IS-WG
Information Services Working Group
http://spaces.internet2.edu/display/ISWG/Home

Aaron Brown, Internet2
Andy Lake, Internet2
Evangelos Chaniotakis, ESnet
Jason Zurawski, Internet2
Jeff Boote, Internet2
Marcos Portnoi, UDel
Martin Swany, UDel
July 5, 2009

Unified Network Information Services for Network Performance Monitoring

Status of this Document

This document provides information to the Grid community regarding the use cases, requirements, and functionalities of a Unified Network Information Services Plane for Network Performance Monitoring architecture.

Copyright Notice

Contents

1 Document History

Name	Date	Comments	Version
Marcos Portnoi	July 05, 2009	First draft.	1.0

2 Introduction

The Internet2 community and its counterparts around the world have been actively engaged in developing new network services, including dynamic circuit networking capabilities and performance tools. Both dynamic circuit allocation using the Internet2 DCN (Dynamic Circuit Network) and network performance tools like perfSONAR use an "Information Services plane" that allows users to discover network topology and the location and capabilities of network services within that topology. As global federation of network services occurs, the standardization and flexibility of the network-centric Information Services becomes even more critical.

In order to help catalyze and focus the development of these common information services, the Internet2 Network Advisory Committee (NTAC) has commissioned the creation of a new Information Services Working Group (IS-WG).

The group will work to further define the role and functionality of Information Services as well as drive design and development. Since these services will require specialized communications protocols, the group will work with and contribute to standardization bodies such as the OGF, GLIF, and the IETF. Discussions regarding the operation and exchange of information in organization federations will also be an important consideration.

3 Use Cases

3.1 Control Plane Use Cases

This section portrays cases specific to the world of control plane functionality. Note that some of these cases can easily be generalized to other domains (e.g., performance monitoring, distributed computing), but are presented through this community.

3.1.1 Internet2 DCN (Dynamic Circuit Network) - Edge mapping

Dynamic circuit networks implementing the Interdomain Controller (IDC) protocol require requesters of circuits to specify the edge links of a dynamic connection. The IDC protocol uses the NMWG topology schema to describe those links as Network Uniform Resource Names (NURNs). A NURN describes a hierarchy of domain, node, port, and link that is well suited for machine processing, but can result in rather long identifiers. In practice, requesters that need to manually enter endpoints (i.e., via a form on a web page) found these identifiers hard to remember and prone to typos. Similar to how DNS maps human-readable names to IP addresses; the Information Service can alleviate this issue by mapping human-readable names to NURNs. The figure below shows an example of a dynamic connection between two hosts both with human-readable identifiers registered with the Information Service:

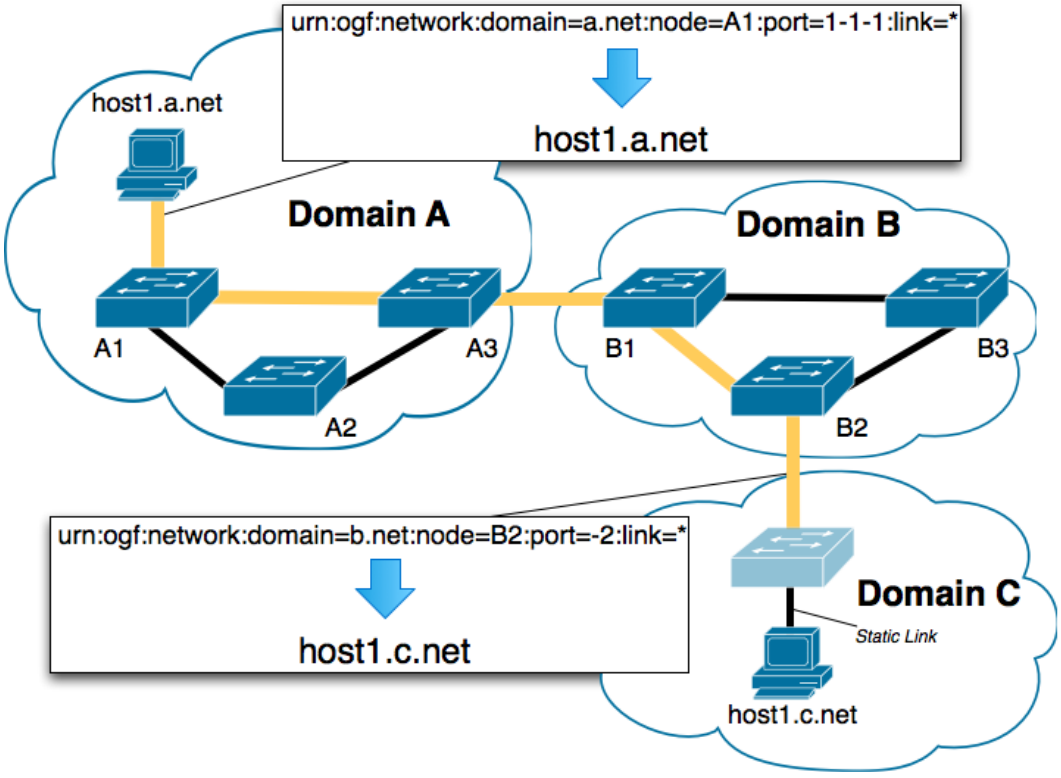

Figure 1: Dynamic connection between two hosts.

The diagram demonstrates a few important points of this use case. The thick orange line represents the dynamic portion of the connection. Some hosts are directly connected to a node that can be dynamically provisioned such as the host in Domain A. In this case we can generate the human-readable name *host1.a.net* that maps to an NURN identifying the link between the host and the dynamically configured node. It should be noted that the name *host1.a.net* looks like a DNS name but it may or may not be registered in DNS. DNS-style names are used because they are common on the web and

therefore familiar to users. Also, DNS-style names provide a concise structure that can be summarized for global discovery by a distributed set of Information Services.

The host in Domain C represents a more complicated yet common case in current dynamic circuit networks. Domain B is capable of dynamic configuration, but the node in Domain C is not; therefore, the host in Domain C is directly connected to node C1 with a static connection. The last dynamic link is that between Domain B and Domain C (as shown by the orange line). When we create a name like *host1.c.net*, we do not want to map it to the static link, since the dynamic circuit software cannot provision that link. Instead, we want to map it to the last dynamic connection- the link between domain B and C. A side-effect of this approach is that if domain C ever becomes dynamically configurable the mapping of the name can be changed to the link between the host and node on domain C transparently to the requester. The requester will continue to use *host1.c.net* to provision the circuit but it will automatically map to the new link.

The cases above work well currently but a few questions still remain such as:

- Can user-friendly names be used with non-edge link to aid with processes such as edge discovery?
- Names and their NURN mappings are currently kept in the Lookup Service and information about the link that the NURN identifies is in the Topology Service. Would it be beneficial to have both of these pieces of information in an integrated Information Service?

3.1.1.1 Specification

Use Case Name	Edge Mapping
Actors	Network devices that receive human-readable names as data.
Description	Maps human-readable names to NURNs.
Preconditions	Have, as input, a human-readable name instead of a NURN.
Post Conditions	Correctly translated human-readable name into a NURN.
Priority	
Difficulty	
Triggers	UNIS modules requesting a translation.
Frequency of Use	
Normal Course Steps	1. A module receives one or more human-readable names as part of their input. 2. The module builds a message to the Edge Mapping service containing the names and request for mapping. 3. The Edge Mapping service returns a message with the mappings or error if non-existent.
Alternative Course Steps	
Exceptions	Non-existent mapping.

Includes	
Extends	
Assumptions	
Notes and Issues	1. Can user-friendly names be used with non-edge link to aid with processes such as edge discovery? 2. Names and their NURN mappings are currently kept in the Lookup Service and information about the link that the NURN identifies is in the Topology Service. Would it be beneficial to have both of these pieces of information in an integrated Information Service?

3.1.2 Internet2 DCN – IDC and Notification Broker Service Discovery

The Interdomain Controller (IDC) protocol requires that messages be passed between services in different domains. The OSCARS IDC implementation requires a domain to interact with two services: it must forward messages to another Interdomain Controller and it must subscribe to notifications from a Notification Broker. The IDC sends these messages to a URL and the IDC must be able to discover this URL. The figure below shows an IDC in Domain A that needs to interact with services in Domain B but does not know their locations:

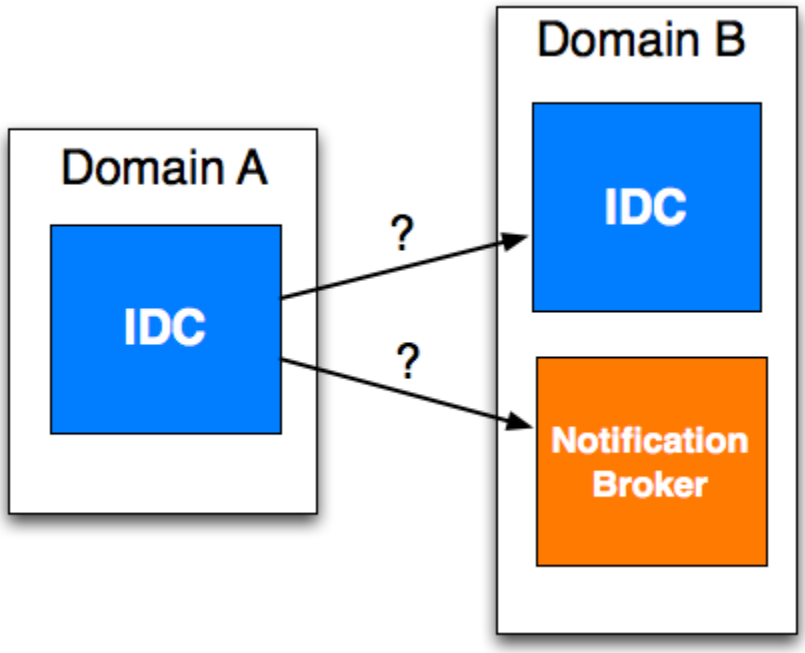

Figure 2: IDC in domain A wishes to interact with IDC and Notification Broker in domain B, but does not know their network location.

The current discovery options are as follows:

- Manually enter the URL of each neighboring IDCs and Notification Broker.
- Each service registers with the Lookup Service and neighbors automatically discover the URL of the registered services.

The second case is clearly more desirable as it is easier to add new neighbors, discover changes to a URL if a service moves, and avoid human error. In practice it has eliminated some of the more confusing configuration steps required when deploying an IDC.

The Lookup Service not only allows a domain to ask questions like "Where is my neighbor's IDC service?" but can also provide other useful information about the neighbor's service(s). For example, as the IDC protocol continues to change and interact with other projects it becomes increasingly important to determine **what protocol** (or version of the IDC protocol) another domain accepts. Current IDC implementations do not check the protocol but they likely will in the very near future as all the facilities to do so exist in the current Lookup Service. For example, if an IDC discovers another domain speaks an older version of a protocol, it may send a different message than it would otherwise. Information like this will continue to be important as the project grows.

3.1.2.1 Specification

Use Case Name	IDC and Notification Broker Service Discovery
Actors	IDCs; Notification Brokers; Users.
Description	IDCs interact with other IDCs and Notification Brokers. In order to do this, the IDC must discover the destination IDC's and Notification Broker's URL. This service provides the IDCs' and Notification Brokers' URLs for queries, and also provides registering capabilities for both.
Preconditions	1. Service's database must have valid IDC's and Notification Broker's URLs for the case of queries.
Post Conditions	1. For queries: the service returns locations for queried IDCs and Notification Brokers, or error if not existent. 2. For registration: service integrates the IDC's or Notification Broker's URL into the database.
Priority	
Difficulty	
Triggers	1. IDCs and Notification Brokers must register their location to the service. 2. IDC desires to find location of another IDC and/or Notification Broker.
Frequency of Use	
Normal Course Steps	1. For query: a. IDC constructs query message to service, requesting location of another IDC and/or Notification Broker. b. Service responds with message containing locations or error, if not found. 2. For registration: a. User constructs proper command to access service database and passes information about IDC and Notification Broker locations.

	b. Service integrates the information into its database.
Alternative Course Steps	
Exceptions	Location of IDC or Notification Broker not existent in service database.
Includes	
Extends	
Assumptions	
Notes and Issues	This service can be implemented and handled by the Lookup Service.

3.2 Performance Monitoring Use Cases

These are cases specific to the world of performance monitoring. Note that some cases here can easily be generalized to other domains (e.g., control plane, distributed computing) but are presented through this community.

3.2.1 perfSONAR - Measurement Archive Queries

perfSONAR Measurement Archives (MAs) are services designed to store performance measurement data over time. Typical MA design allows a single type (or several related types) of measurement information to be stored in the same facility, usually dictated by back-end storage. Some examples:

1. RRD MA: Back-end storage consists of Round Robin Databases [5] which normally store network data collected from SNMP. Data types include utilization, discards, errors and other related types.
2. SQL MA: Back-end storage consists of SQL typed database (e.g., MySQL [6], PostgreSQL [7], SQLite [8]) which can store almost any type of measurement data. Structure is limited by table complexity.
3. perfSONAR-BUOY MA: Back-end is a MySQL database and measurement data can be OWAMP [3] or BWCTL [2] data.

The query interface of all current perfSONAR services is highly dependent on the XML representation as dictated by the standards discussed in the OGF's [4] NM-WG [9] . For example an RRD MA stores information in different RRD files for each interface of a network device, e.g., for network device **somerouter** on the **Internet2** network there will be many interfaces. Gigabit Ethernet port 3/2 would be stored in file:

```
/data/somerouter.internet2.edu--ge-3_2_0.rrd
```

The MA service needs a way to tie the description of this specific interface and device to this file. The well formed XML would be similar to:

```
<nmwg:metadata xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/" id="m-in-
netutil-1">
    <netutil:subject
xmlns:netutil="http://ggf.org/ns/nmwg/characteristic/utilization/2.0/" id="s-
in-netutil-1">
      <nmwgt:interface xmlns:nmwgt="http://ggf.org/ns/nmwg/topology/2.0/">
        <nmwgt:ifAddress type="ipv4">10.10.10.1</nmwgt:ifAddress>
        <nmwgt:hostName>somerouter.internet2.edu</nmwgt:hostName>
        <nmwgt:ifName>ge-3/2/0</nmwgt:ifName>
        <nmwgt:ifIndex>2</nmwgt:ifIndex>
        <nmwgt:direction>in</nmwgt:direction>
        <nmwgt:capacity>1000000000</nmwgt:capacity>
      </nmwgt:interface>
    </netutil:subject>

<nmwg:eventType>http://ggf.org/ns/nmwg/characteristic/utilization/2.0</nmwg:e
ventType>
    <nmwg:eventType>http://ggf.org/ns/nmwg/tools/snmp/2.0</nmwg:eventType>
  </nmwg:metadata>

  <nmwg:data xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/" id="d-in-netutil-
1" metadataIdRef="m-in-netutil-1">
    <nmwg:key id="k-in-netutil-1">
      <nmwg:parameters id="pk-in-netutil-1">
        <nmwg:parameter
name="eventType">http://ggf.org/ns/nmwg/tools/snmp/2.0</nmwg:parameter>
        <nmwg:parameter
name="eventType">http://ggf.org/ns/nmwg/characteristic/utilization/2.0</nmwg:
parameter>
        <nmwg:parameter name="type">rrd</nmwg:parameter>
        <nmwg:parameter name="file">/data/somerouter.internet2.edu--ge-
3_2_0.rrd</nmwg:parameter>
        <nmwg:parameter name="valueUnits">Bps</nmwg:parameter>
        <nmwg:parameter name="dataSource">ifinoctets</nmwg:parameter>
      </nmwg:parameters>
    </nmwg:key>
  </nmwg:data>
```

The ***metadata*** portion contains the higher level description, namely how we could identify this specific network functionality from a very high level view. It is a simple interface on some network device and this specific description only cares about interface utilization (e.g., normally collected via SNMP and reading the ifInOctets/ifOutOctets counters). The ***data*** portion is more concerned in describing how we may reach the archived measurement observations. In this case, the RRD file has a name, and we are able to describe some of the internal portions (e.g., the ***datasource*** name, and the units we are measuring).

This format takes advantage of some of the structure of the XML to provide the semantics on how to link the information together: namely through the use of **ids** and **idReferences**. In the example above, the metadata (e.g., ***m-in-netutil-1***) is linked to the data (e.g., ***d-in-netutil-1***) through the use of these references:

```
<nmwg:metadata xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/" id="m-in-
netutil-1" />

<nmwg:data xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/" id="d-in-netutil-1"
metadataIdRef="m-in-netutil-1" />
```

All perfSONAR MA services feature a simple XML based query system to locate the stored information. Request messages (also constructed in XML) are used to locate information for a given service. To build upon the previous example, an interested client application may want to know about all interface utilization on **somerouter2.internet2.edu** and may think that a certain MA has this information. A request will be sent similar to this:

```
<?xml version="1.0" encoding="UTF-8"?>
<nmwg:message type="SetupDataRequest" id="setupDataRequest1"

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:snmp="http://ggf.org/ns/nmwg/tools/snmp/2.0/">

  <nmwg:metadata id="meta1" xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/">
    <netutil:subject
xmlns:netutil="http://ggf.org/ns/nmwg/characteristic/utilization/2.0/" id="s-
in-netutil-1">
      <nmwgt:interface xmlns:nmwgt="http://ggf.org/ns/nmwg/topology/2.0/">
        <nmwgt:hostName>somerouter2.internet2.edu</nmwgt:hostName>
      </nmwgt:interface>
    </netutil:subject>

<nmwg:eventType>http://ggf.org/ns/nmwg/characteristic/utilization/2.0</nmwg:e
ventType>
  </nmwg:metadata>

  <nmwg:data id="data1" metadataIdRef="meta1"
xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/"/>

</nmwg:message>
```

The service will perform simple matching against the internal storage to see if there is anything matching this request. Missing fields are assumed to be ***wild cards*** in this system. There are several advantages to this query interface:

1. Easy to construct APIs that are capable of constructing the proper message format
2. Flexible: allows arbitrarily complex queries based on several field names
3. Simple: requests and subsequent responses from the service are easy to interpret for results

Drawbacks also exist, many that limit the usefulness of perfSONAR MA services:

1. Must be aware of internal storage formats: requires knowledge of XML schemata

2. Each MA will use a different schema; APIs help but this requires knowing many XML dialects
3. Cannot construct queries based on higher level needs, e.g., query information on domain ***internet2.edu*** or IP range ***192.168.0.0/16***

3.2.1.1 Specification

Use Case Name	Measurement Archive Query
Actors	Users; MPs.
Description	Stores performance and measurement information into database, and provides access to this database for other services and users.
Preconditions	1. Location of MA is known or obtainable from Lookup Service. 2. Performance data must be existent into the MA database for a successful query.
Post Conditions	Storage of performance data, and retrieval of performance data into well formed XML messages.
Priority	
Difficulty	
Triggers	1. Measurement tool in MP desires to store performance data. 2. User or other service needs to access previously stored performance data.
Frequency of Use	
Normal Course Steps	1. For storage: a. Location of desired MA is obtained from Lookup Service, if not previously known. b. Measurement tool in MP constructs proper XML message containing the performance data and submits it to MA. c. MA responds success, or not, of operation. 2. For query: a. Location of desired MA is obtained from Lookup Service, if not previously known. b. Service or user constructs proper XML message requesting the needed data and queries the MA. c. MA responds with message data, if existent; error, otherwise (or activate alternative course).
Alternative Course Steps	1. If the MA does not have the desired performance information, it then activates the competent MPs to obtain the measurement. a. Select the MPs from the actor's original message, or request them from the Closest MP Service.

	b. Build message for each MP and request measurement. c. Obtain measurement from MPs and store them into database. d. Respond with performance data to the actor.
Exceptions	Data may not be available in desired MA for queries; database may fail to store information (e.g., space overflow).
Includes	
Extends	
Assumptions	
Notes and Issues	1. Missing fields are assumed to be ***wild cards*** in this system. 2. Must be aware of internal storage formats: requires knowledge of XML schemata. 3. Each MA will use a different schema; APIs help but this requires knowing many XML dialects. 4. Cannot construct queries based on higher level needs, e.g., query information on domain ***internet2.edu*** or IP range ***192.168.0.0/16.***

3.2.2 perfSONAR - Lookup Service Queries

The perfSONAR Lookup Service comes in two flavors:

- **Home Lookup Service**
- **Global Lookup Service**

A brief overview of each follows:

3.2.2.1 Home Lookup Service

The Home Lookup service (or ***hLS***) is meant to serve as a *lightning rod* to attract complex queries away from individual MA and MP services. For instance, if a client is looking for measurement data in the **internet2.edu** domain there may be 4 MAs to choose from. How does this client know which to search? There are two ways to achieve the goal:

1. Search **all** services. This will consume lots of resources.
2. Search the **correct** service. This is more efficient, but knowing which service is correct also takes some domain knowledge.

The hLS accepts *registrations* from all services offering perfSONAR data: simply put this is a copy of the metadata storage that each service maintains. This replication of information enables the hLS to know the availability of all data for a specific domain. Each hLS is able to combine the information from these registrations into a single ***summary***. This process tries to extract some important bits of information from each XML registration for a given service:

1. Data Types - These are pulled via the eventType information.
2. Domains - pulled from host specification.
3. IP Addresses - pulled from address specification (v4 and v6). We aim to perform CIDR summaries on this when all are collected so we can advertise *IP ranges*. E.g., if we have ***192.168.0.1***, ***192.168.0.2***, and ***192.168.0.3***, we can advertise having information on ***192.168.0.0/30.***
4. Keywords - User defined *tags* that describe a dataset. For instance, if the data was collected for a project such as ***USATLAS*** or on a network such as ***ESnet***.

The query interface to the hLS is similar to that of the perfSONAR MA and MP interface. At the core it is an XML message but there are two distinct kinds of queries:

1. ***Classic Query***: Relies on knowledge of the internal storage structure of the hLS which is really just an XML database. This query allows client applications to pass raw XPath and XQuery statements
2. ***Discovery Query***: Utilizes the summarization process described above to search for specific elements that may be present in the database.

The original query message has a format similar to this example:

```
<?xml version='1.0' encoding='UTF-8'?>
<nmwg:message type="LSQueryRequest"
              id="msg1"
              xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/"

xmlns:xquery="http://ggf.org/ns/nmwg/tools/org/perfsonar/service/lookup/xquer
y/1.0/">

  <nmwg:metadata id="meta1">

    <xquery:subject id="sub1">

  declare namespace nmwg="http://ggf.org/ns/nmwg/base/2.0/";
  /nmwg:store[@type="LSStore"]/nmwg:metadata

    </xquery:subject>

<nmwg:eventType>http://ggf.org/ns/nmwg/tools/org/perfsonar/service/lookup/xqu
ery/1.0</nmwg:eventType>

    <xquery:parameters id="param.1">
      <nmwg:parameter name="lsOutput">native</nmwg:parameter>
    </xquery:parameters>

  </nmwg:metadata>

  <nmwg:data metadataIdRef="meta1" id="d1"/>

</nmwg:message>
```

The goal of this message is to return all metadata elements in the XML database where the ***nmwg:store*** element has a type equal to ***LSStore***. As noted earlier, this is a property of the actual storage. Note that the contents of the ***subject*** may contain any valid XQuery or XPath statement. There are several advantages to this query interface:

1. Easy to construct APIs that are capable of constructing the proper message format.
2. Flexible: allows arbitrarily complex queries based on XQuery and XPath

Drawbacks also exist:

1. Must be aware of internal storage formats: requires knowledge of XML schemata and the organization of the database
2. Due to arbitrary nature, APIs cannot offer specific functions but can simple expose the XPath/XQuery to be sent
3. Cannot construct queries based on higher level needs, e.g., query information on domain ***internet2.edu*** or IP range ***192.168.0.0/16***

To address some of the drawbacks, the Discovery protocol was designed. The Discovery message has a specific format as seen here:

```
<?xml version='1.0' encoding='UTF-8'?>
<nmwg:message type="LSQueryRequest"
              id="LSDiscoveryRequest"

xmlns:perfsonar="http://ggf.org/ns/nmwg/tools/org/perfsonar/1.0/"
              xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/"

xmlns:psservice="http://ggf.org/ns/nmwg/tools/org/perfsonar/service/1.0/"
              xmlns:nmwgt="http://ggf.org/ns/nmwg/topology/2.0/"

xmlns:summary="http://ggf.org/ns/nmwg/tools/org/perfsonar/service/lookup/summ
arization/2.0/"

xmlns:nmtb="http://ogf.org/schema/network/topology/base/20070828/"

xmlns:nmtl3="http://ogf.org/schema/network/topology/l3/20070828/">

  <nmwg:metadata id="meta1">
      <summary:subject
xmlns:summary="http://ggf.org/ns/nmwg/tools/org/perfsonar/service/lookup/summ
arization/2.0/" id="subject.1">

<!-- can have multiples of each, note that this creates an 'or' relationship
-->
        <nmtb:address
xmlns:nmtb="http://ogf.org/schema/network/topology/base/20070828/"
type="ipv4">128.4.133.167</nmtb:address>
        <nmtb:address
xmlns:nmtb="http://ogf.org/schema/network/topology/base/20070828/"type="ipv4"
>128.4.100.45</nmtb:address>
```

```
        <nmtb:domain
xmlns:nmtb="http://ogf.org/schema/network/topology/base/20070828/">
          <nmtb:name type="dns">edu</nmtb:name>
        </nmtb:domain>

        <nmtb:domain
xmlns:nmtb="http://ogf.org/schema/network/topology/base/20070828/">
          <nmtb:name type="dns">udel.edu</nmtb:name>
        </nmtb:domain>


<nmwg:eventType>http://ggf.org/ns/nmwg/characteristic/utilization/2.0</nmwg:e
ventType>

<nmwg:eventType>http://ggf.org/ns/nmwg/characteristic/errors/2.0</nmwg:eventT
ype>

        <summary:parameters>
          <nmwg:parameter name="keyword">project:Geant2</nmwg:parameter>
        </summary:parameters>

<!-- the combination of all things is an 'and' relationship, this entire
subject is therefore:

('128.4.133.167' or '128.4.100.45')  and
('udel.edu' or 'edu') and
('http://ggf.org/ns/nmwg/characteristic/utilization/2.0' or
'http://ggf.org/ns/nmwg/characteristic/errors/2.0') and
('project:Geant2')

-->

      </summary:subject>

    <!-- need this... -->

<nmwg:eventType>http://ogf.org/ns/nmwg/tools/org/perfsonar/service/lookup/dis
covery/summary/2.0</nmwg:eventType>


  </nmwg:metadata>

  <nmwg:data metadataIdRef="meta1" id="d1"/>
</nmwg:message>
```

In general these messages offer a greater expressiveness than is available through XQuery. These two query methods serve an important purpose for the LS infrastructure as well as perfSONAR as whole.

3.2.2.2 Specification

Use Case Name	Home Lookup Service Query
Actors	Other services (MAs, MPs, gLS, etc.).

Description	Serves as a directory service where services register themselves (and their capabilities), to facilitate the discovery of these services by other services. I.e., services need not know the location and/or capabilities of all services they need to contact; they only need to query the proper hLS for the information. Interacts with upper level gLS to pass summarized information about its own database.
Preconditions	1. Services must register their capabilities and controlled data to the hLS for the queries. 2. The hLS must know the location of at least one gLS.
Post Conditions	1. Registration of a service and its controlled data. 2. Retrieval of the location of a desired service and its data types. 3. Synchronization with upper level gLS instances.
Priority	
Difficulty	
Triggers	1. A service wants to register its existence and data types to a hLS. 2. A service wants to find specific performance data, and desires to know the location of this information or tool to query directly.
Frequency of Use	
Normal Course Steps	1. For queries from other services: a. Services build proper XML messages requesting the desired information and submit to the hLS. b. The hLS consults its database and determines whether it knows about the desired information (i.e., if the responsible service has already registered with the hLS). c. If the hLS knows about the information, it constructs an XML response with the proper destination service location; error otherwise. 2. For service registration: a. Services build proper XML messages containing the data types they control and submit this to the hLS. b. The hLS populates its database accordingly. 3. For gLS synchronization: a. The hLS builds summaries of its database with relevant data. b. The hLS constructs XML messages containing the summaries, and submit them to the gLS (may be more than one). c. Repeat this process periodically. 4. For direct database access with XPath and XQuery messages: a. Receive the message from the service.

	b. Performs the database operation. c. Returns a response message, if applicable.
Alternative Course Steps	1. For direct database access with XPath and XQuery messages: a. Receive the message from the service. b. Performs the database operation. c. Returns a response message, if applicable.
Exceptions	Inexistence of the desired service/information in the hLS database for queries. Failure to communicate with upper level gLS instance.
Includes	
Extends	
Assumptions	The hLS may advertise its controlled information to one or more gLS instances.
Notes and Issues	1. Must be aware of internal storage formats: requires knowledge of XML schemata and the organization of the database. 2. Due to arbitrary nature, APIs cannot offer specific functions but can simple expose the XPath/XQuery to be sent. 3. Cannot construct queries based on higher level needs, e.g., query information on domain ***internet2.edu*** or IP range ***192.168.0.0/16.*** 4. Must rely on effective IP summarization heuristic to optimize communication with gLS and gLS storage. To address some of the drawbacks, the Discovery protocol was designed.

3.2.2.3 Global Lookup Service

The Global Lookup service (or ***gLS***) acts as a globally accessible directory of Information Services. The design and internal workings of this service are exactly the same as the Home Lookup Service (***hLS***) with a single exception: MA and MP services register with an hLS and in turn the hLS will register with the gLS. The hLS registration involves taking the summarized data (discussed previously) and sending this off to one or many gLS instances. The gLS instances are then capable of sharing the hLS information amongst themselves to offer complete information coverage.

The two aforementioned query types are available in this service as well, offering a complete solution to information location at the local and global levels.

3.2.2.4 Specification

Use Case Name	Global Lookup Service Query
Actors	hLS and other services.
Description	Acts as a global directory service, containing a summary of existent hLS instances and the respective controlled performance data types. Synchronize

	periodically with hLS to keep this summary current, and responds to queries from other services wishing to locate proper hLS instances for registration or service discovery. The gLS also synchronize its information with other gLS instances.
Preconditions	hLS must synchronize with gLS periodically with summary of registered services and data types. Other gLS instances locations must be known, if gLS synchronization is desired.
Post Conditions	gLS responds with proper hLS in case of service discovery, or updates its summary database in case of hLS synchronization.
Priority	
Difficulty	
Triggers	1. A hLS wishes to synchronize its data with the gLS. 2. A service wants to find the hLS that controls specific performance data.
Frequency of Use	
Normal Course Steps	1. For queries from other services: a. Services build proper XML messages requesting the desired information and submit to the gLS. b. The gLS consults its database and determines whether it knows about the hLS that controls the desired information. c. If the gLS finds the hLS, it constructs an XML response with the proper destination hLS location; error otherwise. 2. For hLS synchronization: a. The hLS builds summaries of its database with relevant data. b. The hLS constructs XML messages containing the summaries, and submit them to the gLS (may be more than one). c. Repeat this process periodically. 3. For gLS synchronization: a. The gLS forms proper XML messages containing data into its database and advertises it to other gLS instances. b. Other gLS instances receive the message and update their database. c. Conversely, one gLS instance may receive a message from another gLS instance containing synchronization data; it receives this message and updates its database.
Alternative Course Steps	1. For direct database access with XPath and XQuery messages: a. Receive the message from the service. b. Performs the database operation. c. Returns a response message, if applicable.

Exceptions	hLS that should control the queried information is not found in gLS database.
Includes	
Extends	
Assumptions	
Notes and Issues	The hLSs must rely on effective IP summarization heuristic to optimize communication with gLS and gLS storage.

3.2.3 perfSONAR - Circuit Monitoring Queries

TBD

3.2.3.1 Specification

Use Case Name	Circuit Monitoring Query
Actors	
Description	
Preconditions	
Post Conditions	
Priority	
Difficulty	
Triggers	
Frequency of Use	
Normal Course Steps	
Alternative Course Steps	
Exceptions	
Includes	
Extends	
Assumptions	
Notes and Issues	

3.2.4 perfSONAR - Finding Closest MP (Measurement Point)

In a network measurement infrastructure such as perfSONAR, Measurement Points (MPs) are the devices responsible for running the proper tools and performing the collection of measurement data. When a certain measurement needs to be executed between two end points, one or more MPs

(depending on the desired kind of measurement) are activated in order to conduct the mensuration. Depending on the network topology and end points involved, the MPs might (a) lie totally in the path between the end points, or (b) lie outside the path.

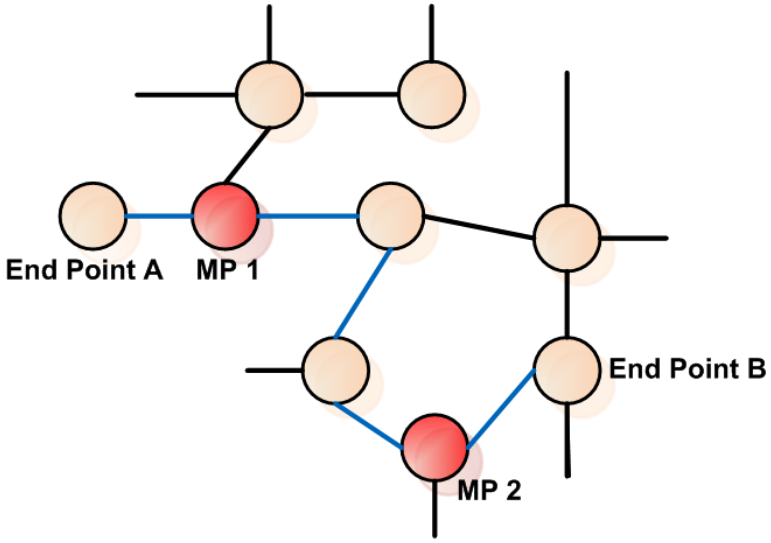

Figure 3: Example topology where MPs line within a path connecting end points.

Case (a) is depicted in Figure 3. The desired measurement between end points A and B is conducted by MPs 1 and 2, which lie within the same path (shown in blue) that connects the end points. In case (b), illustrated in Figure 4, one of the MPs (MP 1) is located outside the path that connects end points A and B. But there is a path that connects MP 1 to MP 2, and this path includes one or more segments of the path that connects end points A and B. This way, the mensuration can still be carried. (Notice, also, that there is more than one path connecting end points A and B.)

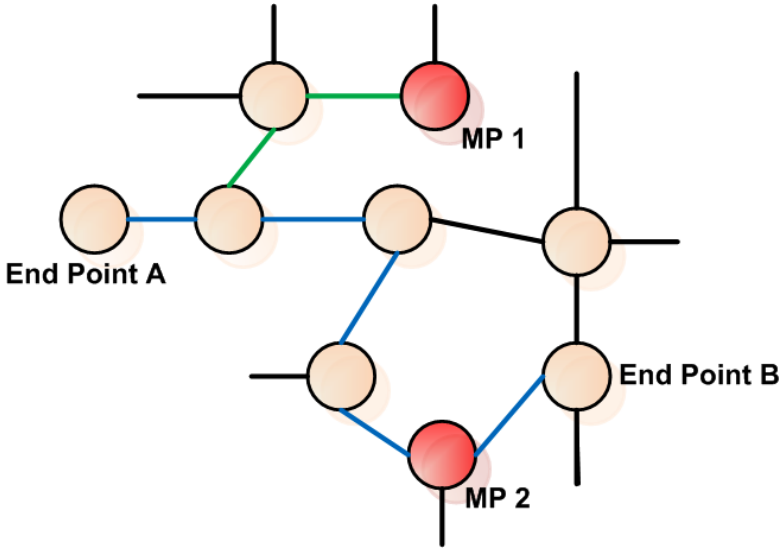

Figure 4: Example topology where one MP lies outside a path connecting end points.

Both cases (a) and (b) can pose their own problems to the measurement being performed. In (a), for example, the path between MPs 1 and 2 does not cover the entire path that connects end points A and B; the segments connecting end point A to MP 1, and end point B to MP 2 are left out of the

measurements. Thus, the influence of these segments, such as latency, bandwidth, or loss, will not be computed in the overall results. In case (b), on the other hand, not only those segments won't be included in the measurement, but extra segments (shown in green in Figure 4) that connect MP 1 to the path between end points A and B will now contribute with their own characteristics to the overall results. Since these extra segments do not belong to the original path between end points A and B, the measurements in case (b) might contain noise. Conversely, it can also be said that the measurements in case (a) will not contain all the expected ***noise***.

An additional case is shown in Figure 5. Here, there are three possible MPs that can conduct the measurements; MPs 2 and 3 lie directly within the path connecting end points A and B, whereas MP 1 lies outside the path. So a choice must be made typically between MPs 1 and 3, and MP 2 remains an obvious choice.

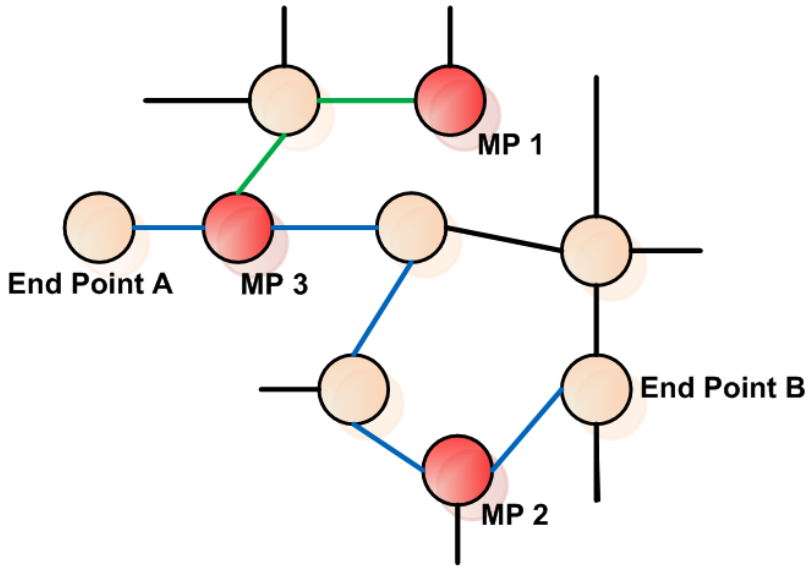

Figure 5: Example topology where several MPs are candidates for conducting a measurement.

In order to manage network measurements in an infrastructure such as perfSONAR, not only MPs are needed, but also their location in respect to the topology must be known, so as to allow choosing the most appropriate MPs in regard to the measurement. In general, the most appropriate MP is the one closest to the end point in the topology. (We can generally assume that the ***closest*** MP is the one with the least number of hops to a certain end point; this metric is flexible, however.)

The proposed service named Closest MP shall fulfill the need to optimize the measurement services performed by MPs, by facilitating the discovery of MPs and detecting their actual location within a topology. Simultaneously, by querying the metrics that can be obtained from each candidate MP (this query is made to the Lookup Service), this may result in a better choice of MPs which will be responsible for collecting the desired measurement.

3.2.4.1 Specification

Use Case Name	Find Closest MP
Actors	User; statistical tools; other performance tools in MPs; MAs.
Description	This service facilitates discovery of optimal located MPs relative to a specific desired measurement.

Preconditions	1. The network topology must be previously known or obtainable to the service. 2. MPs must exist, and that are able to perform the desired measurement.
Post Conditions	Return list of optimal MPs.
Priority	
Difficulty	
Triggers	Actor wishes to know MPs that are optimally located relative to a network measurement to be performed.
Frequency of Use	
Normal Course Steps	1. Actor builds message containing network locations and type of measurement, and submits it to the Closest MP service. 2. The service obtains from a gLS a list of available MPs in the network that are capable of conducting the desired measurement. 3. The service obtains topology information from the proper topology service. 4. The service combines the list of MPs and topology information, and derives (by algorithm) the list of optimal MPs relative to the received locations from actor. 5. Service builds proper XML message and returns with list of optimal MPs.
Alternative Course Steps	
Exceptions	There may be no available MPs capable of conducting the desired measurement.
Includes	
Extends	
Assumptions	There must be at least one MP in the network. The service will return MPs capable of conducting the measurement and that are “closer” to the target locations, even if this “distance” is in fact considerable.
Notes and Issues	One possible parameter is a “maximum distance”, for which the Closest MP service will discard MPs that are topologically too far from the measurement target (as specified by the actor). This might result, however, in no MPs being available.

3.2.5 OSCARS Software Suite - Current OSCARS/perfSONAR Interdomain Pathfinder

The current OSCARS release has two components that interact with the current perfSONAR Topology Service. The IDCs register their topologies with a Topology Service. The IDCs can then query these Topology Services to download the topologies and build an inter-domain graph. This inter-domain

graph can be used to lookup the next domain along the path instead of having static routes configured for each possible destination domain.

The current steps used are sketched out in the following figure.

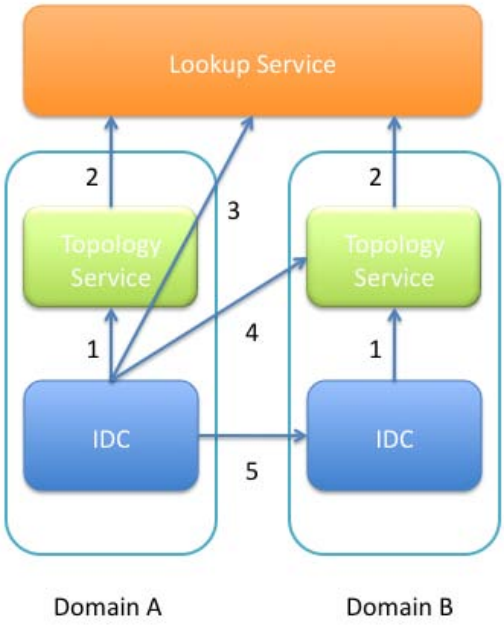

Figure 6: IDC in domain A finds a path to IDC in domain B.

The steps are:

1. The IDCs register their topologies with their respective Topology Services
2. The Topology Services register which domains they contain information about with the Lookup Service infrastructure.
3. When the IDC for domain A gets an incoming request, it begins to do path finding starting with its local topology. When it encounters a link into another domain, the IDC queries the Lookup Service infrastructure to find which Topology Service contains that domain's topology.
4. The IDC contacts that Topology Service and downloads the topology for that domain
5. The IDC continues doing the pathfinding until it finds the appropriate next domain, and connects to that domain.

This approach allows users to deploy their own Topology Service, or to re-use a centralized one. Current deployments are using a centralized Topology Service housed at Internet2, but there's nothing that prevents them from setting up their own.

3.2.5.1 Topology Abstraction

One feature of the current registration component is that it allows the IDC to register a full topology as well as an abstracted topology that does not include any of the internal links. This allows domains who would rather not divulge the structure of their networks to only register the external interfaces. The

pathfinding component, when it sees one of these abstracted domains, assumes that a path can be constructed between any of the external facing interfaces.

3.2.5.2 Performance Considerations

There are a couple features that improve the pathfinding performance. The IDC will grab a domain's entire topology in one go instead of downloading it piecemeal while the pathfinding is being performed. This speeds up the pathfinding at the expense of increased memory use on the local machine. Were the complexity of the topologies too high or were there too many domains to search, the memory use might become excessive, but for the near term, this optimization works well.

Another way the pathfinding component improves performance is by caching the topologies it downloads as well as the services from which it downloaded the topology. Since the pathfinder will likely often calculate paths through a similar set of domains, the pathfinder can shortcut some of the lookups above. It can reuse the cached topology, bypassing any need to contact other hosts, or reuse the saved Topology Service URL allowing it to bypass the Lookup Service queries.

3.2.5.3 Requested Features

There have been some features requested of the pathfinder. The most notable is that there is not a way to create an authoritative representation for a domain. As long as there are no bad actors, this should not pose much of a problem. Longer term, it would be useful to give users or IDCs some way to verify the data they receive from the Topology Service. How this would be done is an open question, but possible options would be to have domains sign their topologies, or to let domains define authoritative Topology Services for their topologies.

Another open question is whether there are better forms of topology abstraction that can be done. It might be useful to be able to register different levels of abstraction for different requesting parties. It might also be useful to register more complex abstract topologies, like one that has abstract internal links defining the possible paths that do exist through the network.

3.2.5.4 Specification

Use Case Name	Interdomain Pathfinder
Actors	IDC
Description	Allows finding of a path between domains using the topology information registered in Topology Services.
Preconditions	1. Topology information for each domain must be properly registered into local Topology Services. 2. Topology Services must be properly registered to Lookup Service.
Post Conditions	A complete path connecting two domains is constructed.
Priority	
Difficulty	
Triggers	Incoming query requesting contact with destination in another domain.
Frequency of Use	

Normal Course Steps	1. The IDCs register their topologies with their respective Topology Services. 2. The Topology Services register which domains they contain information about with the Lookup Service infrastructure. 3. When the IDC for domain A gets an incoming request, it begins to do path finding starting with its local topology. When it encounters a link into another domain, the IDC queries the Lookup Service infrastructure to find which Topology Service contains that domain's topology. 4. The IDC contacts that Topology Service and downloads the topology for that domain. 5. The IDC continues doing the pathfinding until it finds the appropriate next domain, and connects to that domain.
Alternative Course Steps	
Exceptions	Non-accessible domains will return an empty path.
Includes	
Extends	
Assumptions	
Notes and Issues	1. Complete topology is contained in memory. If topology complexity is too high, excessive memory utilization might happen. 2. There is no authorization information. 3. The service may implement different abstractions of the topology respective to the actor's needs or authorization.

3.3 Virtual Organizations - Application monitoring

Specific communities use the network to perform their necessities. This includes scientific groups such as LHC, LIGO, Climate, etc. For these groups the network is simply another resource they want to monitor in a similar way that they monitor their compute-clusters and storage-clusters.

3.3.1 Abstract Location Finder

The abstract locations refer to Pop's, Peering/Exchange points, compute clusters, storage clusters.

From an application's perspective, information such as throughput or latency information can largely be abstracted as a location to location perspective. They want to know how long it takes packets to get from the storage cluster to the compute cluster. They do not care so much about specific hosts in either cluster.

From a network perspective things are similar. In general, Internet2 does not really care which of the hosts within a particular POP was used to perform a throughput test with an end site, or with another POP. The interesting information is the performance from location to location.

3.3.1.1 Requested Features

The aspired ability is to abstract a location without respect to IP/netmask in an only semi-coordinated fashion. A client application should be able to pose abstract queries such as: latency from Internet2/CHIC to Internet2/LOSA. This does not have to be supported directly via the MA. This could easily be a two-stage query where all 'hosts' associated with a particular 'location' are first found, and then queries are made to find all data related to a particular 'location'. The issue here is that an 'abstract' location entity seems to be needed at some level.

3.3.1.2 Specification

Use Case Name	Abstract Location Finder
Actors	MPs; users.
Description	This service translates an abstract location, generally a user view of a location, into a network location. Namely, it translates a query such as "find latency between University of Delaware and ESnet" into "find latency between *specific IP address at University of Delaware* and *specific IP address at ESnet*".
Preconditions	A mapping of an abstract location and its candidate IP addresses must exist.
Post Conditions	Existing IP addresses.
Priority	
Difficulty	
Triggers	An actor wishes performance data regarding abstract locations.
Frequency of Use	
Normal Course Steps	1. Actor submits message containing abstract locations to the service. 2. Service queries its database and obtains a mapping for the abstract location and a network domain or sub-domain. 3. Service queries, if necessary, a Topology Service to obtain IP addresses respective to the domains or sub-domains. 4. Service elects the IP address that "best" represent each abstract location, based on heuristic, and returns the addresses in a properly formatted message.
Alternative Course Steps	
Exceptions	Abstract location is unknown.
Includes	
Extends	
Assumptions	
Notes and Issues	The abstract location mapping to a specific IP address is based on some heuristic that is able to produce the optimal representation of that abstract

	location as an address. This optimality is dependent on the network administrator point of view.

4 Notational Conventions

The keywords “MUST” “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” are to be interpreted as described in RFC2119 [1].

5 Intellectual Property Statement

The OGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claim of rights made available for publication and any assurances of licenses to be made available or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementer or users of this specification can be obtained from the OGF Secretariat. The OGF invites any interested party to bring to its attention any copyrights, patent or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the OGF Executive Director.

6 Disclaimer

This document and the information contained herein is provided on an “As Is” basis and the OGF disclaims all warranties, express or implied, including but not limited to any warranty that the use of the information herein will not infringe any rights or any implied warranties of merchantability or fitness for a particular purpose.

7 Full Copyright Notice

8 References

[1] Bradner, S. Key Words for Use in RFCs to Indicate Requirement Levels. RFC 2119, March 1997.

[2] http://e2epi.internet2.edu/bwctl/

[3] http://e2epi.internet2.edu/owamp/
[4] http://ogf.org/
[5] http://oss.oetiker.ch/rrdtool/
[6] http://www.mysql.com/
[7] http://www.postgresql.org/
[8] http://www.sqlite.org/
[9] https://forge.gridforum.org/projects/nm-wg