Client API

Description

It is worthwhile and very important to have a client API for PerfSONAR which have following functions

- Provide a wrapper arround perfSONAR services
- Provide an interface for each services to fetch data for visualization and analysis.
- Hiding the XML parsing complexity behind request/response

Use Cases

- What are all the interfaces that a service publish its utilization to perfSONAR framework?
- What is the utilization for any specific interface supported by perfSONAR?

Implementation

Overview

There are two ways to implement this

1. Query Lookup service
    - Query the lookup service for the metadata of a specific evenType
    - Lookup service will return all the metadata and its associated service url to query
    - Query the service url for the specific information directly from the service url
2. Static Mapping of Service URL
    - Get the MAs.conf static service url mapping file from perfSONARUI.
    - Query metadata for an eventType e.g. utilization for each service.
    - Query the interface utilization request directly to service

Considering the current deployement status of lookup service, not all the services are registers. There is an implementation of Multi-Lookup service, where each local domain look up service register itself with other domains lookup service to create hierarchy but there is not such heirarchy in place by this date. Hence we are starting implementation without query Lookup service for metadata i.e. Static Mapping of service URL. In future, we will update the API to query Lookup service for metadata instead of Service directly.

Details

If we consider the first use case and wants to fetch all the Interfaces that is supported by the service e.g. EsNET. To get the required information we need the following information to query:

1. The MA Service URL or Lookup Service URL which will give us the MA Service URL
2. Event type supported by the MA which in most cases for utilization is eighter "utilization" for old version and "http://ggf.org/ns/nmwg/characteristic /utilization/2.0" for new version.

The sample request message for all the supported interfaces is shown below:

```
<nmwg:message id="msg4" type="MetadataKeyRequest"
xmlns:netutil="http://ggf.org/ns/nmwg/characteristic/utilization/2.0/"
            xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/"
            xmlns:nmwgt="http://ggf.org/ns/nmwg/topology/2.0/"
            xmlns:select="http://ggf.org/ns/nmwg/ops/select/2.0/">
  <nmwg:metadata id="meta1">
    <netutil:subject id="iusub1">
      <nmwgt:interface>
      </nmwgt:interface>
    </netutil:subject>
<nmwg:eventType>
http://ggf.org/ns/nmwg/characteristic/utilization/2.0
</nmwg:eventType>
  </nmwg:metadata>
<nmwg:data id="data1" metadataIdRef="meta1"/>
</nmwg:message>
```

The above request to Measurement Archive service will return all the interfaces either inbound or outbound with its respective metadata will be returned. To scall down the request e.g. if you are interested in only inbound interfaces. Then just add the following tab inside the <nmwgt:interface> element.

```
<nmwgt:interface><nmwgt:direction>in</nmwgt:direction>
</nmwgt:interface>
```

The response to the aboce request will look like this.

```
<?xml version="1.0" encoding="UTF-8"?>
<nmwg:message id="msg4_resp" messageIdRef="msg4"
  type="MetadataKeyResponse" xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/">
  <nmwg:metadata id="meta31">
    <netutil:subject id="subject31" xmlns:netutil="http://ggf.org/ns/nmwg/characteristic/utilization/2.0/">
      <nmwgt:interface xmlns:nmwgt="http://ggf.org/ns/nmwg/topology/2.0/">
        <nmwgt:hostName>rt1.bud.hu.geant2.net</nmwgt:hostName>
        <nmwgt:ifName>so-0/1/0</nmwgt:ifName>
        <nmwgt:ifDescription>Link to ISTF 1 (Pr:T-Systems Id:041/3002)</nmwgt:ifDescription>
        <nmwgt:ifAddress type="ipv4">62.40.124.109</nmwgt:ifAddress>
        <nmwgt:direction>in</nmwgt:direction>
        <nmwgt:authRealm>GEANT2</nmwgt:authRealm>
        <nmwgt:capacity>155520000</nmwgt:capacity>
      </nmwgt:interface>
    </netutil:subject>
    <nmwg:parameters id="localhost.localdomain.35530a35:111c3d9794a:1360">
      <nmwg:parameter name="supportedEventType">
http://ggf.org/ns/nmwg/characteristic/utilization/2.0
</nmwg:parameter>
    </nmwg:parameters>
  </nmwg:metadata>
  <nmwg:metadata id="meta5">
    <netutil:subject id="subject5" xmlns:netutil="http://ggf.org/ns/nmwg/characteristic/utilization/2.0/">
      <nmwgt:interface xmlns:nmwgt="http://ggf.org/ns/nmwg/topology/2.0/">
        <nmwgt:hostName>rt1.gen.ch.geant2.net</nmwgt:hostName>
        <nmwgt:ifName>ge-7/1/0</nmwgt:ifName>
        <nmwgt:ifDescription>Link to Switch (Pr:x Id:x)</nmwgt:ifDescription>
        <nmwgt:ifAddress type="ipv4">62.40.124.21</nmwgt:ifAddress>
        <nmwgt:direction>in</nmwgt:direction>
        <nmwgt:authRealm>GEANT2</nmwgt:authRealm>
        <nmwgt:capacity>10000000000</nmwgt:capacity>
      </nmwgt:interface>
    </netutil:subject>
    <nmwg:parameters id="localhost.localdomain.35530a35:111c3d9794a:12d1">
      <nmwg:parameter name="supportedEventType">
http://ggf.org/ns/nmwg/characteristic/utilization/2.0
</nmwg:parameter>
    </nmwg:parameters>
  </nmwg:metadata><nmwg:data id="data5" metadataIdRef="meta5">
    <nmwg:key id="localhost.localdomain.35530a35:111c3d9794a:153f">
      <nmwg:parameters id="localhost.localdomain.35530a35:111c3d9794a:1540">
        <nmwg:parameter name="dataSource">ds0</nmwg:parameter>
        <nmwg:parameter name="file">/home/taksometro/cricket-data/ge-1_0_0_0.rrd</nmwg:parameter>
        <nmwg:parameter name="eventType" value="http://ggf.org/ns/nmwg/characteristic/utilization/2.0"/>
        <nmwg:parameter name="valueUnits">Bps</nmwg:parameter>
      </nmwg:parameters>
    </nmwg:key>
  </nmwg:data>
  <nmwg:data id="data31" metadataIdRef="meta31">
    <nmwg:key id="localhost.localdomain.35530a35:111c3d9794a:14fd">
      <nmwg:parameters id="localhost.localdomain.35530a35:111c3d9794a:14fe">
        <nmwg:parameter name="dataSource">ds0</nmwg:parameter>
        <nmwg:parameter name="file">/home/taksometro/rtd</nmwg:parameter>
        <nmwg:parameter name="eventType" value="http://ggf.org/ns/nmwg/characteristic/utilization/2.0"/>
        <nmwg:parameter name="valueUnits">Bps</nmwg:parameter>
      </nmwg:parameters>
    </nmwg:key> </nmwg:message>
```

From metadata, you can get information about the interface e.g. its name, address, description, capacity etc. From data in the metadata key response, you get the key to query for detailed utilization of that specific interface and the unit of the returned data. In the aboce example the unit is shown as Bytes per second (Bps).

Parsing NWMG schema:

If you have already build the Measurement Archive, you will find two jar files in $PERFSONAR_HOME/build/ directory.

1. perfSONAR_generic.jar
2. perfSONAR_sqlma.jar or perfSONAR_rrdma.jar

If you see into the perfSONAR_generic.jar file all the elements and namespaces have its own corresponding Java Bean. E.g. ***http://ggf.org/ns/nmwg/base/Message***, you will find its Java Bean as ***org.ggf.ns.nmwg.base.Message***. You will find all the mapping of elements and namespaces in objects.conf file in the root src directory of perfSONAR i.e. $PERFSONAR_HOME/src/objects.conf

To parse the the message, There is a static method in perfSONAR_generic.jar i.e. **org.perfsonar.commons.util.XMLUtils.convertToMessage().** This method have two arguments, one is the org.w3c.dom.Document which consists of the response/request and the second argument is the file path to objects.conf as mentioned above. The method will return the **org.ggf.ns.nmwg.base.Message** object. From this class you can get all the information what you are required either for request or response.

The example below will fetch the capacity of an interface in response.

```
org.ggf.ns.nmwg.base.Message message =
XMLUtils.convertToMessage(metadata_response_dom, "path/to/objects.conf");

org.ggf.ns.nmwg.base.Metadata metadata =  message.getMetadataArray()[0];
  // Lets assume if we want to show the capacity of
  // the first metadata in response
org.ggf.ns.nmwg.topology.v2_0.Interface iface =
  metadata.getSubject.getInterface();
System.out.println (iface.getCapacityElement().getCapacity);
  // it will return the capacity in String
```