHENTICATION /AUTHORIZATION SERVER
OAUTH TOKEN RESOURCE
HTTP
refresh_token request
Authenticate
user credentials
Authentication Successful
Generate access_token
to OAuth Token resource
HTTP Response with auth_token and refresh_token
on successful validation of the refresh_token

Sequence Diagram

2.1 Headers for refresh_token request

Name	**Type**	**Value**
Authorization	Basic + Base64 encoded string of <userName>+":"+<password>	*Basic cm9vdDpwdWJsaWM=*
WebNMS (Optional)	String	data_type="JSON",version="5.2"

Parameters:

The following parameters must be present in the "application/x-www-form-urlencoded" format with the character encoding of UTF-8 in the HTTP request entity-body.

Type	**Params**	**Values**	**Sample**

POST	grant_type	<String>	refresh_token
POST	refresh_token	<String>	1f3f5dad081a4386a4fcb4c943b8264d

The refresh_token that was provided with the access_token, has to be used to refresh the expired access_token. The refresh_token has to be provided along with the “refresh_token” grant type for the “grant_type” parameter. Along with this the client credentials has to be encrypted using Base64 encryption and has to be set in the “Authorization” header.

Once the request is received, the client credentials are validated. On successful validation, the refresh_token received in the request is checked whether it belongs to the client. If yes, then a new access token is generated and sent back to the client with the refresh_token.

2.2 refresh_token response

When data_type is JSON:

```
{
  "NmsRESTResult": {
    "message": "5.2",
    "timeStamp": "19 Feb, 2016 5:29:18 AM",
    "status": "SUCCESS",
    "token_validity": 1456012758330,
    "expires_in": 1800,
    "token_type": "bearer",
    "refresh_token": "fd5043b1c5514193a69c3234c2ba01b2",
    "access_token": "dcd16149bd98464d96e061d4774759b2"
  }
}
```

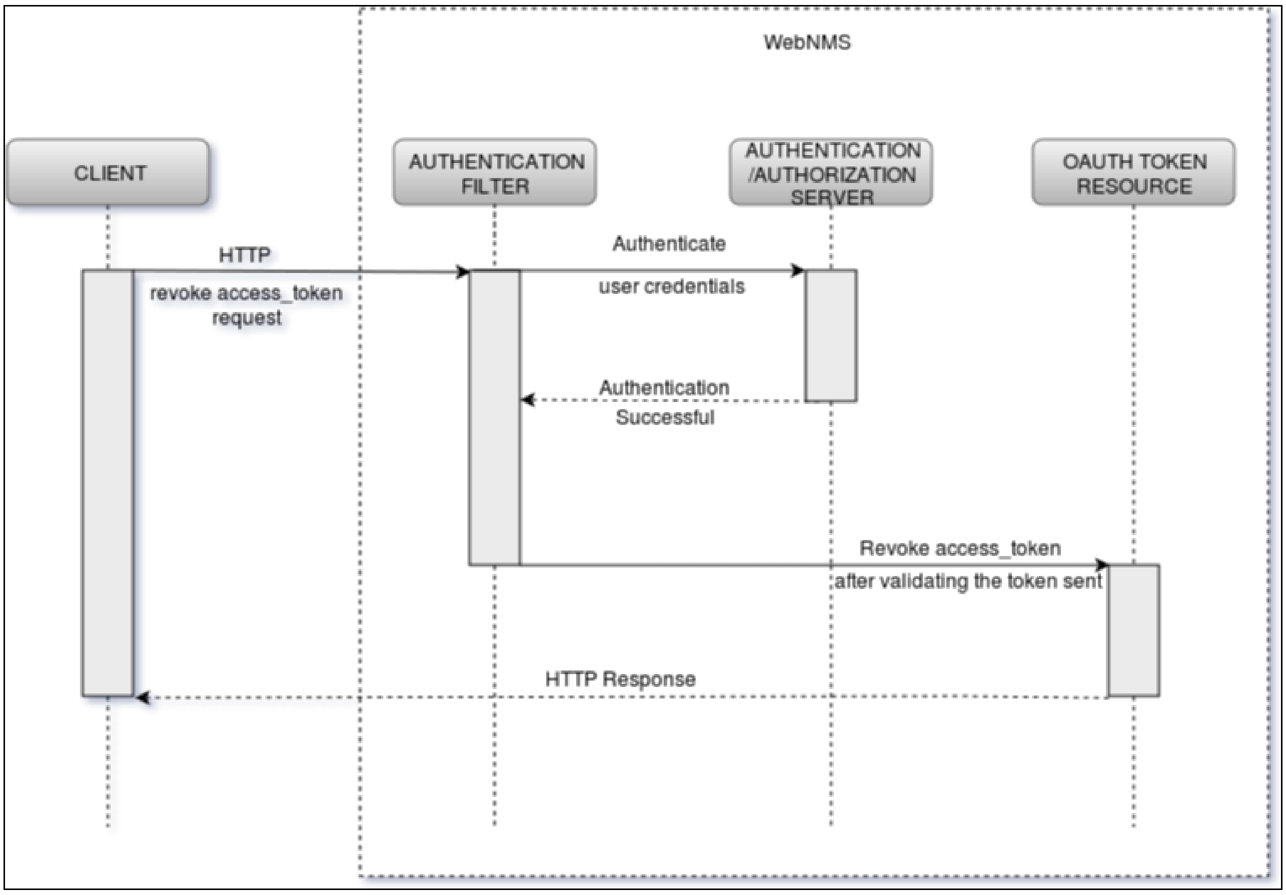
When data_type is XML:

```
<?xml version="1.0" ?>
<NmsRESTResult>
    <status>SUCCESS</status>
    <message>Access token generated successfully</message>
    <timeStamp>19 Feb, 2016 5:39:23 AM</timeStamp>
    <access_token>46a20eb41e7e44059fa4947fca5c48a8</access_token>
    <refresh_token>8a66eaabb4214cd6a8c18c9c6bb54055</refresh_token>
    <token_type>bearer</token_type>
    <expires_in>1240</expires_in>
    <token_validity>1456012802158</token_validity>
</NmsRESTResult>
```

3. Revoke access_token request

URL:

Method	URL
POST	*http://<server-host>:<server-port>/rest/authentication/oauth/revoke_access_token*



Sequence Diagram

3.1 Headers for revoke access_token request

Name	Type	Value
Authorization	Basic + Base64 encoded string of <userName>+”:”+<password>	*Basic cm9vdDpwdWJsaWM=*
WebNMS (Optional)	String	data_type="JSON",version="5.2"

Parameters:

The following parameters must be present in the "application/x-www-form-urlencoded" format with the character encoding of UTF-8 in the HTTP request entity-body.

Type	Params	Values	Sample
POST	token	<String>	705247f38bea4fdaa87d782ac8ffebe2
POST	token_type_hint	<String>	refresh_token or access_token

token (Mandatory):

This parameter should be either the access_token or refresh_token for which the access_token is to be revoked. This is based up on the following parameter.

token_type_hint(Mandatory):

This parameter can be refresh_token or access_token. If the string refresh_token is provided as the value for this parameter, then the <refresh_token>, for which the access_token is to be deleted, has to be provided for the above parameter (token). If the string access_token is provided as the value for this parameter, then the <access_token> to be revoked has to be provided as the input for the above parameter (token).

When the revoke revoke access token is received at the server side with the above parameters, the client authentication is done. On successful client authentication, the server verifies whether the token passed, was issued for the client making the revoke request. If yes then the server revokes the access_token and the refresh token.

3.2 Revoke access_token response

When data_type is for JSON:

```
 {
  "NmsRESTResult": {
    "message": "Successfully revoked the token",
    "resultObject": true,
    "timeStamp": "10 Feb, 2016 2:26:48 PM",
    "status": "SUCCESS"
  }
}
```

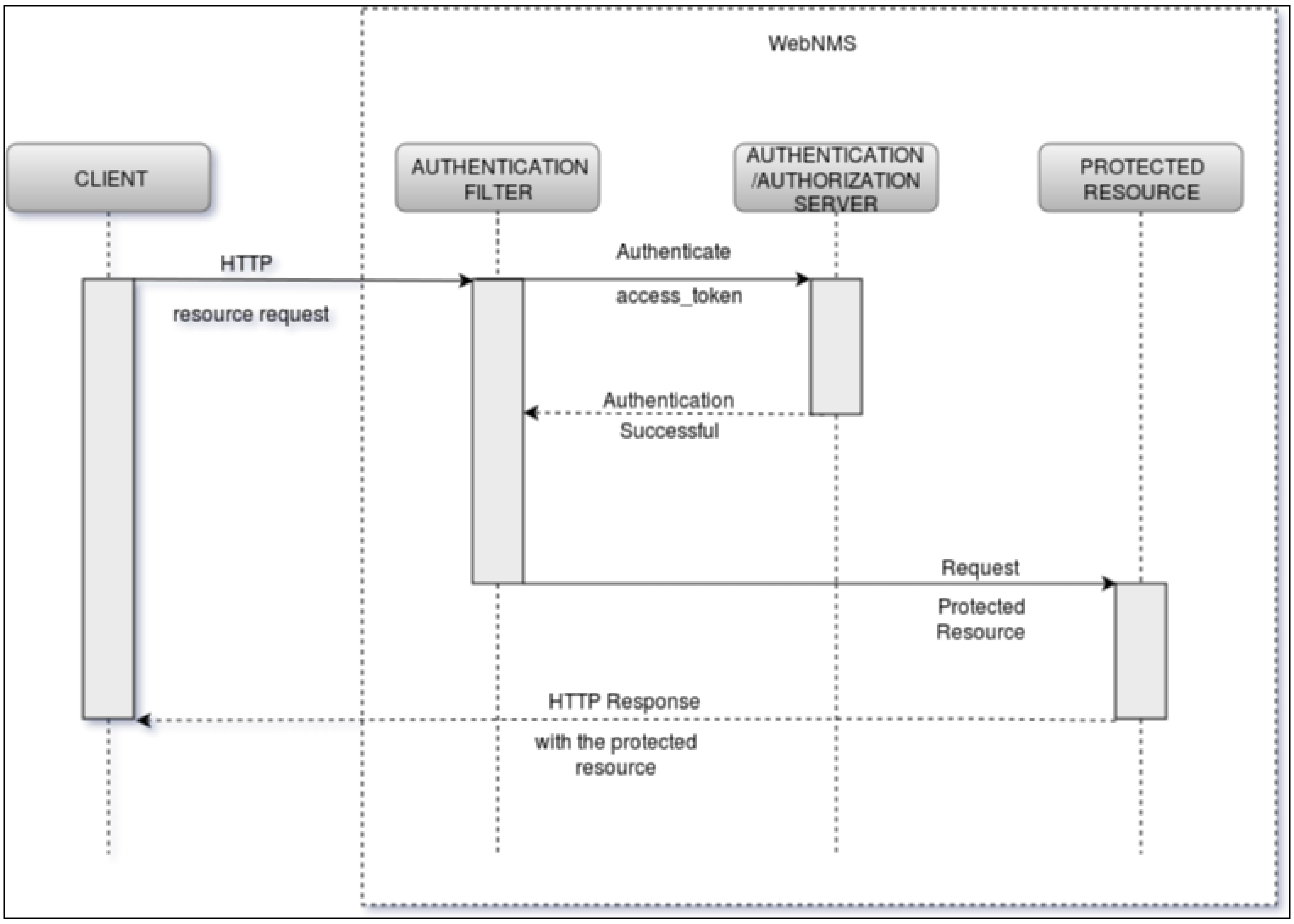
When data_type is for XML :

```
<?xml version="1.0" ?>
<NmsRESTResult>
    <status>SUCCESS</status>
    <message>Successfully revoked the token</message>
    <timeStamp>10 Feb, 2016 2:43:50 PM</timeStamp>
    <resultObject class="boolean">true</resultObject>
</NmsRESTResult>
```

4. Resource Request

URL:

Method	URL
Specific to individual resources.	*http://<server-host>:<server-port>/rest/secure/nms/topoapi/getbyname?name=192.168.208.21*



Sequence Diagram

4.1 Headers for resource request:

Name	Type	Value
Authorization	Bearer <access_token>	*Bearer bc41d30f710147b3b494e5a6fd8f6795*
WebNMS (Optional)	String	data_type="JSON",version="5.2"

Parameters (This depends up on the individual resource requests):

The following parameter is an example.

Type	Params	Values	Sample
GET	name	<String>	192.168.208.21

4.2 Resource Response

When data_type is JSON:

```
{
  "NmsRESTResult": {
    "message": "Successfully retrieved the object with name:192.168.208.21",
    "resultObject": {
      "tester": "max",
      "statusPollEnabled": false,
      "failureCount": 0,
      "type": "HP-Printer",
      "sysName": "NPI258127",
      "statusChangeTime": 1453386155409,
      "baseMibs": "RFC1213-MIB",
      "netmask": "255.255.254.0",
      "isRouter": false,
      "version": "v2",
      "childrenKeys": [],
      "parentId": 0,
      "isGroup": false,
      "sysDescr": "HP ETHERNET MULTI-ENVIRONMENT,ROM none,JETDIRECT,JD149,EEPROM
JDI23260006,CIDATE 06/12/2014",
      "name": "192.168.208.21",
      "isNetwork": false,
      "userName": "",
      "writeCommunity": "public",
      "isContainer": false,
      "members": [],
      "groups": [],
      "contextName": "",
      "ipAddress": "192.168.208.21",
      "pollInterval": 300,
      "userProperties": {},
      "failureThreshold": 1,
      "parentKey": "NULL",
      "sysOID": ".1.3.6.1.4.1.11.2.3.9.1",
      "status": 5,
      "mappedProperties": {},
      "classname": "SnmpNode",
      "community": "public",
      "parentNets": [
        "192.168.208.0"
      ],
      "moid": 141771,
      "isSNMP": true,
      "snmpport": 161,
      "managed": true,
      "isNode": true,
      "isInterface": false,
      "hostNetmask": "255.255.254.0",
      "ipaddrs": [
        "192.168.208.21"
      ],
      "dynamicProps": [],
      "displayName": "192.168.208.21",
      "isDHCP": false,
      "statusUpdateTime": 1453386155409
    },
    "timeStamp": "10 Feb, 2016 2:51:57 PM",
    "status": "SUCCESS"
  }
}
```

When data_type is XML:

```
<?xml version="1.0" ?>
<NmsRESTResult>
    <status>SUCCESS</status>
    <message>Successfully retrieved the object with
name:192.168.208.21</message>
    <timeStamp>10 Feb, 2016 2:52:31 PM</timeStamp>
    <resultObject class="com.adventnet.nms.topodb.SnmpNode">
        <name>192.168.208.21</name>
        <lazyLoaded>false</lazyLoaded>
        <moid>141771</moid>
        <displayName>192.168.208.21</displayName>
        <type>HP-Printer</type>
        <managed>true</managed>
        <statusPollEnabled>false</statusPollEnabled>
        <status>5</status>
        <failureThreshold>1</failureThreshold>
        <failureCount>0</failureCount>
        <pollInterval>300</pollInterval>
        <statusChangeTime>1453386155409</statusChangeTime>
        <statusUpdateTime>1453386155409</statusUpdateTime>
        <tester>max</tester>
        <classname>SnmpNode</classname>
        <groups class="org.hibernate.collection.PersistentSet">
            <initialized>true</initialized>
            <owner class="com.adventnet.nms.topodb.SnmpNode"
reference="../.."></owner>
            <cachedSize>-1</cachedSize>
            <role>com.adventnet.nms.topodb.ManagedObject.groups</role>
            <key class="string">192.168.208.21</key>
            <dirty>false</dirty>
            <storedSnapshot class="map"></storedSnapshot>
            <set></set>
        </groups>
        <members class="org.hibernate.collection.PersistentSet">
            <initialized>true</initialized>
            <owner class="com.adventnet.nms.topodb.SnmpNode"
reference="../.."></owner>
            <cachedSize>-1</cachedSize>
            <role>com.adventnet.nms.topodb.ManagedObject.members</role>
            <key class="string">192.168.208.21</key>
            <dirty>false</dirty>
            <storedSnapshot class="map"></storedSnapshot>
            <set></set>
        </members>
        <parentKey>NULL</parentKey>
        <parentId>0</parentId>
        <childrenKeys></childrenKeys>
        <isGroup>false</isGroup>
        <isContainer>false</isContainer>
        <dynamicProps></dynamicProps>
        <userProperties></userProperties>
        <mappedProperties class="properties"></mappedProperties>
        <ipAddress>192.168.208.21</ipAddress>
        <netmask>255.255.254.0</netmask>
        <community>public</community>
        <version>v2</version>
        <userName></userName>
        <contextName></contextName>
        <writeCommunity>public</writeCommunity>
        <snmpport>161</snmpport>
        <baseMibs>RFC1213-MIB</baseMibs>
        <isDHCP>false</isDHCP>
        <isSNMP>true</isSNMP>
        <isInterface>false</isInterface>
        <isRouter>false</isRouter>
        <isNode>true</isNode>
        <isNetwork>false</isNetwork>
        <parentNets>
            <string>192.168.208.0</string>
        </parentNets>
        <ipaddrs>
            <string>192.168.208.21</string>
        </ipaddrs>
        <hostNetmask>255.255.254.0</hostNetmask>
        <sysDescr>HP ETHERNET MULTI-ENVIRONMENT,ROM none,JETDIRECT,JD149,EEPROM
JDI23260006,CIDATE 06/12/2014</sysDescr>
        <sysName>NPI258127</sysName>
        <sysOID>.1.3.6.1.4.1.11.2.3.9.1</sysOID>
    </resultObject>
</NmsRESTResult>
```

DataType conversion

1. Converting the input data (in the form of XML string) to Java Object and Java Object to XML string is done by XStream.
2. Converting the input data String from JSON to Java Object and Java Object to JSON is done by google-gson.

Since none of the third party application converts XML/JSON to Java Object and Java Object to JSON/XML, we have to use two different applications to perform the above conversions.

Authorization

Authorization for the API methods is done based on the available Operations in WebNMS Operations tree. The following table shows the operations defined for every method in the APIs that has ReST support.

API	Method	Operation Name
TopoAPI	getByName	Topology
	getObjects	Topology
	getObjectPropsWithProps	Topology
	updateObject	Modify Object
	addObject	Topology
	deleteObject	Delete Object
	deleteObjects	Delete Object
EventAPI	getEventByID	Events
	getObjects	Events
	getObjectPropsWithProps	Events
AlertAPI	getObjects	Get Alert Details
	getObjects	Get Alert Details

PollAPI	getObjects	Administrative Operation
	getCollectedData	Administrative Operation
	getCollectedValues	Administrative Operation

Pagination

Pagination feature was introduced for API methods like getObjects, which might return huge data, to avoid problems like OOM, high bandwidth consumption etc. The user can set the startIndex and the endIndex, WebNMS will return the matching objects from startIndex to the endIndex.

To use the Pagination feature apart from the default arguments for the getObjects(), the user has to provide following parameters,

type	- This is only needed for the TopoModule.
needpagination	- To enable the Pagination feature. If set to false all the objects will be returned.
startindex	- The index from which the objects should be fetched.
endindex	- The index upto which the objects should be fetched.

1. Methods with Pagination Support

TopoAPI

1. getObjects
2. getObjectPropsWithProps

EventAPI

1. getObjects
2. getObjectPropsWithProps

AlertAPI

1. getObjects
2. getObjectPropsWithProps

1.1 Pagination Example

getobjects method in TopoAPI.

Method: GET

URL: http://<host>:<port>/rest/secure/nms/topoapi/getobject

Input:

type	-	snmp-node
classname	-	SnmpNode
criteria	-	%7B%22managed%22%3A%22true%22%7D *Note: (A JSON String {“managed”:”true”} URL encoded)*
needpagination	-	true
startindex	-	1
endindex	-	2

<u>When data_type is JSON:</u>

```
{
  "NmsRESTResult": {
    "message": "Successfully retrieved the objects.",
    "resultObject": [
      {
        "tester": "max",
        "statusPollEnabled": "false",
        "failureCount": "0",
        "type": "snmp-node",
        "sysName": "rest-test",
        "baseMibs": "RFC1213-MIB",
        "statusChangeTime": "1438097853874",
        "isRouter": "false",
        "netmask": "255.255.255.0",
        "version": "v2",
        "name": "rest-test",
        "sysDescr": "Linux rest-test 3.13.0-43-generic #72-Ubuntu SMP Mon Dec 8
19:35:06 UTC 2014 x86_64",
        "isGroup": "false",
        "isNetwork": "false",
        "userName": "",
        "writeCommunity": "public",
        "contextName": "",
        "isContainer": "false",
        "ipAddress": "XXX.XXX.XXX.XXX",
        "pollInterval": "300",
        "failureThreshold": "1",
        "parentKey": "",
        "sysOID": ".1.3.6.1.4.1.8072.3.2.10",
        "status": "2",
        "classname": "SnmpNode",
        "community": "public",
        "moid": "7",
        "isSNMP": "true",
        "snmpport": "161",
        "isNode": "true",
        "managed": "true",
        "isInterface": "false",
        "hostNetmask": "",
        "displayName": "rest-test",
        "isDHCP": "false",
        "statusUpdateTime": "1438097853874"
      },
      {
        "tester": "max",
        "statusPollEnabled": "false",
        "failureCount": "0",
        "type": "snmp-node",
        "sysName": "rest-test1",
        "baseMibs": "RFC1213-MIB",
        "statusChangeTime": "1438110500494",
        "isRouter": "false",
        "netmask": "255.255.255.0",
        "version": "v2",
        "name": "rest-test1",
        "sysDescr": "Linux rest-test1 3.8.0-44-generic #66~precise1-Ubuntu SMP
Tue Jul 15 04:01:04 UTC 2014 x86_64",
        "isGroup": "false",
        "isNetwork": "false",
        "userName": "",
        "writeCommunity": "public",
        "contextName": "",
        "isContainer": "false",
        "ipAddress": "YYY.YYY.YYY.YYY",
        "pollInterval": "300",
        "failureThreshold": "1",
        "parentKey": "",
        "sysOID": ".1.3.6.1.4.1.8072.3.2.10",
        "status": "2",
        "classname": "SnmpNode",
        "community": "public",
        "moid": "15",
        "isSNMP": "true",
        "snmpport": "161",
        "isNode": "true",
        "managed": "true",
        "isInterface": "false",
        "hostNetmask": "",
        "displayName": "rest-test1",
        "isDHCP": "false",
        "statusUpdateTime": "1438110500494"
      }
    ],
    "timeStamp": "29 Jul, 2015 6:50:59 PM",
    "status": "SUCCESS",
    "pagination": {
      "totalLength": 6,
      "startIndex": 1,
      "nextURL":
"http://<host>:<port>/rest/secure/nms/topoapi/getobjects?needpagination=true&sta
rtindex=3&endindex=4&classname=SnmpNode&match=%7B%22managed%22%3A%22true%22%7D&t
ype=snmp-node",
      "endIndex": 2
    }
  }
}
```

In the response only the objects from 1 to 5, that matches the match criteria provided will be fetched from the DB and sent in the response. Apart from the default parameters, the response for the API methods that has the Pagination support has the following extra parameters.

```
 "pagination": {
      "totalLength": 6,
      "startIndex": 1,
      "nextURL":
"http://<host>:<port>/rest/nms/topoapi/getobjects?needpagination=true&startindex=6&endindex=10&classname=SnmpNode&match=%7B%22managed%22%3A%22true%22%7D&type=snmp-node",
      "endIndex": 5
    }
```

pagination	- The root node for the pagination parameters.
totalLength	- The total number of objects present in the DB that matches the given match criteria.
startIndex	- The startIndex.
endIndex	- The endIndex.
nextURL	- The URL with all the parameters, like the criteria, classname, type, needpagination etc along with the new startIndex and the endIndex. The user can use the nexURL until, the endIndex matches the totalLength.

<u>When data_type is XML:</u>

```
<?xml version="1.0" ?>
<NmsRESTResult>
    <status>SUCCESS</status>
    <message>Successfully retrieved the objects.</message>
    <timeStamp>29 Jul, 2015 7:32:03 PM</timeStamp>
    <resultObject class="string">
        <?xml version="1.0" ?>
        <Vector>
            <properties>
                <property name="classname" value="SnmpNode"></property>
                <property name="statusChangeTime"
value="1438097853874"></property>
                <property name="managed" value="true"></property>
                <property name="statusUpdateTime"
value="1438097853874"></property>
                <property name="isNode" value="true"></property>
                <property name="community" value="public"></property>
                <property name="failureCount" value="0"></property>
                <property name="ipAddress" value="XXX.XXX.XXX.XXX"></property>
                <property name="name" value="rest-test"></property>
                <property name="parentKey" value=""></property>
                <property name="sysName" value="rest-test"></property>
                <property name="netmask" value="255.255.255.0"></property>
                <property name="userName" value=""></property>
                <property name="writeCommunity" value="public"></property>
                <property name="moid" value="7"></property>
                <property name="isDHCP" value="false"></property>
                <property name="snmpport" value="161"></property>
                <property name="status" value="2"></property>
                <property name="displayName" value="rest-test"></property>
                <property name="statusPollEnabled" value="false"></property>
                <property name="sysDescr" value="Linux rest-test 3.13.0-43-
generic #72-Ubuntu SMP Mon Dec 8 19:35:06 UTC 2014 x86_64"></property>
                <property name="isGroup" value="false"></property>
                <property name="sysOID"
value=".1.3.6.1.4.1.8072.3.2.10"></property>
                <property name="isContainer" value="false"></property>
                <property name="isSNMP" value="true"></property>
                <property name="contextName" value=""></property>
                <property name="failureThreshold" value="1"></property>
                <property name="pollInterval" value="300"></property>
                <property name="tester" value="max"></property>
                <property name="isRouter" value="false"></property>
                <property name="baseMibs" value="RFC1213-MIB"></property>
                <property name="version" value="v2"></property>
                <property name="isInterface" value="false"></property>
                <property name="hostNetmask" value=""></property>
                <property name="isNetwork" value="false"></property>
                <property name="type" value="snmp-node"></property>
            </properties>
            <properties>
                <property name="classname" value="SnmpNode"></property>
                <property name="statusChangeTime"
value="1438110500494"></property>
                <property name="managed" value="true"></property>
                <property name="statusUpdateTime"
value="1438110500494"></property>
                <property name="isNode" value="true"></property>
                <property name="community" value="public"></property>
                <property name="failureCount" value="0"></property>
                <property name="ipAddress" value="YYY.YYY.YYY.YYY"></property>
                <property name="name" value="rest-test1"></property>
                <property name="parentKey" value=""></property>
                <property name="sysName" value="rest-test1"></property>
                <property name="netmask" value="255.255.255.0"></property>
                <property name="userName" value=""></property>
                <property name="writeCommunity" value="public"></property>
                <property name="moid" value="15"></property>
                <property name="isDHCP" value="false"></property>
                <property name="snmpport" value="161"></property>
                <property name="status" value="2"></property>
                <property name="displayName" value="rest-test1"></property>
                <property name="statusPollEnabled" value="false"></property>
                <property name="sysDescr" value="Linux rest-test1 3.8.0-44-
generic #66~precise1-Ubuntu SMP Tue Jul 15 04:01:04 UTC 2014 x86_64"></property>
                <property name="isGroup" value="false"></property>
                <property name="sysOID"
value=".1.3.6.1.4.1.8072.3.2.10"></property>
                <property name="isContainer" value="false"></property>
                <property name="isSNMP" value="true"></property>
                <property name="contextName" value=""></property>
                <property name="failureThreshold" value="1"></property>
                <property name="pollInterval" value="300"></property>
                <property name="tester" value="max"></property>
                <property name="isRouter" value="false"></property>
                <property name="baseMibs" value="RFC1213-MIB"></property>
                <property name="version" value="v2"></property>
                <property name="isInterface" value="false"></property>
                <property name="hostNetmask" value=""></property>
                <property name="isNetwork" value="false"></property>
                <property name="type" value="snmp-node"></property>
            </properties>
        </Vector>
    </resultObject>
    <pagination>
        <startIndex>1</startIndex>
        <endIndex>2</endIndex>

<nextURL>http://<host>:<port>/rest/secure/nms/topoapi/getobjects?needpagination=
true&startindex=3&endindex=4&classname=SnmpNode&match=%3Cproperties%3E%3Cpropert
y%20name%3D%22managed%22%20value%3D%22true%22%3E%3C%2Fproperty%3E%3C%2Fpropertie
s%3E&type=snmp-node</nextURL>
        <totalLength>6</totalLength>
    </pagination>
</NmsRESTResult>
```

ReST Support for Custom APIs

The following are the main components in the ReST implementation for an API.

1. A resource class.
2. A handler class.
3. An entry for the resource class in the web.xml file

1. Resource Class

A resource class is the one from which the Jersey server identifies the URLs for different methods defined in the resource class.

So a resource class should have the following:

1. A class level path (for ex: @Path("/secure/testing"))
2. Methods for which ReST support has to be given.

   For each method the following should be present

   1. Http method type annotation (for ex: @GET)
   2. Method level path (for ex: @Path("/getData"))
   3. Consume annotation (for ex:@Consumes({MediaType.*APPLICATION_JSON*, MediaType.*APPLICATION_XML*}))
   4. The method should have the annotations to inject the arguments to the java method (for ex: @QueryParam("input"))

Example :

```
package test;

import javax.ws.rs.Consumes;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.QueryParam;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

@Path("/secure/testing")//the class level path
```

```
public class TestRestResourceClass
{
	//The handler class
	TestRestHandler restHandler = new TestRestHandler();


	@GET//http method annotation
	@Path("/getData")//No I18N The method level path
	@Consumes({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})

	public Response getData(@QueryParam("input") String input)
	{
		String responseStr = "";
		responseStr = restHandler.processGetData(input);
		return Response.status(200).entity(responseStr).build();
	}
}
```

2. Handler Class

It contains the business logic (API call/logic) for all the methods present in the resource class.

Example :

```
package test;

public class TestRestHandler
{
	public String processGetData(String name)
	{
		String toReturn = "Response string to be returned";

		/****************************************/
		//
		//The following are to be taken care here.
		// 1. Input Validation.
		// 2. Conversion of input argument string to
		//       objects etc.
		// 3. Customer logic.
		// 4. Error handling.
		// 5. Converting the result object to either
		//    JSON or XML String.
		//
		/****************************************/
		TestAPI testAPI = NmsUtil.getAPI("TestAPI");
		toReturn = testAPI.getData();

		return toReturn;
	}
}
```

3. web.xml file Entry

Entry for the resource class in the web.xml files present in <WebNMS_Home>/WEB-INF/ directory.

The following servlet entry will be present in the web.xml file. The customer has to provide the resource class entry in the param-value tag as shown below,

```
<servlet>
     <servlet-name>RESTFulWebNMS servlet </servlet-name>
     <servlet-class> org.glassfish.jersey.servlet.ServletContainer </servlet-
class>
      <init-param>
             <param-name> javax.ws.rs.Application</param-name>
             <param-
value>com.webnms.rest.webclient.ConfigurationInitializer</param-value>
              </init-param>
        <init-param>
        <param-name>jersey.config.server.provider.classnames</param-name>
                        <param-
value>com.webnms.rest.service.fault.AlertAPIResource;com.webnms.rest.service.fau
lt.EventAPIResource;com.webnms.rest.service.topo.TopoAPIResource;com.webnms.rest
.service.performance.PollAPIResource;com.webnms.rest.service.provisioning.NmsPro
visioningAPIResource;com.webnms.rest.sse.SSEService;org.glassfish.jersey.media.s
se.SseFeature,test.TestRestResourceClass</param-value>
       </init-param>
        <load-on-startup> 1 </load-on-startup>
        <async-supported> true </async-supported>
</servlet>
<servlet-mapping>
<servlet-name> RESTFulWebNMS servlet </servlet-name>
<url-pattern> /rest/* </url-pattern>
</servlet-mapping>
```

Testing ReST APIs using PostMan Client

1. Select the method type from the drop down list.
2. Enter the URL in the URL bar.
3. Select the Headers tab. In the headers tab, enter the following headers as key value pairs.

Key	Value
Authorization	Bearer 70c316602c264c438a204bc9c335d7e1
WebNMS	version="5.2",data_type="JSON"

4. Input:

For GET methods:

Click the Params tab and enter the data as key value pairs. For example: for the getobject method in TopoAPI the values are enterd as follows:

Key	Value
name	test-rest1

For other methods with FormParams:

1. Select the respective method type from the drop down list.
2. Enter the headers as mentioned above.
3. Enter the URL in the URL tab
4. Select the Body tab.
5. Enter the data as key value pairs.

5. Click the **Send** button. The response/error will be printed in the Response window.

Sample Java code to get access tokens and send ReST requests:

```java
package test;

import java.io.UnsupportedEncodingException;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.Form;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
import javax.xml.bind.DatatypeConverter;

import org.json.JSONObject;

public class RestThroughCode
{
	static String hostIp = "localhost";
	static String port = "9090";
	static String commonPart = "http://"+hostIp+":"+port;

	static String toUpdateObject = "";

	public static void main(String[] args)
	{
		String accessToken = getAccessToken(hostIp,port);
		String object = testGetByName(accessToken);
		testUpdateObject(accessToken,object);
	}

	private static void testUpdateObject(String accessToken, String
updateObjString) {

		String url = commonPart +"/rest/secure/nms/topoapi/updateobject";

		String authHeader = "Bearer "+accessToken;

		String webNmsHeader = "data_type=\"JSON\"";

		Client client = ClientBuilder.newBuilder().build();
		JSONObject jObj = null;

		try
		{
```

```
                jObj = new JSONObject(updateObjString);
                jObj.put("managed", "false");
            }
            catch(Exception e)
            {
                e.printStackTrace();
            }

            Form form = new Form();
            form.param("obj", jObj.toString());//The JSONString of the object to
be updated
            form.param("retainuserproperties", "true");
            form.param("dealwithlocks", "true");
            form.param("fullclassname", "com.adventnet.nms.topodb.Node");

            WebTarget target = client.target(url);

            Response response = target.request(MediaType.APPLICATION_JSON)
                        .header("Authorization", authHeader)
                        .header("WebNMS", webNmsHeader)
                        .method("PUT", Entity.form(form));

            System.out.println("UpdateObject response
===>"+response.readEntity(String.class));

        }

        private static String testGetByName(String accessToken)
        {
            String url = commonPart +"/rest/secure/nms/topoapi/getbyname";

            String authHeader = "Bearer "+accessToken;

            String webNmsHeader = "data_type=\"JSON\"";

            Client client = ClientBuilder.newBuilder().build();

            WebTarget target = client.target(url);
            String getObjectResponse = target.queryParam("name",
"192.168.208.1").request(MediaType.APPLICATION_JSON_TYPE)
                        .header("Authorization", authHeader)
                        .header("WebNMS", webNmsHeader)
                        .get(String.class);

            return getResultObject(getObjectResponse);
      }

      private static String getAccessToken(String hostIp, String port)
      {
            String accessToken = "";
            String accessURL =
commonPart+"/rest/authentication/oauth/access_token";
            String userName = "root";
            String password = "public";
            String encodedString = null;

            try
            {
                  encodedString  =
DatatypeConverter.printBase64Binary((userName+":"+password).getBytes("UTF-8"));
            }
            catch (UnsupportedEncodingException e1)
            {
                  e1.printStackTrace();
            }

            String authHeader = "Basic "+encodedString;
            String webNmsHeader = "data_type=\"JSON\"";
            Client client = ClientBuilder.newBuilder().build();

            WebTarget target = client.target(accessURL);
            Form form = new Form();
            form.param("grant_type","client_credentials");

            Response response = target.request(MediaType.TEXT_PLAIN).
```

```
                        header("Authorization",
authHeader).acceptEncoding(MediaType.APPLICATION_FORM_URLENCODED)
                        .header("WebNMS", webNmsHeader)
                        .method("POST", Entity.form(form));

            String accessResponse = response.readEntity(String.class);
            String accessTokenJSON =
getAccessObjectFromResponse(accessResponse);

            try
            {
                  JSONObject jObj = new JSONObject(accessTokenJSON);

                  accessToken = jObj.getString("access_token");
            }
            catch(Exception e)
            {
                  e.printStackTrace();
            }

            return accessToken;
      }

      public static String getAccessObjectFromResponse(String inputString)
      {
            String result = "";
            try
            {
                  JSONObject jObj = new JSONObject(inputString);
                  result  = jObj.getString("NmsRESTResult");

            }
            catch(Exception e)
            {
                  e.printStackTrace();
            }
            return result;
      }
```

```
	public static String getResultObject(String inputString)
	{
		String result = "";
		try
		{
			JSONObject jObj = new JSONObject(inputString);
			String nmsRestResult  = jObj.getString("NmsRESTResult");
			JSONObject resObj = new JSONObject(nmsRestResult);
			result = resObj.getString("resultObject");
		}
		catch(Exception e)
		{
			e.printStackTrace();
		}
		return result;
	}
}
```