PingER service Interface Specification

Authors	Maxim Grigoriev
Date	04-01-08
Current Version	1.0

Document Change Log

As SA3-WI15 Document			
Version number	**Date**	**Description of change**	**People**
1.0	04-01-08	First draft issued	Maxim Grigoriev

Table of Contents

1 General Information

Service Name: perfSONAR-PS-PingER-1.0

Service Type: comprehensive (Measurement Archive and Measurement Point)

Version/release: 1.0

Service Description: This service provides the capability to publish data from the Measurement Archive, interact with remote Lookup Service and schedule and execute ping measurements and store measured data and metadata into the service storage SQL database.

Contact Person(s): Maxim Grigoriev, Yee-Ting Li

Contact Information: maxim at fnal dot gov, ytl at slac dot stanford dot edu

2 Introduction

This document presents interface specification of the PingER service. In the following text the interfaces of the PingER Measurement Archive and Measurement Point will be described and examples of various request/response messages will be shown. The following document is based on the sample-interface-specification-RRDMA.doc and re-uses some parts of the ***geant2_java-sql-ma/doc/Interface_Specification.doc***.
The documented interface is an implementation of the functionality, described in the PingER functionality specification [1]. For detailed protocol specifications see "PingER service protocol specification" [2].

3 Measurement Archive Interface

3.1 MA Functionality – Handle-metadata

3.1.1 Introduction

The *Handle-metadata* functionality aims to allow users to perform searches on the metadata information stored within the installed service. With the help of such searches, the user can easily check what data can be retrieved via the service.

This functionality is made available to the users with the help of *MetadataKeyRequest* and *MetadataKeyResponse* messages. Whenever the user makes a *MetadataKeyRequest*, the response from the service is always a *MetadataKeyResponse* (not considering underlying transport protocol errors). If the service encountered one or more problems, they are reported using error codes. These error codes are contained within the *MetadataKeyResponse* as well. Currently MA service supports requests with metadata supporting only one eventType - "http://ggf.org/ns/nmwg/tools/pinger/2.0/", registered under *pinger* namespace.

Number of request messages supported: 1
Number of possible error-free response message types: 1

3.1.2 MetadataKeyRequest

The .rnc file for the request message can be found in Appendix I. This .rnc file is based on NMWG v2 (base 2) xml protocol definitions and completely describes request message.
At a high level, the request message may consist of multiple *metadata* elements and mupltiple *data* elements where each *data* element must refer to the *metadata* element by *metadataIdRef*. The data element with missing metadata element will be marked as orphaned and will be skipped in the message processing. The error code will be returned for such element. Multiple metadata elements may refer to each other and the complete metadata sections can be assembled by chaining. Chaining will work only for the metadata elements with the same *evenType* elements. The presence of the *key* element will result in the immediate query to the metadata storage engine.
The number of the data elements in response message might be limited by optional message parameter from the nmwg:parameters namespace with name *setLimit.* Also, two types of time format are supported – *"iso"* and *"unix"* by setting *timeType* parameter for the whole message.

3.1.3 MetadataKeyResponse

The .rnc file for the response message can be found in Appendix I. This .rnc file is based on NMWG v2 xml protocol definitions and completely describes response message.
The *MetadataKeyResponse* message will contain at least one metadata-data chain under all circumstances (apart from transport layer errors). Each chain will contain a metadata element and data element. There can be as many chains as there are matches for the given *MetadataKeyRequest* message. If the system error was encountered, there will be only one chain in the response and this chain will contain the error code.
The metadata element can also contain a *key*. This happens when a *key* was passed in the metadataKeyRequest. In such a case the original *key* will be passed back and the same *key* will be passed inside of the data element.
If a result code in the metadata element is passed back, it is contained in an *eventType* element directly in the metadata block. The data block will also contain a datum element

(*nmwgr* namespace) and the datum element will provide more information about the error.
If no errors occurred and no error codes were passed back, the linked data block will contain a *key*. This key element contains a metaID primary key from the metaData table of the storage SQL database. This key might be encrypted by the server in the future.

3.1.4 Examples

A simple *MetadataKeyRequest* is in the first example below with partial subject for broader search criteria and *setLimit* parameter set to 10

```
<nmwg:message id="message1"
              type="MetadataKeyRequest"
              xmlns="http://ggf.org/ns/nmwg/base/2.0/"
              xmlns:pinger="http://ggf.org/ns/nmwg/tools/pinger/2.0/"
              xmlns:nmwgt="http://ggf.org/ns/nmwg/topology/2.0/"
              xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/">

 <nmwg:parameters>
         <nmwg:parameter name="setLimit" value="10"/>
         <nmwg:parameter name="timeType" value="unix"/>
 </nmwg:parameters>
 <nmwg:metadata id="meta1">
    <pinger:subject  id="subject1">
      <nmwgt:endPointPair>
        <nmwgt:dst type="hostname" value="pinger.fnal.gov"/>
     </nmwgt:endPointPair>
    </pinger:subject>

<nmwg:eventType>http://ggf.org/ns/nmwg/tools/pinger/2.0/</nmwg:eventTyp
e>
  </nmwg:metadata>
  <nmwg:data id="data1" metadataIdRef="meta1"/>
</nmwg:message>
```

A *MetadataKeyResponse* for the above request is in the example below:

```
 <nmwg:message type="MetadataKeyResponse"
              id="message.2455708"
              xmlns:nmwgt="http://ggf.org/ns/nmwg/topology/2.0/"
              xmlns:pinger="http://ggf.org/ns/nmwg/tools/pinger/2.0/"
              xmlns:select="http://ggf.org/ns/nmwg/ops/select/2.0/"
              xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/">
<nmwg:parameters>
         <nmwg:parameter name="setLimit" value="10"/>
         <nmwg:parameter name="timeType" value="unix"/>
 </nmwg:parameters>
<nmwg:metadata id="meta1">
  <pinger:subject id="subj330229">
    <nmwgt:endPointPair>
        <nmwgt:src value="lhcopnmon1-mgm.fnal.gov" type="hostname"/>
        <nmwgt:dst value="pinger.fnal.gov" type="hostname"/>
```

```
    </nmwgt:endPointPair>
</pinger:subject>
<nmwg:key id="meta330229">
   <pinger:parameters id="params330229">
      <nmwg:parameter value="10" name="count"/>
      <nmwg:parameter value="1000" name="packetSize"/>
      <nmwg:parameter value="32" name="ttl"/>
      <nmwg:parameter value="ICMP" name="transport"/>
      <nmwg:parameter value="1" name="packetInterval"/>
   </pinger:parameters>
</nmwg:key>
<nmwg:eventType>http://ggf.org/ns/nmwg/tools/pinger/2.0/</nmwg:eventTyp
e>
</nmwg:metadata>
<nmwg:data metadataIdRef="meta1" id="data1">
   <nmwg:key id="330229"/>
</nmwg:data>
</nmwg:message>
```

3.2 MA Functionality – Handle-data

3.2.1 Introduction

The Handle Data functionality provides the capability to request the service for retrieval of measurement data stored in backend storage.
There are many ways in which data can be retrieved. The service also supports bulk data retrievals wherein many requests can be put into one request whereby many overheads can be avoided. It also supports usage of 'keys' in requests. These keys would have to be retrieved previously using 'handle-metadata' functionality. Such measures when used will help in increasing the efficiency of the server and also increasing the speed with which data is available for consumption by the client. There is one interface message supported - *SetupDataRequest* and it returns *SetupDataResponse*.
Currently MA service supports requests with metadata supporting only one eventType - "http://ggf.org/ns/nmwg/tools/pinger/2.0/", registered under *pinger* namespace.
If the service encountered one or more problems, they are reported using error codes. These error codes are contained within the *SetupDataResponse* as well.

Number of request messages supported: 1
Number of possible error-free response message types: 1

3.2.2 SetupDataRequest

The .rnc file for the request message can be found in Appendix II. This .rnc file is based on NMWG v2 (base 2) xml protocol definitions and completely describes request message.

The set of rules from building *MetaDataRequest* applied here as well. The same metadata-data and multiple metadata chaining is supported. Filter chaining is supported with additional time range parameters - *'startTime'* and *'endTime'*.
There are several use cases considered for the requests.
Rrequests without keys – there is one metadata element and one data element. Time range select can be inserted as select parameters of the metadata. If time range will me missing then default time range is 24 hours past current time in UTC.
Requests with keys – A simple request for data can also be made by providing a *'key'* in the metadata section of the request with 'id' attribute set to the actual *metaID* primary key of the particular metadata in the metadata storage. The usage of the keys is preferred in case of the bulk requests.
Requests with filters – both types of requests from above can be utilized with additional metadata element and linked by *metadataIdRef.*
A response for such filter chained request will contain the main metadata section with found metadata, requested time range in form of the linked metadata element and linked data element with collection of *nmwg:commonTime* sub-elements identified by the unique timestamp.
All requests can be extensively chained with unlimited depth by matching the same *eventType* in the chained metadata. In this case the same parameters from the metadata elements will be overwritten.
The number of either metadata or data elements in response message might be limited by optional message parameter from the nmwg:parameters namespace with name *setLimit.*
Also, two types of time format are supported – *"iso"* and *"unix"* by setting *timeType* parameter for the whole message.

3.2.3 SetupDataResponse

The .rnc file for the request message can be found in Appendix II. This .rnc file is based on NMWG v2 (base 2) xml protocol definitions and completely describes response message.
A Typical Response Message of type *SetupDataResponse* may contain multiple metadata elements paired with data elements and actual measured data will be contained as multiple datums inside of *nmwg:commonTime* sub-elements.
If a match for the given metadata was found on the service, the service will return all known metadata elements and will fill up the data section with data values from the data table of the storage SQL database for the matched metadata. If data is unavailable for the given metadata or there is no metadata element in the request then an error code will be sent back in the response. If no matching metadata were found, a result error message will be sent back saying *'no metadata found'*. If service storage does not contain datum for requested metadata and time range then *'no data found'* error message will be returned. In both cases *eventType* of the metadata will signify the error nature of the result datum.

3.2.4 Examples

SetupDataRequest with additional *setLimit* parameter set to *10* and extra filter chained metadata with specified time range select

```
<nmwg:message id="message1"
            type="SetupDataRequest"
            xmlns="http://ggf.org/ns/nmwg/base/2.0/"
            xmlns:pinger="http://ggf.org/ns/nmwg/tools/pinger/2.0/"
            xmlns:nmwgt="http://ggf.org/ns/nmwg/topology/2.0/"
            xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/"
            xmlns:select="http://ggf.org/ns/nmwg/ops/select/2.0/">
<nmwg:parameters>
        <nmwg:parameter name=”setLimit” value=”10”/>
        <nmwg:parameter name=”timeType” value=”unix”/>
 </nmwg:parameters>
  <nmwg:metadata id="meta426">
    <pinger:subject id="subject426">
      <nmwgt:endPointPair>
        <nmwgt:src type="hostname" value="pinger.ictp.it"/>
        <nmwgt:dst type="hostname" value="foundation.bw"/>
      </nmwgt:endPointPair>
      <pinger:parameters>
         <nmwg:parameter name="packetSize">1000</nmwg:parameter>
          <nmwg:parameter name="count">10</nmwg:parameter>
       </pinger:parameters>
    </pinger:subject>
    <nmwg:eventType>http://ggf.org/ns/nmwg/tools/pinger/2.0/</nmwg:eventType>
  </nmwg:metadata>
 <nmwg:metadata id="meta427" metadataIdRef="meta426">
    <select:parameters id="param2c" >
      <nmwg:parameter name="startTime">1193876292</nmwg:parameter>
      <nmwg:parameter name="endTime">1194045045</nmwg:parameter>
    </select:parameters>
    <nmwg:eventType>http://ggf.org/ns/nmwg/tools/pinger/2.0/</nmwg:eventType>
  </nmwg:metadata>
  <nmwg:data id="data426" metadataIdRef="meta427"/>
</nmwg:message>
```

And corresponded SetupDataResponse message:

```
<nmwg:message
                xmlns:nmwgt="http://ggf.org/ns/nmwg/topology/2.0/"
                xmlns:select="http://ggf.org/ns/nmwg/ops/select/2.0/"
                xmlns:pinger="http://ggf.org/ns/nmwg/tools/pinger/2.0/"
                 xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/"
                type="SetupDataResponse" id="message.2455708">
```

```
<nmwg:parameters>
       <nmwg:parameter name=”setLimit” value=”10”/>
       <nmwg:parameter name=”timeType” value=”unix”/>
 </nmwg:parameters>
<nmwg:metadata id="meta1">
    <select:subject id="subj1"/>
    <select:parameters id="params1">
           <nmwg:parameter value="1193876292" name="startTime"/>
           <nmwg:parameter value="1194045045" name="endTime"/>
   </select:parameters>
   <nmwg:eventType>http://ggf.org/ns/nmwg/ops/select/2.0/</nmwg:eventType>
</nmwg:metadata>
<nmwg:metadata metadataIdRef="1" id="meta2">
    <pinger:subject id="subj25150">
       <nmwgt:endPointPair>
            <nmwgt:src value="pinger.ictp.it" type="hostname"/>
            <nmwgt:dst value="foundation.bw" type="hostname"/>
        </nmwgt:endPointPair>
    </pinger:subject>
     <nmwg:key id="meta25150">
          <pinger:parameters id="params25150">
             <nmwg:parameter value="10" name="count"/>
              <nmwg:parameter value="100" name="packetSize"/>
              <nmwg:parameter value="ICMP" name="transport"/>
             <nmwg:parameter value="1" name="packetInterval"/>
         </pinger:parameters>
     </nmwg:key>
     <nmwg:eventType>http://ggf.org/ns/nmwg/tools/pinger/2.0/</nmwg:eventType>
</nmwg:metadata>
 <nmwg:data metadataIdRef="meta2" id="data1">
    <nmwg:commonTime value="1193877091" type="unix">
           <pinger:datum value="315" name="minRtt"/>
           <pinger:datum value="323.9" name="maxRtt"/>
           <pinger:datum value="320" name="medianRtt"/>
           <pinger:datum value="339" name="meanRtt"/>
           <pinger:datum value="10" name="iqrIpd"/>
           <pinger:datum value="20" name="maxIpd"/>
           <pinger:datum value="1" name="minIpd"/>
           <pinger:datum value="7.88889" name="meanIpd"/>
     </nmwg:commonTime>
  </nmwg:data>
</nmwg:message>
```

3.2.5 MeasurementArchiveStoreRequest

Not supported at present time

3.2.6 MeasurementArchiveStoreResponse

Not supported at present time

4 General PingER Service Interface

The PingER service is capable of registering information about itself and its capabilities with the Lookup Service. In order to do so, it communicates with the LS (Lookup Service) by sending messages of registration, de-registration, etc.

4.1 Service Functionality – Handle-LS-interaction

4.1.1 RequestMessage – LSRegisterRequest

4.1.2 ResponseMessage – LSRegisterResponse for LSRegisterRequest

4.1.3 RequestMessage – LSDeregisterRequest

4.1.4 ResponseMessage – LSDeregisterResponse

4.2 Service Functionality – Handle-Echo

The PingER service provides a 'echo' feature with which clients and any other users interested in finding out the status of a service can send an echo message and get back a echo reply. If no 'success.echo' reply was got back then the client/user can assume that the service is down (even though there might not be a 404 error response). Depending on the requested endpoint the service will return echo response for MA or MP.

5 Measurement Point Interface

5.1 MP Functionality – Schedule measurement

The PingER MP is designed to provide interface to the ping facility, ie schedule ping measurements, place requests for such measurements and store measured data and newly acquired metadata into the service storage database.

5.1.1 MakeMeasurementRequest

Not supported at present time

5.1.2 MakeMeasurementResponse

Not supported at present time

6 References

1. “PingER service functional specification”, by Maxim Grigoriev and Yee-Ting Li
2. “PingER service protocol specification”, by Maxim Grigoriev and Yee-Ting Li

7 Appendix I

This appendix contains Relax NG compact schema definitions for the handle-metadata functionality interface.

7.1 MetadataKeyRequest schema

```
namespace nmwg=”http://ggf.org/ns/nmwg/base/2.0/”
namespace pinger="http://ggf.org/ns/nmwg/tools/pinger/2.0/"
namespace nmtl3="http://ggf.org/schema/network/topology/l3/20070707/"
namespace nmtl4=”http://ggf.org/schema/network/topology/l4/20070707/"
namespace nmwgt="http://ggf.org/ns/nmwg/topology/2.0/"

include “nmtopo_l4.rnc”
include “nmtopo.rnc”

start = element nmwg:message { MessageContent }

MessageContent =
      Identifier? &
      attribute messageIdRef { xsd:string }? &
      attribute type { “MetadataKeyRequest” } &
      MessageParameters? &
      (
            Metadata+,
            Data
      )+
MessageParameters = element nmwg:parameters {
            Identifier &
            MessageParameter+

MessageParameter =    element nmwg:parameter {
            attribute name {   "setLimit"  |  “timeType”  } &
            (
                  attribute value { text } |
                  text
            )
      }
Identifier =  attribute id { xsd:string }
MetadataIdentifierRef =   attribute metadataIdRef { xsd:string }
Metadata =  element nmwg:metadata {
                  Identifier &
                  MetadataIdentifierRef? &
                    ( PingerSubject  |
```

```
                    ( MetadataKeyContent |
                      PingERParameters)?
                    )?  &
                EventType?
        }
PingerSubject =      element pinger:subject {
            Identifier &
            MetadataIdentifierRef? &
        (
           EndpointPair |
           L4EndpointPair
      )
       }
 PingERParameters =      element pinger:parameters {
            Identifier &
            PingERParameter+
       }
PingERParameter =    element nmwg:parameter {
            attribute name {
                        "count" | "packetInterval" |
                        "packetSize" | "ttl" |
                        "valueUnits" | “deadline” | “protocol”|
                        “transport”
                        } &
            (
                  attribute value { text } |
                  text
            )
       }
MetadataKeyContent =     element nmwg:key     {
                  attribute id {xsd:string}?,
                  PingERParameters
         }
EventType =   element nmwg:eventType {  “http://ggf.org/ns/nmwg/tools/pinger/2.0/” }
Data =     element nmwg:data { Identifier &
                              MetadataIdentifierRef }
```

7.2 MetadataKeyResponse schema

```
namespace nmwg=”http://ggf.org/ns/nmwg/base/2.0/”
namespace pinger="http://ggf.org/ns/nmwg/tools/pinger/2.0/"
namespace nmtl3="http://ggf.org/schema/network/topology/l3/20070707/"
namespace nmtl4=”http://ggf.org/schema/network/topology/l4/20070707/"
namespace nmwgt="http://ggf.org/ns/nmwg/topology/2.0/"
namespace nmwgr="http://ggf.org/ns/nmwg/result/2.0/"
```

```
include “nmtopo_l4.rnc”
include “nmtopo.rnc”

start = element nmwg:message { MessageContent }

MessageContent =
      Identifier? &
      attribute messageIdRef { xsd:string }? &
      attribute type { “MetadataKeyResponse” } &
      MessageParameters? &
      (
            Metadata,
            Data
      )+
MessageParameters = element nmwg:parameters {
            Identifier &
            MessageParameter+

MessageParameter =    element nmwg:parameter {
            attribute name {   "setLimit"  |  “timeType”  } &
            (
                  attribute value { text } |
                  text
            )
      }
Identifier =   attribute id { xsd:string }
Metadata =  element nmwg:metadata {
                  Identifier &
                  MetadataIdentifierRef? &
                   ( PingerSubject  |
                       ( MetadataKeyContent |
                         PingERParameters)?
                    )? &
                   EventType?
       }
PingerSubject =      element pinger:subject {
            Identifier &
            MetadataIdentifierRef? &
       (
          EndpointPair |
          L4EndpointPair
     )
      }
PingERParameters =
        element pinger:parameters {
                Identifier &
```

```
                PingERParameter+
        }
PingERParameter =
        element nmwg:parameter {
                attribute name {
                                "count" | "packetInterval" |
                                "packetSize" | "ttl" |
                                "valueUnits" | "deadline" | "protocol"|
                                "transport"
                                } &
                (
                        attribute value { text } |
                        text
                )
        }
MetadataKeyContent =
        element nmwg:key
        {
                attribute id {xsd:string}?,
                PingERParameters
        }
EventType =   element nmwg:eventType {  xsd:string  }
Data = (KeyData|ResultCodeData)
KeyData = element nmwg:data
    {
        Identifier &
        MetadataIdentifierRef &
        MetadataKeyContent
    }
ResultCodeData =
    element nmwg:data {
        attribute id {xsd:string},
        attribute metadataIdRef {xsd:string},
        element nmwgr:datum { text }
    }
```

8 Appendix II

This appendix contains Relax NG compact schema definitions for the handle-data functionality interface.

8.1 SetupDataRequest schema

```
namespace nmwg="http://ggf.org/ns/nmwg/base/2.0/"
namespace pinger="http://ggf.org/ns/nmwg/tools/pinger/2.0/"
namespace nmtl3="http://ggf.org/schema/network/topology/l3/20070707/"
```

```
namespace nmtl4=”http://ggf.org/schema/network/topology/l4/20070707/"
namespace nmwgt="http://ggf.org/ns/nmwg/topology/2.0/"
namespace select=”http://ggf.org/ns/nmwg/ops/select/2.0/”

include “nmtopo_l4.rnc”
include “nmtopo.rnc”

start = element nmwg:message { MessageContent }

MessageContent =
      Identifier? &
      attribute messageIdRef { xsd:string }? &
      attribute type { “SetupDataRequest” } &
      MessageParameters? &
      (
            Metadata+,
            Data
      )+
MessageParameters = element nmwg:parameters {
            Identifier &
            MessageParameter+

MessageParameter =    element nmwg:parameter {
            attribute name {   "setLimit"  |  “timeType”  } &
            (
                  attribute value { text } |
                  text
            )
      }
Identifier =  attribute id { xsd:string }
MetadataIdentifierRef =   attribute metadataIdRef { xsd:string }
Metadata =  element nmwg:metadata {
                  Identifier &
                  MetadataIdentifierRef? &
                  (
                    ( PingerSubject  |
                      ( MetadataKeyContent |
                        PingERParameters)?
                     )? &
                   FilterMetadataContent?
                  ) &
                  EventType?
        }
PingerSubject =      element pinger:subject {
            Identifier &
            MetadataIdentifierRef? &
```

```
        (
            EndpointPair |
            L4EndpointPair
       )
        }
FilterMetadataContent =     element select:subject     {
            Identifier  &
            MetadataIdentifierRef?
        },
        element select:parameters { SelectParametersContent }
 SelectParametersContent =
            Identifier &
                        MetadataIdentifierRef?
            element nmwg:parameter {
                attribute name {   "startTime" | "endTime"  } &
                (
                  attribute value { text } |
                  text
                )
             }

 PingERParameters =      element pinger:parameters {
            Identifier &
            PingERParameter+
        }
PingERParameter =    element nmwg:parameter {
            attribute name {
                        "count" | "packetInterval" |
                        "packetSize" | "ttl" |
                        "valueUnits" | “deadline” | “protocol”|
                        “transport”
                        } &
            (
                  attribute value { text } |
                  text
            )
        }
MetadataKeyContent =     element nmwg:key     {
                  attribute id {xsd:string}?,
                  PingERParameters
         }
EventType =   element nmwg:eventType {
                        ”http://ggf.org/ns/nmwg/ops/select/2.0/” |
                        “http://ggf.org/ns/nmwg/tools/pinger/2.0/”
                  }
Data =     element nmwg:data { Identifier &
```

```
MetadataIdentifierRef }
```

8.2 SetupDataResponse schema

```
namespace nmwg=”http://ggf.org/ns/nmwg/base/2.0/”
namespace pinger="http://ggf.org/ns/nmwg/tools/pinger/2.0/"
namespace nmtl3="http://ggf.org/schema/network/topology/l3/20070707/"
namespace nmtl4=”http://ggf.org/schema/network/topology/l4/20070707/"
namespace nmwgt="http://ggf.org/ns/nmwg/topology/2.0/"
namespace nmwgr=http://ggf.org/ns/nmwg/result/2.0/
namespace select =  ”http://ggf.org/ns/nmwg/ops/select/2.0/”

include “nmtopo_l4.rnc”
include “nmtopo.rnc”

start = element nmwg:message { MessageContent }

MessageContent =
      Identifier? &
      attribute messageIdRef { xsd:string }? &
      attribute type { “MetadataKeyResponse” } &
      MessageParameters? &
      (
            Metadata,
            Data
      )+
MessageParameters = element nmwg:parameters {
            Identifier &
            MessageParameter+

MessageParameter =    element nmwg:parameter {
            attribute name {   "setLimit"  |  “timeType”  } &
            (
                  attribute value { text } |
                  text
            )
      }
Identifier =   attribute id { xsd:string }
Metadata =  element nmwg:metadata {
                  Identifier &
                  MetadataIdentifierRef? &
                  (
                    ( PingerSubject  |
                      ( MetadataKeyContent |
```

```
                    PingERParameters)?
                  )? &
                 FilterMetadataContent?
               ) &
               EventType?
       }
PingerSubject =      element pinger:subject {
           Identifier &
           MetadataIdentifierRef? &
       (
          EndpointPair |
          L4EndpointPair
     )
      }
FilterMetadataContent =     element select:subject     {
           Identifier  &
           MetadataIdentifierRef?
      },
      element select:parameters { SelectParametersContent }
 SelectParametersContent =
           Identifier &
                    MetadataIdentifierRef?
           element nmwg:parameter {
              attribute name {   "startTime" | "endTime"  } &
              (
                attribute value { text } |
                text
              )
            }
PingERParameters =
        element pinger:parameters {
                Identifier &
                PingERParameter+
        }
PingERParameter =
        element nmwg:parameter {
                attribute name {
                                "count" | "packetInterval" |
                                "packetSize" | "ttl" |
                                "valueUnits" | "deadline" | "protocol"|
                                "transport"
                                } &
                (
                        attribute value { text } |
                        text
                )
        }
MetadataKeyContent =
       element nmwg:key
          {
```

```
                attribute id {xsd:string}?,
                PingERParameters
        }
EventType =   element nmwg:eventType {  xsd:string  }
Data = (ResultData|ResultCodeData)
ResultData = element nmwg:data      {
        Identifier &
        MetadataIdentifierRef &
        PingERCommonTime+
    }
PingERCommonTime =    element nmwg:commonTime {
        attribute type { “unix” }  &
        attribute value { xsd:string },
        PingERDatum+
    }
ResultCodeData =   element nmwg:data {
        attribute id {xsd:string},
        attribute metadataIdRef {xsd:string},
        element nmwgr:datum { text }
    }
PingERDatum =       element pinger:datum {
        (
          (
            attribute value { xsd:float } &
            attribute valueUnits { xsd:string }? &
            attribute seqNum { xsd:int }? &
            attribute numBytes { xsd:int }? &
            attribute ttl { xsd:int }?
          ) |
          (
            attribute name {  "minRtt" | "maxRtt" |
                              "meanRtt" | "medianRtt" |
                              "lossPercent" | "clp" |
                              "minIpd" | "maxIpd" |
                              "iqrIpd" | "meanIpd" } &
            attribute value { xsd:float } &
            attribute valueUnits { xsd:string }?
          ) |
          (
            attribute name { "outOfOrder" | "duplicates" } &
            attribute value { xsd:boolean }
          )
        ) &

    }
```

8.3 MeasurementArchiveStoreRequest schema

8.4 MeasurementArchiveStoreResponse schema

9 Appendix III

9.1 LSRegisterRequet schema

9.2 LSRegisterResponse schema

9.3 LSDeregisterRequest schema

9.4 LSDeregisterResponse schema