

Network Measurements Request Schema: INFORMAL Requirements Document

References

1. GGF NM-WG "Schemas for Exchanging Network Measurements with OGSF": http://www-didc.lbl.gov/NMWG/schemas/NM-WG_Schemas_for_OGSF.html
2. GGF NM-WG "A Hierarchy of Network Performance Characteristics for Grid Applications and Services": <http://www-didc.lbl.gov/NMWG/docs/draft-ggf-nmwg-hierarchy-01.pdf>
3. GGF DAMED "An analysis of "Top N" Event Descriptions": http://www-didc.lbl.gov/damed/documents/TopN_Events_final_draft.pdf
4. IETF "RFC 958: Network Time Protocol (NTP)": <http://www.ietf.org/rfc/rfc0958.txt?number=958>
5. Iperf: <http://dast.nlanr.net/Projects/Iperf/>
6. GGF Grid High Performance Networking Research Group (GHPN-RG): <http://forge.gridforum.org/projects/ghpn-rg/>

Background

At GGF9 (Chicago, 5-8th September 2003) a possible requirement was identified for a data/test request schema to complement the existing NM-WG data publishing schema [1]. An informal working group was subsequently formed, headed by Eric Boyd, to define the requirements for such a schema.

To provide a firmer base for further discussion the group went beyond its original remit, and in addition to this document of requirements, has produced a sample schema and examples of its possible use.

Initial group members are given in table 1.

Loukik Kudarimoti	Dante
Nicolas Simar	
Mark Leese	Daresbury Laboratory
Jeff Boote	Internet2
Eric Boyd	
Matt Zekauskas	
Paul Mealor	University College London

Table 1: Informal "request schema" group members

Since this time, many people have contributed to face-to-face and telephone meetings reviewing the document.

This revision includes accompanying business logic in green text - the overall aim of the request schema is for a NM-WG compliant requestor to be able to drive predictable behaviour from a compliant responder. This compels the requirements to either directly or indirectly define a set of rules of the form, such that:

"IF you receive X, THEN do Y"

This is business logic, in the same way that rules can be associated with a database to encode business policies (such as automatically generating a request for a reminder letter if a customer hasn't paid their gas bill).

Overall Requirement

The basic requirement is for an XML schema that can be used to generate messages to be sent to network monitoring systems, to make requests for historical data and the running of network monitoring tests (whether immediate or future). We envisage a test request and report mechanism as shown in figure 1.

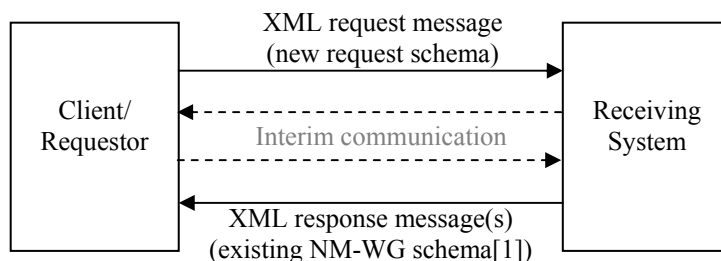


Figure 1: test request and report mechanism

Note: If the request is for the running of a test, it is expected that some interim communication may take place between the requestor and receiving system. For example, the receiving system may acknowledge a request with “Yes, I will run your test, and will return the results when they are available”. The request will therefore be the first part of any communication, and the NM-WG publication schema [1] response the last. These interim communications are, so far, undefined, and are considered beyond the scope of the request schema.

The schema should allow the specification of the following information for tests that they wish performed, or test data they wish to retrieve:

What: measurement/metric information
 Where: source & destination identifiers
 When: time information
 How: how to run tests/queries and return results

Detailed Requirements

General Requirements:

1. Where possible and appropriate, use should be made of equivalent schema element names and structures to those contained in the NMWG response schema [1].
2. It should be possible to specify a query (albeit a simple one) with a minimum amount of information. This would be a test start and end point, a time, and a test type (metric/measurement). **Note:** this is a definition of the minimum set of information required to request a test or historic data, not the minimum set of values to be contained in a message based on the schema. Default values could be defined for non-specified information.

Although obviously not a direct requirement of a request schema, it is nevertheless expected that a minimally specified request should result in a correspondingly minimal response, e.g. containing time information and a measurement value only.

What - Measurement/Metric Information:

1. The measurement to be made/metric data to provided (characteristic) should be specified using a dotted textual notation as described in the NM-WG hierarchy of characteristics document [2], and often referred to as a DAMED style name [3]. Care must be taken however, that characteristic names cannot cause contention with other areas of the schema. For example, a characteristic name previously used in the publication schema [1], *path.bandwidth.achievable.TCP*, could cause contention if the transmission protocol (TCP) could also be specified elsewhere in schema.
2. It should be possible to request more than one characteristic in each message based on the schema. In making this requirement we have tried to balance the need for efficiency in making requests against the danger of heavily loading request satisfying systems.

3. In addition to basic metrics, it should also be possible to request statistical data concerning a characteristic (e.g. min, max and average values), with or without the raw data up on which the summary is based.

As a continuation of point 2, it should be possible to request multiple statistical summaries in the same request. To find the minimum value of a statistic over a past period would require a system to look over all the data from that past period. If the requestor then wanted the maximum value, the process would have to be repeated. It is therefore sensible to give the responder the option of calculating multiple summaries concurrently.

Accompanying business logic: A request for a statistical summary can be rejected for any reason, without explanation. This may be desirable when, for example, calculating a statistical summary would require an unacceptable amount of processing for the responder. If you wish to give a reason, a list of suggested responses is:

- statistic not supported (responder can't do it)
- statistic disallowed (the responder will not provide the requested data e.g. it would involve too much processing)
- statistic unavailable (e.g. temporarily unavailable, because the system is out-of-order or currently has too much other work)

Particular responses could be communicated to the requestor using a set of pre-defined fault codes.

4. The schema should be extensible to allow requests to be made for statistical data that is currently unsupported by the NM-WG publication schema [1].

Accompanying business logic: If a request relates to a statistical summary which is unrecognised by the responder, then the request should be rejected with an appropriate reason, such as "not understood".

5. A means should be provided to specify a sample interval on which statistical analyses should be made. For example, if a user required the daily averages for one-way delay over a period of one month, they would request one-way delay results for that month, with a sample interval of one day.

The sample interval should default to the whole time period for which a request has been made, and use the same units as all other time related parameters, as discussed in a later section.

Accompanying business logic:

- If a requested sample interval is bigger than the corresponding time period, then the default of sample interval = time period should not be assumed. If this were allowed, users could assume any returned data to be what was requested. The request should instead be rejected, with explanation.
 - Further, if the requested sample interval is not a "good fit" within the time period (e.g. the time period is 30 minutes, and the sample interval 20) then the receiver should start its statistical calculations at the start of the time period and generate as many statistical summaries as possible. Any data following the last statistical summary should be ignored. So, continuing with the 30-20 minute example, one statistical summary would be provided based on the first 20 minutes of data, with the remaining 10 minutes of raw data being ignored.
6. To prevent the requestor being overloaded when a query results in the responder having "excessive" amounts of data to return, the schema should include a value which sets a limit on the number of results the responder can return.

Accompanying business logic: Requests where the maximum number of results is ≤ 0 or an illegal keyword should be rejected with an appropriate reason, such as "malformed request: ..."

In the event that there are more results to return than the defined maximum, the N values closest to target time should be returned (see "When" section for definition of target time). Further, by using the positive and negative time tolerance values (see "When") to control where target time lies in relation to the measurement period (start, middle or end), it will be possible to control from which interval of a measurement period the closest values are taken. This is perhaps best illustrated with an example. If a query was made for the time period 12:00 to 16:00, the target time and time

values can be set as shown in Table 2 to put target time at the beginning, middle or end of the period.

Take closest results from...			
	...start of period	...middle of period	...end of period
negative time tolerance	0	2 hours	4 hours
target time	12:00	14:00	16:00
positive time tolerance	4 hours	2 hours	0 hours

Table 2: setting the location of target time in relation to the query period

Allowed values are all positive real numbers, and the keyword “all”. When not specified, this value should default to one, which when combined with the defaults for time information (see the “When” section) result in a sensible default request of “give me your last result for characteristic X”.

Where - Source & Destination Information:

1. The method of specifying addresses for test source and destination points should be flexible, to allow IP addresses (both IPv4 and v6), hostnames, and even textual names to be used. As an example of the latter, a user may want to refer to a test source and destination as “core router” and “edge router”, thus hiding the IP addresses. This would however require the requestor and receiver to have prior knowledge of such names. **Note** that multiple addresses (whether referring to different devices or different interfaces on the same device) are being excluded to avoid making the schema unnecessarily complex. Support will only be added if, at a later stage, sufficient use cases are identified which require them.

Accompanying business logic: If a request is received with an unknown name the request should be rejected with an explanation (error message) such as “don't understand name xyz” or “don't know xyz”. Further, it would seem sensible to extend this to “unknown” hostnames.

2. The schema should reflect the fact that source address information (whether IP, hostname or textual name) should always be specified in requests.

Accompanying business logic: Response messages should contain exactly the same address information as their corresponding requests. If for example, a request is received containing host or textual names, then the response should contain those same names, and not a resolved IP address.

When - Time Information:

1. The schema should provide one primary means of specifying the time period for which tests should be run, or data should be retrieved. This should be achieved via:
 - a. a target time variable, which is either an absolute time, or takes the value “now”, and
 - b. negative and positive time tolerances, which are relative to the target time, with the negative time tolerance also able to adopt the value “infinity”

These values delimit a time period, running from (target time – negative time tolerance) to (target time + positive time tolerance). When combined with the variable restricting the number of results to be returned (see “What”, point 6), two basic queries against historic data are supported:

- i) With the maximum results set to “all”, the query is essentially to supply all matching measurements in the time period
- ii) With the maximum results set to one, the time tolerance values essentially define the period for which a measurement is considered to be acceptable. For example, if a user is only interested in a value if it is within 10 minutes of 14:00, target time can be set to a value representing 14:00, with the positive and negative time tolerances set to values representing 10 minutes. Preference is always given to “closeness” to target time, whether referring to matches against historic data, or the running of new tests.

In terms of requesting tests, these time values together with the testing interval described in point 5, define the period over which tests should be run.

Target time defaults to “now” if not specified, with negative time tolerance defaulting to “infinity” and the positive time tolerances to zero. When combined with the variable restricting the number of results that can be returned (see “What”, point 6), this provides a sensible default request of “give me your last result for characteristic X”.

One advantage of the positive time tolerance being relative is that when used with a target time of “now” it ensures that the query period will always be “positive time tolerance” long, no matter when the query is processed.

2. For ease of use, and as in the first draft of the requirements, it should also be possible for a time period to be specified using absolute start and end times, for which all matching results are returned. Start and end time should be implemented as pseudo variables however, and should be mapped at the receiver to the primary time type, as shown below:

```
target time           = start time
negative time tolerance = 0
positive time tolerance = end time - start time
maximum number of results = all
```

Start and end times must adopt the value “now”, or be absolute times. As with the target time combination, start and end time can be used to request the scheduling of tests, if the times are at all in the future.

An additional advantage of the start and end time combination is that, because end time can be specified as an absolute, it can be used to set a clear cut off time, after which no data is considered or tests run, no matter what the value of the start time.

3. Absolute times should only be specified as:
 - a. seconds since midnight on 1st January 1970 UTC, or
 - b. an appropriate pre-defined XML type for textual representation of dates and times, or
 - c. an NTP timestamp [4]

Accompanying business logic: publication of results should use the same timestamp format as that used in the corresponding request. Further, timestamps should not be mixed, e.g. a start time should not be specified using one format with the end time in another.

4. Variables taking the value “now”, if evaluated (converted to a real time), should be done so as late as possible, in terms of both:
 - d. time – “now” should be evaluated when a request is about to be processed, not when it arrives), and
 - e. location – if the first system to receive a request is not the one to process it (e.g. it is acting as a proxy or wrapper for another system) it should, where possible, be the final system that evaluates “now”

It is conceivable that “now” need not be converted to a timestamp. For example, if the following query was made:

```
Characteristic           = [any_metric]
target time               = "now"
negative time tolerance   = "infinity"
positive time tolerance   = 0
maximum number of results = 1
```

The receiving system could interpret it as “give me your last result for [any_metric]” without ever needing to evaluate “now”.

5. It should also be possible to specify a testing interval to control how often future tests should be run between a beginning and end of a test period. For example, with start time set to 14:00 and end time at 15:00 with a testing interval of 30 minutes, tests should be run at (or close to) 14:00, 14:30, and 15:00.

6. The testing and sample intervals should both be specified in seconds using the timestamps given in point 3.
7. Where times (point 3) and time periods (point 6) are given in seconds, accuracy to a least μ S must be supported.

How - Test Parameters:

1. To satisfy certain requests, the receiving system can decide to run on-demand tests. There may be instances however, when this is undesirable to the requestor. They may not be prepared to wait for a test to be run for example, or may not want to run a test which could place extra load on the network. It should therefore be possible for the requestor to indicate whether a request should ever result in a test being run. This can be facilitated by a Boolean type, which should be entirely optional, and default to allowing tests, if not specified.

In cases where a request for an on-demand test is made, and the system is either incapable of performing, or disallows that test, an appropriate response should be returned to the requestor.

2. The schema should allow test and tool parameters to be specified in a similar way to the existing NM-WG publication schema [1], with a pre-defined set of tags, such as:

```
<packetTypeParam>TCP</packetTypeParam>
```

The pre-defined list should be a subset of those tags used in the publication schema [1]. To avoid making the request schema overly complex, tags that add little value to a request (e.g. operating system and version) should be omitted.

Accompanying business logic: It will be mandatory for schema compliant systems to recognise and accept parameter specific tags.

Parameter values can be specified as raw values (e.g. 1048576) or values with ISO quantities and units (e.g. "1024" with the ISO standard for "Mega" and "Byte") [ISO reference required].

3.
 - a. In addition to allowing parameters to be specified using separate XML tags, the schema should also allow test (tool) parameters to be specified with strings of command-line style parameters. For example, if using the Iperf tool [5], a request could contain elements such as:

```
<sourceParameterString>-c -p 5001</sourceParameterString>
<destParameterString>-s -w 1024K</destParameterString>
```

Two command-line style "fields" are required, with one for each end of a possible test path. Further, these elements should be lists of parameters not tool command lines. The tool name can be specified elsewhere in the schema, and it should be left to the receiving system to decide on an appropriate executable, including the special case where the executable to be used differs depending on whether it is to be the source or destination of a test.

Care should be taken however, as it will be optional for schema compliant systems to accept t will be optional should be noted however, th

Accompanying business logic:

- It will be optional for systems to support parameter strings.
- If a request using parameter strings is sent to a system that does not support them, then the responding system should reply with some form of "I do not support parameter strings" message.
- Implementers should consider the potential security risks of providing support for parameter strings. It would be unwise for example, to allow unchecked parameter strings, as they could allow damaging affects (e.g "rm *.*").
- Implementers should also be aware that parameter strings do not support all of the features associated with parameter specific tags. The following are not supported:
 - Reporting of parameters used in tests
 - Setting the use of parameters as "optional" or "required"

- o Setting ranges or preferred values for parameters
- b. To avoid possible contention between tool parameters specified using parameter specific tags, and in parameter strings, the two must be mutually exclusive. Schema users will only be able to specify parameter values in one or the other, not both. In the event that both are used, the parameter string should take preference, and any values given in parameter specific tags should be ignored.

Contention could also take place between the source and destination address information and the parameter strings. To prevent this, the source and destination address fields should be ignored when a parameter string is used, for all uses but populating a response with address information.

- c. For situations in which destination parameters make no sense, for example, in the case of “ping” testing where the remote end should respond automatically, any supplied parameters should be ignored.

Accompanying business logic: by the time the reader has finished this section, it will be clear that using parameter strings publication of results should use the same timestamp format as that used in the corresponding request. Further, timestamps should not be mixed, e.g. a start time should not be specified using one format with the end time in another.

- 4. The order in which parameters are specified in messages based on the schema must be maintained between the message and actual system calls to test tools. This is to prevent situations arising where the behaviour of a tool could be altered by the order in which its parameters are specified (whether this is the intended operation or through a software bug). This should apply whether the parameters are specified in parameter specific tags, or in a parameter string.

If for example, a request message was made for an iperf [5] test using port 5001 and a TCP window size of 1048576, part of the request could look like ...

```
<toolName>iperf</toolName>
<port>5001</port>
<tcpWindowSize>1048576</tcpWindowSize>
```

...in which case the resulting system should look something like...

```
iperf -p 5001 -w 1048576
```

...which gives the port first and TCP window size second, as in the request message.

- 5. When a request is received for the running of an on-demand or future test, any test parameters for which no value is specified should use the receiving system’s default value for that parameter. If the request results in a search of historic data, then unspecified parameters should match against any value of those parameters.
- 6. The schema should allow measurement parameters given using parameter specific tags to be specified as being required or optional. If a “required” parameter cannot be supported, then the request should be rejected. “optional” indicates a requestor’s preference to a particular value or set of values.

This applies to requests that may result in tests being made in the future and those which are queries against historical data. In the case of the latter, specified parameters act as a filter for returned data.

Accompanying business logic: For situations in which a request contains an extension to the schema that is unrecognised by the responder:

- IF the unknown extension is marked as “optional”, THEN it should be silently ignored by the responder. The responder should reply to the best of its ability based on the known elements of the request.
- IF the unknown extension is marked as “required”, THEN it should be rejected with an appropriate reason, such as “not understood”. A rejection fault code or series of codes will be required.

The feature should not be available for parameter strings, as this would make it a requirement for the receiver to parse the strings to determine each requested option. The requirement for parsing

would apply to at least the “required” case, as the receiver would need to identify and then compare the requested values against its known capabilities, before running any tests.

7. A facility should be provided for requesting that the parameters used to make a measurement are reported in the response. This applies to requests relating to historic data and on-demand or future tests. This feature should be available on a per parameter basis, and also as a single flag used to request reporting of all parameters.

In cases where parameter information is unavailable, the receiving system is expected to report “unknown” for optional parameters, and reject the request completely if any parameters are marked as required.

The facility should not be available for parameter strings, as this would make it a requirement for the receiver to parse the strings to identify each requested option, so that it could establish which parameters to report.

8. The schema should provide the ability to set ranges and preferential orders for tool parameters. However, this only applies to parameters specified in parameter specific tags.

This is perhaps best illustrated with two examples, both using entirely fictional XML tags. For a range, the following ...

```
<tcpBufferSize range="max">4194304</tcpBufferSize>
<tcpBufferSize range="min">1048576</tcpBufferSize>
```

...could be used to indicate that TCP buffer size should be between 1 and 4 Megabytes. While, for a preferential order, and by saying that order is significant, the following ...

```
<tcpBufferSize>4194304</tcpBufferSize>
<tcpBufferSize>1048576</tcpBufferSize>
```

...could meaning use 4Mb for TCP buffer size if possible, and 1Mb if not.

9. The specification of range and/or preferential order should work in conjunction with parameters being required or optional. Again this only applies to parameters specified in parameter specific tags.
10. To help in situations where a request results in there being a large amount of data to be returned, batching of results will be needed. The request schema must therefore provide the ability to specify a batch size, which will define the maximum number of results that can be returned in a single message.

Accompanying business logic: For example, if there are 32 results and the batch size is ten, three batches of ten results and one batch containing two results should be returned.

Specifying a batch size should be optional, and the default should be “all data in one response” (corresponding to a batch size of “max”).

Supporting batching may require the introduction of transaction ID fields, to enable the association of batches of results with their corresponding request.

11. It is envisaged that results will be returned using the NM-WG publication schema [1]. However, the schema should provide the ability to specify the return language and transport mechanism for responses to requests. This prevents users from being tied into the NM-WG publication schema, and will allow the best method for the particular purpose to be used. For example, in small, closed network domain, it may be more efficient to transfer a file of results by FTP.

Desirable Features

1. XML is extensible, and so users should be able to add custom tags to messages based on any schemas resulting from this document. This would be useful if, for example, the pre-defined list of parameter specific tags did not meet a user’s requirements. However, advice needs to be sought on the mechanisms for adding custom tags, and where in a request messages this would be appropriate. Until this advice has been gained, addition of custom tags has been dropped as a hard requirement.

Non-Requirements

1. NM-WG contributors have expressed great interest in developing a mechanism by which the monitoring and data publishing capabilities of receiving systems can be discovered. Examples include the ability to ask a system such questions as:
 - i) what network monitoring data do you have available?
 - ii) what tools are available?
 - iii) what are their versions?
 - iv) what parameters can be used with them?

There is no requirement however for the request schema to support this mechanism. Capability discovery is viewed as a facility to be provided either by technologies under development by the wider web/Grid services community, or by a separate NM-WG schema.

Use Cases

To be added.

Assumptions

1. It is assumed that this activity does not contradict or duplicate network monitoring schema work being undertaken by any other GGF groups, such as the GHPN-RG [6].
2. It is also assumed that the schema need not concern itself with aspects of security, such as user authorisation and authentication, and that this will be carried out by a function of the communication mechanism.
3. It is expected that systems will record or be aware of the parameters:
 - a. they have used in the running of tests for which they have stored historic data, and that
 - b. they will use for tests to be run in the future for which they have been requested to report the actual parameters usedSee also, the "How" section, point 7.
4. As previously mentioned in the Non-Requirements section, it is assumed that the schema need not provide any support for the discovery of a receiving system's capabilities.

Outstanding Issues

1. Interim communications, as described in the "Overall Requirement" section, and any associated schemas are yet to be defined. It is recognised that the proposed WSRF [WSRF reference required] standards may provide solutions in this area, but it is also recognised that production quality WSRF implementations are not yet available. NM-WG may therefore need to make recommendations for interim solutions, to be produced on the understanding that they may later be retired.
2. The possibility has been raised for requiring the request schema to support more complex messages, combining requests for both raw measurements and statistical summaries. As an example, "Give me the last five raw values of this characteristic and the average over the last day". This requires discussion.
3. It is a requirement that "Where possible and appropriate, use should be made of equivalent schema element names and structures to those contained in the NMWG response schema." This will inevitably lead to both schemas containing some identical, or near identical sections. In the interests of maintenance and avoiding error-prone duplication, efforts need to be made to share duplicate components between the request and publication schemas. Indeed, a hierarchy of components has been suggested, from which relevant components are selected to make up varying types of NM-WG schemas. This issue requires further study.
4. The possibility of using wildcards for characteristic names has been raised. This would allow matches against more than one characteristic per request. For example, "path.delay.*" could match

against “path.delay.oneWay” and “path.delay.roundTrip”. A decision on whether to include this as a requirement is still to be reached.

5. The pre-defined set of tags for specifying test (tool) parameters, e.g. <tcpBufferSize>, is yet to be defined.
6. Discussion is required as to whether two lists of parameters specific tags are required, with one for each end of a possible test path (source and destination). Any decision is likely to centre on whether a majority of the parameters in the pre-defined list of tags (see point 5) are source and destination specific, or common to the connection between the two. In the latter case, a single list could be used, including common cases and source and destination specific tags, as exemplified below:

```
<protocol>TCP</protocol>          <!-- common to source & dest -->
<sourcePort>5001</sourcePort>    <!-- specific to source end -->
<destPort>5002</destPort>        <!-- specific to dest end -->
```

7. Discussion is required on whether it should be possible to specify source parameters using parameter specific tags, and destination parameters using a command-line style parameter list, and vice-versa.
8. Advice needs to be sought on the mechanisms for allowing custom tags to be added to messages.
9. Discuss the inclusion of a field for transaction IDs in both the request and existing publication schema [1], to allow better tracking of interactions.

Mark Leese,
9th August 2004.