Network Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: March 14, 2020

M. Bagnulo
UC3M
B. Claise
Cisco Systems, Inc.
P. Eardley
BT
A. Morton
AT&T Labs
A. Akhter
Consultant

September 11, 2019

Registry for Performance Metrics
draft-ietf-ippm-metric-registry-20

Abstract

This document defines the format for the IANA Performance Metrics Registry. This document also gives a set of guidelines for Registered Performance Metric requesters and reviewers.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 14, 2020.

Copyright Notice

Table of Contents

1. Introduction

The IETF specifies and uses Performance Metrics of protocols and applications transported over its protocols. Performance metrics are such an important part of the operations of IETF protocols that [RFC6390] specifies guidelines for their development.

The definition and use of Performance Metrics in the IETF happens in various working groups (WG), most notably:

The "IP Performance Metrics" (IPPM) WG is the WG primarily focusing on Performance Metrics definition at the IETF.

The "Metric Blocks for use with RTCP's Extended Report Framework" (XRBLOCK) WG recently specified many Performance Metrics related to "RTP Control Protocol Extended Reports (RTCP XR)" [RFC3611], which establishes a framework to allow new information to be conveyed in RTCP, supplementing the original report blocks defined in "RTP: A Transport Protocol for Real-Time Applications", [RFC3550].

The "Benchmarking Methodology" WG (BMWG) defined many Performance Metrics for use in laboratory benchmarking of inter-networking technologies.

The "IP Flow Information eXport" (IPFIX) concluded WG specified an IANA process for new Information Elements. Some Performance Metrics related Information Elements are proposed on regular basis.

The "Performance Metrics for Other Layers" (PMOL) a concluded WG, defined some Performance Metrics related to Session Initiation Protocol (SIP) voice quality [RFC6035].

It is expected that more Performance Metrics will be defined in the future, not only IP-based metrics, but also metrics which are protocol-specific and application-specific.

Despite the importance of Performance Metrics, there are two related problems for the industry. First, ensuring that when one party requests another party to measure (or report or in some way act on) a particular Performance Metric, then both parties have exactly the same understanding of what Performance Metric is being referred to. Second, discovering which Performance Metrics have been specified, to avoid developing a new Performance Metric that is very similar, but not quite inter-operable. These problems can be addressed by creating a registry of performance metrics. The usual way in which the IETF organizes registries is with Internet Assigned Numbers Authority (IANA), and there is currently no Performance Metrics Registry maintained by the IANA.

This document requests that IANA create and maintain a Performance Metrics Registry, according to the maintenance procedures and the Performance Metrics Registry format defined in this memo. The resulting Performance Metrics Registry is for use by the IETF and others. Although the Registry formatting specifications herein are primarily for registry creation by IANA, any other organization that wishes to create a Performance Metrics Registry MAY use the same formatting specifications for their purposes. The authors make no guarantee of the registry format's applicability to any possible set of Performance Metrics envisaged by other organizations, but encourage others to apply it. In the remainder of this document, unless we explicitly say otherwise, we will refer to the IANA-maintained Performance Metrics Registry as simply the Performance Metrics Registry.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Performance Metric: A Performance Metric is a quantitative measure of performance, targeted to an IETF-specified protocol or targeted to an application transported over an IETF-specified protocol. Examples of Performance Metrics are the FTP response time for a

complete file download, the DNS response time to resolve the IP address, a database logging time, etc. This definition is consistent with the definition of metric in [RFC2330] and broader than the definition of performance metric in [RFC6390].

Registered Performance Metric: A Registered Performance Metric is a Performance Metric expressed as an entry in the Performance Metrics Registry, administered by IANA. Such a performance metric has met all the registry review criteria defined in this document in order to included in the registry.

Performance Metrics Registry: The IANA registry containing Registered Performance Metrics.

Proprietary Registry: A set of metrics that are registered in a proprietary registry, as opposed to Performance Metrics Registry.

Performance Metrics Experts: The Performance Metrics Experts is a group of designated experts [RFC8126] selected by the IESG to validate the Performance Metrics before updating the Performance Metrics Registry. The Performance Metrics Experts work closely with IANA.

Parameter: A Parameter is an input factor defined as a variable in the definition of a Performance Metric. A Parameter is a numerical or other specified factor forming one of a set that defines a metric or sets the conditions of its operation. All Parameters must be known to measure using a metric and interpret the results. There are two types of Parameters: Fixed and Run-time parameters. For the Fixed Parameters, the value of the variable is specified in the Performance Metrics Registry entry and different Fixed Parameter values results in different Registered Performance Metrics. For the Run-time Parameters, the value of the variable is defined when the metric measurement method is executed and a given Registered Performance Metric supports multiple values for the parameter. Although Run-time Parameters do not change the fundamental nature of the Performance Metric's definition, some have substantial influence on the network property being assessed and interpretation of the results.

Note: Consider the case of packet loss in the following two Active Measurement Method cases. The first case is packet loss as background loss where the Run-time Parameter set includes a very sparse Poisson stream, and only characterizes the times when packets were lost. Actual user streams likely see much higher loss at these times, due to tail drop or radio errors. The second case is packet loss as inverse of throughput where the Run-time Parameter set includes a very dense, bursty

stream, and characterizes the loss experienced by a stream that approximates a user stream. These are both "loss metrics", but the difference in interpretation of the results is highly dependent on the Run-time Parameters (at least), to the extreme where we are actually using loss to infer its compliment: delivered throughput.

Active Measurement Method: Methods of Measurement conducted on traffic which serves only the purpose of measurement and is generated for that reason alone, and whose traffic characteristics are known a priori. The complete definition of Active Methods is specified in section 3.4 of[RFC7799]. Examples of Active Measurement Methods are the measurement methods for the One way delay metric defined in [RFC7679] and the one for round trip delay defined in [RFC2681].

Passive Measurement Method: Methods of Measurement conducted on network traffic, generated either from the end users or from network elements that would exist regardless whether the measurement was being conducted or not. The complete definition of Passive Methods is specified in section 3.6 of [RFC7799]. One characteristic of Passive Measurement Methods is that sensitive information may be observed, and as a consequence, stored in the measurement system.

Hybrid Measurement Method: Hybrid Methods are Methods of Measurement that use a combination of Active Methods and Passive Methods, to assess Active Metrics, Passive Metrics, or new metrics derived from the a priori knowledge and observations of the stream of interest. The complete definition of Hybrid Methods is specified in section 3.8 of [RFC7799].

3. Scope

This document is intended for two different audiences:

1. For those defining new Registered Performance Metrics, it provides specifications and best practices to be used in deciding which Registered Performance Metrics are useful for a measurement study, instructions for writing the text for each column of the Registered Performance Metrics, and information on the supporting documentation required for the new Performance Metrics Registry entry (up to and including the publication of one or more RFCs or I-Ds describing it).

2. For the appointed Performance Metrics Experts and for IANA personnel administering the new IANA Performance Metrics Registry, it defines a set of acceptance criteria against which

these proposed Registered Performance Metrics should be evaluated.

In addition, this document may be useful for other organizations who are defining a Performance Metric registry of their own, and may re-use the features of the Performance Metrics Registry defined in this document.

This Performance Metrics Registry is applicable to Performance Metrics issued from Active Measurement, Passive Measurement, and any other form of Performance Metric. This registry is designed to encompass Performance Metrics developed throughout the IETF and especially for the technologies specified in the following working groups: IPPM, XRBLOCK, IPFIX, and BMWG. This document analyzes an prior attempt to set up a Performance Metrics Registry, and the reasons why this design was inadequate [RFC6248]. Finally, this document gives a set of guidelines for requesters and expert reviewers of candidate Registered Performance Metrics.

This document makes no attempt to populate the Performance Metrics Registry with initial entries.

Based on [RFC8126] Section 4.3, this document is processed as Best Current Practice (BCP) [RFC2026].

4. Motivation for a Performance Metrics Registry

In this section, we detail several motivations for the Performance Metrics Registry.

4.1. Interoperability

As any IETF registry, the primary use for a registry is to manage a registry for its use within one or more protocols. In the particular case of the Performance Metrics Registry, there are two types of protocols that will use the Performance Metrics in the Performance Metrics Registry during their operation (by referring to the Index values):

- Control protocol: This type of protocol used to allow one entity to request another entity to perform a measurement using a specific metric defined by the Performance Metrics Registry. One particular example is the LMAP framework [RFC7594]. Using the LMAP terminology, the Performance Metrics Registry is used in the LMAP Control protocol to allow a Controller to request a measurement task to one or more Measurement Agents. In order to enable this use case, the entries of the Performance Metrics Registry must be sufficiently defined to allow a Measurement Agent

implementation to trigger a specific measurement task upon the reception of a control protocol message. This requirement heavily constrains the type of entries that are acceptable for the Performance Metrics Registry.

- Report protocol: This type of protocol is used to allow an entity to report measurement results to another entity. By referencing to a specific Performance Metrics Registry, it is possible to properly characterize the measurement result data being reported. Using the LMAP terminology, the Performance Metrics Registry is used in the Report protocol to allow a Measurement Agent to report measurement results to a Collector.

It should be noted that the LMAP framework explicitly allows for using not only the IANA-maintained Performance Metrics Registry but also other registries containing Performance Metrics, either defined by other organizations or private ones. However, others who are creating Registries to be used in the context of an LMAP framework are encouraged to use the Registry format defined in this document, because this makes it easier for developers of LMAP Measurement Agents (MAs) to programmatically use information found in those other Registries' entries.

4.2. Single point of reference for Performance Metrics

A Performance Metrics Registry serves as a single point of reference for Performance Metrics defined in different working groups in the IETF. As we mentioned earlier, there are several WGs that define Performance Metrics in the IETF and it is hard to keep track of all them. This results in multiple definitions of similar Performance Metrics that attempt to measure the same phenomena but in slightly different (and incompatible) ways. Having a registry would allow both the IETF community and external people to have a single list of relevant Performance Metrics defined by the IETF (and others, where appropriate). The single list is also an essential aspect of communication about Performance Metrics, where different entities that request measurements, execute measurements, and report the results can benefit from a common understanding of the referenced Performance Metric.

4.3. Side benefits

There are a couple of side benefits of having such a registry. First, the Performance Metrics Registry could serve as an inventory of useful and used Performance Metrics, that are normally supported by different implementations of measurement agents. Second, the results of measurements using the Performance Metrics should be comparable even if they are performed by different implementations

and in different networks, as the Performance Metric is properly defined. BCP 176 [RFC6576] examines whether the results produced by independent implementations are equivalent in the context of evaluating the completeness and clarity of metric specifications. This BCP defines the standards track advancement testing for (active) IPPM metrics, and the same process will likely suffice to determine whether Registered Performance Metrics are sufficiently well specified to result in comparable (or equivalent) results. Registered Performance Metrics which have undergone such testing SHOULD be noted, with a reference to the test results.

5. Criteria for Performance Metrics Registration

It is neither possible nor desirable to populate the Performance Metrics Registry with all combinations of Parameters of all Performance Metrics. The Registered Performance Metrics should be:

1. interpretable by the user.

2. implementable by the software designer,

3. deployable by network operators,

4. accurate, for interoperability and deployment across vendors,

5. Operationally useful, so that it has significant industry interest and/or has seen deployment,

6. Sufficiently tightly defined, so that different values for the Run-time Parameters does not change the fundamental nature of the measurement, nor change the practicality of its implementation.

In essence, there needs to be evidence that a candidate Registered Performance Metric has significant industry interest, or has seen deployment, and there is agreement that the candidate Registered Performance Metric serves its intended purpose.

6. Performance Metric Registry: Prior attempt

There was a previous attempt to define a metric registry RFC 4148 [RFC4148]. However, it was obsoleted by RFC 6248 [RFC6248] because it was "found to be insufficiently detailed to uniquely identify IPPM metrics... [there was too much] variability possible when characterizing a metric exactly" which led to the RFC4148 registry having "very few users, if any".

A couple of interesting additional quotes from RFC 6248 might help understand the issues related to that registry.

1. "It is not believed to be feasible or even useful to register every possible combination of Type P, metric parameters, and Stream parameters using the current structure of the IPPM Metrics Registry."

2. "The registry structure has been found to be insufficiently detailed to uniquely identify IPPM metrics."

3. "Despite apparent efforts to find current or even future users, no one responded to the call for interest in the RFC 4148 registry during the second half of 2010."

The current approach learns from this by tightly defining each Registered Performance Metric with only a few variable (Run-time) Parameters to be specified by the measurement designer, if any. The idea is that entries in the Performance Metrics Registry stem from different measurement methods which require input (Run-time) parameters to set factors like source and destination addresses (which do not change the fundamental nature of the measurement). The downside of this approach is that it could result in a large number of entries in the Performance Metrics Registry. There is agreement that less is more in this context - it is better to have a reduced set of useful metrics rather than a large set of metrics, some with with questionable usefulness.

6.1. Why this Attempt Will Succeed

As mentioned in the previous section, one of the main issues with the previous registry was that the metrics contained in the registry were too generic to be useful. This document specifies stricter criteria for performance metric registration (see section 6), and imposes a group of Performance Metrics Experts that will provide guidelines to assess if a Performance Metric is properly specified.

Another key difference between this attempt and the previous one is that in this case there is at least one clear user for the Performance Metrics Registry: the LMAP framework and protocol. Because the LMAP protocol will use the Performance Metrics Registry values in its operation, this actually helps to determine if a metric is properly defined. In particular, since we expect that the LMAP control protocol will enable a controller to request a measurement agent to perform a measurement using a given metric by embedding the Performance Metrics Registry value in the protocol, a metric is properly specified if it is defined well-enough so that it is possible (and practical) to implement the metric in the measurement agent. This was the failure of the previous attempt: a registry entry with an undefined Type-P (section 13 of RFC 2330 [RFC2330]) allows implementation to be ambiguous.

7. Definition of the Performance Metric Registry

This Performance Metrics Registry is applicable to Performance Metrics used for Active Measurement, Passive Measurement, and any other form of Performance Metric. Each category of measurement has unique properties, so some of the columns defined below are not applicable for a given metric category. In this case, the column(s) SHOULD be populated with the "NA" value (Non Applicable). However, the "NA" value MUST NOT be used by any metric in the following columns: Identifier, Name, URI, Status, Requester, Revision, Revision Date, Description. In the future, a new category of metrics could require additional columns, and adding new columns is a recognized form of registry extension. The specification defining the new column(s) MUST give guidelines to populate the new column(s) for existing entries (in general).

The columns of the Performance Metrics Registry are defined below. The columns are grouped into "Categories" to facilitate the use of the registry. Categories are described at the 7.x heading level, and columns are at the 7.x.y heading level. The Figure below illustrates this organization. An entry (row) therefore gives a complete description of a Registered Performance Metric.

Each column serves as a check-list item and helps to avoid omissions during registration and expert review.

Registry Categories and Columns, shown as

```
Category
------------------
Column |  Column |

Summary
-------------------------------------------------------------------------
Identifier | Name | URIs | Desc. | Reference | Change Controller | Ver |

Metric Definition
-----------------------------------------
Reference Definition | Fixed Parameters |

Method of Measurement
-----------------------------------------------------------------------
Reference | Packet     | Traffic | Sampling     | Run-time   | Role |
Method    | Stream     | Filter  | Distribution | Parameters |      |
          | Generation |
Output
-----------------------------------------
Type | Reference  | Units | Calibration |
     | Definition |       |             |

Administrative Information
----------------------------------
Status |Request | Rev | Rev.Date |

Comments and Remarks
--------------------
```

7.1. Summary Category

7.1.1. Identifier

A numeric identifier for the Registered Performance Metric. This identifier MUST be unique within the Performance Metrics Registry.

The Registered Performance Metric unique identifier is an unbounded integer (range 0 to infinity).

The Identifier 0 should be Reserved. The Identifier values from 64512 to 65536 are reserved for private use.

When adding newly Registered Performance Metrics to the Performance Metrics Registry, IANA SHOULD assign the lowest available identifier to the new Registered Performance Metric.

If a Performance Metrics Expert providing review determines that there is a reason to assign a specific numeric identifier, possibly leaving a temporary gap in the numbering, then the Performance Expert SHALL inform IANA of this decision.

7.1.2. Name

As the name of a Registered Performance Metric is the first thing a potential human implementor will use when determining whether it is suitable for their measurement study, it is important to be as precise and descriptive as possible. In future, users will review the names to determine if the metric they want to measure has already been registered, or if a similar entry is available as a basis for creating a new entry.

Names are composed of the following elements, separated by an underscore character "_":

MetricType_Method_SubTypeMethod_... Spec_Units_Output

- MetricType: a combination of the directional properties and the metric measured, such as:

  RTDelay (Round Trip Delay)

  RTDNS (Response Time Domain Name Service)

  RLDNS (Response Loss Domain Name Service)

  OWDelay (One Way Delay)

  RTLoss (Round Trip Loss)

  OWLoss (One Way Loss)

  OWPDV (One Way Packet Delay Variation)

  OWIPDV (One Way Inter-Packet Delay Variation)

  OWReorder (One Way Packet Reordering)

  OWDuplic (One Way Packet Duplication)

  OWBTC (One Way Bulk Transport Capacity)

  OWMBM (One Way Model Based Metric)

  SPMonitor (Single Point Monitor)

MPMonitor (Multi-Point Monitor)

- Method: One of the methods defined in [RFC7799], such as:

  Active (depends on a dedicated measurement packet stream and observations of the stream)

  Passive (depends *solely* on observation of one or more existing packet streams)

  HybridType1 (obervations on one stream that combine both active and passive methods)

  HybridType2 (obervations on two or more streams that combine both active and passive methods)

  Spatial (Spatial Metric of RFC5644)

- SubTypeMethod: One or more sub-types to further describe the features of the entry, such as:

  ICMP (Internet Control Message Protocol)

  IP (Internet Protocol)

  DSCPxx (where xx is replaced by a Diffserv code point)

  UDP (User Datagram Protocol)

  TCP (Transport Control Protocol)

  QUIC (QUIC transport protocol)

  HS (Hand-Shake, such as TCP's 3-way HS)

  Poisson (Packet generation using Poisson distribution)

  Periodic (Periodic packet generation)

  SendOnRcv (Sender keeps one packet in-transit by sending when previous packet arrives)

  PayloadxxxxB (where xxxx is replaced by an integer, the number of octets in the Payload))

  SustainedBurst (Capacity test, worst case)

  StandingQueue (test of bottleneck queue behavior)

SubTypeMethod values are separated by a hyphen "-" character, which indicates that they belong to this element, and that their order is unimportant when considering name uniqueness.

- Spec: RFC number and major section number that specifies this Registry entry in the form RFCXXXXsecY, such as RFC7799sec3. Note: the RFC number is not the Primary Reference specification for the metric definition, such as [RFC7679] for One-way Delay; it will contain the placeholder "RFCXXXXsecY" until the RFC number is assigned to the specifying document, and would remain blank in private registry entries without a corresponding RFC.

- Units: The units of measurement for the output, such as:

  Seconds

  Ratio (unitless)

  Percent (value multiplied by 100)

  Logical (1 or 0)

  Packets

  BPS (Bits per Second)

  PPS (Packets per Second)

  EventTotal (for unit-less counts)

  Multiple (more than one type of unit)

  Enumerated (a list of outcomes)

  Unitless

- Output: The type of output resulting from measurement, such as:

  Singleton

  Raw (multiple Singletons)

  Count

  Minimum

  Maximum

Median

Mean

95Percentile (95th Percentile)

99Percentile (99th Percentile)

StdDev (Standard Deviation)

Variance

PFI (Pass, Fail, Inconclusive)

FlowRecords (descriptions of flows observed)

LossRatio (lost packets to total packets, <=1)

An example is:

RTDelay_Active_IP-UDP-Periodic_RFCXXXXsecY_Seconds_95Percentile

as described in section 4 of [I-D.ietf-ippm-initial-registry].

Note that private registries following the format described here SHOULD use the prefix "Priv_" on any name to avoid unintended conflicts (further considerations are described in section 10). Private registry entries usually have no specifying RFC, thus the Spec: element has no clear interpretation.

7.1.3. URIs

The URIs column MUST contain a URL [RFC3986] that uniquely identifies and locates the metric entry so it is accessible through the Internet. The URL points to a file containing all the human-readable information for one registry entry. The URL SHALL reference a target file that is HTML-formated and contains URLs to referenced sections of HTML-ized RFCs. These target files for different entries can be more easily edited and re-used when preparing new entries. The exact form of the URL for each target file will be determined by IANA and reside on "iana.org". The major sections of [I-D.ietf-ippm-initial-registry] provide an example of a target file in HTML form (sections 4 and higher).

7.1.4. Description

A Registered Performance Metric description is a written representation of a particular Performance Metrics Registry entry. It supplements the Registered Performance Metric name to help Performance Metrics Registry users select relevant Registered Performance Metrics.

7.1.5. Reference

This entry gives the specification containing the candidate registry entry which was reviewed and agreed, if such an RFC or other specification exists.

7.1.6. Change Controller

This entry names the entity responsible for approving revisions to the registry entry, and SHALL provide contact information (for an individual, where appropriate).

7.1.7. Version (of Registry Format)

This entry gives the version number for the registry format used. Formats complying with this memo MUST use 1.0. The version number SHALL NOT change unless a new RFC is published that changes the registry format.

7.2. Metric Definition Category

This category includes columns to prompt all necessary details related to the metric definition, including the RFC reference and values of input factors, called fixed parameters, which are left open in the RFC but have a particular value defined by the performance metric.

7.2.1. Reference Definition

This entry provides a reference (or references) to the relevant section(s) of the document(s) that define the metric, as well as any supplemental information needed to ensure an unambiguous definition for implementations. The reference needs to be an immutable document, such as an RFC; for other standards bodies, it is likely to be necessary to reference a specific, dated version of a specification.

7.2.2. Fixed Parameters

Fixed Parameters are Parameters whose value must be specified in the Performance Metrics Registry. The measurement system uses these values.

Where referenced metrics supply a list of Parameters as part of their descriptive template, a sub-set of the Parameters will be designated as Fixed Parameters. As an example for active metrics, Fixed Parameters determine most or all of the IPPM Framework convention "packets of Type-P" as described in [RFC2330], such as transport protocol, payload length, TTL, etc. An example for passive metrics is for RTP packet loss calculation that relies on the validation of a packet as RTP which is a multi-packet validation controlled by MIN_SEQUENTIAL as defined by [RFC3550]. Varying MIN_SEQUENTIAL values can alter the loss report and this value could be set as a Fixed Parameter.

Parameters MUST have well-defined names. For human readers, the hanging indent style is preferred, and any Parameter names and definitions that do not appear in the Reference Method Specification MUST appear in this column (or Run-time Parameters column).

Parameters MUST have a well-specified data format.

A Parameter which is a Fixed Parameter for one Performance Metrics Registry entry may be designated as a Run-time Parameter for another Performance Metrics Registry entry.

7.3. Method of Measurement Category

This category includes columns for references to relevant sections of the RFC(s) and any supplemental information needed to ensure an unambiguous method for implementations.

7.3.1. Reference Method

This entry provides references to relevant sections of the RFC(s) describing the method of measurement, as well as any supplemental information needed to ensure unambiguous interpretation for implementations referring to the RFC text.

Specifically, this section should include pointers to pseudocode or actual code that could be used for an unambigious implementation.

7.3.2. Packet Stream Generation

This column applies to Performance Metrics that generate traffic as part of their Measurement Method, including but not necessarily limited to Active metrics. The generated traffic is referred as a stream and this column describes its characteristics.

Each entry for this column contains the following information:

- Value: The name of the packet stream scheduling discipline
- Reference: the specification where the parameters of the stream are defined

The packet generation stream may require parameters such as the average packet rate and distribution truncation value for streams with Poisson-distributed inter-packet sending times. In case such parameters are needed, they should be included either in the Fixed parameter column or in the run time parameter column, depending on wether they will be fixed or will be an input for the metric.

The simplest example of stream specification is Singleton scheduling (see [RFC2330]), where a single atomic measurement is conducted. Each atomic measurement could consist of sending a single packet (such as a DNS request) or sending several packets (for example, to request a webpage). Other streams support a series of atomic measurements in a "sample", with a schedule defining the timing between each transmitted packet and subsequent measurement. Principally, two different streams are used in IPPM metrics, Poisson distributed as described in [RFC2330] and Periodic as described in [RFC3432]. Both Poisson and Periodic have their own unique parameters, and the relevant set of parameters names and values should be included either in the Fixed Parameters column or in the Run-time parameter column.

7.3.3. Traffic Filter

This column applies to Performance Metrics that observe packets flowing through (the device with) the measurement agent i.e. that is not necessarily addressed to the measurement agent. This includes but is not limited to Passive Metrics. The filter specifies the traffic that is measured. This includes protocol field values/ ranges, such as address ranges, and flow or session identifiers.

The traffic filter itself depends on needs of the metric itself and a balance of an operator’s measurement needs and a user’s need for privacy. Mechanics for conveying the filter criteria might be the BPF (Berkley Packet Filter) or PSAMP [RFC5475] Property Match

Filtering which reuses IPFIX [RFC7012]. An example BPF string for matching TCP/80 traffic to remote destination net 192.0.2.0/24 would be "dst net 192.0.2.0/24 and tcp dst port 80". More complex filter engines might be supported by the implementation that might allow for matching using Deep Packet Inspection (DPI) technology.

The traffic filter includes the following information:

Type: the type of traffic filter used, e.g. BPF, PSAMP, OpenFlow rule, etc. as defined by a normative reference

Value: the actual set of rules expressed

7.3.4. Sampling Distribution

The sampling distribution defines out of all the packets that match the traffic filter, which one of those are actually used for the measurement. One possibility is "all" which implies that all packets matching the Traffic filter are considered, but there may be other sampling strategies. It includes the following information:

Value: the name of the sampling distribution

Reference definition: pointer to the specification where the sampling distribution is properly defined.

The sampling distribution may require parameters. In case such parameters are needed, they should be included either in the Fixed parameter column or in the run time parameter column, depending on whether they will be fixed or will be an input for the metric.

Sampling and Filtering Techniques for IP Packet Selection are documented in the PSAMP (Packet Sampling) [RFC5475], while the Framework for Packet Selection and Reporting, [RFC5474] provides more background information. The sampling distribution parameters might be expressed in terms of the Information Model for Packet Sampling Exports, [RFC5477], and the Flow Selection Techniques, [RFC7014].

7.3.5. Run-time Parameters

Run-Time Parameters are Parameters that must be determined, configured into the measurement system, and reported with the results for the context to be complete. However, the values of these parameters is not specified in the Performance Metrics Registry (like the Fixed Parameters), rather these parameters are listed as an aid to the measurement system implementer or user (they must be left as variables, and supplied on execution).

Where metrics supply a list of Parameters as part of their descriptive template, a sub-set of the Parameters will be designated as Run-Time Parameters.

Parameters MUST have well defined names. For human readers, the hanging indent style is preferred, and the names and definitions that do not appear in the Reference Method Specification MUST appear in this column.

A Data Format for each Run-time Parameter MUST be specified in this column, to simplify the control and implementation of measurement devices. For example, parameters that include an IPv4 address can be encoded as a 32 bit integer (i.e. binary base64 encoded value) or ip-address as defined in [RFC6991]. The actual encoding(s) used must be explicitly defined for each Run-time parameter. IPv6 addresses and options MUST be accomodated, allowing Registered Metrics to be used in either address family.

Examples of Run-time Parameters include IP addresses, measurement point designations, start times and end times for measurement, and other information essential to the method of measurement.

7.3.6. Role

In some methods of measurement, there may be several roles defined, e.g., for a one-way packet delay active measurement there is one measurement agent that generates the packets and another agent that receives the packets. This column contains the name of the Role(s) for this particular entry. In the one-way delay example above, there should be two entries in the Role registry column, one for each Role (Source and Destination). When a measurement agent is instructed to perform the "Source" Role for one-way delay metric, the agent knows that it is required to generate packets. The values for this field are defined in the reference method of measurement (and this frequently results in abbreviated role names such as "Src").

When the Role column of a registry entry defines more than one Role, then the Role SHALL be treated as a Run-time Parameter and supplied for execution. It should be noted that the LMAP framework [RFC7594] distinguishes the Role from other Run-time Parameters, and defines a special parameter "Roles" inside the registry-grouping function list in the LMAP YANG model[RFC8194].

7.4. Output Category

For entries which involve a stream and many singleton measurements, a statistic may be specified in this column to summarize the results to

a single value. If the complete set of measured singletons is output, this will be specified here.

Some metrics embed one specific statistic in the reference metric definition, while others allow several output types or statistics.

7.4.1. Type

This column contains the name of the output type. The output type defines a single type of result that the metric produces. It can be the raw results (packet send times and singleton metrics), or it can be a summary statistic. The specification of the output type MUST define the format of the output. In some systems, format specifications will simplify both measurement implementation and collection/storage tasks. Note that if two different statistics are required from a single measurement (for example, both "Xth percentile mean" and "Raw"), then a new output type must be defined ("Xth percentile mean AND Raw"). See the Naming section above for a list of Output Types.

7.4.2. Reference Definition

This column contains a pointer to the specification(s) where the output type and format are defined.

7.4.3. Metric Units

The measured results must be expressed using some standard dimension or units of measure. This column provides the units.

When a sample of singletons (see Section 11 of[RFC2330] for definitions of these terms) is collected, this entry will specify the units for each measured value.

7.4.4. Calibration

Some specifications for Methods of Measurement include the possibility to perform an error calibration. Section 3.7.3 of [RFC7679] is one example. In the registry entry, this field will identify a method of calibration for the metric, and when available, the measurement system SHOULD perform the calibration when requested and produce the output with an indication that it is the result of a calbration method. In-situ calibration could be enabled with an internal loopback that includes as much of the measurement system as possible, performs address manipulation as needed, and provides some form of isolation (e.g., deterministic delay) to avoid send-receive interface contention. Some portion of the random and systematic error can be characterized this way.

For one-way delay measurements, the error calibration must include an assessment of the internal clock synchronization with its external reference (this internal clock is supplying timestamps for measurement). In practice, the time offsets of clocks at both the source and destination are needed to estimate the systematic error due to imperfect clock synchronization (the time offsets are smoothed, thus the random variation is not usually represented in the results).

Both internal loopback calibration and clock synchronization can be used to estimate the *available accuracy* of the Output Metric Units. For example, repeated loopback delay measurements will reveal the portion of the Output result resolution which is the result of system noise, and thus inaccurate.

7.5. Administrative information

7.5.1. Status

The status of the specification of this Registered Performance Metric. Allowed values are 'current' and 'deprecated'. All newly defined Information Elements have 'current' status.

7.5.2. Requester

The requester for the Registered Performance Metric. The requester MAY be a document, such as RFC, or person.

7.5.3. Revision

The revision number of a Registered Performance Metric, starting at 0 for Registered Performance Metrics at time of definition and incremented by one for each revision.

7.5.4. Revision Date

The date of acceptance or the most recent revision for the Registered Performance Metric. The date SHALL be determined by the reviewing Performance Metrics Expert in the case of Expert Review, or by IANA in the case of Standards Action.

7.6. Comments and Remarks

Besides providing additional details which do not appear in other categories, this open Category (single column) allows for unforeseen issues to be addressed by simply updating this informational entry.

8. The Life-Cycle of Registered Performance Metrics

Once a Performance Metric or set of Performance Metrics has been identified for a given application, candidate Performance Metrics Registry entry specifications prepared in accordance with Section 7 should be submitted to IANA to follow the process for review by the Performance Metric Experts, as defined below. This process is also used for other changes to the Performance Metrics Registry, such as deprecation or revision, as described later in this section.

It is desirable that the author(s) of a candidate Performance Metrics Registry entry seek review in the relevant IETF working group, or offer the opportunity for review on the working group mailing list.

8.1. Adding new Performance Metrics to the Performance Metrics Registry

Requests to add Registered Performance Metrics in the Performance Metrics Registry SHALL be submitted to IANA, which forwards the request to a designated group of experts (Performance Metric Experts) appointed by the IESG; these are the reviewers called for by the Expert Review [RFC8126]policy defined for the Performance Metrics Registry. The Performance Metric Experts review the request for such things as compliance with this document, compliance with other applicable Performance Metric-related RFCs, and consistency with the currently defined set of Registered Performance Metrics.

Submission to IANA MAY be during IESG review (leading to IETF Standards Action), where an Internet Draft proposes one or more Registered Performance Metrics to be added to the Performance Metrics Registry, including the text of the proposed Registered Performance Metric(s).

Authors of proposed Registered Performance Metrics SHOULD review compliance with the specifications in this document to check their submissions before sending them to IANA.

At least one Performance Metric Expert should endeavor to complete referred reviews in a timely manner. If the request is acceptable, the Performance Metric Experts signify their approval to IANA, and IANA updates the Performance Metrics Registry. If the request is not acceptable, the Performance Metric Experts MAY coordinate with the requester to change the request to be compliant, otherwise IANA SHALL coordinate resolution of issues on behalf of the expert. The Performance Metric Experts MAY choose to reject clearly frivolous or inappropriate change requests outright, but such exceptional circumstances should be rare.

This process should not in any way be construed as allowing the Performance Metric Experts to overrule IETF consensus. Specifically, any Registered Performance Metrics that were added to the Performance Metrics Registry with IETF consensus require IETF consensus for revision or deprecation.

Decisions by the Performance Metric Experts may be appealed as in Section 7 of [RFC8126].

8.2. Revising Registered Performance Metrics

A request for Revision is only permitted when the requested changes maintain backward-compatibility with implementations of the prior Performance Metrics Registry entry describing a Registered Performance Metric (entries with lower revision numbers, but the same Identifier and Name).

The purpose of the Status field in the Performance Metrics Registry is to indicate whether the entry for a Registered Performance Metric is 'current' or 'deprecated'.

In addition, no policy is defined for revising the Performance Metric entries in the IANA Regsirty or addressing errors therein. To be clear, changes and deprecations within the Performance Metrics Registry are not encouraged, and should be avoided to the extent possible. However, in recognition that change is inevitable, the provisions of this section address the need for revisions.

Revisions are initiated by sending a candidate Registered Performance Metric definition to IANA, as in Section 8.1, identifying the existing Performance Metrics Registry entry, and explaining how and why the existing entry should be revised.

The primary requirement in the definition of procedures for managing changes to existing Registered Performance Metrics is avoidance of measurement interoperability problems; the Performance Metric Experts must work to maintain interoperability above all else. Changes to Registered Performance Metrics may only be done in an interoperable way; necessary changes that cannot be done in a way to allow interoperability with unchanged implementations MUST result in the creation of a new Registered Performance Metric (with a new Name, replacing the RFCXXXXsecY portion of the name) and possibly the deprecation of the earlier metric.

A change to a Registered Performance Metric SHALL be determined to be backward-compatible only when:

1. it involves the correction of an error that is obviously only editorial; or

2. it corrects an ambiguity in the Registered Performance Metric's definition, which itself leads to issues severe enough to prevent the Registered Performance Metric's usage as originally defined; or

3. it corrects missing information in the metric definition without changing its meaning (e.g., the explicit definition of 'quantity' semantics for numeric fields without a Data Type Semantics value); or

4. it harmonizes with an external reference that was itself corrected.

If a Performance Metric revision is deemed permissible and backward-compatible by the Performance Metric Experts, according to the rules in this document, IANA SHOULD execute the change(s) in the Performance Metrics Registry. The requester of the change is appended to the original requester in the Performance Metrics Registry. The Name of the revised Registered Performance Metric, including the RFCXXXXsecY portion of the name, SHALL remain unchamged (even when the change is the result of IETF Standards Action; the revised registry entry SHOULD reference the new RFC in an appropriate category and column).

Each Registered Performance Metric in the Performance Metrics Registry has a revision number, starting at zero. Each change to a Registered Performance Metric following this process increments the revision number by one.

When a revised Registered Performance Metric is accepted into the Performance Metrics Registry, the date of acceptance of the most recent revision is placed into the revision Date column of the registry for that Registered Performance Metric.

Where applicable, additions to Registered Performance Metrics in the form of text Comments or Remarks should include the date, but such additions may not constitute a revision according to this process.

Older version(s) of the updated metric entries are kept in the registry for archival purposes. The older entries are kept with all fields unmodified (version, revision date) except for the status field that SHALL be changed to "Deprecated".

8.3. Deprecating Registered Performance Metrics

Changes that are not permissible by the above criteria for Registered Performance Metric's revision may only be handled by deprecation. A Registered Performance Metric MAY be deprecated and replaced when:

1. the Registered Performance Metric definition has an error or shortcoming that cannot be permissibly changed as in Section 8.2 Revising Registered Performance Metrics; or

2. the deprecation harmonizes with an external reference that was itself deprecated through that reference's accepted deprecation method.

A request for deprecation is sent to IANA, which passes it to the Performance Metric Experts for review. When deprecating an Performance Metric, the Performance Metric description in the Performance Metrics Registry must be updated to explain the deprecation, as well as to refer to any new Performance Metrics created to replace the deprecated Performance Metric.

The revision number of a Registered Performance Metric is incremented upon deprecation, and the revision Date updated, as with any revision.

The use of deprecated Registered Performance Metrics should result in a log entry or human-readable warning by the respective application.

Names and Metric IDs of deprecated Registered Performance Metrics must not be reused.

The deprecated entries are kept with all fields unmodified, except the version, revision date, and the status field (changed to "Deprecated").

9. Security considerations

This draft defines a registry structure, and does not itself introduce any new security considerations for the Internet. The definition of Performance Metrics for this registry may introduce some security concerns, but the mandatory references should have their own considerations for secuity, and such definitions should be reviewed with security in mind if the security considerations are not covered by one or more reference standards.

10. IANA Considerations

With the background and processes described in earlier sections, this document requests the following IANA Actions. Note that mock-ups of the implementation of this set of requests have been prepared with IANA's help during development of this memo, and have been captured in the Proceedings of IPPM working group sessions.

10.1. Registry Group

The new registry group SHALL be named, "PERFORMANCE METRICS Group".

10.2. Performance Metric Name Elements

This document specifies the procedure for Performance Metrics Name Element Registry setup. IANA is requested to create a new set of registries for Performance Metric Name Elements called "Registered Performance Metric Name Elements". Each Registry, whose names are listed below:

MetricType:

Method:

SubTypeMethod:

Spec:

Units:

Output:

will contain the current set of possibilities for Performance Metrics Registry Entry Names.

To populate the Registered Performance Metric Name Elements at creation, the IANA is asked to use the lists of values for each name element listed in Section 7.1.2. The Name Elements in each registry are case-sensitive.

When preparing a Metric entry for Registration, the developer SHOULD choose Name elements from among the registered elements. However, if the proposed metric is unique in a significant way, it may be necessary to propose a new Name element to properly describe the metric, as described below.

A candidate Metric Entry RFC or document for Expert Review would propose one or more new element values required to describe the

unique entry, and the new name element(s) would be reviewed along with the metric entry. New assignments for Registered Performance Metric Name Elements will be administered by IANA through Expert Review [RFC8126], i.e., review by one of a group of experts, the Performance Metric Experts, who are appointed by the IESG upon recommendation of the Transport Area Directors.

10.3. New Performance Metrics Registry

This document specifies the procedure for Performance Metrics Registry setup. IANA is requested to create a new registry for Performance Metrics called "Performance Metrics Registry". This Registry will contain the following Summary columns:

Identifier:

Name:

URIs:

Description:

Reference:

Change Controller:

Version:

Descriptions of these columns and additional information found in the template for registry entries (categories and columns) are further defined in section Section 7.

The "Identifier" 0 should be Reserved. "The Identifier" values from 64512 to 65536 are reserved for private use.

Names starting with the prefix Priv_ are reserved for private use, and are not considered for registration. The "Name" column entries are further defined in section Section 7.

The "URIs" column will have a URL to the full template of each registry entry. The Registry Entry text SHALL be HTML-ized to aid the reader, with links to reference RFCs (similar to the way that Internet Drafts are HTML-ized, the same tool can perform the function).

The "Reference" column will include an RFC number, an approved specification designator from another standards body, or the contact person.

New assignments for Performance Metrics Registry will be administered by IANA through Expert Review [RFC8126], i.e., review by one of a group of experts, the Performance Metric Experts, who are appointed by the IESG upon recommendation of the Transport Area Directors, or by Standards Action. The experts can be initially drawn from the Working Group Chairs, document editors, and members of the Performance Metrics Directorate, among other sources of experts.

Extensions of the Performance Metrics Registry require IETF Standards Action. Only one form of registry extension is envisaged:

1. Adding columns, or both categories and columns, to accommodate unanticipated aspects of new measurements and metric categories.

If the Performance Metrics Registry is extended in this way, the Version number of future entries complying with the extension SHALL be incremented (either in the unit or tenths digit, depending on the degree of extension.

11. Acknowledgments

Thanks to Brian Trammell and Bill Cerveny, IPPM chairs, for leading some brainstorming sessions on this topic. Thanks to Barbara Stark and Juergen Schoenwaelder for the detailed feedback and suggestions. Thanks to Andrew McGregor for suggestions on metric naming. Thanks to Michelle Cotton for her early IANA review, and to Amanda Barber for answering questions related to the presentation of the registry and accessibility of the complete template via URL.

12. References

12.1. Normative References

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, DOI 10.17487/RFC2026, October 1996, <https://www.rfc-editor.org/info/rfc2026>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2330] Paxson, V., Almes, G., Mahdavi, J., and M. Mathis, "Framework for IP Performance Metrics", RFC 2330, DOI 10.17487/RFC2330, May 1998, <https://www.rfc-editor.org/info/rfc2330>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC6390] Clark, A. and B. Claise, "Guidelines for Considering New Performance Metric Development", BCP 170, RFC 6390, DOI 10.17487/RFC6390, October 2011, <https://www.rfc-editor.org/info/rfc6390>.

[RFC6576] Geib, R., Ed., Morton, A., Fardid, R., and A. Steinmitz, "IP Performance Metrics (IPPM) Standard Advancement Testing", BCP 176, RFC 6576, DOI 10.17487/RFC6576, March 2012, <https://www.rfc-editor.org/info/rfc6576>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

12.2. Informative References

[I-D.ietf-ippm-initial-registry] Morton, A., Bagnulo, M., Eardley, P., and K. D'Souza, "Initial Performance Metrics Registry Entries", draft-ietf-ippm-initial-registry-11 (work in progress), March 2019.

[RFC2681] Almes, G., Kalidindi, S., and M. Zekauskas, "A Round-trip Delay Metric for IPPM", RFC 2681, DOI 10.17487/RFC2681, September 1999, <https://www.rfc-editor.org/info/rfc2681>.

[RFC3432] Raisanen, V., Grotefeld, G., and A. Morton, "Network performance measurement with periodic streams", RFC 3432, DOI 10.17487/RFC3432, November 2002, <https://www.rfc-editor.org/info/rfc3432>.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <https://www.rfc-editor.org/info/rfc3550>.

[RFC3611] Friedman, T., Ed., Caceres, R., Ed., and A. Clark, Ed., "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, DOI 10.17487/RFC3611, November 2003, <https://www.rfc-editor.org/info/rfc3611>.

[RFC4148] Stephan, E., "IP Performance Metrics (IPPM) Metrics Registry", BCP 108, RFC 4148, DOI 10.17487/RFC4148, August 2005, <https://www.rfc-editor.org/info/rfc4148>.

[RFC5474] Duffield, N., Ed., Chiou, D., Claise, B., Greenberg, A., Grossglauser, M., and J. Rexford, "A Framework for Packet Selection and Reporting", RFC 5474, DOI 10.17487/RFC5474, March 2009, <https://www.rfc-editor.org/info/rfc5474>.

[RFC5475] Zseby, T., Molina, M., Duffield, N., Niccolini, S., and F. Raspall, "Sampling and Filtering Techniques for IP Packet Selection", RFC 5475, DOI 10.17487/RFC5475, March 2009, <https://www.rfc-editor.org/info/rfc5475>.

[RFC5477] Dietz, T., Claise, B., Aitken, P., Dressler, F., and G. Carle, "Information Model for Packet Sampling Exports", RFC 5477, DOI 10.17487/RFC5477, March 2009, <https://www.rfc-editor.org/info/rfc5477>.

[RFC6035] Pendleton, A., Clark, A., Johnston, A., and H. Sinnreich, "Session Initiation Protocol Event Package for Voice Quality Reporting", RFC 6035, DOI 10.17487/RFC6035, November 2010, <https://www.rfc-editor.org/info/rfc6035>.

[RFC6248] Morton, A., "RFC 4148 and the IP Performance Metrics (IPPM) Registry of Metrics Are Obsolete", RFC 6248, DOI 10.17487/RFC6248, April 2011, <https://www.rfc-editor.org/info/rfc6248>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7012] Claise, B., Ed. and B. Trammell, Ed., "Information Model for IP Flow Information Export (IPFIX)", RFC 7012, DOI 10.17487/RFC7012, September 2013, <https://www.rfc-editor.org/info/rfc7012>.

[RFC7014] D'Antonio, S., Zseby, T., Henke, C., and L. Peluso, "Flow Selection Techniques", RFC 7014, DOI 10.17487/RFC7014, September 2013, <https://www.rfc-editor.org/info/rfc7014>.

[RFC7594] Eardley, P., Morton, A., Bagnulo, M., Burbridge, T., Aitken, P., and A. Akhter, "A Framework for Large-Scale Measurement of Broadband Performance (LMAP)", RFC 7594, DOI 10.17487/RFC7594, September 2015, <https://www.rfc-editor.org/info/rfc7594>.

[RFC7679] Almes, G., Kalidindi, S., Zekauskas, M., and A. Morton, Ed., "A One-Way Delay Metric for IP Performance Metrics (IPPM)", STD 81, RFC 7679, DOI 10.17487/RFC7679, January 2016, <https://www.rfc-editor.org/info/rfc7679>.

[RFC7799] Morton, A., "Active and Passive Metrics and Methods (with Hybrid Types In-Between)", RFC 7799, DOI 10.17487/RFC7799, May 2016, <https://www.rfc-editor.org/info/rfc7799>.

[RFC8194] Schoenwaelder, J. and V. Bajpai, "A YANG Data Model for LMAP Measurement Agents", RFC 8194, DOI 10.17487/RFC8194, August 2017, <https://www.rfc-editor.org/info/rfc8194>.

Authors' Addresses

Marcelo Bagnulo
Universidad Carlos III de Madrid
Av. Universidad 30
Leganes, Madrid 28911
SPAIN

Phone: 34 91 6249500
Email: marcelo@it.uc3m.es
URI: http://www.it.uc3m.es

Benoit Claise
Cisco Systems, Inc.
De Kleetlaan 6a b1
1831 Diegem
Belgium

Email: bclaise@cisco.com

Philip Eardley
BT
Adastral Park, Martlesham Heath
Ipswich
ENGLAND

Email: philip.eardley@bt.com

Al Morton
AT&T Labs
200 Laurel Avenue South
Middletown, NJ
USA

Email: acmorton@att.com

Aamer Akhter
Consultant
118 Timber Hitch
Cary, NC
USA

Email: aakhter@gmail.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 23, 2017

T. Burbridge
P. Eardley
BT
M. Bagnulo
Universidad Carlos III de Madrid
J. Schoenwaelder
Jacobs University Bremen

April 21, 2017

Information Model for Large-Scale Measurement Platforms (LMAP)
draft-ietf-lmap-information-model-18

Abstract

This Information Model applies to the Measurement Agent within a Large-Scale Measurement Platform. As such it outlines the information that is (pre-)configured on the Measurement Agent or exists in communications with a Controller or Collector within an LMAP framework. The purpose of such an Information Model is to provide a protocol and device independent view of the Measurement Agent that can be implemented via one or more Control and Report protocols.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

Table of Contents

1. Introduction

A large-scale measurement platform is a collection of components that work in a coordinated fashion to perform measurements from a large number of vantage points. A typical use case is the execution of broadband measurements [RFC7536]. The main components of a large-scale measurement platform are the Measurement Agents (hereafter MAs), the Controller(s) and the Collector(s).

The MAs are the elements actually performing the measurements. The MAs are controlled by exactly one Controller at a time and the

Collectors gather the results generated by the MAs. In a nutshell, the normal operation of a large-scale measurement platform starts with the Controller instructing a set of one or more MAs to perform a set of one or more Measurement Tasks at a certain point in time. The MAs execute the instructions from a Controller, and once they have done so, they report the results of the measurements to one or more Collectors. The overall framework for a large-scale measurement platform as used in this document is described in detail in [RFC7594].

A large-scale measurement platform involves basically three types of protocols, namely, a Control protocol (or protocols) between a Controller and the MAs, a Report protocol (or protocols) between the MAs and the Collector(s) and several measurement protocols between the MAs and Measurement Peers (MPs), used to actually perform the measurements. In addition some information is required to be configured on the MA prior to any communication with a Controller.

This document defines the information model for both Control and the Report protocols along with pre-configuration information that is required on the MA before communicating with the Controller, broadly named as the LMAP Information Model. The measurement protocols are out of the scope of this document.

As defined in [RFC3444], the LMAP Information Model defines the concepts involved in a large-scale measurement platform at a high level of abstraction, independent of any specific implementation or actual protocol used to exchange the information. It is expected that the proposed information model can be used with different protocols in different measurement platform architectures and across different types of MA devices (e.g., home gateway, smartphone, PC, router). A YANG data model implementing the information model can be found in [I-D.ietf-lmap-yang].

The definition of an Information Model serves a number of purposes:

1. To guide the standardisation of one or more Control and Report protocols and data models

2. To enable high-level inter-operability between different Control and Report protocols by facilitating translation between their respective data models such that a Controller could instruct sub-populations of MAs using different protocols

3. To form agreement of what information needs to be held by an MA and passed over the Control and Report interfaces and support the functionality described in the LMAP framework

4. To enable existing protocols and data models to be assessed for their suitability as part of a large-scale measurement system

2. Notation

This document uses a programming language-like notation to define the properties of the objects of the information model. An optional property is enclosed by square brackets, [ ], and a list property is indicated by two numbers in angle brackets, <m..n>, where m indicates the minimal number of values, and n is the maximum. The symbol * for n means no upper bound.

The object definitions use a couple of base types that are defined as follows:

int A type representing signed or unsigned integer numbers. This information model does not define a precision nor does it make a distinction between signed and unsigned number ranges. This type is also used to represent enumerations.

boolean A type representing a boolean value.

string A type representing a human-readable string consisting of a (possibly restricted) subset of Unicode and ISO/IEC 10646 [ISO.10646] characters.

datetime A type representing a date and time using the Gregorian calendar. The datetime format MUST conform to RFC 3339 [RFC3339].

uuid A type representing Universally Unique IDentifier (UUID(as defined in RFC 4122 [RFC4122]. The UUID values are expected to be unique within an installation of a large-scale measurement system.

uri A type representing a Uniform Resource Identifier as defined in STD 66 [RFC3986].

ip-address A type representing an IP address. This type supports both IPv4 and IPv6 addresses.

counter A non-negative integer that monotonically increases. Counters may have discontinuities and they are not expected to persist across restarts.

credentials An opaque type representing credentials needed by a cryptographic mechanism to secure communication. Data

models must expand this opaque type as needed and required by the security protocols utilized.

data An opaque type representing data obtained from measurements.

Names of objects are generally assumed to be unique within an implementation.

3. LMAP Information Model

The information described herein relates to the information stored, received or transmitted by a Measurement Agent as described within the LMAP framework [RFC7594]. As such, some subsets of this information model are applicable to the measurement Controller, Collector and any device management system that pre-configures the Measurement Agent. The information described in these models will be transmitted by protocols using interfaces between the Measurement Agent and such systems according to a Data Model.

The information model is divided into six aspects. Firstly the grouping of information facilitates reader understanding. Secondly, the particular groupings chosen are expected to map to different protocols or different transmissions within those protocols.

1. Pre-Configuration Information. Information pre-configured on the Measurement Agent prior to any communication with other components of the LMAP architecture (i.e., the Controller, Collector and Measurement Peers), specifically detailing how to communicate with a Controller and whether the device is enabled to participate as an MA.

2. Configuration Information. Update of the pre-configuration information during the registration of the MA or subsequent communication with the Controller, along with the configuration of further parameters about the MA (rather than the Measurement Tasks it should perform) that were not mandatory for the initial communication between the MA and a Controller.

3. Instruction Information. Information that is received by the MA from the Controller pertaining to the Measurement Tasks that should be executed. This includes the task execution Schedules (other than the Controller communication Schedule supplied as (pre)configuration information) and related information such as the Task Configuration, communication Channels to Collectors and schedule Event and Timing information. It also includes Task Suppression information that is used to over-ride normal Task execution.

4. Logging Information. Information transmitted from the MA to the Controller detailing the results of any configuration operations along with error and status information from the operation of the MA.

5. Capability and Status Information. Information on the general status and capabilities of the MA. For example, the set of measurements that are supported on the device.

6. Reporting Information. Information transmitted from the MA to one or more Collectors including measurement results and the context in which they were conducted.

In addition the MA may hold further information not described herein, and which may be optionally transferred to or from other systems including the Controller and Collector. One example of information in this category is subscriber or line information that may be extracted by a task and reported by the MA in the reporting communication to a Collector.

It should also be noted that the MA may be in communication with other management systems which may be responsible for configuring and retrieving information from the MA device. Such systems, where available, can perform an important role in transferring the pre-configuration information to the MA or enabling/disabling the measurement functionality of the MA.

The granularity of data transmitted in each operation of the Control and Report Protocols is not dictated by the Information Model. For example, the Instruction object may be delivered in a single operation. Alternatively, Schedules and Task Configurations may be separated or even each Schedule/Task Configuration may be delivered individually. Similarly the Information Model does not dictate whether data is read, write, or read/write. For example, some Control Protocols may have the ability to read back Configuration and Instruction information which have been previously set on the MA. Lastly, while some protocols may simply overwrite information (for example refreshing the entire Instruction Information), other protocols may have the ability to update or delete selected items of information.

The information modeled by the six aspects of the information model is supported by a number of common information objects. These objects are also described later in this document and comprise of:

a. Schedules. A set of Schedules tells the MA to execute Actions. An Action of a Schedule leads to the execution of a Task. Without a Schedule no Task (including measurements or reporting

or communicating with the Controller) is ever executed. Schedules are used within the Instruction to specify what tasks should be performed, when, and how to direct their results. A Schedule is also used within the pre-Configuration and Configuration information in order to execute the Task or Tasks required to communicate with the Controller. A specific Schedule can only be active once. Attempts to start a Schedule while the same Schedule is still running will fail.

b. Channels. A set of Channel objects are used to communicate with a number of endpoints (i.e., the Controller and Collectors). Each Channel object contains the information required for the communication with a single endpoint such as the target location and security details.

c. Task Configurations. A set of Task Configurations is used to configure the Tasks that are run by the MA. This includes the registry entries for the Task and any configuration parameters, represented as Task Options. Task Configurations are referenced from a Schedule in order to specify what Tasks the MA should execute.

d. Events. A set of Event objects that can be referenced from the Schedules. Each Schedule always references exactly one Event object that determines when the schedule is executed. An Event object specifies either a singleton or series of events that indicate when Tasks should be executed. A commonly used kind of Event objects are Timing objects. For Event objects specifying a series of events, it is generally a good idea to configure an end time and to refresh the end time as needed to ensure that MAs that loose connectivity to their controller do not continue executing Schedules forever.

Figure 1 illustrates the structure in which these common information objects are referenced. The references are achieved by each object (Task Configuration, Event) being given a short textual name that is used by other objects. The objects shown in parenthesis are part of the internal object structure of a Schedule. Channels are not shown in the diagram since they are only used as an option by selected Task Configurations but are similarly referenced using a short text name.

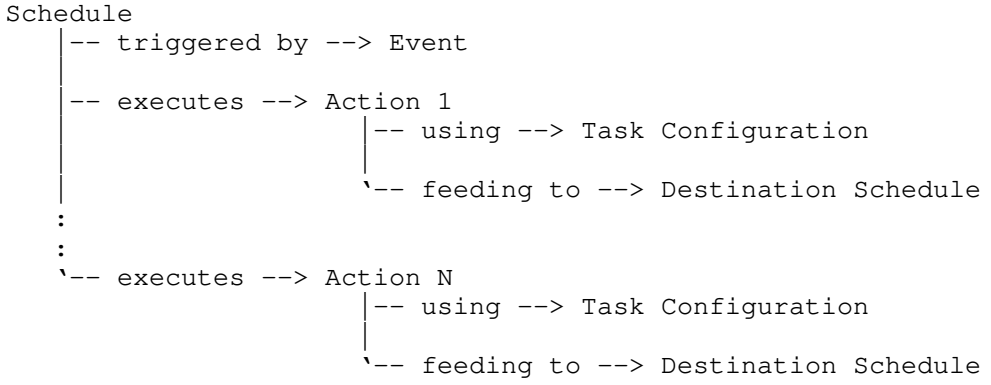

Figure 1: Relationship between Schedules, Events, Actions, Task Configurations, and Destination Schedules

The primary function of an MA is to execute Schedules. A Schedule, which is triggered by an Event, executes a number of Actions. An Action refers to a Configured Task and it may feed results to a Destination Schedule. Both, Actions and Configured Tasks can provide parameters, represented as Action Options and Task Options.

Tasks can implement a variety of different functions. While in terms of the Information Model, all Tasks have the same structure, it can help conceptually to think of different Task categories:

1. Measurement Tasks measure some aspect of network performance or traffic. They may also capture contextual information from the MA device or network interfaces such as the device type or interface speed.

2. Data Transfer Tasks support the communication with a Controller and Collectors:

   A. Reporting Tasks report the results of Measurement Tasks to Collectors

   B. Control Task(s) implement the Control Protocol and communicate with the Controller.

3. Data Analysis Tasks can exist to analyse data from other Measurement Tasks locally on the MA

4. Data Management Tasks may exist to clean-up, filter or compress data on the MA such as Measurement Task results

Figure 1 indicates that Actions can produce data that is fed into Destination Schedules. This can by used by Actions implementing Measurement Tasks to feed measurement results to a Schedule that triggers Actions implementing Reporting Tasks. Data fed to a Destination Schedule is consumed by the first Action of the Destination Schedule if the Destination Schedule is using sequential or pipelined execution mode and it is consumed by all Actions of the Destination Schedule if the Destination Schedule is using parallel execution mode.

3.1. Pre-Configuration Information

This information is the minimal information that needs to be pre-configured to the MA in order for it to successfully communicate with a Controller during the registration process. Some of the Pre-Configuration Information elements are repeated in the Configuration Information in order to allow an LMAP Controller to update these items. The pre-configuration information also contains some elements that are not under the control of the LMAP framework (such as the device identifier and device security credentials).

This Pre-Configuration Information needs to include a URL of the initial Controller from where configuration information can be communicated along with the security information required for the communication including the certificate of the Controller (or the certificate of the Certification Authority which was used to issue the certificate for the Controller). All this is expressed as a Channel. While multiple Channels may be provided in the Pre-Configuration Information they must all be associated with a single Controller (e.g., over different interfaces or network protocols).

Where the MA pulls information from the Controller, the Pre-Configuration Information also needs to contain the timing of the communication with the Controller as well as the nature of the communication itself (such as the protocol and data to be transferred). The timing is represented as an Event that invokes a Schedule that executes the Task(s) responsible for communication with the Controller. It is this Task (or Tasks) that implement the Control protocol between the MA and the Controller and utilises the Channel information. The Task(s) may take additional parameters, as defined by a Task Configuration.

Even where information is pushed to the MA from the Controller (rather than pulled by the MA), a Schedule still needs to be supplied. In this case the Schedule will simply execute a Controller listener Task when the MA is started. A Channel is still required for the MA to establish secure communication with the Controller.

It can be seen that these Channels, Schedules and Task Configurations for the initial MA-Controller communication are no different in terms of the Information Model to any other Channel, Schedule or Task Configuration that might execute a Measurement Task or report the measurement results (as described later).

The MA may be pre-configured with an MA ID, or may use a Device ID in the first Controller contact before it is assigned an MA ID. The Device ID may be a MAC address or some other device identifier expressed as a URI. If the MA ID is not provided at this stage, then it must be provided by the Controller during Configuration.

3.1.1. Definition of ma-preconfig-obj

```
object {
    [uuid                ma-preconfig-agent-id;]
     ma-task-obj         ma-preconfig-control-tasks<1..*>;
     ma-channel-obj      ma-preconfig-control-channels<1..*>;
     ma-schedule-obj     ma-preconfig-control-schedules<1..*>;
    [uri                 ma-preconfig-device-id;]
     credentials         ma-preconfig-credentials;
} ma-preconfig-obj;
```

The ma-preconfig-obj describes information that needs to be available to the MA in order to bootstrap communication with a Controller. The ma-preconfig-obj consists of the following elements:

ma-preconfig-agent-id: An optional uuid uniquely identifying the measurement agent.

ma-preconfig-control-tasks: An unordered set of task objects.

ma-preconfig-control-channels: An unordered set of channel objects.

ma-preconfig-control-schedules: An unordered set of scheduling objects.

ma-preconfig-device-id: An optional identifier for the device.

ma-preconfig-credentials: The security credentials used by the measurement agent.

3.2. Configuration Information

During registration or at any later point at which the MA contacts the Controller (or vice-versa), the choice of Controller, details for the timing of communication with the Controller or parameters for the

communication Task(s) can be changed (as captured by the Channels, Schedules and Task Configurations objects). For example the pre-configured Controller (specified as a Channel or Channels) may be over-ridden with a specific Controller that is more appropriate to the MA device type, location or characteristics of the network (e.g., access technology type or broadband product). The initial communication Schedule may be over-ridden with one more relevant to routine communications between the MA and the Controller.

While some Control protocols may only use a single Schedule, other protocols may use several Schedules (and related data transfer Tasks) to update the Configuration Information, transfer the Instruction Information, transfer Capability and Status Information and send other information to the Controller such as log or error notifications. Multiple Channels may be used to communicate with the same Controller over multiple interfaces (e.g., to send logging information over a different network).

In addition the MA will be given further items of information that relate specifically to the MA rather than the measurements it is to conduct or how to report results. The assignment of an ID to the MA is mandatory. If the MA Agent ID was not optionally provided during the pre-configuration then one must be provided by the Controller during Configuration. Optionally a Group ID may also be given which identifies a group of interest to which that MA belongs. For example the group could represent an ISP, broadband product, technology, market classification, geographic region, or a combination of multiple such characteristics. Additional flags control whether the MA ID or the Group ID are included in Reports. The reporting of a Group ID without the MA ID may allow the MA to remain anonymous, which may be particularly useful to prevent tracking of mobile MA devices.

Optionally an MA can also be configured to stop executing any Instruction Schedule if the Controller is unreachable. This can be used as a fail-safe to stop Measurement and other Tasks being conducted when there is doubt that the Instruction Information is still valid. This is simply represented as a time window in seconds since the last communication with the Controller after which an Event is generated that can trigger the suspension of Instruction Schedules. The appropriate value of the time window will depend on the specified communication Schedule with the Controller and the duration for which the system is willing to tolerate continued operation with potentially stale Instruction Information.

While Pre-Configuration Information is persistent upon device reset or power cycle, the persistency of the Configuration Information may be device dependent. Some devices may revert back to their pre-

configuration state upon reboot or factory reset, while other devices may store all Configuration and Instruction information in persistent storage. A Controller can check whether an MA has the latest Configuration and Instruction information by examining the Capability and Status information for the MA.

3.2.1. Definition of ma-config-obj

```
object {
    uuid                ma-config-agent-id;
    ma-task-obj         ma-config-control-tasks<1..*>;
    ma-channel-obj      ma-config-control-channels<1..*>;
    ma-schedule-obj     ma-config-control-schedules<1..*>;
    credentials         ma-config-credentials;
   [string              ma-config-group-id;]
   [string              ma-config-measurement-point;]
   [boolean             ma-config-report-agent-id;]
   [boolean             ma-config-report-group-id;]
   [boolean             ma-config-report-measurement-point;]
   [int                 ma-config-controller-timeout;]
} ma-config-obj;
```

The ma-config-obj consists of the following elements:

ma-config-agent-id: A uuid uniquely identifying the measurement agent.

ma-config-control-tasks: An unordered set of task objects.

ma-config-control-channels: An unordered set of channel objects.

ma-config-control-schedules: An unordered set of scheduling objects.

ma-config-credentials: The security credentials used by the measurement agent.

ma-config-group-id: An optional identifier of the group of measurement agents this measurement agent belongs to.

ma-config-measurement-point: An optional identifier for the measurement point indicating where the measurement agent is located on a path (see [RFC7398] for further details).

ma-config-report-agent-id:	An optional flag indicating whether the agent identifier (ma-config-agent-id) is included in reports. The default value is true.
ma-config-report-group-id:	An optional flag indicating whether the group identifier (ma-config-group-id) is included in reports. The default value is false.
ma-config-report-measurement-point:	An optional flag indicating whether the measurement point (ma-config-measurement-point) should be included in reports. The default value is false.
ma-config-controller-timeout:	A timer is started after each successful contact with a controller. When the timer reaches the controller-timeout (measured in seconds), an event is raised indicating that connectivity to the controller has been lost (see ma-controller-lost-obj).

3.3. Instruction Information

The Instruction information model has four sub-elements:

1. Instruction Task Configurations
2. Report Channels
3. Instruction Schedules
4. Suppression

The Instruction supports the execution of all Tasks on the MA except those that deal with communication with the Controller (specified in (pre-)configuration information). The Tasks are configured in Instruction Task Configurations and included by reference in the Actions of Instruction Schedules that specify when to execute them. The results can be communicated to other Schedules or a Task may implement a Reporting Protocol and communicate results over Report Channels. Suppression is used to temporarily stop the execution of

new Tasks as specified by the Instruction Schedules (and optionally to stop ongoing Tasks).

A Task Configuration is used to configure the mandatory and optional parameters of a Task. It also serves to instruct the MA about the Task including the ability to resolve the Task to an executable and specifying the schema for the Task parameters.

A Report Channel defines how to communicate with a single remote system specified by a URL. A Report Channel is used to send results to a single Collector but is no different in terms of the Information Model to the Control Channel used to transfer information between the MA and the Controller. Several Report Channels can be defined to enable results to be split or duplicated across different destinations. A single Channel can be used by multiple (reporting) Task Configurations to transfer data to the same Collector. A single Reporting Task Configuration can also be included in multiple Schedules. E.g., a single Collector may receive data at three different cycle rates, one Schedule reporting hourly, another reporting daily and a third specifying that results should be sent immediately for on-demand measurement tasks. Alternatively multiple Report Channels can be used to send Measurement Task results to different Collectors. The details of the Channel element is described later as it is common to several objects.

Instruction Schedules specify which Actions to execute according to a given triggering Event. An Action extends a Configured Task with additional specific parameters. An Event can trigger the execution of a single Action or it can trigger a repeated series of Actions. The Schedule also specifies how to link Tasks output data to other Schedules.

Measurement Suppression information is used to over-ride the Instruction Schedule and temporarily stop measurements or other Tasks from running on the MA for a defined or indefinite period. While conceptually measurements can be stopped by simply removing them from the Measurement Schedule, splitting out separate information on Measurement Suppression allows this information to be updated on the MA on a different timing cycle or protocol implementation to the Measurement Schedule. It is also considered that it will be easier for a human operator to implement a temporary explicit suppression rather than having to move to a reduced Schedule and then roll-back at a later time.

It should be noted that control schedules and tasks cannot be suppressed as evidenced by the lack of suppression information in the Configuration. The control schedule must only reference tasks listed as control tasks (i.e., within the Configuration information).

A single Suppression object is able to enable/disable a set of Instruction Tasks that are tagged for suppression. This enables fine grained control on which Tasks are suppressed. Suppression of both matching Actions and Measurement Schedules is supported. Support for disabling specific Actions allows malfunctioning or mis-configured Tasks or Actions that have an impact on a particular part of the network infrastructure (e.g., a particular Measurement Peer) to be targeted. Support for disabling specific Schedules allows for particularly heavy cycles or sets of less essential Measurement Tasks to be suppressed quickly and effectively. Note that Suppression has no effect on either Controller Tasks or Controller Schedules.

Suppression stops new Tasks from executing. In addition, the Suppression information also supports an additional Boolean that is used to select whether on-going tasks are also to be terminated.

Unsuppression is achieved through either overwriting the Measurement Suppression information (e.g., changing 'enabled' to False) or through the use of an End time such that the Measurement Suppression will no longer be in effect beyond this time.

The goal when defining these four different elements is to allow each part of the information model to change without affecting the other three elements. For example it is envisaged that the Report Channels and the set of Task Configurations will be relatively static. The Instruction Schedule, on the other hand, is likely to be more dynamic, as the measurement panel and test frequency are changed for various business goals. Another example is that measurements can be suppressed with a Suppression command without removing the existing Instruction Schedules that would continue to apply after the Suppression expires or is removed. In terms of the Controller-MA communication this can reduce the data overhead. It also encourages the re-use of the same standard Task Configurations and Reporting Channels to help ensure consistency and reduce errors.

3.3.1. Definition of ma-instruction-obj

```
object {
    ma-task-obj         ma-instruction-tasks<0..*>;
    ma-channel-obj      ma-instruction-channels<0..*>;
    ma-schedule-obj     ma-instruction-schedules<0..*>;
   [ma-suppression-obj  ma-instruction-suppressions<0..*>;]
} ma-instruction-obj;
```

An ma-instruction-obj consists of the following elements:

ma-instruction-tasks: A possibly empty unordered set of task objects.

ma-instruction-channels: A possibly empty unordered set of channel objects.

ma-instruction-schedules: A possibly empty unordered set of schedule objects.

ma-instruction-suppressions: An optional possibly empty unordered set of suppression objects.

3.3.2. Definition of ma-suppression-obj

```
object {
    string              ma-suppression-name;
   [ma-event-obj        ma-suppression-start;]
   [ma-event-obj        ma-suppression-end;]
   [string              ma-suppression-match<0..*>;]
   [boolean             ma-suppression-stop-running;]
} ma-suppression-obj;
```

The ma-suppression-obj controls the suppression of schedules or actions and consists of the following elements:

ma-suppression-name: A name uniquely identifying a suppression.

ma-suppression-start: The optional event indicating when suppression starts. If not present, the suppression starts immediately, i.e., as if the value would be 'immediate'.

ma-suppression-end: The optional event indicating when suppression ends. If not present, the suppression does not have a defined end, i.e., the suppression remains for an indefinite period of time.

ma-suppression-match: An optional and possibly empty unordered set of match patterns. The suppression will apply to all schedules (and their actions) that have a matching value in their ma-schedule-suppression-tags and all actions that have a matching value in their ma-action-suppression-tags. Pattern matching is done using glob style pattern (see below).

ma-suppression-stop-running: An optional boolean indicating whether suppression will stop any running matching schedules or actions. The default value for this boolean is false.

Glob style pattern matching is following POSIX.2 fnmatch() [POSIX.2] without special treatment of file paths:

*	matches a sequence of characters
?	matches a single character
[seq]	matches any character in seq
[!seq]	matches any character not in seq

A backslash followed by a character matches the following character. In particular:

*	matches *
\?	matches ?
\\	matches \

A sequence seq may be a sequence of characters (e.g., [abc] or a range of characters (e.g., [a-c]).

3.4. Logging Information

The MA may report on the success or failure of Configuration or Instruction communications from the Controller. In addition further operational logs may be produced during the operation of the MA and updates to capabilities may also be reported. Reporting this information is achieved in exactly the same manner as scheduling any other Task. We make no distinction between a Measurement Task conducting an active or passive network measurement and one which solely retrieves static or dynamic information from the MA such as capabilities or logging information. One or more logging tasks can be programmed or configured to capture subsets of the Logging Information. These logging tasks are then executed by Schedules which also specify that the resultant data is to be transferred over the Controller Channels.

The type of Logging Information will fall into three different categories:

1. Success/failure/warning messages in response to information updates from the Controller. Failure messages could be produced due to some inability to receive or parse the Controller communication, or if the MA is not able to act as instructed. For example:

* "Measurement Schedules updated OK"

* "Unable to parse JSON"

* "Missing mandatory element: Measurement Timing"

* "'Start' does not conform to schema - expected datetime"

* "Date specified is in the past"

* "'Hour' must be in the range 1..24"

* "Schedule A refers to non-existent Measurement Task Configuration"

* "Measurement Task Configuration X registry entry Y not found"

* "Updated Measurement Task Configurations do not include M used by Measurement Schedule N"

2. Operational updates from the MA. For example:

* "Out of memory: cannot record result"

* "Collector 'collector.example.com' not responding"

* "Unexpected restart"

* "Suppression timeout"

* "Failed to execute Measurement Task Configuration H"

3. Status updates from the MA. For example:

* "Device interface added: eth3"

* "Supported measurements updated"

* "New IP address on eth0: xxx.xxx.xxx.xxx"

This Information Model document does not detail the precise format of logging information since it is to a large extent protocol and MA specific. However, some common information can be identified.

3.4.1. Definition of ma-log-obj

```
object {
    uuid                ma-log-agent-id;
    datetime            ma-log-event-time;
    int                 ma-log-code;
    string              ma-log-description;
} ma-log-obj;
```

The ma-log-obj models the generic aspects of a logging object and consists of the following elements:

ma-log-agent-id: A uuid uniquely identifying the measurement agent.

ma-log-event-time: The date and time of the event reported in the logging object.

ma-log-code: A machine readable code describing the event.

ma-log-description: A human readable description of the event.

3.5. Capability and Status Information

The MA will hold Capability Information that can be retrieved by a Controller. Capabilities include the device interface details available to Measurement Tasks as well as the set of Measurement Tasks/Roles (specified by registry entries) that are actually installed or available on the MA. Status information includes the times that operations were last performed such as contacting the Controller or producing Reports.

3.5.1. Definition of ma-capability-obj

```
object {
    string                  ma-capability-hardware;
    string                  ma-capability-firmware;
    string                  ma-capability-version;
   [string                  ma-capability-tags<0..*>;]
   [ma-capability-task-obj  ma-capability-tasks<0..*>;]
} ma-capability-obj;
```

The ma-capability-obj provides information about the capabilities of the measurement agent and consists of the following elements:

ma-capability-hardware: A description of the hardware of the device the measurement agent is running on.

ma-capability-firmware: A description of the firmware of the device the measurement agent is running on.

ma-capability-version: The version of the measurement agent.

ma-capability-tags: An optional unordered set of tags that provide additional information about the capabilities of the measurement agent.

ma-capability-tasks: An optional unordered set of capability objects for each supported task.

3.5.2. Definition of ma-capability-task-obj

```
object {
    string              ma-capability-task-name;
    ma-registry-obj     ma-capability-task-functions<0..*>;
    string              ma-capability-task-version;
} ma-capability-task-obj;
```

The ma-capability-task-obj provides information about the capability of a task and consists of the following elements:

ma-capability-task-name: A name uniquely identifying a task.

ma-capability-task-functions: A possibly empty unordered set of registry entries identifying functions this task implements.

ma-capability-task-version: The version of the measurement task.

3.5.3. Definition of ma-status-obj

```
object {
    uuid                        ma-status-agent-id;
   [uri                         ma-status-device-id;]
    datetime                    ma-status-last-started;
    ma-status-interface-obj     ma-status-interfaces<0..*>;
   [ma-status-schedule-obj      ma-status-schedules<0..*>;]
   [ma-status-suppression-obj   ma-status-suppressions<0..*>;]
} ma-status-obj;
```

The ma-status-obj provides status information about the measurement agent and consists of the following elements:

ma-status-agent-id: A uuid uniquely identifying the measurement agent.

ma-status-device-id: A URI identifying the device.

ma-status-last-started: The date and time the measurement agent last started.

ma-status-interfaces: An unordered set of network interfaces available on the device.

ma-status-schedules: An optional unordered set of status objects for each schedule.

ma-status-suppressions: An optional unordered set of status objects for each suppression.

3.5.4. Definition of ma-status-schedule-obj

```
object {
    string                  ma-status-schedule-name;
    string                  ma-status-schedule-state;
    int                     ma-status-schedule-storage;
    counter                 ma-status-schedule-invocations;
    counter                 ma-status-schedule-suppressions;
    counter                 ma-status-schedule-overlaps;
    counter                 ma-status-schedule-failures;
    datetime                ma-status-schedule-last-invocation;
   [ma-status-action-obj    ma-status-schedule-actions<0..*>;]
} ma-status-schedule-obj;
```

The ma-status-schedule-obj provides status information about the status of a schedule and consists of the following elements:

ma-status-schedule-name: The name of the schedule this status object refers to.

ma-status-schedule-state: The state of the schedule. The value 'enabled' indicates that the schedule is currently enabled. The value 'suppressed' indicates that the schedule is currently suppressed. The value 'disabled' indicates that the schedule is currently disabled. The value 'running' indicates that the schedule is currently running.

ma-status-schedule-storage: The amount of secondary storage (e.g., allocated in a file

	system) holding temporary data allocated to the schedule in bytes. This object reports the amount of allocated physical storage and not the storage used by logical data records. Data models should use a 64-bit integer type.
ma-status-schedule-invocations	Number of invocations of this schedule. This counter does not include suppressed invocations or invocations that were prevented due to an overlap with a previous invocation of this schedule.
ma-status-schedule-suppressions	Number of suppressed executions of this schedule.
ma-status-schedule-overlaps	Number of executions prevented due to overlaps with a previous invocation of this schedule.
ma-status-schedule-failures	Number of failed executions of this schedule. A failed execution is an execution where at least one action failed.
ma-status-schedule-last-invocation:	The date and time of the last invocation of this schedule.
ma-status-schedule-actions:	An optional ordered list of status objects for each action of the schedule.

3.5.5. Definition of ma-status-action-obj

```
object {
    string              ma-status-action-name;
    string              ma-status-action-state;
    int                 ma-status-action-storage;
    counter             ma-status-action-invocations;
    counter             ma-status-action-suppressions;
    counter             ma-status-action-overlaps;
    counter             ma-status-action-failures;
    datetime            ma-status-action-last-invocation;
    datetime            ma-status-action-last-completion;
    int                 ma-status-action-last-status;
    string              ma-status-action-last-message;
    datetime            ma-status-action-last-failed-completion;
    int                 ma-status-action-last-failed-status;
    string              ma-status-action-last-failed-message;
} ma-status-action-obj;
```

The ma-status-action-obj provides status information about an action of a schedule and consists of the following elements:

ma-status-action-name: The name of the action of a schedule this status object refers to.

ma-status-action-state: The state of the action. The value 'enabled' indicates that the action is currently enabled. The value 'suppressed' indicates that the action is currently suppressed. The value 'disabled' indicates that the action is currently disabled. The value 'running' indicates that the action is currently running.

ma-status-action-storage: The amount of secondary storage (e.g., allocated in a file system) holding temporary data allocated to the action in bytes. This object reports the amount of allocated physical storage and not the storage used by logical data records. Data models should use a 64-bit integer type.

ma-status-action-invocations	Number of invocations of this action. This counter does not include suppressed invocations or invocations that were prevented due to an overlap with a previous invocation of this action.
ma-status-action-suppressions	Number of suppressed executions of this action.
ma-status-action-overlaps	Number of executions prevented due to overlaps with a previous invocation of this action.
ma-status-action-failures	Number of failed executions of this action.
ma-status-action-last-invocation:	The date and time of the last invocation of this action.
ma-status-action-last-completion:	The date and time of the last completion of this action.
ma-status-action-last-status:	The status code returned by the last execution of this action.
ma-status-action-last-message:	The status message produced by the last execution of this action.
ma-status-action-last-failed-completion:	The date and time of the last failed completion of this action.
ma-status-action-last-failed-status:	The status code returned by the last failed execution of this action.
ma-status-action-last-failed-message:	The status message produced by the last failed execution of this action.

3.5.6. Definition of ma-status-suppression-obj

```
object {
    string                  ma-status-suppression-name;
    string                  ma-status-suppression-state;
} ma-status-suppression-obj;
```

The ma-status-suppression-obj provides status information about that status of a suppression and consists of the following elements:

ma-status-suppression-name: The name of the suppression this status object refers to.

ma-status-suppression-state: The state of the suppression. The value 'enabled' indicates that the suppression is currently enabled. The value 'active indicates that the suppression is currently active. The value 'disabled' indicates that the suppression is currently disabled.

3.5.7. Definition of ma-status-interface-obj

```
object {
    string              ma-status-interface-name;
    string              ma-status-interface-type;
   [int                 ma-status-interface-speed;]
   [string              ma-status-interface-link-layer-address;]
   [ip-address          ma-status-interface-ip-addresses<0..*>;]
   [ip-address          ma-status-interface-gateways<0..*>;]
   [ip-address          ma-status-interface-dns-servers<0..*>;]
} ma-status-interface-obj;
```

The ma-status-interface-obj provides status information about network interfaces and consists of the following elements:

ma-status-interface-name: A name uniquely identifying a network interface.

ma-status-interface-type: The type of the network interface.

ma-status-interface-speed: An optional indication of the speed of the interface (measured in bits-per-second).

ma-status-interface-link-layer-address:	An optional link-layer address of the interface.
ma-status-interface-ip-addresses:	An optional ordered list of IP addresses assigned to the interface.
ma-status-interface-gateways:	An optional ordered list of gateways assigned to the interface.
ma-status-interface-dns-servers:	An optional ordered list of DNS servers assigned to the interface.

3.6. Reporting Information

At a point in time specified by a Schedule, the MA will execute tasks that communicate a set of measurement results to the Collector. These Reporting Tasks will be configured to transmit task results over a specified Report Channel to a Collector.

It should be noted that the output from Tasks does not need to be sent to communication Channels. It can alternatively, or additionally, be sent to other Tasks on the MA. This facilitates using a first Measurement Task to control the operation of a later Measurement Task (such as first probing available line speed and then adjusting the operation of a video testing measurement) and also to allow local processing of data to output alarms (e.g., when performance drops from earlier levels). Of course, subsequent Tasks also include Tasks that implement the reporting protocol(s) and transfer data to one or more Collector(s).

The Report generated by a Reporting Task is structured hierarchically to avoid repetition of report header and Measurement Task Configuration information. The report starts with the timestamp of the report generation on the MA and details about the MA including the optional Measurement Agent ID and Group ID (controlled by the Configuration Information).

Much of the report Information is optional and will depend on the implementation of the Reporting Task and any parameters defined in the Task Configuration for the Reporting Task. For example some Reporting Tasks may choose not to include the Measurement Task Configuration or Action parameters, while others may do so dependent on the Controller setting a configurable parameter in the Task Configuration.

It is possible for a Reporting Task to send just the Report header (datetime and optional agent ID and/or Group ID) if no measurement data is available. Whether to send such empty reports again is dependent on the implementation of the Reporting Task and potential Task Configuration parameter.

The handling of measurement data on the MA before generating a Report and transfer from the MA to the Collector is dependent on the implementation of the device, MA and/or scheduled Tasks and not defined by the LMAP standards. Such decisions may include limits to the measurement data storage and what to do when such available storage becomes depleted. It is generally suggested that implementations running out of storage stop executing new measurement tasks and retain old measurement data.

No context information, such as line speed or broadband product are included within the report header information as this data is reported by individual tasks at the time they execute. Either a Measurement Task can report contextual parameters that are relevant to that particular measurement, or specific tasks can be used to gather a set of contextual and environmental data at certain times independent of the reporting schedule.

After the report header information the results are reported grouped according to different Measurement Task Configurations. Each Task section optionally starts with replicating the Measurement Task Configuration information before the result headers (titles for data columns) and the result data rows. The Options reported are those used for the scheduled execution of the Measurement Task and therefore include the Options specified in the Task Configuration as well as additional Options specified in the Action. The Action Options are appended to the Task Configuration Options in exactly the same order as they were provided to the Task during execution.

The result row data includes a time for the start of the measurement and optionally an end time where the duration also needs to be considered in the data analysis.

Some Measurement Tasks may optionally include an indication of the cross-traffic although the definition of cross-traffic is left up to each individual Measurement Task. Some Measurement Tasks may also output other environmental measures in addition to cross-traffic such as CPU utlilisation or interface speed.

Whereas the Configuration and Instruction information represent information transmitted via the Control Protocol, the Report represents the information that is transmitted via the Report Protocol. It is constructed at the time of sending a report and

represents the inherent structure of the information that is sent to the Collector.

3.6.1. Definition of ma-report-obj

```
  object {
      datetime              ma-report-date;
     [uuid                  ma-report-agent-id;]
     [string                ma-report-group-id;]
     [string                ma-report-measurement-point;]
     [ma-report-result-obj  ma-report-results<0..*>;]
  } ma-report-obj;
```

The ma-report-obj provides the meta-data of a single report and consists of the following elements:

ma-report-date: The date and time when the report was sent to a collector.

ma-report-agent-id: An optional uuid uniquely identifying the measurement agent.

ma-report-group-id: An optional identifier of the group of measurement agents this measurement agent belongs to.

ma-report-measurement-point: An optional identifier for the measurement point indicating where the measurement agent is located on a path (see [RFC7398] for further details).

ma-report-results: An optional and possibly empty unordered set of result objects.

3.6.2. Definition of ma-report-result-obj

```
object {
    string                  ma-report-result-schedule-name;
    string                  ma-report-result-action-name;
    string                  ma-report-result-task-name;
   [ma-option-obj           ma-report-result-options<0..*>;]
   [string                  ma-report-result-tags<0..*>;]
    datetime                ma-report-result-event-time;
    datetime                ma-report-result-start-time;
   [datetime                ma-report-result-end-time;]
   [string                  ma-report-result-cycle-number;]
    int                     ma-report-result-status;
   [ma-report-conflict-obj  ma-report-result-conflicts<0..*>;]
   [ma-report-table-obj     ma-report-result-tables<0..*>;]
} ma-report-result-obj;
```

The ma-report-result-obj provides the meta-data of a result report of a single executed action. It consists of the following elements:

ma-report-result-schedule-name: The name of the schedule that produced the result.

ma-report-result-action-name: The name of the action in the schedule that produced the result.

ma-report-result-task-name: The name of the task that produced the result.

ma-report-result-options: An optional ordered joined list of options provided by the task object and the action object when the action was started.

ma-report-result-tags: An optional unordered set of tags. This is the joined set of tags provided by the task object and the action object and schedule object when the action was started.

ma-report-result-event-time: The date and time of the event that triggered the schedule of the action that produced the reported result values. The date and time does not include any added randomization.

ma-report-result-start-time: The date and time of the start of the action that produced the reported result values.

ma-report-result-end-time: An optional date and time indicating when the action finished.

ma-report-result-cycle-number: An optional cycle number derived from ma-report-result-event-time. It is the time closest to ma-report-result-event-time that is a multiple of the ma-event-cycle-interval of the event that triggered the execution of the schedule. The value is only present in an ma-report-result-obj if the event that triggered the execution of the schedule has a defined ma-event-cycle-interval. The cycle number is represented in the format YYYYMMDD.HHMMSS where YYYY represents the year, MM the month (1..12), DD the day of the months (01..31), HH the hour (00..23), MM the minute (00..59), and SS the second (00..59). The cycle number is using Coordinated Universal Time (UTC).

ma-report-result-status: The status code returned by the execution of the action.

ma-report-result-conflicts: A possibly empty set of conflict actions that might have impacted the measurement results being reported.

ma-report-result-tables: An optional and possibly empty unordered set of result tables.

3.6.3. Definition of ma-report-conflict-obj

```
object {
    string  ma-report-conflict-schedule-name;
    string  ma-report-conflict-action-name;
    string  ma-report-conflict-task-name;
} ma-report-conflict-obj;
```

The ma-report-conflict-obj provides the information about conflicting action that might have impacted the measurement results. It consists of the following elements:

ma-report-result-schedule-name: The name of the schedule that may have impacted the result.

ma-report-result-action-name: The name of the action in the schedule that may have impacted the result.

ma-report-result-task-name: The name of the task that may have impacted the result.

3.6.4. Definition of ma-report-table-obj

```
object {
   [ma-registry-obj     ma-report-table-functions<0..*>;]
   [string]             ma-report-table-column-labels<0..*>;]
   [ma-report-row-obj   ma-report-table-rows<0..*>;]
} ma-report-table-obj;
```

The ma-report-table-obj represents a result table and consists of the following elements:

ma-report-table-functions: An optional and possibly empty unordered set of registry entries identifying the functions for which results that are reported.

ma-report-table-column-labels: An optional and possibly empty ordered list of column labels.

ma-report-table-rows: A possibly empty ordered list of result rows.

3.6.5. Definition of ma-report-row-obj

```
object {
    data                ma-report-row-values<0..*>;
} ma-report-row-obj;
```

The ma-report-row-obj represents a result row and consists of the following elements:

ma-report-row-values: A possibly empty ordered list of result values. When present, it contains an ordered list of values that align to the set of column labels for the report.

3.7. Common Objects: Schedules

A Schedule specifies the execution of a single or repeated series of Actions. An Action extends a Configured Task with additional specific parameters. Each Schedule contains basically two elements:

an ordered list of Actions to be executed and an Event object triggering the execution of the Schedule. The Schedule states what Actions to run (with what configuration) and when to run the Actions. A Schedule may optionally have an Event that stops the execution of the Schedule or a maximum duration after which a schedule is stopped.

Multiple Actions contained as an ordered list of a single Measurement Schedule will be executed according to the execution mode of the Schedule. In sequential mode, Actions will be executed sequentially and in parallel mode, all Actions will be executed concurrently. In pipelined mode, data produced by one Action is passed to the subsequent Action. Actions contained in different Schedules execute in parallel with such conflicts being reported in the Reporting Information where necessary. If two or more Schedules have the same start time, then the two will execute in parallel. There is no mechanism to prioritise one schedule over another or to mutex scheduled tasks.

As well as specifying which Actions to execute, the Schedule also specifies how to link the data outputs from each Action to other Schedules. Specifying this within the Schedule allows the highest level of flexibility since it is even possible to send the output from different executions of the same Task Configuration to different destinations. A single Task producing multiple different outputs is expected to properly tag the different result. An Action receiving the output can then filter the results based on the tag if necessary. For example, a Measurement Task might report routine results to a data Reporting Task in a Schedule that communicates hourly via the Broadband PPP interface, but also outputs emergency conditions via an alarm Reporting Task in a different Schedule communicating immediately over a GPRS channel. Note that task-to-task data transfer is always specified in association with the scheduled execution of the sending task - there is no need for a corresponding input specification for the receiving task. While it is likely that an MA implementation will use a queue mechanism between the Schedules or Actions, this Information Model does not mandate or define a queue. The Information Model, however, reports the storage allocated to Schedules and Actions so that storage usage can be monitored. Furthermore, it is recommended that MA implementations by default retain old data and stop the execution of new measurement tasks if the MA runs out of storage capacity.

When specifying the task to execute within the Schedule, i.e., creating an Action, it is possible to add to the Action option parameters. This allows the Task Configuration to determine the common characteristics of a Task, while selected parameters (e.g., the test target URL) are defined within as option parameters of the Action in the schedule. A single Tasks Configuration can even be

used multiple times in the same schedule with different additional parameters. This allows for efficiency in creating and transferring the Instruction. Note that the semantics of what happens if an option is defined multiple times (either in the Task Configuration, Action or in both) is not standardised and will depend upon the Task. For example, some tasks may legitimately take multiple values for a single parameter.

Where Options are specified in both the Action and the Task Configuration, the Action Options are appended to those specified in the Task Configuration.

Example: An Action of a Schedule references a single Measurement Task Configuration for measuring UDP latency. It specifies that results are to be sent to a Schedule with a Reporting Action. This Reporting Task of the Reporting Action is executed by a separate Schedule that specifies that it should run hourly at 5 minutes past the hour. When run this Reporting Action takes the data generated by the UDP latency Measurement Task as well as any other data to be included in the hourly report and transfers it to the Collector over the Report Channel specified within its own Schedule.

Schedules and Actions may optionally also be given tags that are included in result reports sent to a Collector. In addition, schedules can be given suppression tags that may be used to select Schedules and Actions for suppression.

3.7.1. Definition of ma-schedule-obj

```
  object {
      string              ma-schedule-name;
      ma-event-obj        ma-schedule-start;
     [ma-event-obj        ma-schedule-end;]
     [int                 ma-schedule-duration;]
      ma-action-obj       ma-schedule-actions<0..*>;
      string              ma-schedule-execution-mode;
     [string              ma-schedule-tags<0..*>;]
     [string              ma-schedule-suppression-tags<0..*>;]
  } ma-schedule-obj;
```

The ma-schedule-obj is the main scheduling object. It consists of the following elements:

ma-schedule-name: A name uniquely identifying a scheduling object.

ma-schedule-start:	An event object indicating when the schedule starts.
ma-schedule-end:	An optional event object controlling the forceful termination of scheduled actions. When the event occurs, all actions of the schedule will be forced to terminate gracefully.
ma-schedule-duration:	An optional duration in seconds for the schedule. All actions of the schedule will be forced to terminate gracefully after the duration number of seconds past the start of the schedule.
ma-schedule-actions:	A possibly empty ordered list of actions to invoke when the schedule starts.
ma-schedule-execution-mode:	Indicates whether the actions should be executed sequentially, in parallel, or in a pipelined mode (where data produced by one action is passed to the subsequent action). The default execution mode is pipelined.
ma-schedule-tags:	An optional unordered set of tags that are reported together with the measurement results to a collector.
ma-schedule-suppression-tags:	An optional unordered set of suppression tags that are used to select schedules to be suppressed.

3.7.2. Definition of ma-action-obj

```
  object {
      string              ma-action-name;
      string              ma-action-config-task-name;
     [ma-option-obj       ma-action-task-options<0..*>;]
     [string              ma-action-destinations<0..*>;]
     [string              ma-action-tags<0..*>;]
     [string              ma-action-suppression-tags<0..*>;]
   } ma-action-obj;
```

The ma-action-obj models a task together with its schedule specific task options and destination schedules. It consists of the following elements:

ma-action-name:	A name uniquely identifying an action of a scheduling object.
ma-action-config-task-name:	A name identifying the configured task to be invoked by the action.
ma-action-task-options:	An optional and possibly empty ordered list of options (name-value pairs) that are passed to the task by appending them to the options configured for the task object.
ma-action-destinations:	An optional and possibly empty unordered set of names of destination schedules that consume output produced by this action.
ma-action-tags:	An optional unordered set of tags that are reported together with the measurement results to a collector.
ma-action-suppression-tags:	An optional unordered set of suppression tags that are used to select actions to be suppressed.

3.8. Common Objects: Channels

A Channel defines a bi-directional communication mechanism between the MA and a Controller or Collector. Multiple Channels can be defined to enable results to be split or duplicated across different Collectors.

Each Channel contains the details of the remote endpoint (including location and security credential information such as a certificate). The timing of when to communicate over a Channel is specified by the Schedule which executes the corresponding Control or Reporting Task. The certificate can be the digital certificate associated to the FQDN in the URL or it can be the certificate of the Certification Authority that was used to issue the certificate for the FQDN (Fully Qualified Domain Name) of the target URL (which will be retrieved later on using a communication protocol such as TLS). In order to establish a secure channel, the MA will use its own security credentials (in the Configuration Information) and the given credentials for the individual Channel end-point.

As with the Task Configurations, each Channel is also given a text name by which it can be referenced as a Task Option.

Although the same in terms of information, Channels used for communication with the Controller are referred to as Control Channels whereas Channels to Collectors are referred to as Report Channels. Hence Control Channels will be referenced from Control Tasks executed by a Control Schedule, whereas Report Channels will be referenced from within Reporting Tasks executed by an Instruction Schedule.

Multiple interfaces are also supported. For example the Reporting Task could be configured to send some results over GPRS. This is especially useful when such results indicate the loss of connectivity on a different network interface.

Example: A Channel used for reporting results may specify that results are to be sent to the URL (https://collector.example.org/report/), using the appropriate digital certificate to establish a secure channel.

3.8.1. Definition of ma-channel-obj

```
object {
    string              ma-channel-name;
    url                 ma-channel-target;
    credentials         ma-channel-credentials;
    [string             ma-channel-interface-name;]
} ma-channel-obj;
```

The ma-channel-obj consists of the following elements:

ma-channel-name: A unique name identifying the channel object.

ma-channel-target: A URL identifying the target channel endpoint.

ma-channel-credentials: The security credentials needed to establish a secure channel.

ma-channel-interface-name: An optional name of the network interface to be used. If not present, the IP protocol stack will select a suitable interface.

3.9. Common Objects: Task Configurations

Conceptually each Task Configuration defines the parameters of a Task that the Measurement Agent (MA) may perform at some point in time. It does not by itself actually instruct the MA to perform them at any particular time (this is done by a Schedule). Tasks can be

Measurement Tasks (i.e., those Tasks actually performing some type of passive or active measurement) or any other scheduled activity performed by the MA such as transferring information to or from the Controller and Collectors. Other examples of Tasks may include data manipulation or processing Tasks conducted on the MA.

A Measurement Task Configuration is the same in information terms to any other Task Configuration. Both measurement and non-measurement Tasks may have registry entries to enable the MA to uniquely identify the Task it should execute and retrieve the schema for any parameters that may be passed to the Task. Registry entries are specified as a URI and can therefore be used to identify the Task within a namespace or point to a web or local file location for the Task information. As mentioned previously, these URIs may be used to identify the Measurement Task in a public namespace [I-D.ietf-ippm-metric-registry].

Example: A Measurement Task Configuration may configure a single Measurement Task for measuring UDP latency. The Measurement Task Configuration could define the destination port and address for the measurement as well as the duration, internal packet timing strategy and other parameters (for example a stream for one hour and sending one packet every 500 ms). It may also define the output type and possible parameters (for example the output type can be the 95th percentile mean) where the measurement task accepts such parameters. It does not define when the task starts (this is defined by the Schedule element), so it does not by itself instruct the MA to actually perform this Measurement Task.

The Task Configuration will include a local short name for reference by a Schedule. Task Configurations may also refer to registry entries as described above. In addition the Task can be configured through a set of configuration Options. The nature and number of these Options will depend upon the Task. These options are expressed as name-value pairs although the 'value' may be a structured object instead of a simple string or numeric value. The implementation of these name-value pairs will vary between data models.

An Option that must be present for Reporting Tasks is the Channel reference specifying how to communicate with a Collector. This is included in the task options and will have a value that matches a channel name that has been defined in the Instruction. Similarly Control Tasks will have a similar option with the value set to a specified Control Channel.

A Reporting Task might also have a flag parameter, defined as an Option, to indicate whether to send a report without measurement results if there is no measurement result data pending to be

transferred to the Collector. In addition many tasks will also take as a parameter which interface to operate over.

In addition the Task Configuration may optionally also be given tags that can carry a Measurement Cycle ID. The purpose of this ID is to easily identify a set of measurement results that have been produced by Measurement Tasks with comparable Options. This ID could be manually incremented or otherwise changed when an Option change is implemented which could mean that two sets of results should not be directly compared.

3.9.1. Definition of ma-task-obj

```
object {
    string              ma-task-name;
    ma-registry-obj     ma-task-functions<0..*>;
   [ma-option-obj       ma-task-options<0..*>;]
   [string              ma-task-tags<0..*>;]
} ma-task-obj;
```

The ma-task-obj defines a configured task that can be invoked as part of an action. A configured task can be referenced by its name and it contains a possibly empty set of URIs to link to registry entries. Options allow the configuration of task parameters (in the form of name-value pairs). The ma-task-obj consists of the following elements:

ma-task-name: A name uniquely identifying a configured task object.

ma-task-functions: A possibly empty unordered set of registry entries identifying the functions of the configured task.

ma-task-options: An optional and possibly empty ordered list of options (name-value pairs) that are passed to the configured task.

ma-task-tags: An optional unordered set of tags that are reported together with the measurement results to a collector.

3.9.2. Definition of ma-option-obj

```
object {
    string              ma-option-name;
   [object              ma-option-value;]
} ma-option-obj;
```

The ma-option-obj models a name-value pair and consists of the following elements:

ma-option-name: The name of the option.

ma-option-value: The optional value of the option.

The ma-option-obj is used to define Task Configuration Options. Task Configuration Options are generally task specific. For tasks associated with an entry in a registry, the registry may define well-known option names (e.g., the so-called parameters in the IPPM metric registry [I-D.ietf-ippm-metric-registry]). Control and Reporting Tasks need to know the Channel they are going to use. The common option name for specifying the channel is "channel" where the option's value refers to the name of an ma-channel-obj.

3.10. Common Objects: Registry Information

Tasks and actions can be associated with entries in a registry. A registry object refers to an entry in a registry (identified by a URI) and it may define a set of roles.

3.10.1. Definition of ma-registry-obj

```
object {
    uri                 ma-registry-uri;
   [string              ma-registry-role<0..*>;]
} ma-registry-obj;
```

The ma-registry-obj refers to an entry of a registry and it defines the associated role(s). The ma-registry-obj consists of the following elements:

ma-registry-uri: A URI identifying an entry in a registry.

ma-registry-role: An optional and possibly empty unordered set of roles for the identified registry entry.

3.11. Common Objects: Event Information

The Event information object used throughout the information models can initially take one of several different forms. Additional forms may be defined later in order to bind the execution of schedules to additional events. The initially defined Event forms are:

1. Periodic Timing: Emits multiple events periodically according to an interval time defined in seconds

2. Calendar Timing: Emits multiple events according to a calendar based pattern, e.g., 22 minutes past each hour of the day on weekdays

3. One Off Timing: Emits one event at a specific date and time

4. Immediate: Emits one event as soon as possible

5. Startup: Emits an event whenever the MA is started (e.g., at device startup)

6. Controller Lost: Emits an event when connectivity to the controller has been lost

7. Controller Connected: Emits an event when connectivity to the controller has been (re-)established

Optionally each of the Event options may also specify a randomness that should be evaluated and applied separately to each indicated event. This randomness parameter defines a uniform interval in seconds over which the start of the task is delayed from the starting times specified by the event object.

Both the Periodic and Calendar timing objects allow for a series of Actions to be executed. While both have an optional end time, it is best practice to always configure an end time and refresh the information periodically to ensure that lost MAs do not continue their tasks forever.

Startup events are only created on device startup, not when a new Instruction is transferred to the MA. If scheduled task execution is desired both on the transfer of the Instruction and on device restart then both the Immediate and Startup timing needs to be used in conjunction.

The datetime format used for all elements in the information model MUST conform to RFC 3339 [RFC3339].

3.11.1. Definition of ma-event-obj

```
object {
   string               ma-event-name;
   union {
       ma-periodic-obj             ma-event-periodic;
       ma-calendar-obj             ma-event-calendar;
       ma-one-off-obj              ma-event-one-off;
       ma-immediate-obj            ma-event-immediate;
       ma-startup-obj              ma-event-startup;
       ma-controller-lost-obj      ma-event-controller-lost;
       ma-controller-connected-obj ma-event-controller-connected;
   }
   [int                 ma-event-random-spread;]
   [int                 ma-event-cycle-interval;]
} ma-event-obj;
```

The ma-event-obj is the main event object. Event objects are identified by a name. A generic event object itself contains a more specific event object. The set of specific event objects should be extensible. The initial set of specific event objects is further described below. The ma-event-obj also includes an optional uniform random spread that can be used to randomize the start times of schedules triggered by an event. The ma-event-obj consists of the following elements:

ma-event-name: The name uniquely identifies an event object. Schedules refer to event objects by this name.

ma-event-periodic: The ma-event-periodic is present for periodic timing objects.

ma-event-calendar: The ma-event-calendar is present for calendar timing objects.

ma-event-one-off: The ma-event-one-off is present for one-off timing objects.

ma-event-immediate: The ma-event-immediate is present for immediate event objects.

ma-event-startup: The ma-event-startup is present for startup event objects.

ma-event-controller-lost: The ma-event-controller-lost is present for connectivity to controller lost event objects.

ma-event-controller-connected: The ma-event-controller-connected is present for connectivity to a controller established event objects.

ma-event-random-spread: The optional ma-event-random-spread adds a random delay defined in seconds to the event object. No random delay is added if ma-event-random-spread does not exist.

ma-event-cycle-interval: The optional ma-event-cycle-interval defines the duration of the time interval in seconds that is used to calculate cycle numbers. No cycle number is calculated if ma-event-cycle-interval does not exist.

3.11.2. Definition of ma-periodic-obj

```
object {
   [datetime            ma-periodic-start;]
   [datetime            ma-periodic-end;]
    int                 ma-periodic-interval;
} ma-periodic-obj;
```

The ma-periodic-obj timing object has an optional start and an optional end time plus a periodic interval. Schedules using an ma-periodic-obj are started periodically between the start and end time. The ma-periodic-obj consists of the following elements:

ma-periodic-start: The optional date and time at which Schedules using this object are first started. If not present it defaults to immediate.

ma-periodic-end: The optional date and time at which Schedules using this object are last started. If not present it defaults to indefinite.

ma-periodic-interval: The interval defines the time in seconds between two consecutive starts of tasks.

3.11.3. Definition of ma-calendar-obj

Calendar Timing supports the routine execution of Schedules at specific times and/or on specific dates. It can support more flexible timing than Periodic Timing since the execution of Schedules

does not have to be uniformly spaced. For example a Calendar Timing could support the execution of a Measurement Task every hour between 6pm and midnight on weekdays only.

Calendar Timing is also required to perform measurements at meaningful times in relation to network usage (e.g., at peak times). If the optional timezone offset is not supplied then local system time is assumed. This is essential in some use cases to ensure consistent peak-time measurements as well as supporting MA devices that may be in an unknown timezone or roam between different timezones (but know their own timezone information such as through the mobile network).

The calendar elements within the Calendar Timing do not have defaults in order to avoid accidental high-frequency execution of Tasks. If all possible values for an element are desired then the wildcard * is used.

```
  object {
     [datetime            ma-calendar-start;]
     [datetime            ma-calendar-end;]
     [string              ma-calendar-months<0..*>;]
     [string              ma-calendar-days-of-week<0..*>;]
     [string              ma-calendar-days-of-month<0..*>;]
     [string              ma-calendar-hours<0..*>;]
     [string              ma-calendar-minutes<0..*>;]
     [string              ma-calendar-seconds<0..*>;]
     [int                 ma-calendar-timezone-offset;]
  } ma-calendar-obj;
```

ma-calendar-start: The optional date and time at which Schedules using this object are first started. If not present it defaults to immediate.

ma-calendar-end: The optional date and time at which Schedules using this object are last started. If not present it defaults to indefinite.

ma-calendar-months: The optional set of months (1-12) on which tasks scheduled using this object are started. The wildcard * means all months. If not present, it defaults to no months.

ma-calendar-days-of-week: The optional set of days of a week ("Mon", "Tue", "Wed", "Thu", "Fri",

	"Sat", "Sun") on which tasks scheduled using this object are started. The wildcard * means all days of the week. If not present, it defaults to no days.
ma-calendar-days-of-month:	The optional set of days of a months (1-31) on which tasks scheduled using this object are started. The wildcard * means all days of a months. If not present, it defaults to no days.
ma-calendar-hours:	The optional set of hours (0-23) on which tasks scheduled using this object are started. The wildcard * means all hours of a day. If not present, it defaults to no hours.
ma-calendar-minutes:	The optional set of minutes (0-59) on which tasks scheduled using this object are started. The wildcard * means all minutes of an hour. If not present, it defaults to no hours.
ma-calendar-seconds:	The optional set of seconds (0-59) on which tasks scheduled using this object are started. The wildcard * means all seconds of an hour. If not present, it defaults to no seconds.
ma-calendar-timezone-offset:	The optional timezone offest in hours. If not present, it defaults to the system's local timezone.

If a day of the month is specified that does not exist in the month (e.g., 29th of Feburary) then those values are ignored.

3.11.4. Definition of ma-one-off-obj

```
object {
    datetime            ma-one-off-time;
} ma-one-off-obj;
```

The ma-one-off-obj timing object specifies a fixed point in time. Schedules using an ma-one-off-obj are started once at the specified date and time. The ma-one-off-obj consists of the following elements:

ma-one-off-time: The date and time at which Schedules using this object are started.

3.11.5. Definition of ma-immediate-obj

```
object {
                        // empty
} ma-immediate-obj;
```

The ma-immediate-obj event object has no further information elements. Schedules using an ma-immediate-obj are started as soon as possible.

3.11.6. Definition of ma-startup-obj

```
object {
                        // empty
} ma-startup-obj;
```

The ma-startup-obj event object has no further information elements. Schedules or suppressions using an ma-startup-obj are started at MA initialization time.

3.11.7. Definition of ma-controller-lost-obj

```
object {
                        // empty
} ma-controller-lost-obj;
```

The ma-controller-lost-obj event object has no further information elements. The ma-controller-lost-obj indicates that connectivity to the controller has been lost. This is determined by a timer started after each successful contact with a controller. When the timer reaches the controller-timeout (measured in seconds), an ma-controller-lost-obj event is generated. This event may be used to start a suppression.

3.11.8. Definition of ma-controller-connected-obj

```
object {
                        // empty
} ma-controller-connected-obj;
```

The ma-controller-connected-obj event object has no further information elements. The ma-controller-connected-obj indicates that connectivity to the controller has been established again after it was lost. This event may be used to end a suppression.

4. Example Execution

The example execution has two event sources E1 and E2 and three schedules S1, S2, and S3. The schedule S3 is started by events of event source E2 while the schedules S1 and S2 are both started by events of the event source E1. The schedules S1 and S2 have two actions each and schedule S3 has a single action. The event source E2 has no randomization while the event source E1 has the randomization r.

Figure 2 shows a possible timeline of an execution. The time T is progressing downwards. The dotted vertial line indicates progress of time while a dotted horizontal line indicates which schedule are triggered by an event. Tilded lines indicate data flowing from an action to another schedule. Actions within a schedule are named A1, A2, etc.

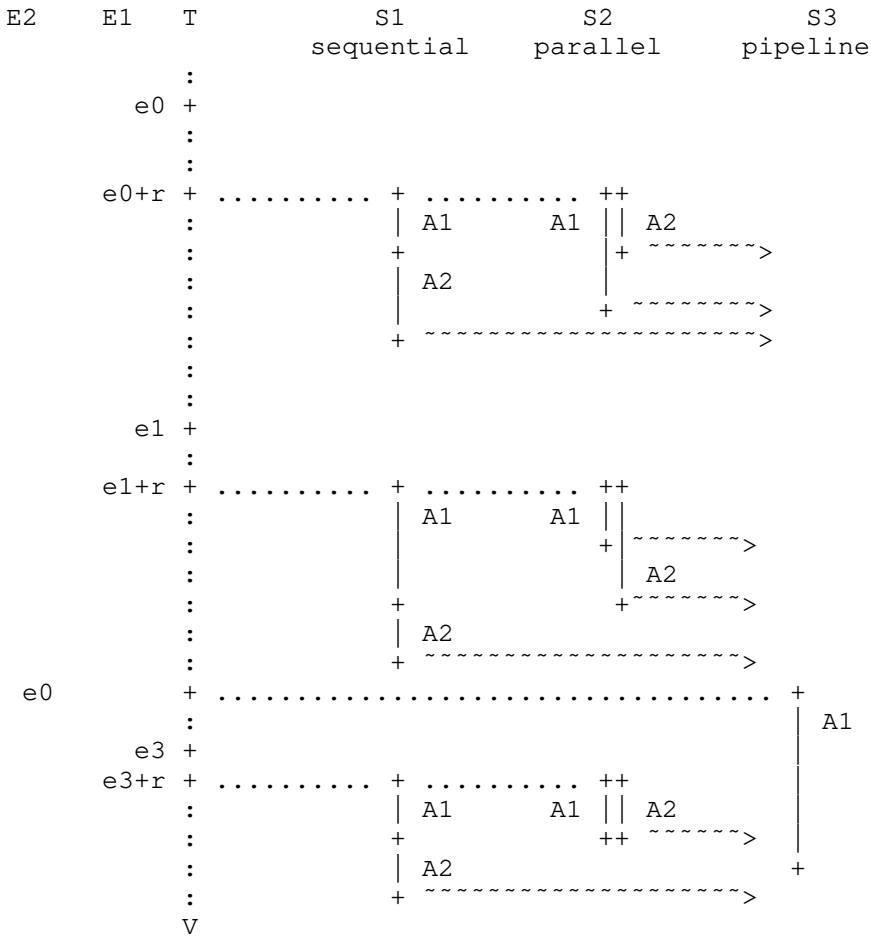

Figure 2: Example Execution

Note that implementations must handle possible concurrency issues. In the example execution, action A1 of schedule S3 is consuming the data that has been forwarded to schedule S3 while additional data is arriving from action A2 of schedule S2.

5. IANA Considerations

This document makes no request of IANA.

Note to the RFC Editor: this section may be removed on publication as an RFC.

6. Security Considerations

This Information Model deals with information about the control and reporting of the Measurement Agent. There are broadly two security considerations for such an Information Model. Firstly the Information Model has to be sufficient to establish secure communication channels to the Controller and Collector such that other information can be sent and received securely. Additionally, any mechanisms that the Network Operator or other device administrator employs to pre-configure the MA must also be secure to protect unauthorized parties from modifying pre-configuration information. These mechanisms are important to ensure that the MA cannot be hijacked, for example to participate in a distributed denial of service attack.

The second consideration is that no mandated information items should pose a risk to confidentiality or privacy given such secure communication channels. For this latter reason items such as the MA context and MA ID are left optional and can be excluded from some deployments. This may, for example, allow the MA to remain anonymous and for information about location or other context that might be used to identify or track the MA to be omitted or blurred. Implementations and deployments should also be careful about exposing device-ids when this is not strictly needed.

An implementation of this Information Model should support all the security and privacy requirements associated with the LMAP Framework [RFC7594]. In addition, users of this Information Model are advised to choose identifiers for Group IDs, tags or names of information model objects (e.g., configured tasks, schedules or actions) that do not reveal any sensitive information to people authorized to process measurement results but who are not authorized to know details about the Measurement Agents that were used to perform the measurement.

7. Acknowledgements

Several people contributed to this specification by reviewing early versions and actively participating in the LMAP working group (apologies to those unintentionally omitted): Vaibhav Bajpai, Michael Bugenhagen, Timothy Carey, Alissa Cooper, Kenneth Ko, Al Morton, Dan Romascanu, Henning Schulzrinne, Andrea Soppera, Barbara Stark, and Jason Weil.

Trevor Burbridge, Philip Eardley, Marcelo Bagnulo and Juergen Schoenwaelder worked in part on the Leone research project, which received funding from the European Union Seventh Framework Programme [FP7/2007-2013] under grant agreement number 317647.

Juergen Schoenwaelder was partly funded by Flamingo, a Network of Excellence project (ICT-318488) supported by the European Commission under its Seventh Framework Programme.

8. References

8.1. Normative References

[ISO.10646] International Organization for Standardization, "Information Technology - Universal Multiple-Octet Coded Character Set (UCS)", ISO Standard 10646:2014, 2014.

[POSIX.2] The IEEE and The Open Group, "The Open Group Base Specifications Issue 7", IEEE Standard 1003.1-2008, 2016.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <http://www.rfc-editor.org/info/rfc3339>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <http://www.rfc-editor.org/info/rfc4122>.

8.2. Informative References

[I-D.ietf-ippm-metric-registry] Bagnulo, M., Claise, B., Eardley, P., Morton, A., and A. Akhter, "Registry for Performance Metrics", draft-ietf-ippm-metric-registry-10 (work in progress), November 2016.

[I-D.ietf-lmap-yang] Schoenwaelder, J. and V. Bajpai, "A YANG Data Model for LMAP Measurement Agents", draft-ietf-lmap-yang-10 (work in progress), January 2017.

[RFC3444] Pras, A. and J. Schoenwaelder, "On the Difference between Information Models and Data Models", RFC 3444, DOI 10.17487/RFC3444, January 2003, <http://www.rfc-editor.org/info/rfc3444>.

[RFC7398] Bagnulo, M., Burbridge, T., Crawford, S., Eardley, P., and A. Morton, "A Reference Path and Measurement Points for Large-Scale Measurement of Broadband Performance", RFC 7398, DOI 10.17487/RFC7398, February 2015, <http://www.rfc-editor.org/info/rfc7398>.

[RFC7536] Linsner, M., Eardley, P., Burbridge, T., and F. Sorensen, "Large-Scale Broadband Measurement Use Cases", RFC 7536, DOI 10.17487/RFC7536, May 2015, <http://www.rfc-editor.org/info/rfc7536>.

[RFC7594] Eardley, P., Morton, A., Bagnulo, M., Burbridge, T., Aitken, P., and A. Akhter, "A Framework for Large-Scale Measurement of Broadband Performance (LMAP)", RFC 7594, DOI 10.17487/RFC7594, September 2015, <http://www.rfc-editor.org/info/rfc7594>.

Appendix A. Change History

Note to the RFC Editor: this section should be removed on publication as an RFC.

A.1. Non-editorial changes since -17

- The information model is subdivided into aspects and not sections.
- Changes to address the GEN-ART review comments.

A.2. Non-editorial changes since -16

- Addressing Alissa Cooper's review comments.

A.3. Non-editorial changes since -15

- The reference to the framework is now informational.

A.4. Non-editorial changes since -14

- Clarified that the cycle number is in UTC.

A.5. Non-editorial changes since -13

- Removed the ma-config-device-id from the ma-config-obj.
- Added ma-config-report-group-id and clarified how two flags ma-config-report-agent-id and ma-config-report-group-id work.

A.6. Non-editorial changes since -12

- Renamed the ma-metrics-registry-obj to ma-registry-obj since tasks may refer to different registries (not just a metrics registry).
- Clarifications and bug fixes.

A.7. Non-editorial changes since -11

- Clarifications and bug fixes.

A.8. Non-editorial changes since -10

- Rewrote the text concerning the well-known "channel" option name.
- Added ma-report-result-event-time, ma-report-result-cycle-number, and ma-event-cycle-interval.
- Added ma-capability-tags.
- Added a new section showing an example execution.
- Several clarifications and bug fixes.

A.9. Non-editorial changes since -09

- Added ma-status-schedule-storage and ma-status-action-storage.
- Removed suppress-by-default.
- Moved ma-report-result-metrics of the ma-report-result-obj to ma-report-table-metrics of the ma-report-table-obj so that the relationship between metrics and result tables is clear.
- Added ma-report-conflict-obj.
- Added ma-report-result-status to ma-report-result-obj.
- Several clarifications and bug fixes.

A.10. Non-editorial changes since -08

- Refactored the ma-report-task-obj into the ma-report-result-obj.
- Introduced the ma-report-table-obj so that a result can contain multiple tables.
- Report schedule, action, and task name as part of the ma-report-result-obj.
- Report conflicts per ma-report-result-obj and not per ma-report-row-obj.
- Report the start/end time as part of the ma-report-result-obj.

A.11. Non-editorial changes since -07

- Added ma-schedule-end and ma-schedule-duration.
- Changed the granularity of scheduler timings to seconds.
- Added ma-status-suppression-obj to report the status of suppressions as done in the YANG data model.
- Added counters to schedule and action status objects to match the counters in the YANG data model.
- Using tags to pass information such as a measurement cycle identifier to the collector.
- Using suppression tags and glob-style matching to select schedules and actions to be suppressed.

A.12. Non-editorial changes since -06

- The default execution mode is pipelined (LI12)
- Added text to define which action consumes data in sequential, pipelines, and parallel execution mode (LI11)
- Added ma-config-measurement-point, ma-report-measurement-point, and ma-config-report-measurement-point to configure and report the measurement point (LI10)
- Turned ma-suppression-obj into a list that uses a start event and a stop event to define the start and end of suppression; this unifies the handling of suppression and loss of controller connectivity (LI09)

- Added ma-controller-lost-obj and ma-controller-ok-obj event objects (LI09)
- Added ma-status-schedule-obj to report the status of a schedule and refactored ma-task-status-obj into ma-status-action-obj to report the status of an action (LI07, LI08)
- Introduced a common ma-metric-registry-obj that identifies a metric and a set of associated roles and added this object to expose metric capabilities and to support the configuration of metrics and to report the metrics used (LI06)
- Introduced ma-capability-obj and ma-capability-task-obj to expose the capabilities of a measurement agent (LI05)
- Use 'ordered list' or 'unordered set' instead of list, collection, etc. (LI02)
- Clarification that Actions are part of a Schedule (LI03)
- Deleted terms that are not strictly needed (LI04)

A.13. Non-editorial changes since -05

- A task can now reference multiply registry entries.
- Consistent usage of the term Action and Task.
- Schedules are triggered by Events instead of Timings; Timings are just one of many possible event sources.
- Actions feed into other Schedules (instead of Actions within other Schedules).
- Removed the notion of multiple task outputs.
- Support for sequential, parallel, and pipelined execution of Actions.

Authors' Addresses

Trevor Burbridge
BT
Adastral Park, Martlesham Heath
Ipswich IP5 3RE
United Kingdom

Email: trevor.burbridge@bt.com

Philip Eardley
BT
Adastral Park, Martlesham Heath
Ipswich IP5 3RE
United Kingdom

Email: philip.eardley@bt.com

Marcelo Bagnulo
Universidad Carlos III de Madrid
Av. Universidad 30
Leganes, Madrid 28911
Spain

Email: marcelo@it.uc3m.es

Juergen Schoenwaelder
Jacobs University Bremen
Campus Ring 1
Bremen 28759
Germany

Email: j.schoenwaelder@jacobs-university.de

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 4, 2017

J. Schoenwaelder
Jacobs University Bremen
V. Bajpai
Technical University Munich
June 2, 2017

Using RESTCONF with LMAP Measurement Agents
draft-ietf-lmap-restconf-04.txt

Abstract

This document describes how RESTCONF can be used with a YANG data model for Large-Scale Measurement Platforms (LMAP).

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 4, 2017.

Copyright Notice

Table of Contents

1. Introduction

The Framework for Large-Scale Measurement of Broadband Performance (LMAP) [RFC7594] describes an overall framework for large-scale broadband performance measurement systems. The standardization work in the IETF is restricted to the interaction between Measurement Agents and their Controllers and between Measurement Agents and result Collectors (see Figure 1 in [RFC7594]).

The protocol selection process within the LMAP working group of the IETF gave preference to a solution that reuses existing IETF protocols rather than inventing new ones. In addition, there was a preference to use a protocol that is layered on top of HTTP since this allows to reuse implementations already widely available.

This document discusses how the RESTCONF protocol [RFC8040] can be used to facilitate the communication between components implementing the LMAP framework. In particular, this document discusses how RESTCONF can be used as a Control Protocol to deliver Instruction(s) from a Controller to a Measurement Agent, and as a Report Protocol delivering Report(s) from a Measurement Agent to a Collector.

Measurement Agents may be deployed as separate hardware devices or as functions embedded in consumer electronic devices and home routers or as pure software solutions that can be installed on off-the-shelf computing equipment. Measurement Agents receive instructions from a Controller about when and how to conduct measurements (the Measurement Schedule) and how and when to report measurement results to a data Collector (the Report Schedule). Further information about the interaction between Measurement Agents and Controllers and Collectors can be found in [RFC7594].

The LMAP information model [I-D.ietf-lmap-information-model] defines in a conceptual and protocol-independent way the information exchanged between a Controller and a Measurement Agent as well as the information exchanged between a Measurement Agent and a Collector. A concrete YANG [RFC7950] data model derived from the conceptual information model is defined in [I-D.ietf-lmap-yang].

This document uses the LMAP terminology defined in [RFC7594].

2. Overview of RESTCONF

The RESTCONF protocol [RFC8040] provides an HTTP-based protocol for accessing data defined in YANG [RFC7950]. The basic idea behind RESTCONF is to expose YANG-defined data as a collection of Web resources that can be accessed and manipulated using standard HTTP [RFC7230] GET, DELETE, PATCH, POST, and PUT methods. The resource hierarchy is derived from the nesting structure of the YANG schema tree, leading to a so-called data-model-driven application programming interface (API).

RESTCONF is essentially a convention how to use HTTP over TLS to access a resources representing YANG-defined data. The resources are represented using either XML encoding (according to [RFC7950]) or JSON encoding (according to [RFC7951]). The examples shown in this document use the JSON encoding.

The normal mode of operation is that the RESTCONF client initiates a secure transport to the RESTCONF server. For devices located behind a middlebox (e.g., a network address translator or a firewall), a so called Call Home mechanism has been defined [RFC8071]. The Call Home mechanism allows the RESTCONF server to initiate a secure transport to a RESTCONF client. Note that Call Home only changes the TCP connection establishment, the TLS and HTTP client/server roles do not change. The policy used to control the Call Home mechanism can be configured through a configuration data model [I-D.ietf-netconf-restconf-client-server]. This model provides a mechanism to configure a list of redundant endpoints and it provides control over Call Home parameters (e.g, frequency of Call Home attempts, idle-timers, keep-alive timers).

3. RESTCONF as LMAP Control Protocol

The LMAP Control Protocol delivers Instruction(s) from a Controller to a Measurement Agent. The LMAP Control Protocol is realized by running a RESTCONF server on the Measurement Agent and a RESTCONF client on the Controller. Figure 1 depicts how the connection and the secure transport is established when the Measurement Agent is directly reachable from the Controller, i.e., the Measurement Agent

has a well-known name or address and is directly reachable from the Controller.

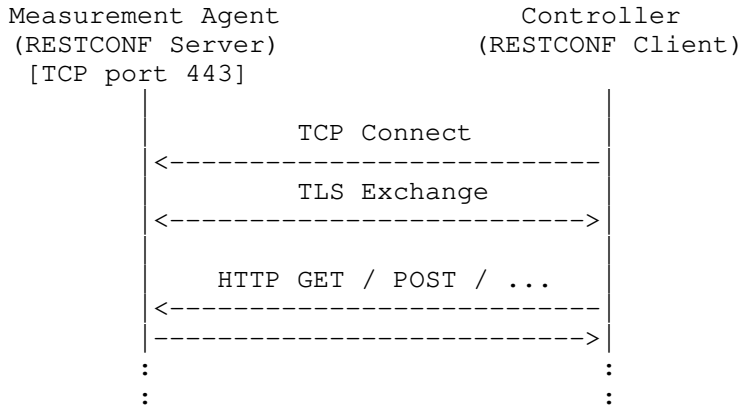

Figure 1: RESTCONF as Control protocol (without Call Home)

In several deployment scenarios, it will not be possible for the Controller to initiate a connection to the Measurement Agent due to the presence of middleboxes such as network address translators and firewalls. In such a situation, the Measurement Agent running a RESTCONF server will Call Home to the Controller running a RESTCONF client as shown in Figure 2.

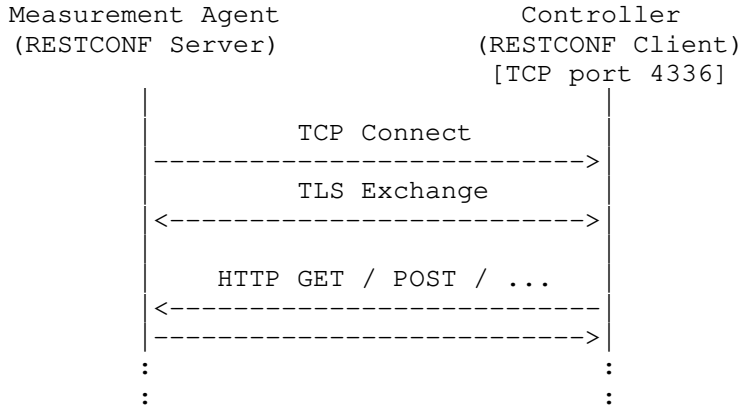

Figure 2: RESTCONF as Control Protocol (with Call Home)

Note that the Call Home mechanism only 'reverses' the way the underlying TCP connection is established. The subsequent TLS

exchange has the TLS server role on the RESTCONF server side and the TLS client role on the RESTCONF client side.

The YANG data model [I-D.ietf-lmap-yang], derived from the underlying information model [I-D.ietf-lmap-information-model], translates into a collection of RESTCONF resources that can be accessed and manipulated at various levels of granularity using HTTP GET, DELETE, PATCH, POST, and PUT methods.

An example exchange showing how a schedule object is installed on a Measurement Agent is shown in Appendix A.

[[CREF1: Move the example inline, update it to be aligned to the final YANG model and use JSON encoding.]]

4. RESTCONF as LMAP Report Protocol

The LMAP Report Protocol delivers Report(s) from a Measurement Agent to a Collector. The LMAP Report Protocol is realized by running a RESTCONF server on the Collector and a RESTCONF client on the Measurement Agent. Figure 3 depicts how the connection and the secure transport is established and how reports are delivered to the Controller. Note that it is generally assumed that the Controller is directly reachable from the Measurement Agent. (In situations where this may not be true, RESTCONF Call Home can be used as well but this is not shown here.)

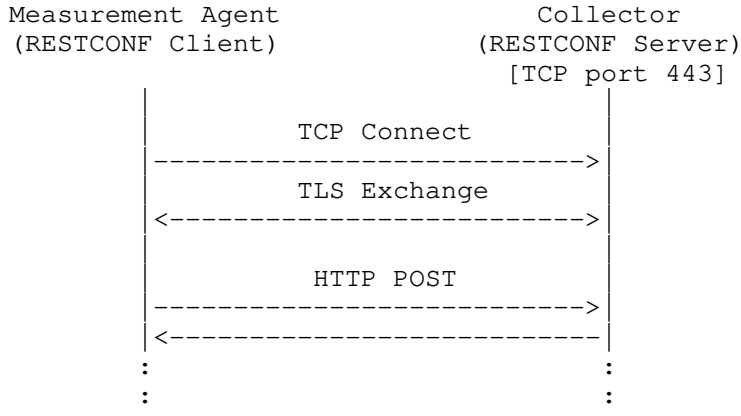

Figure 3: RESTCONF as Report Protocol

Note that the Measurement Agent pushes results to the Collector by invoking an operation on the Controller. This maps to an HTTP POST

in RESTCONF. Hence, pushing results can effectively be done by posting a the result to a specific RESTCONF resource.

An example exchange showing how results are reported to a Controller is shown in Appendix B.

[[CREF2: Move the example inline, update it to be aligned to the final YANG model and use JSON encoding.]]

5. RESTCONF Configuration for LMAP

[[CREF3: This section could explain how an LMAP implementation needs to be configured to make use of the Call Home mechanism and how report tasks refer to the configuration (if any standardized) needed to obtain the necessary credentials to report results. Is this necessary are can we simply refer to the I-Ds that have the details? Note that these I-Ds are not stable yet.]]

6. Security Considerations

Security and privacy aspects of the LMAP framework are discussed in Sections 7 and 8 of [RFC7594]. Section 12 of [RFC8040] and Section 5 of [RFC8071] discuss the security aspects of RESTCONF and the RESTCONF Call Home mechanism.

The security considerations specific to the LMAP information model and the YANG data model can be found in Section 6 of [I-D.ietf-lmap-information-model] and Section 5 of [I-D.ietf-lmap-yang].

7. IANA Considerations

This document has no requests for IANA.

8. Acknowledgements

Juergen Schoenwaelder and Vaibhav Bajpai worked in part on the Leone research project, which received funding from the European Union Seventh Framework Programme [FP7/2007-2013] under grant agreement number 317647.

Juergen Schoenwaelder and Vaibhav Bajpai were partly funded by Flamingo, a Network of Excellence project (ICT-318488) supported by the European Commission under its Seventh Framework Programme.

9. References

9.1. Normative References

[I-D.ietf-lmap-information-model]
Burbridge, T., Eardley, P., Bagnulo, M., and J. Schoenwaelder, "Information Model for Large-Scale Measurement Platforms (LMAP)", draft-ietf-lmap-information-model-16 (work in progress), January 2017.

[I-D.ietf-lmap-yang]
Schoenwaelder, J. and V. Bajpai, "A YANG Data Model for LMAP Measurement Agents", draft-ietf-lmap-yang-10 (work in progress), January 2017.

[RFC7594] Eardley, P., Morton, A., Bagnulo, M., Burbridge, T., Aitken, P., and A. Akhter, "A Framework for Large-Scale Measurement of Broadband Performance (LMAP)", RFC 7594, DOI 10.17487/RFC7594, September 2015, <http://www.rfc-editor.org/info/rfc7594>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <http://www.rfc-editor.org/info/rfc8040>.

[RFC8071] Watsen, K., "NETCONF Call Home and RESTCONF Call Home", RFC 8071, DOI 10.17487/RFC8071, February 2017, <http://www.rfc-editor.org/info/rfc8071>.

9.2. Informative References

[I-D.ietf-netconf-restconf-client-server]
Watsen, K. and J. Schoenwaelder, "RESTCONF Client and Server Models", draft-ietf-netconf-restconf-client-server-01 (work in progress), November 2016.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <http://www.rfc-editor.org/info/rfc7230>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <http://www.rfc-editor.org/info/rfc7950>.

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <http://www.rfc-editor.org/info/rfc7951>.

Appendix A. Example RESTCONF Control Protocol Exchange

Below is a YANG tree diagram of a part of the data model covering schedules. This is taken from [I-D.ietf-lmap-yang].

```
module: ietf-lmap-control
   +--rw lmap
      +--rw schedules
         +--rw schedule* [name]
            +--rw name      lmap:identifier
            +--rw event     event-ref
            +--rw execution-mode   enumeration
            +--rw action* [name]
               +--rw name           string
               +--rw task           task-ref
               +--rw option* [name]
               |  +--rw id          lmap:identifier
               |  +--rw name?       string
               |  +--rw value?      string
               +--rw destination*  leafref
```

Below is an XML representation of instance data conforming to the YANG data model is shown below. Note that some of the strings are references to other portions of the instance data not show here. This is again taken from [I-D.ietf-lmap-yang].

```
<lmap xmlns="urn:ietf:params:xml:ns:yang:ietf-lmap">
  <schedules>
    <schedule>
      <name>hourly-schedule</name>
      <event>hourly</event>
      <execution-mode>sequential</execution-mode>
      <action>
        <name>icmp-latency-hourly</name>
        <task>icmp-latency-measurement</task>
        <destination>daily</destination>
      </action>
    </schedule>
  </schedules>
</lmap>
```

Below is an example showing how RESTCONF can be used to create the above schedule. The prefix C: indicates the Controller, the prefix M: indicates the Measurement Agent. This example uses a JSON encoding (and note that much of the white-space can be removed, this is only there to help with readability).

```
C: POST /restconf/data/ietf-lmap-control:lmap/schedules HTTP/1.1
C: Host: example.com
C: Content-Type: application/yang.data+json
C:
C:   {
C:     "ietf-lmap-control:schedule": {
C:       "name": "hourly-schedule",
C:       "event": "hourly",
C:       "execution-mode": "sequential",
C:       "action": [
C:         {
C:           "name": "icmp-latency-hourly",
C:           "task": "icmp-latency-measurement",
C:           "destination": "daily",
C:         }
C:       ]
C:     }
C:   }

M: HTTP/1.1 201 Created
M: Date: Mon, 26 Mar 2015 17:01:00 GMT
M: Server: example-server
M: Location: https://example.com/restconf/data
M:           /ietf-lmap-control:lmap/schedules/schedule=hourly-schedule
M: Last-Modified: Mon, 26 Mar 2015 17:01:00 GMT
M: ETag: b3a3e673be2
```

Appendix B. Example RESTCONF Report Protocol Exchange

Below is an example showing how a Measurement Agent can submit results to a Collector running an RESTCONF server. The prefix C: indicates the Collector, the prefix M: indicates the Measurement Agent.

```
M: POST /restconf/operations/ietf-lmap-report:report HTTP/1.1
M: Host: example.com
M: Content-Type: application/yang.operation+xml
M:
M: <input xmlns="urn:ietf:params:xml:ns:yang:ietf-lmap-report">
M:   <date>2015-10-28T13:27:42+02:00</date>
M:   <agent-id>550e8400-e29b-41d4-a716-446655440000</agent-id>
M:   <group-id>wireless measurement at the north-pole</group-id>
M:   <result>
M:     <schedule-name>pinger</schedule-name>
M:     <action-name>fping</action-name>
M:     <task-name>fping</task-name>
M:     <option>
```

```
M:        <id>display-address</id>
M:        <name>-A</name>
M:      </option>
M:      <option>
M:        <id>display-DNS-lookup</id>
M:        <name>-d</name>
M:      </option>
M:      <option>
M:        <id>number-of-packets</id>
M:        <name>-C</name>
M:        <value>5</value>
M:      </option>
M:      <option>
M:        <id>quiet</id>
M:        <name>-q</name>
M:      </option>
M:      <option>
M:        <id>www.example.org</id>
M:        <name>www.example.org</name>
M:      </option>
M:      <option>
M:        <id>mail.example.com</id>
M:        <name>mail.example.com</name>
M:      </option>
M:      <start>2016-03-21T10:48:55+01:00</start>
M:      <end>2016-03-21T10:48:57+01:00</end>
M:      <status>0</status>
M:      <table>
M:        <column>target</column>
M:        <column>ip</column>
M:        <column>rtt-1</column>
M:        <column>rtt-2</column>
M:        <column>rtt-3</column>
M:        <column>rtt-4</column>
M:        <column>rtt-5</column>
M:        <row>
M:          <value>www.example.org</value>
M:          <value>2001:db8::1</value>
M:          <value>14.15</value>
M:          <value>14.14</value>
M:          <value>14.09</value>
M:          <value>14.17</value>
M:          <value>14.51</value>
M:        </row>
M:        <row>
M:          <value>mail.example.org</value>
M:          <value>2001:db8::2</value>
M:          <value>12.24</value>
```

```
M:          <value>11.99</value>
M:          <value>12.49</value>
M:          <value>11.87</value>
M:          <value>12.45</value>
M:        </row>
M:      </table>
M:    </result>
M: </input>

C: HTTP/1.1 200 OK
```

Authors' Addresses

Juergen Schoenwaelder
Jacobs University Bremen

Email: j.schoenwaelder@jacobs-university.de

Vaibhav Bajpai
Technical University Munich

Email: bajpaiv@in.tum.de

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 23, 2017

J. Schoenwaelder
V. Bajpai
Jacobs University Bremen
April 21, 2017

A YANG Data Model for LMAP Measurement Agents
draft-ietf-lmap-yang-12.txt

Abstract

This document defines a data model for Large-Scale Measurement Platforms (LMAP). The data model is defined using the YANG data modeling language.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 23, 2017.

Table of Contents

1. Introduction

This document defines a data model for Large-Scale Measurement Platforms (LMAP) [RFC7594]. The data model is defined using the YANG [RFC7950] data modeling language. It is based on the LMAP Information Model [I-D.ietf-lmap-information-model].

1.1. Terminology

This document uses the LMAP terminology defined in [RFC7594].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1.2. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams is as follows:

- Brackets "[" and "]" enclose list keys.
- Abbreviations before data node names: "rw" means configuration (read-write), "ro" means state data (read-only), and "w" means RPC input data (write-only).
- Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.
- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- Ellipsis ("...") stands for contents of subtrees that are not shown.

2. Data Model Overview

The LMAP framework has three basic elements: Measurement Agents, Controllers, and Collectors. Measurement Agents initiate the actual measurements, which are called Measurement Tasks in the LMAP terminology. The Controller instructs one or more MAs and communicates the set of Measurement Tasks an MA should perform and when. The Collector accepts Reports from the MAs with the Results from their Measurement Tasks.

The YANG data model for LMAP has been split into three modules:

1. The module ietf-lmap-common.yang provides common definitions such as LMAP specific data types.
2. The module ietf-lmap-control.yang defines the data structures exchanged between a Controller and Measurement Agents.
3. The module ietf-lmap-report.yang defines the data structures exchanged between Measurement Agents and Collectors.

As shown in Figure 1, a Controller, implementing ietf-lmap-common.yang and ietf-lmap-control.yang as a client, will instruct Measurement Agents, which implement ietf-lmap-common.yang and ietf-lmap-control.yang as servers. A Measurement Agent, implementing ietf-lmap-common.yang and ietf-lmap-report.yang, will send results to

a Collector, which implements ietf-lmap-common.yang and ietf-lmap-report.yang as a server.

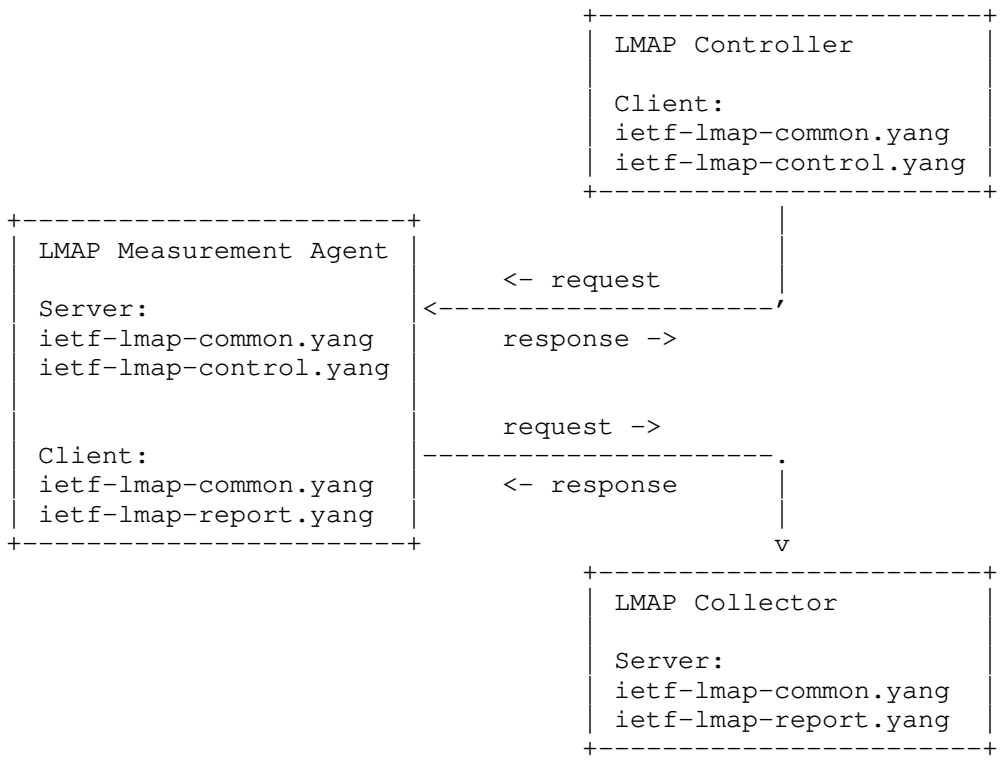

Figure 1: LMAP Controller, Measurement Agents, and Collector and the YANG modules they implement as client or server

The tree diagram below shows the structure of the control data model.

```
module: ietf-lmap-control
    +--rw lmap
       +--ro capabilities
       |  +--ro version    string
       |  +--ro tag*       lmap:tag
       |  +--ro tasks
       |     +--ro task* [name]
       |        +--ro name        lmap:identifier
       |        +--ro function* [uri]
       |        |  +--ro uri     inet:uri
       |        |  +--ro role*   string
       |        +--ro version?    string
```

```
|        +--ro program?    string
+--rw agent
|  +--rw agent-id?                   yang:uuid
|  +--rw group-id?                   string
|  +--rw measurement-point?          string
|  +--rw report-agent-id?            boolean
|  +--rw report-group-id?            boolean
|  +--rw report-measurement-point?   boolean
|  +--rw controller-timeout?         uint32
|  +--ro last-started                yang:date-and-time
+--rw tasks
|  +--rw task* [name]
|     +--rw name        lmap:identifier
|     +--rw function* [uri]
|     |  +--rw uri     inet:uri
|     |  +--rw role*   string
|     +--rw program?    string
|     +--rw option* [id]
|     |  +--rw id       lmap:identifier
|     |  +--rw name?    string
|     |  +--rw value?   string
|     +--rw tag*        lmap:identifier
+--rw schedules
|  +--rw schedule* [name]
|     +--rw name               lmap:identifier
|     +--rw start              event-ref
|     +--rw (stop)?
|     |  +--:(end)
|     |  |  +--rw end?               event-ref
|     |  +--:(duration)
|     |     +--rw duration?         uint32
|     +--rw execution-mode?    enumeration
|     +--rw tag*               lmap:tag
|     +--rw suppression-tag*   lmap:tag
|     +--ro state              enumeration
|     +--ro storage            yang:gauge64
|     +--ro invocations        yang:counter32
|     +--ro suppressions       yang:counter32
|     +--ro overlaps           yang:counter32
|     +--ro failures           yang:counter32
|     +--ro last-invocation?   yang:date-and-time
|     +--rw action* [name]
|        +--rw name                      lmap:identifier
|        +--rw task                      task-ref
|        +--rw parameters
|        |  +--rw (extension)?
|        +--rw option* [id]
|        |  +--rw id       lmap:identifier
```

```
|          |  +--rw name?    string
|          |  +--rw value?   string
|          +--rw destination*              schedule-ref
|          +--rw tag*                      lmap:tag
|          +--rw suppression-tag*          lmap:tag
|          +--ro state                     enumeration
|          +--ro storage                   yang:gauge64
|          +--ro invocations               yang:counter32
|          +--ro suppressions              yang:counter32
|          +--ro overlaps                  yang:counter32
|          +--ro failures                  yang:counter32
|          +--ro last-invocation           yang:date-and-time
|          +--ro last-completion           yang:date-and-time
|          +--ro last-status               lmap:status-code
|          +--ro last-message              string
|          +--ro last-failed-completion    yang:date-and-time
|          +--ro last-failed-status        lmap:status-code
|          +--ro last-failed-message       string
|       +--rw suppressions
|       |  +--rw suppression* [name]
|       |     +--rw name            lmap:identifier
|       |     +--rw start?          event-ref
|       |     +--rw end?            event-ref
|       |     +--rw match*          lmap:glob-pattern
|       |     +--rw stop-running?   boolean
|       |     +--ro state           enumeration
|       +--rw events
|          +--rw event* [name]
|             +--rw name                    lmap:identifier
|             +--rw random-spread?          uint32
|             +--rw cycle-interval?         uint32
|             +--rw (event-type)?
|                +--:(periodic)
|                |  +--rw periodic
|                |     +--rw interval    uint32
|                |     +--rw start?      yang:date-and-time
|                |     +--rw end?        yang:date-and-time
|                +--:(calendar)
|                   +--rw calendar
|                      +--rw month*              lmap:month-or-all
|                      +--rw day-of-month*       lmap:day-of-months-or-all
|                      +--rw day-of-week*        lmap:weekday-or-all
|                      +--rw hour*               lmap:hour-or-all
|                      +--rw minute*             lmap:minute-or-all
|                      +--rw second*             lmap:second-or-all
|                      +--rw timezone-offset?    lmap:timezone-offset
|                      +--rw start?              yang:date-and-time
|                      +--rw end?                yang:date-and-time
```

```
               +--:(one-off)
               |  +--rw one-off
               |     +--rw time    yang:date-and-time
               +--:(immediate)
               |  +--rw immediate               empty
               +--:(startup)
               |  +--rw startup                 empty
               +--:(controller-lost)
               |  +--rw controller-lost         empty
               +--:(controller-connected)
                  +--rw controller-connected    empty
```

The tree diagram below shows the structure of the reporting data model.

```
module: ietf-lmap-report

  rpcs:
    +---x report
       +---w input
          +---w date                  yang:date-and-time
          +---w agent-id?             yang:uuid
          +---w group-id?             string
          +---w measurement-point?    string
          +---w result*
             +---w schedule?       lmap:identifier
             +---w action?         lmap:identifier
             +---w task?           lmap:identifier
             +---w parameters
             |  +---w (extension)?
             +---w option* [id]
             |  +---w id       lmap:identifier
             |  +---w name?    string
             |  +---w value?   string
             +---w tag*            lmap:tag
             +---w event?          yang:date-and-time
             +---w start           yang:date-and-time
             +---w end?            yang:date-and-time
             +---w cycle-number?   lmap:cycle-number
             +---w status          lmap:status-code
             +---w conflict*
             |  +---w schedule-name?   lmap:identifier
             |  +---w action-name?     lmap:identifier
             |  +---w task-name?       lmap:identifier
             +---w table*
                +---w function* [uri]
                |  +---w uri     inet:uri
                |  +---w role*   string
                +---w column*     string
                +---w row*
                   +---w value*   string
```

3. Relationship to the Information Model

The LMAP information model [I-D.ietf-lmap-information-model] is divided into six aspects. They are mapped into the YANG data model as explained below:

- Pre-Configuration Information: This is not modeled explicitly since bootstrapping information is outside the scope of this data model. Implementations may use some of the Configuration Information also for bootstrapping purposes.

- Configuration Information: This is modeled in the /lmap/agent subtree, the /lmap/schedules subtree, and the /lmap/tasks subtree described below. Some items have been left out because they are expected to be dealt with by the underlying protocol.

- Instruction Information: This is modeled in the /lmap/suppressions subtree, the /lmap/schedules subtree, and the /lmap/tasks subtree described below.

- Logging Information: Some of the logging information, in particular 'success/failure/warning messages in response to information updates from the Controller', will be handled by the protocol used to manipulate the lmap specific configuration. The LMAP data model defined in this document assumes that runtime logging information will be communicated using protocols that do not require a formal data model, e.g., the Syslog protocol defined in [RFC5424].

- Capability and Status Information: Some of the capability and status information is modeled in the /lmap/capability subtree. The list of supported tasks is modeled in the /lmap/capabilities/task list. Status information about schedules and actions is included in the /lmap/schedules subtree. Information about network interfaces can be obtained from the ietf-interfaces YANG data model [RFC7223]. Information about the hardware and the firmware can be obtained from the ietf-system YANG data model [RFC7317]. A device identifier can be obtained from the ietf-hardware YANG data model [I-D.ietf-netmod-entity].

- Reporting Information: This is modeled by the report data model to be implemented by the Collector. Measurement Agents send results to the Collector by invoking an RPC on the Collector.

These six information model aspects use a collection of common information objects. These common information objects are represented in the YANG data model as follows:

- Schedules: Schedules are modeled in the /lmap/schedules subtree.

- Channels: Channels are not modeled since the NETCONF server configuration data model [I-D.ietf-netconf-netconf-client-server] already provides a mechanism to configure NETCONF server channels.

- Task Configurations: Configured tasks are modeled in the /lmap/tasks subtree.

- Event Information: Event definitions are modeled in the /lmap/events subtree.

4. YANG Modules

4.1. LMAP Common YANG Module

This module imports definitions from [RFC6536] and it references [ISO-8601].

```
<CODE BEGINS> file "ietf-lmap-common@2017-04-21.yang"
module ietf-lmap-common {

  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-lmap-common";
  prefix "lmap";

  import ietf-inet-types {
    prefix inet;
  }

  organization
    "IETF Large-Scale Measurement Platforms Working Group";

  contact
    "WG Web:   <http://tools.ietf.org/wg/lmap/>
     WG List:  <mailto:lmap@ietf.org>

     Editor:   Juergen Schoenwaelder
               <j.schoenwaelder@jacobs-university.de>

     Editor:   Vaibhav Bajpai
               <v.bajpai@jacobs-university.de>";

  description
    "This module provides common definitions used by the data
     models written for Large-Scale Measurement Platforms (LMAP).
     This module defines typedefs and groupings but no schema
     tree elements.";

  revision "2017-04-21" {
    description
      "Initial version";
    reference
      "RFC XXXX: A YANG Data Model for LMAP Measurement Agents";
  }

  /*
   * Typedefs
   */
```

```
  typedef identifier {
    type string {
      length "1..max";
    }
    description
      "An string value used to name something.";
  }

  typedef tag {
    type string {
      length "1..max";
    }
    description
      "A tag consists of at least one character.";
  }

  typedef glob-pattern {
    type string {
      length "1..max";
    }
    description
      'A glob style pattern (following POSIX.2 fnmatch() without
       special treatment of file paths):

         *         matches a sequence of characters
         ?         matches a single character
         [seq]     matches any character in seq
         [!seq]    matches any character not in seq

       A backslash followed by a character matches the following
       character. In particular:

         \*        matches *
         \?        matches ?
         \\        matches \

       A sequence seq may be a sequence of characters (e.g., [abc]
       or a range of characters (e.g., [a-c]).';
  }

  typedef wildcard {
    type string {
      pattern '\*';
    }
    description
      "A wildcard for calendar scheduling entries.";
  }
```

```
  typedef cycle-number {
    type string {
      pattern '[0-9]{8}\.[0-9]{6}';
    }
    description
      "A cycle number represented in the format YYYYMMDD.HHMMSS
       where YYYY represents the year, MM the month (1..12), DD
       the day of the months (01..31), HH the hour (00..23), MM
       the minute (00..59), and SS the second (00..59). The cycle
       number is using Coordinated Universal Time (UTC).";
  }

  typedef month {
    type enumeration {
      enum january {
        value 1;
        description
          "January of the Gregorian calendar.";
      }
      enum february {
        value 2;
        description
          "February of the Gregorian calendar.";
      }
      enum march {
        value 3;
        description
          "March of the Gregorian calendar.";
      }
      enum april {
        value 4;
        description
          "April of the Gregorian calendar.";
      }
      enum may {
        value 5;
        description
          "May of the Gregorian calendar.";
      }
      enum june {
        value 6;
        description
          "June of the Gregorian calendar.";
      }
      enum july {
        value 7;
        description
          "July of the Gregorian calendar.";
```

```
        }
        enum august {
          value 8;
          description
            "August of the Gregorian calendar.";
        }
        enum september {
          value 9;
          description
            "September of the Gregorian calendar.";
        }
        enum october {
          value 10;
          description
            "October of the Gregorian calendar.";
        }
        enum november {
          value 11;
          description
            "November of the Gregorian calendar.";
        }
        enum december {
          value 12;
          description
            "December of the Gregorian calendar.";
        }
      }
      description
        "A type modeling the month in the Gregorian calendar.";
    }

    typedef month-or-all {
      type union {
        type month;
        type wildcard;
      }
      description
        "A month or a wildcard indicating all twelve months.";
    }

    typedef day-of-month {
      type uint8 { range "1..31"; }
      description
        "A day of a month of the Gregorian calendar.";
    }

    typedef day-of-months-or-all {
      type union {
```

```
        type day-of-month;
        type wildcard;
      }
      description
        "A day of a months or a wildcard indicating all days
         of a month.";
    }

    typedef weekday {
      type enumeration {
        enum monday {
          value 1;
          description
            "Monday of the Gregorian calendar.";
        }
        enum tuesday {
          value 2;
          description
            "Tuesday of the Gregorian calendar.";
        }
        enum wednesday {
          value 3;
          description
            "Wednesday of the Gregorian calendar.";
        }
        enum thursday {
          value 4;
          description
            "Thursday of the Gregorian calendar.";
        }
        enum friday {
          value 5;
          description
            "Friday of the Gregorian calendar.";
        }
        enum saturday {
          value 6;
          description
            "Saturday of the Gregorian calendar.";
        }
        enum sunday {
          value 7;
          description
            "Sunday of the Gregorian calendar.";
        }
      }
      description
       "A type modeling the weekdays in the Gregorian calendar.
```

```
        The numbering follows the ISO 8601 scheme.";
      reference
       "ISO 8601:2004: Data elements and interchange formats --
                       Information interchange -- Representation
                       of dates and times";
    }

    typedef weekday-or-all {
      type union {
        type weekday;
        type wildcard;
      }
      description
        "A weekday or a wildcard indicating all seven weekdays.";
    }

    typedef hour {
      type uint8 { range "0..23"; }
      description
        "An hour of a day.";
    }

    typedef hour-or-all {
      type union {
        type hour;
        type wildcard;
      }
      description
        "An hour of a day or a wildcard indicating all hours
         of a day.";
    }

    typedef minute {
      type uint8 { range "0..59"; }
      description
        "A minute of an hour.";
    }

    typedef minute-or-all {
      type union {
        type minute;
        type wildcard;
      }
      description
        "A minute of an hour or a wildcard indicating all
         minutes of an hour.";
    }
```

```
  typedef second {
    type uint8 { range "0..59"; }
    description
      "A second of a minute.";
  }

  typedef second-or-all {
    type union {
      type second;
      type wildcard;
    }
    description
      "A second of a minute or a wildcard indicating all
       seconds of a minute.";
  }

  typedef status-code {
    type int32;
    description
      "A status code returned by the execution of a task. Note
       that the actual range is implementation dependent but it
       should be portable to use values in the range 0..127 for
       regular exit codes. By convention, 0 indicates successful
       termination. Negative values may be used to indicate
       abnormal termination due to a signal; the absolute value
       may identify the signal number in this case.";
  }

  typedef timezone-offset {
    type string {
      pattern 'Z|[\+\-]\d{2}:\d{2}';
    }
    description
      "A timezone-offset as it is used by the date-and-time type
       defined in the ietf-yang-types module. The value Z is
       equivalent to +00:00. The value -00:00 indicates and
       unknown time-offset.";
    reference
      "RFC 6991: Common YANG Data Types";
  }

  /*
   * Groupings
   */

  grouping registry-grouping {
    description
      "This grouping models a list of entries in a registry
```

```
          that identify functions of a tasks.";

        list function {
          key uri;
          description
            "A list of entries in a registry identifying functions.";

          leaf uri {
            type inet:uri;
            description
              "A URI identifying an entry in a registry.";
          }

          leaf-list role {
            type string;
            description
              "A set of roles for the identified registry entry.";
          }
        }
      }

      grouping options-grouping {
        description
          "A list of options of a task. Each option is a name/value
           pair (where the value may be absent).";

        list option {
          key "id";
          ordered-by user;
          description
            "A list of options passed to the task. It is a list of
             key / value pairs and may be used to model options.
             Options may be used to identify the role of a task
             or to pass a channel name to a task.";

          leaf id {
            type lmap:identifier;
            description
              "An identifier uniquely identifying an option. This
               identifier is required by YANG to uniquely identify
               a name value pair but it otherwise has no semantic
               value";
          }

          leaf name {
            type string;
            description
              "The name of the option.";
```

```
        }

        leaf value {
          type string;
          description
            "The value of the option.";
        }
      }
    }
  }
  <CODE ENDS>
```

4.2. LMAP Control YANG Module

This module imports definitions from [RFC6536], [RFC6991] and the common LMAP module and it references [RFC7398].

```
  <CODE BEGINS> file "ietf-lmap-control@2017-04-21.yang"
  module ietf-lmap-control {

    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-lmap-control";
    prefix "lmapc";

    import ietf-yang-types {
      prefix yang;
    }
    import ietf-netconf-acm {
      prefix nacm;
    }
    import ietf-lmap-common {
      prefix lmap;
    }

    organization
      "IETF Large-Scale Measurement Platforms Working Group";

    contact
      "WG Web:   <http://tools.ietf.org/wg/lmap/>
       WG List:  <mailto:lmap@ietf.org>

       Editor:   Juergen Schoenwaelder
                 <j.schoenwaelder@jacobs-university.de>

       Editor:   Vaibhav Bajpai
                 <v.bajpai@jacobs-university.de>";
```

```
  description
    "This module defines a data model for controlling measurement
     agents that are part of a Large-Scale Measurement Platform
     (LMAP). This data model is expected to be implemented by a
     measurement agent.";

  revision "2017-04-21" {
    description
      "Initial version";
    reference
      "RFC XXXX: A YANG Data Model for LMAP Measurement Agents";
  }

  /*
   * Typedefs
   */

  typedef event-ref {
    type leafref {
      path "/lmap/events/event/name";
    }
    description
      "This type is used by data models that need to reference
       a configured event source.";
  }

  typedef task-ref {
    type leafref {
      path "/lmap/tasks/task/name";
    }
    description
      "This type is used by data models that need to reference
       a configured task.";
  }

  typedef schedule-ref {
    type leafref {
      path "/lmap/schedules/schedule/name";
    }
    description
      "This type is used by data models that need to reference
       a configured schedule.";
  }

  /*
   * Groupings
   */
```

```
  grouping start-end-grouping {
    description
      "A grouping that provides start and end times for
       event objects.";
    leaf start {
      type yang:date-and-time;
      description
        "The date and time when the event object
         starts to create triggers.";
    }
    leaf end {
      type yang:date-and-time;
      description
        "The date and time when the event object
         stops to create triggers.

         It is generally a good idea to always configure
         an end time and to refresh the end time as needed
         to ensure that agents that lose connectivity to
         their controller do not continue executing schedules
         forever.";
    }
  }

  /*
   * Capability, configuration and state data nodes
   */

  container lmap {
    description
      "Configuration and control of an LMAP agent.";

    container capabilities {
      config false;
      description
        "Agent capabilities including a list of supported tasks.";

      leaf version {
        type string;
        config false;
        mandatory true;
        description
          "A short description of the software implementing the
           measurement agent. This should include the version
           number of the measurement agent software.";
      }

      leaf-list tag {
```

```
          type lmap:tag;
          config false;
          description
            "An optional unordered set of tags that provide
             additional information about the capabilities of
             the measurement agent.";
        }

        container tasks {
          description
            "A list of tasks that the measurement agent supports.";

          list task {
            key name;
            description
              "The list of tasks supported by the LMAP agent.";

            leaf name {
              type lmap:identifier;
              description
                "The unique name of a task capability.";
            }

            uses lmap:registry-grouping;

            leaf version {
              type string;
              description
                "A short description of the software implementing
                 the task. This should include the version
                 number of the measurement task software.";
            }

            leaf program {
              type string;
              description
                "The (local) program to invoke in order to execute
                 the task.";
            }
          }
        }
      }

      /*
       * Agent Configuration
       */

      container agent {
```

```
        description
          "Configuration of parameters affecting the whole
           measurement agent.";

        leaf agent-id {
          type yang:uuid;
          description
            "The agent-id identifies a measurement agent with
             a very low probability of collision. In certain
             deployments, the agent-id may be considered
             sensitive and hence this object is optional.";
        }

        leaf group-id {
          type string;
          description
            "The group-id identifies a group of measurement
             agents. In certain deployments, the group-id
             may be considered less sensitive than the
             agent-id.";
        }

        leaf measurement-point {
          type string;
          description
            "The measurement point indicating where the
              measurement agent is located on a path.";
          reference
            "RFC 7398: A Reference Path and Measurement Points
                       for Large-Scale Measurement of Broadband
                       Performance";
        }

        leaf report-agent-id {
          type boolean;
          must '. != "true" or ../agent-id' {
            description
              "An agent-id must exist for this to be set
               to true.";
          }
          default false;
          description
            "The 'report-agent-id' controls whether the
             'agent-id' is reported to collectors.";
        }

        leaf report-group-id {
          type boolean;
```

```
          must '. != "true" or ../group-id' {
            description
              "A group-id must exist for this to be set
               to true.";
          }
          default false;
          description
            "The 'report-group-id' controls whether the
             'group-id' is reported to collectors.";
        }

        leaf report-measurement-point {
          type boolean;
          must '. != "true" or ../measurement-point' {
            description
              "A measurement-point must exist for this to be
               set to true.";
          }
          default false;
          description
            "The 'report-measurement-point' controls whether
             the 'measurement-point' is reported to collectors.";
        }

        leaf controller-timeout {
          type uint32;
          units "seconds";
          description
            "A timer is started after each successful contact
             with a controller. When the timer reaches the
             controller-timeout, an event (controller-lost) is
             raised indicating that connectivity to the controller
             has been lost.";
        }

        leaf last-started {
          type yang:date-and-time;
          config false;
          mandatory true;
          description
            "The date and time the measurement agent last started.";
        }
      }

      /*
       * Task Configuration
       */
```

```
      container tasks {
        description
          "Configuration of LMAP tasks.";

        list task {
          key name;
          description
            "The list of tasks configured on the LMAP agent. Note
             that a configured task MUST resolve to a task listed
             in the capabilities. Attempts to execute a configured
             task that is not listed in the capabilities result in
             a runtime execution error.";

          leaf name {
            type lmap:identifier;
            description
              "The unique name of a task.";
          }

          uses lmap:registry-grouping;

          leaf program {
            type string;
            nacm:default-deny-write;
            description
              "The (local) program to invoke in order to execute
               the task. If this leaf is not set, then the system
               will try to identify a suitable program based on
               the registry information present.";
          }

          uses lmap:options-grouping {
            description
              "The list of task specific options.";
          }

          leaf-list tag {
            type lmap:identifier;
            description
              "A set of task specific tags that are reported
               together with the measurement results to a collector.
               A tag can be used, for example, to carry the
               Measurement Cycle ID.";
          }
        }
      }

      /*
```

```
        * Schedule Instructions
        */

       container schedules {
         description
           "Configuration of LMAP schedules. Schedules control
            which tasks are executed by the LMAP implementation.";

         list schedule {
           key name;
           description
             "Configuration of a particular schedule.";

           leaf name {
             type lmap:identifier;
             description
               "The locally-unique, administratively assigned name
                for this schedule.";
           }

           leaf start {
             type event-ref;
             mandatory true;
             description
               "The event source controlling the start of the
                scheduled actions.";
           }

           choice stop {
             description
               "This choice contains optional leafs that control the
                graceful forced termination of scheduled actions.
                When the end has been reached, the scheduled actions
                should be forced to terminate the measurements.
                This may involve being active some additional time in
                order to properly finish the action's activity (e.g.,
                waiting for any still outstanding messages).";

             leaf end {
               type event-ref;
               description
                 "The event source controlling the graceful
                  forced termination of the scheduled actions.";
             }

             leaf duration {
               type uint32;
               units "seconds";
```

```
        description
          "The duration controlling the graceful forced
           termination of the scheduled actions.";
      }
    }

    leaf execution-mode {
      type enumeration {
        enum sequential {
          value 1;
          description
            "The actions of the schedule are executed
             sequentially.";
        }
        enum parallel {
          value 2;
          description
            "The actions of the schedule are executed
             concurrently";
        }
        enum pipelined {
          value 3;
          description
            "The actions of the schedule are executed in a
             pipelined mode. Output created by an action is
             passed as input to the subsequent action.";
        }
      }
      default pipelined;
      description
        "The execution mode of this schedule determines in
         which order the actions of the schedule are executed.";
    }

    leaf-list tag {
      type lmap:tag;
      description
        "A set of schedule specific tags that are reported
         together with the measurement results to a collector.";
    }

    leaf-list suppression-tag {
      type lmap:tag;
      description
        "A set of suppression tags that are used to select
         schedules to be suppressed.";
    }
```

```
          leaf state {
            type enumeration {
              enum enabled {
                value 1;
                description
                  "The value 'enabled' indicates that the
                   schedule is currently enabled.";
              }
              enum disabled {
                value 2;
                description
                  "The value 'disabled' indicates that the
                   schedule is currently disabled.";
              }
              enum running {
                value 3;
                description
                  "The value 'running' indicates that the
                   schedule is currently running.";
              }
              enum suppressed {
                value 4;
                description
                  "The value 'suppressed' indicates that the
                   schedule is currently suppressed.";
              }
            }
            config false;
            mandatory true;
            description
              "The current state of the schedule.";
          }

          leaf storage {
            type yang:gauge64;
            units "bytes";
            config false;
            mandatory true;
            description
              "The amount of secondary storage (e.g., allocated in a
               file system) holding temporary data allocated to the
               schedule in bytes. This object reports the amount of
               allocated physical storage and not the storage used
               by logical data records.";
          }

          leaf invocations {
            type yang:counter32;
```

```
            config false;
            mandatory true;
            description
              "Number of invocations of this schedule. This counter
               does not include suppressed invocations or invocations
               that were prevented due to an overlap with a previous
               invocation of this schedule.";
          }

          leaf suppressions {
            type yang:counter32;
            config false;
            mandatory true;
            description
              "Number of suppressed executions of this schedule.";
          }

          leaf overlaps {
            type yang:counter32;
            config false;
            mandatory true;
            description
              "Number of executions prevented due to overlaps with
               a previous invocation of this schedule.";
          }

          leaf failures {
            type yang:counter32;
            config false;
            mandatory true;
            description
              "Number of failed executions of this schedule. A
               failed execution is an execution where at least
               one action failed.";
          }

          leaf last-invocation {
            type yang:date-and-time;
            config false;
            description
              "The date and time of the last invocation of
               this schedule.";
          }

          list action {
            key name;
            description
              "An action describes a task that is invoked by the
```

```
           schedule. Multiple actions are invoked according to
           the execution-mode of the schedule.";

        leaf name {
          type lmap:identifier;
          description
            "The unique identifier for this action.";
        }

        leaf task {
          type task-ref;
          mandatory true;
          description
            "The task invoked by this action.";
        }

        container parameters {
          description
            "This container is a place-holder for run-time
             parameters defined in task-specific data models
             augmenting the base lmap control data model.";

          choice extension {
            description
              "This choice is provided to augment in different
               sets of parameters.";
          }
        }

        uses lmap:options-grouping {
          description
            "The list of action specific options that are
             appended to the list of task specific options.";
        }

        leaf-list destination {
          type schedule-ref;
          description
            "A set of schedules receiving the output produced
             by this action. The output is stored temporarily
             since the destination schedules will in general
             not be running when output is passed to them. The
             behaviour of an action passing data to its own
             schedule is implementation specific.

             Data passed to a sequential or pipelined schedule
             is received by the schedule's first action. Data
             passed to a parallel schedule is received by all
```

```
          actions of the schedule.";
      }

      leaf-list tag {
        type lmap:tag;
        description
          "A set of action specific tags that are reported
           together with the measurement results to a
           collector.";
      }

      leaf-list suppression-tag {
        type lmap:tag;
        description
          "A set of suppression tags that are used to select
           actions to be suppressed.";
      }

      leaf state {
        type enumeration {
          enum enabled {
            value 1;
            description
              "The value 'enabled' indicates that the
               action is currently enabled.";
          }
          enum disabled {
            value 2;
            description
              "The value 'disabled' indicates that the
               action is currently disabled.";
          }
          enum running {
            value 3;
            description
              "The value 'running' indicates that the
               action is currently runnning.";
          }
          enum suppressed {
            value 4;
            description
              "The value 'suppressed' indicates that the
               action is currently suppressed.";
          }
        }
        config false;
        mandatory true;
        description
```

```
          "The current state of the action.";
        }

        leaf storage {
          type yang:gauge64;
          units "bytes";
          config false;
          mandatory true;
          description
            "The amount of secondary storage (e.g., allocated in a
             file system) holding temporary data allocated to the
             schedule in bytes. This object reports the amount of
             allocated physical storage and not the storage used
             by logical data records.";
        }

        leaf invocations {
          type yang:counter32;
          config false;
          mandatory true;
          description
            "Number of invocations of this action. This counter
             does not include suppressed invocations or invocations
             that were prevented due to an overlap with a previous
             invocation of this action.";
        }

        leaf suppressions {
          type yang:counter32;
          config false;
          mandatory true;
          description
            "Number of suppressed executions of this action.";
        }

        leaf overlaps {
          type yang:counter32;
          config false;
          mandatory true;
          description
            "Number of executions prevented due to overlaps with
             a previous invocation of this action.";
        }

        leaf failures {
          type yang:counter32;
          config false;
          mandatory true;
```

```
        description
          "Number of failed executions of this action.";
      }

      leaf last-invocation {
        type yang:date-and-time;
        config false;
        mandatory true;
        description
          "The date and time of the last invocation of
           this action.";
      }

      leaf last-completion {
        type yang:date-and-time;
        config false;
        mandatory true;
        description
          "The date and time of the last completion of
           this action.";
      }

      leaf last-status {
        type lmap:status-code;
        config false;
        mandatory true;
        description
          "The status code returned by the last execution of
           this action.";
      }

      leaf last-message {
        type string;
        config false;
        mandatory true;
        description
          "The status message produced by the last execution
           of this action.";
      }

      leaf last-failed-completion {
        type yang:date-and-time;
        config false;
        mandatory true;
        description
          "The date and time of the last failed completion
           of this action.";
      }
```

```
            leaf last-failed-status {
              type lmap:status-code;
              config false;
              mandatory true;
              description
                "The status code returned by the last failed
                 execution of this action.";
            }

            leaf last-failed-message {
              type string;
              config false;
              mandatory true;
              description
                "The status message produced by the last failed
                 execution of this action.";
            }
          }
        }
      }

      /*
       * Suppression Instructions
       */

      container suppressions {
        description
          "Suppression information to prevent schedules or
           certain actions from starting.";

        list suppression {
          key name;
          description
            "Configuration of a particular suppression.";

          leaf name {
            type lmap:identifier;
            description
              "The locally-unique, administratively assigned name
               for this suppression.";
          }

          leaf start {
            type event-ref;
            description
              "The event source controlling the start of the
               suppression period.";
          }
```

```
          leaf end {
            type event-ref;
            description
              "The event source controlling the end of the
               suppression period. If not present, supression
               continues indefinitely.";
          }

          leaf-list match {
            type lmap:glob-pattern;
            description
              "A set of suppression match pattern. The suppression
               will apply to all schedules (and their actions) that
               have a matching value in their suppression-tags
               and to all actions that have a matching value in
               their suppression-tags.";
          }

          leaf stop-running {
            type boolean;
            default false;
            description
              "If 'stop-running' is true, running schedules and
               actions matching the suppression will be terminated
               when suppression is activated. If 'stop-running' is
               false, running schedules and actions will not be
               affected if suppression is activated.";
          }

          leaf state {
            type enumeration {
              enum enabled {
                value 1;
                description
                  "The value 'enabled' indicates that the
                   suppression is currently enabled.";
              }
              enum disabled {
                value 2;
                description
                  "The value 'disabled' indicates that the
                   suppression is currently disabled.";
              }
              enum active {
                value 3;
                description
                  "The value 'active' indicates that the
                   suppression is currently active.";
```

```
            }
          }
          config false;
          mandatory true;
          description
            "The current state of the suppression.";
        }
      }
    }

    /*
     * Event Instructions
     */

    container events {
      description
        "Configuration of LMAP events.

         Implementations may be forced to delay acting
         upon the occurance of events in the face of local
         constraints. An action triggered by an event
         therefore should not rely on the accuracy
         provided by the scheduler implementation.";

      list event {
        key name;
        description
          "The list of event sources configured on the
           LMAP agent.";

        leaf name {
          type lmap:identifier;
          description
            "The unique name of an event source.";
        }

        leaf random-spread {
          type uint32;
          units seconds;
          description
            "This optional leaf adds a random spread to the
             computation of the event's trigger time. The
             random spread is a uniformly distributed random
             number taken from the interval [0:random-spread].";
        }

        leaf cycle-interval {
          type uint32;
```

```
            units seconds;
            description
              "The optional cycle-interval defines the duration
               of the time interval in seconds that is used to
               calculate cycle numbers. No cycle number is
               calculated if the optional cycle-interval does
               not exist.";
          }

          choice event-type {
            description
              "Different types of events are handled by
               different branches of this choice. Note that
               this choice can be extended via augmentations.";

            case periodic {
              container periodic {
                description
                  "A periodic timing object triggers periodically
                   according to a regular interval.";

                leaf interval {
                  type uint32 {
                    range "1..max";
                  }
                  units "seconds";
                  mandatory true;
                  description
                    "The number of seconds between two triggers
                     generated by this periodic timing object.";
                }
                uses start-end-grouping;
              }
            }

            case calendar {
              container calendar {
                description
                  "A calendar timing object triggers based on the
                   current calendar date and time.";

                leaf-list month {
                  type lmap:month-or-all;
                  min-elements 1;
                  description
                    "A set of months at which this calendar timing
                     will trigger. The wildcard means all months.";
                }
```

```
                leaf-list day-of-month {
                  type lmap:day-of-months-or-all;
                  min-elements 1;
                  description
                    "A set of days of the month at which this
                     calendar timing will trigger. The wildcard means
                     all days of a month.";
                }

                leaf-list day-of-week {
                  type lmap:weekday-or-all;
                  min-elements 1;
                  description
                    "A set of weekdays at which this calendar timing
                     will trigger. The wildcard means all weekdays.";
                }

                leaf-list hour {
                  type lmap:hour-or-all;
                  min-elements 1;
                  description
                    "A set of hours at which this calendar timing will
                     trigger. The wildcard means all hours of a day.";
                }

                leaf-list minute {
                  type lmap:minute-or-all;
                  min-elements 1;
                  description
                    "A set of minutes at which this calendar timing
                     will trigger. The wildcard means all minutes of
                     an hour.";
                }

                leaf-list second {
                  type lmap:second-or-all;
                  min-elements 1;
                  description
                    "A set of seconds at which this calendar timing
                     will trigger. The wildcard means all seconds of
                     a minute.";
                }

                leaf timezone-offset {
                  type lmap:timezone-offset;
                  description
                    "The timezone in which this calendar timing
                     object will be evaluated. If not present,
```

```
                the systems' local timezone will be used.";
            }
            uses start-end-grouping;
          }
        }

        case one-off {
          container one-off {
            description
              "A one-off timing object triggers exactly once.";

            leaf time {
              type yang:date-and-time;
              mandatory true;
              description
                "This one-off timing object triggers once at
                 the configured date and time.";
            }
          }
        }

        case immediate {
          leaf immediate {
            type empty;
            mandatory true;
            description
              "This immediate event object triggers immediately
               when it is configured.";
          }
        }

        case startup {
          leaf startup {
            type empty;
            mandatory true;
            description
              "This startup event object triggers whenever the
               LMAP agent (re)starts.";
          }
        }

        case controller-lost {
          leaf controller-lost {
            type empty;
            mandatory true;
            description
              "The controller-lost event object triggers when
               the connectivity to the controller has been lost
```

```
                    for at least 'controller-timeout' seconds.";
                }
              }

              case controller-connected {
                leaf controller-connected {
                  type empty;
                  mandatory true;
                  description
                    "The controller-connected event object triggers
                     when the connectivity to the controller has been
                     restored after it was lost for at least
                     'controller-timeout' seconds.";
                }
              }
            }

          }
        }
      }
    }
    <CODE ENDS>
```

4.3. LMAP Report YANG Module

This module imports definitions from [RFC6536] and the common LMAP module.

```
<CODE BEGINS> file "ietf-lmap-report@2017-04-21.yang"
module ietf-lmap-report {

  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-lmap-report";
  prefix "lmapr";

  import ietf-yang-types {
    prefix yang;
  }
  import ietf-lmap-common {
    prefix lmap;
  }

  organization
    "IETF Large-Scale Measurement Platforms Working Group";

  contact
    "WG Web:   <http://tools.ietf.org/wg/lmap/>
```

```
        WG List:  <mailto:lmap@ietf.org>

        Editor:   Juergen Schoenwaelder
                  <j.schoenwaelder@jacobs-university.de>

        Editor:   Vaibhav Bajpai
                  <v.bajpai@jacobs-university.de>";

      description
        "This module defines a data model for reporting results from
         measurement agents, which are part of a Large-Scale Measurement
         Platform (LMAP), to result data collectors. This data model is
         expected to be implemented by a collector.";

      revision "2017-04-21" {
        description
          "Initial version";
        reference
          "RFC XXXX: A YANG Data Model for LMAP Measurement Agents";
      }

      rpc report {
        description
          "The report operation is used by an LMAP measurement agent to
           submit measurement results produced by measurement tasks to
           a collector.";

        input {

          leaf date {
            type yang:date-and-time;
            mandatory true;
            description
              "The date and time when this result report was sent to
               a collector.";
          }

          leaf agent-id {
            type yang:uuid;
            description
              "The agent-id of the agent from which this
               report originates.";
          }

          leaf group-id {
            type string;
            description
              "The group-id of the agent from which this
```

```
            report originates.";
        }

        leaf measurement-point {
          type string;
          description
            "The measurement-point of the agent from which this
             report originates.";
        }

        list result {
          description
            "The list of tasks for which results are reported.";

          leaf schedule {
            type lmap:identifier;
            description
              "The name of the schedule that produced the result.";
          }

          leaf action {
            type lmap:identifier;
            description
              "The name of the action in the schedule that produced
               the result.";
          }

          leaf task {
            type lmap:identifier;
            description
              "The name of the task that produced the result.";
          }

          container parameters {
            description
              "This container is a place-holder for run-time
               parameters defined in task-specific data models
               augmenting the base lmap report data model.";

            choice extension {
              description
                "This choice is provided to augment in different
                 sets of parameters.";
            }
          }

          uses lmap:options-grouping {
            description
```

```
          "The list of options there were in use then the
           measurement was performed. This list must include
           both the task specific options as well as the action
           specific options.";
        }

        leaf-list tag {
          type lmap:tag;
          description
            "A tag contains additional information that is passed
             with the result record to the collector. This is the
             joined set of tags defined for the task object, the
             schedule object, and the action object. A tag can be
             used to carry the Measurement Cycle ID.";
        }

        leaf event {
          type yang:date-and-time;
          description
            "The date and time of the event that triggered the
             schedule of the action that produced the reported
             result values. The date and time does not include
             any added randomization.";
        }

        leaf start {
          type yang:date-and-time;
          mandatory true;
          description
            "The date and time when the task producing
             this result started.";
        }

        leaf end {
          type yang:date-and-time;
          description
            "The date and time when the task producing
             this result finished.";
        }

        leaf cycle-number {
          type lmap:cycle-number;
          description
            "The optional cycle number is the time closest to
             the time reported in the event leaf that is a multiple
             of the cycle-interval of the event that triggered the
             execution of the schedule. The value is only present
             if the event that triggered the execution of the
```

```
            schedule has a defined cycle-interval.";
        }

        leaf status {
          type lmap:status-code;
          mandatory true;
          description
            "The status code returned by the execution of this
             action.";
        }

        list conflict {
          description
            "The names of tasks overlapping with the execution
             of the task that has produced this result.";

          leaf schedule-name {
            type lmap:identifier;
            description
              "The name of a schedule that might have impacted
               the execution of the task that has produced this
               result.";
          }

          leaf action-name {
            type lmap:identifier;
            description
              "The name of an action within the schedule that
               might have impacted the execution of the task that
               has produced this result.";
          }

          leaf task-name {
            type lmap:identifier;
            description
              "The name of the task executed by an action within
               the schedule that might have impacted the execution
               of the task that has produced this result.";
          }
        }

        list table {
          description
            "A list of result tables.";

          uses lmap:registry-grouping;

          leaf-list column {
```

```
              type string;
              description
                "An ordered list of column labels. The order is
                 determined by the system and must match the order
                 of the columns in the result rows.";
            }

            list row {
              description
                "The rows of a result table.";

              leaf-list value {
                type string;
                description
                  "The value of a cell in the result row.";
              }
            }
          }
        }
      }
    }
  }
  <CODE ENDS>
```

5. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer, and the mandatory to implement secure transport is SSH [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF or RESTCONF users to a pre-configured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in this YANG module which are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

/lmap/agent This subtree configures general properties of the measurement agent such as its identity, its

	measurement point or controller timeout. This subtree should only have write access for the system responsible to configure the measurement agent.
/lmap/tasks	This subtree configures the tasks that can be invoked by a controller. This subtree should only have write access for the system responsible to configure the measurement agent. Care must be taken to not expose tasks to a controller that can cause damage to the system or the network.
/lmap/schedules	This subtree is used by a controller to define the schedules and actions that are executed when certain events occur. Unauthorized access can cause unwanted load on the device or network or it might direct measurement traffic to targets that become victims of an attack.
/lmap/suppressions	This subtree is used by a controller to define suppressions that can temporarily disable the execution of schedules or actions. Unauthorized access can either disable measurements that should normally take place or it can cause measurements to take place during times when normally no measurements should take place.
/lmap/events	This subtree is used by a controller to define events that trigger the execution of schedules and actions. Unauthorized access can either disable measurements that should normally take place or it can cause measurements to take place during times when normally no measurements should take place or at frequency that is higher than normally expected.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

/lmap/agent	This subtree provides information about the measurement agent. This information may be used to select specific targets for attacks.

/lmap/capabilities	This subtree provides information about the capabilities of the measurement agent, including its software version number and the tasks that it supports. This information may be used to execute targeted attacks against specific implementations.
/lmap/schedules	This subtree provides information about the schedules and their associated actions executed on the measurement agent. This information may be used to check whether attacks against the implementation are effective.
/lmap/suppressions	This subtree provides information about the suppressions that can be active on the measurement agent. This information may be used to predict time periods where measurements take place (or do not take place).

Some of the RPC operations in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. These are the operations and their sensitivity/vulnerability:

/report	The report operation is used to send locally collected measurement results to a remote collector. Unauthorized access may leak measurement results, including from passive measurements.

The data model uses a number of identifiers that are set by the controller. Implementors may find these identifiers useful for the identification of resources, e.g., to identify objects in a filesystem providing temporary storage. Since the identifiers used by the YANG data model may allow characters that may be given special interpretation in a specific context, implementations must ensure that identifiers are properly mapped into safe identifiers.

The data model allows to specify options in the form of name value pairs that are passed to programs. Implementers ought to take care that option names and values are passed literally to programs. In particular, shell expansions that may alter option names and values must not be performed.

6. IANA Considerations

This document registers three URIs in the "IETF XML Registry" [RFC3688]. Following the format in RFC 3688, the following registrations have been made.

```
URI: urn:ietf:params:xml:ns:yang:ietf-lmap-common
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lmap-control
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lmap-report
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.
```

This document registers three YANG modules in the "YANG Module Names" registry [RFC6020].

```
name: ietf-lmap-common
namespace: urn:ietf:params:xml:ns:yang:ietf-lmap-common
prefix: lmap
reference: RFC XXXX

name: ietf-lmap-control
namespace: urn:ietf:params:xml:ns:yang:ietf-lmap-control
prefix: lmapc
reference: RFC XXXX

name: ietf-lmap-report
namespace: urn:ietf:params:xml:ns:yang:ietf-lmap-report
prefix: lmapr
reference: RFC XXXX
```

7. Acknowledgements

Several people contributed to this specification by reviewing early versions and actively participating in the LMAP working group (apologies to those unintentionally omitted): Marcelo Bagnulo, Martin Bjorklund, Trevor Burbridge, Timothy Carey, Alissa Cooper, Philip Eardley, Al Morton, Dan Romascanu, Andrea Soppera, Barbara Stark, and Qin Wu.

Juergen Schoenwaelder and Vaibhav Bajpai worked in part on the Leone research project, which received funding from the European Union Seventh Framework Programme [FP7/2007-2013] under grant agreement number 317647.

Juergen Schoenwaelder and Vaibhav Bajpai were partly funded by Flamingo, a Network of Excellence project (ICT-318488) supported by the European Commission under its Seventh Framework Programme.

8. References

8.1. Normative References

[I-D.ietf-lmap-information-model]
Burbridge, T., Eardley, P., Bagnulo, M., and J. Schoenwaelder, "Information Model for Large-Scale Measurement Platforms (LMAP)", draft-ietf-lmap-information-model-16 (work in progress), January 2017.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <http://www.rfc-editor.org/info/rfc6991>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <http://www.rfc-editor.org/info/rfc7950>.

8.2. Informative References

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., Wu, G., and J. Schoenwaelder, "NETCONF Client and Server Models", draft-ietf-netconf-netconf-client-server-01 (work in progress), November 2016.

[I-D.ietf-netmod-entity]
Bierman, A., Bjorklund, M., Dong, J., and D. Romascanu, "A YANG Data Model for Hardware Management", draft-ietf-netmod-entity-02 (work in progress), January 2017.

[ISO-8601]
International Organization for Standardization, "Data elements and interchange formats -- Information interchange -- Representation of dates and times", ISO Standard 8601:2004, 2004.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <http://www.rfc-editor.org/info/rfc3688>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[RFC5424] Gerhards, R., "The Syslog Protocol", RFC 5424, DOI 10.17487/RFC5424, March 2009, <http://www.rfc-editor.org/info/rfc5424>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <http://www.rfc-editor.org/info/rfc6242>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <http://www.rfc-editor.org/info/rfc6536>.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <http://www.rfc-editor.org/info/rfc7223>.

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <http://www.rfc-editor.org/info/rfc7317>.

[RFC7398] Bagnulo, M., Burbridge, T., Crawford, S., Eardley, P., and A. Morton, "A Reference Path and Measurement Points for Large-Scale Measurement of Broadband Performance", RFC 7398, DOI 10.17487/RFC7398, February 2015, <http://www.rfc-editor.org/info/rfc7398>.

[RFC7594] Eardley, P., Morton, A., Bagnulo, M., Burbridge, T., Aitken, P., and A. Akhter, "A Framework for Large-Scale Measurement of Broadband Performance (LMAP)", RFC 7594, DOI 10.17487/RFC7594, September 2015, <http://www.rfc-editor.org/info/rfc7594>.

Appendix A. Example Parameter Extension Module

Sometimes tasks may require complicated parameters that cannot easily be fit into options, i.e., a list of name/value pairs. In such a situation, it is possible to augment the ietf-lmap-control.yang and ietf-lmap-report.yang data models with definitions for more complex parameters. The following example module demontrates this idea using the parameters of UDP latency metrics as an example (although UDP latency metric parameters do not really need such an extension module).

```
module example-ietf-ippm-udp-latency {

  namespace "urn:example:ietf-ippm-udp-latency";
  prefix "ippm-udp-latency";

  import ietf-inet-types {
    prefix inet;
  }
  import ietf-lmap-control {
    prefix "lmapc";
  }
  import ietf-lmap-report {
    prefix "lmapr";
  }

  grouping ippm-udp-latency-parameter-grouping {
    leaf src-ip {
      type inet:ip-address;
      description
        "The source IP address of the UDP measurement traffic.";
    }

    leaf src-port {
      type inet:port-number;
      description
        "The source port number of the UDP measurement traffic.";
    }

    leaf dst-ip {
      type inet:ip-address;
      description
```

```
          "The destination IP address of the UDP measurement traffic.";
      }

      leaf dst-port {
        type inet:port-number;
        description
          "The destination port number of the UDP measurement traffic.";
      }

      leaf poisson-lambda {
        type decimal64 {
          fraction-digits 4;
        }
        units "seconds";
        default 1.0000;
        description
          "The average interval for the poisson stream with a resolution
           of 0.0001 seconds (0.1 ms).";
      }

      leaf poisson-limit {
        type decimal64 {
          fraction-digits 4;
        }
        units "seconds";
        default 30.0000;
        description
          "The upper limit on the poisson distribution with a resolution
           of 0.0001 seconds (0.1 ms).";
      }
    }

    augment "/lmapc:lmap/lmapc:schedules/lmapc:schedule/lmapc:action"
          + "/lmapc:parameters/lmapc:extension" {
      description
        "This augmentation adds parameters specific to IPPM UDP
         latency metrics to actions.";

      case "ietf-ippm-udp-latency" {
        uses ippm-udp-latency-parameter-grouping;
      }
    }

    augment "/lmapr:report/lmapr:input/lmapr:result"
          + "/lmapr:parameters/lmapr:extension" {
      description
        "This augmentation adds parameters specific to IPPM UDP
         latency metrics to reports.";
```

```
      case "ietf-ippm-udp-latency" {
        uses ippm-udp-latency-parameter-grouping;
      }
    }

  }
```

Appendix B. Example Configuration

```
   <config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <lmap xmlns="urn:ietf:params:xml:ns:yang:ietf-lmap-control">

       <agent>
         <agent-id>550e8400-e29b-41d4-a716-446655440000</agent-id>
         <report-agent-id>true</report-agent-id>
       </agent>

       <schedules>
         <!-- The schedule S1 first updates a list of ping targets
              and subsequently sends a ping to all targets. -->
         <schedule>
           <name>S1</name>
           <start>E1</start>
           <execution-mode>sequential</execution-mode>
           <action>
             <name>A1</name>
             <task>update-ping-targets</task>
           </action>
           <action>
             <name>A2</name>
             <task>ping-all-targets</task>
             <destination>S3</destination>
           </action>
           <suppression-tag>measurement:ping</suppression-tag>
         </schedule>
         <!-- The schedule S2 executes two traceroutes concurrently. -->
         <schedule>
           <name>S2</name>
           <start>E1</start>
           <execution-mode>parallel</execution-mode>
           <action>
             <name>A1</name>
             <task>traceroute</task>
             <option>
               <id>target</id>
               <name>target</name>
               <value>2001:db8::1</value>
```

```
            </option>
            <destination>S3</destination>
          </action>
          <action>
            <name>A2</name>
            <task>traceroute</task>
            <option>
              <id>target</id>
              <name>target</name>
              <value>2001:db8::2</value>
            </option>
            <destination>S3</destination>
          </action>
          <suppression-tag>measurement:traceroute</suppression-tag>
        </schedule>
        <!-- The schedule S3 sends measurement data to a collector. -->
        <schedule>
          <name>S3</name>
          <start>E2</start>
          <action>
            <name>A1</name>
            <task>report</task>
            <option>
              <id>collector</id>
              <name>collector</name>
              <value>https://collector.example.com/</value>
            </option>
          </action>
        </schedule>
      </schedules>

      <suppressions>
        <!-- stop all measurements if we got orphaned -->
        <suppression>
          <name>orphaned</name>
          <start>controller-lost</start>
          <end>controller-connected</end>
          <match>measurement:*</match>
        </suppression>
      </suppressions>

      <tasks>
        <!-- configuration of an update-ping-targets task -->
        <task>
          <name>update-ping-targets</name>
          <program>fping-update-targets</program>
        </task>
        <!-- configuration of a ping-all-targets task -->
```

```
        <task>
          <name>ping-all-targets</name>
          <program>fping</program>
        </task>
        <!-- configuration of a traceroute task -->
        <task>
          <name>traceroute</name>
          <program>mtr</program>
          <option>
            <id>csv</id>
            <name>--csv</name>
          </option>
        </task>
        <!-- configuration of a reporter task -->
        <task>
          <name>report</name>
          <program>lmap-report</program>
        </task>

        <task>
          <name>ippm-udp-latency-client</name>
          <program>ippm-udp-latency</program>
          <function>
            <uri>urn:example:tbd</uri>
            <role>client</role>
          </function>
          <tag>active</tag>
        </task>
      </tasks>

      <events>
        <!-- The event E1 triggers every hour during September 2016
             with a random spread of one minute. -->
        <event>
          <name>E1</name>
          <random-spread>60</random-spread>   <!-- seconds -->
          <periodic>
            <interval>3600000</interval>
            <start>2016-09-01T00:00:00+00:00</start>
            <end>2016-11-01T00:00:00+00:00</end>
          </periodic>
        </event>
        <!-- The event E2 triggers on Mondays at 4am UTC -->
        <event>
          <name>E2</name>
          <calendar>
            <month>*</month>
            <day-of-week>monday</day-of-week>
```

```
            <day-of-month>*</day-of-month>
            <hour>4</hour>
            <minute>0</minute>
            <second>0</second>
            <timezone-offset>+00:00</timezone-offset>
          </calendar>
        </event>
        <!-- The event contoller-lost triggers when we lost
             connectivity with the controller. -->
        <event>
          <name>controller-lost</name>
          <controller-lost/>
        </event>
        <!-- The event contoller-connected triggers when we
             (re)established connectivity with the controller. -->
        <event>
          <name>controller-connected</name>
          <controller-connected/>
        </event>
      </events>
    </lmap>
  </config>
```

Appendix C. Example Report

```
  <rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
       message-id="1">
    <report xmlns="urn:ietf:params:xml:ns:yang:ietf-lmap-report">
      <date>2015-10-28T13:27:42+02:00</date>
      <agent-id>550e8400-e29b-41d4-a716-446655440000</agent-id>
      <result>
        <schedule>S1</schedule>
        <action>A1</action>
        <task>update-ping-targets</task>
        <start>2016-03-21T10:48:55+01:00</start>
        <end>2016-03-21T10:48:57+01:00</end>
        <status>0</status>
      </result>
      <result>
        <schedule>S1</schedule>
        <action>A2</action>
        <task>ping-all-targets</task>
        <start>2016-03-21T10:48:55+01:00</start>
        <end>2016-03-21T10:48:57+01:00</end>
        <status>0</status>
        <table>
          <column>target</column>
```

```
          <column>rtt</column>
          <row>
            <value>2001:db8::1</value>
            <value>42</value>
          </row>
          <row>
            <value>2001:db8::2</value>
            <value>24</value>
          </row>
        </table>
      </result>
      <result>
        <schedule>S2</schedule>
        <action>A1</action>
        <task>traceroute</task>
        <option>
          <id>target</id>
          <name>target</name>
          <value>2001:db8::1</value>
        </option>
        <option>
          <id>csv</id>
          <name>--csv</name>
        </option>
        <start>2016-03-21T10:48:55+01:00</start>
        <end>2016-03-21T10:48:57+01:00</end>
        <status>1</status>
        <table>
          <column>hop</column>
          <column>ip</column>
          <column>rtt</column>
          <row>
            <value>1</value>
            <value>2001:638:709:5::1</value>
            <value>10.5</value>
          </row>
          <row>
            <value>2</value>
            <value>?</value>
            <value></value>
          </row>
        </table>
      </result>
      <result>
        <schedule>S2</schedule>
        <action>A2</action>
        <task>traceroute</task>
        <option>
```

```
          <id>target</id>
          <name>target</name>
          <value>2001:db8::2</value>
        </option>
        <option>
          <id>csv</id>
          <name>--csv</name>
        </option>
        <start>2016-03-21T10:48:55+01:00</start>
        <end>2016-03-21T10:48:57+01:00</end>
        <status>1</status>
        <table>
          <column>hop</column>
          <column>ip</column>
          <column>rtt</column>
          <row>
            <value>1</value>
            <value>2001:638:709:5::1</value>
            <value>11.8</value>
          </row>
          <row>
            <value>2</value>
            <value>?</value>
            <value></value>
          </row>
        </table>
      </result>
    </report>
  </rpc>
```

Appendix D. Change History

Note to the RFC Editor: this section should be removed on publication as an RFC.

D.1. Non-editorial Changes since -07

- Require yang-version 1.1 since we need leaf-lists supporting non-unique values in the report.
- Merged the /lmap-state tree into the /lmap tree.
- Marked state objects as mandatory.
- Added /lmap/agent/report-group-id.

D.2. Non-editorial Changes since -06

- Removed /lmap/agent/device-id and /lmap-state/agent/device-id, added pointer to the ietf-hardware YANG model.
- Removed /lmap-state/agent/{hardware,firmware}, added pointer to the ietf-system YANG model.

D.3. Non-editorial Changes since -05

- Update the example in an attempt to aligned it with the example in the information model.
- Added an extension hook to reports so that task-specific parameters can be echoed back to the collector. Updated the example extension module accordingly.
- Added text and Figure 1 to describe the function and purpose of the three YANG modules.
- Added a cycle-number type definition.
- Added the optional cycle-interval to event definitions.
- Added tags that report additional capabilities of the measurement agent.
- Added event time and cycle-number to the result report.
- Renamed the metrics-grouping to registry-grouping.
- Removed JSON encoding of the examples (they will go into the RESTCONF document).

D.4. Non-editorial Changes since -04

- Tagged /lmap/tasks/task/program with nacm:default-deny-write.
- Added /lmap-state/schedules/schedule/storage and /lmap-state/schedules/schedule/action/storage.
- Removed suppress-by-default.
- Moved the metric list from /report/result into /report/result/table.
- Conflicts are now reported as a triple (schedule, action, task).

- Replaced IPv4 address in the examples with IPv6 addresses.
- Added result/status.

D.5. Non-editorial Changes since -03

- Reworked the reporting data model to align it with the changes in the information model.

D.6. Non-editorial Changes since -02

- Added a mechanism to enforce a runtime limit for schedules.
- Added security considerations text warning about possible shell expansions of options.
- Restricted all user-defined names and tags to lmap:identifier. Added security considerations text to make implementors aware of possible security issues if identifiers are naively mapped to say filesystem paths.
- Schedules and actions now have tags (echoed to the collector) and suppression tags (used for suppression selection).
- Introduced glob-style pattern to match tags.
- Added an example module for IPPM udp latency metrics to demonstrate the usage of the extension mechanism.
- Introduced parameters, an extension point for task/metric specific parameters defined in augmenting YANG modules.
- Introduced the typedefs event-ref, task-ref, and schedule-ref.
- Changed schedule/event to schedule/start and added the optional schedule/stop and schedule/duration leafs.

D.7. Non-editorial Changes since -01

- Updated and split examples (config vs state vs report).
- Refactored the definitions so that common definitions used by both the control and report data models are in the new module ietf-lmap-common.
- A report is submitted via an RPC operation instead of using a notification.

- The default execution mode is pipelined.
- Clarified which action consumes data in sequential, pipelines, and parallel execution mode.
- Added /lmap/agent/measurement-point, /lmap/agent/report-measurement-point, and /report/measurement-point to configure and report the measurement point.
- Turned /lmap/suppression into a list /lmap/suppressions/suppression that uses a start and stop event to define the beginning and end of a suppression period.
- Added controller-lost an controller-ok event choices to /lmap/events/event.
- Added a metrics-grouping to identify entries in a metric registry and associated roles.
- Added /lmap-state/schedules to report the status of schedules and their actions. Refactored /lmap-state/tasks to only report the task capabilities.

D.8. Non-editorial Changes since -00

- A task can now reference multiply registry entries.
- Schedules are triggered by Events instead of Timings; Timings are just one of many possible event sources.
- Actions feed into other Schedules (instead of Actions within other Schedules).
- Removed the notion of multiple task outputs.
- Support for sequential, parallel, and pipelined execution of Actions.

Authors' Addresses

Juergen Schoenwaelder
Jacobs University Bremen

Email: j.schoenwaelder@jacobs-university.de

Vaibhav Bajpai
Jacobs University Bremen

Email: v.bajpai@jacobs-university.de

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 24, 2016

A. Morton
AT&T Labs
M. Bagnulo
UC3M
P. Eardley
BT
K. D'Souza
AT&T Labs

February 21, 2016

Initial Performance Metric Registry Entries
draft-morton-ippm-initial-registry-04

Abstract

This memo defines the Initial Entries for the Performance Metrics Registry.

Version 04 * All section 4 parameters reference YANG types for alternate data formats. * Discussion has concluded that usecase(s) for machine parse-able registry columns are not needed.

Still need: * suggestion of standard naming format for parameters. * revisions that follow section 4 changes in other proposed metrics.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

Table of Contents

1. Introduction

Note: Efforts to synchronize structure and terminology with [I-D.ietf-ippm-metric-registry] will likely be incomplete until both drafts are stable.

This memo proposes an initial set of entries for the Performance Metric Registry. It uses terms and definitions from the IPPM literature, primarily [RFC2330]. Proponents of Passive Performance Metrics are encouraged to develop a similar document.

Although there are several standard templates for organizing specifications of performance metrics (see [RFC2679] for an example of the traditional IPPM template, based to large extent on the Benchmarking Methodology Working Group's traditional template in [RFC1242], and see [RFC6390] for a similar template), none of these templates were intended to become the basis for the columns of an IETF-wide registry of metrics. While examinating aspects of metric specifications which need to be registered, it became clear that none of the existing metric templates fully satisfies the particular needs of a registry.

Therefore, [I-D.ietf-ippm-metric-registry] defines the overall format for a Performance Metric Registry. Section 5 of [I-D.ietf-ippm-metric-registry] also gives guidelines for those requesting registration of a Metric, that is the creation of entry(s) in the Performance Metric Registry: "In essence, there needs to be evidence that a candidate Registered Performance Metric has significant industry interest, or has seen deployment, and there is agreement that the candidate Registered Performance Metric serves its intended purpose." The process in [I-D.ietf-ippm-metric-registry] also requires that new entries are administered by IANA through Expert Review, which will ensure that the metrics are tightly defined.

2. Scope

This document defines the initial set of Performance Metrics Registry entries, for which IETF approval (following development in the IP Performance Metrics (IPPM) Working Group) will satisfy the requirement for Expert Review. Note that all are Active Performance Metrics, which are based on RFCs prepared in the IPPM working group of the IETF, according to their framework [RFC2330] and its updates.

3. Registry Categories and Columns

This section provides the categories and columns of the registry, for easy reference. An entry (row) therefore gives a complete description of a Registered Metric.

Registry Categories and Columns, shown as

	Category	
	Column	Column

Summary

ID	Name	URIs	Description

Metric Definition

Reference Definition	Fixed Parameters

Method of Measurement

Reference Method	Packet Generation Stream	Traffic Filter	Sampling dist.	Run-time Param	Role

Output

Type	Reference Definition	Units

Administrative information

Status	Request	Rev	Rev.Date

Comments and Remarks

4. UDP Round-trip Latency Registry Entry

This section gives an initial registry entry for the UDP Round-trip Latency.

Note: Each Registry entry only produces a "raw" output or a statistical summary. To describe both "raw" and one or more statistics efficiently, the Identifier, Name, and Output Categories can be split and this section can become two or more closely-related metrics. See Section 7 for an example specifying multiple Registry entries with many common columns.

4.1. Summary

This category includes multiple indexes to the registry entry: the element ID and metric name.

4.1.1. ID (Identifier)

<insert a numeric identifier, an integer, TBD>

4.1.2. Name

<insert name according to metric naming convention>

Act_IP_UDP_Round-trip_Delay_Poisson_95th-percentile

4.1.3. URIs

URN: Prefix urn:ietf:params:performance:metric...<name>

URL: http://<TBD by IANA>/<name>

4.1.4. Description

This metric assesses the delay of a stream of packets exchanged between two hosts (which are the two measurement points), and the Output is the Round-trip delay for all successfully exchanged packets expressed as the 95th percentile of their conditional delay distribution.

4.2. Metric Definition

This category includes columns to prompt the entry of all necessary details related to the metric definition, including the RFC reference and values of input factors, called fixed parameters.

4.2.1. Reference Definition

<Full bibliographic reference to an immutable doc.>

Almes, G., Kalidindi, S., and M. Zekauskas, "A Round-trip Delay Metric for IPPM", RFC 2681, September 1999.

[RFC2681]

<specific section reference and additional clarifications, if needed>

Section 2.4 of [RFC2681] provides the reference definition of the singleton (single value) Round-trip delay metric. Section 3.4 of [RFC2681] provides the reference definition expanded to cover a multi-value sample. Note that terms such as singleton and sample are defined in Section 11 of [RFC2330].

Note that although the definition of "Round-trip-Delay between Src and Dst" is directionally ambiguous in the text, this metric tightens the definition further to recognize that the host in the "Src" role will send the first packet to "Dst", and ultimately receive the corresponding return packet from "Dst" (when neither are lost).

Finally, note that the variable "dT" is used in [RFC2681] to refer to the value of Round-trip delay in metric definitions and methods. The variable "dT" has been re-used in other IPPM literature to refer to different quantities, and cannot be used as a global variable name.

4.2.2. Fixed Parameters

<list and specify Fixed Parameters, input factors that must be determined and embedded in the measurement system for use when needed>

Type-P:

- IPv4 header values:
  - DSCP: set to 0
  - TTL: set to 255
  - Protocol: Set to 17 (UDP)
- IPv6 header values:
  - DSCP: set to 0
  - Hop Count: set to 255
  - Protocol: Set to 17 (UDP)
- UDP header values:
  - Checksum: the checksum MUST be calculated

- UDP Payload
  - total of 9 bytes

Other measurement parameters:

- Tmax: a loss threshold waiting time
  - 3.0, expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 5 (see section 9.3 of [RFC6020]) and with resolution of 0.0001 seconds (0.1 ms), with lossless conversion to/from the 32-bit NTP timestamp as per section 6 of [RFC5905].

4.3. Method of Measurement

This category includes columns for references to relevant sections of the RFC(s) and any supplemental information needed to ensure an unambiguous methods for implementations.

4.3.1. Reference Method

<for metric, insert relevant section references and supplemental info>

The methodology for this metric is defined as Type-P-Round-trip-Delay-Poisson-Stream in section 2.6 of RFC 2681 [RFC2681] and section 3.6 of RFC 2681 [RFC2681] using the Type-P and Tmax defined under Fixed Parameters.

The reference method distinguishes between long-delayed packets and lost packets by implementing a maximum waiting time for packet arrival. Tmax is the waiting time used as the threshold to declare a packet lost. Lost packets SHALL be designated as having undefined delay.

The calculations on the delay (RTT) SHALL be performed on the conditional distribution, conditioned on successful packet arrival within Tmax. Also, when all packet delays are stored, the process which calculates the RTT value MAY enforce the Tmax threshold on stored values before calculations. See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

The reference method requires some way to distinguish between different packets in a stream to establish correspondence between sending times and receiving times for each successfully-arriving

packet. Sequence numbers or other send-order identification MUST be retained at the Src or included with each packet to dis-ambiguate packet reordering if it occurs.

If a standard measurement protocol is employed, then the measurement process will determine the sequence numbers or timestamps applied to test packets after the Fixed and Runtime parameters are passed to that process. The chosen measurement protocol will dictate the format of sequence numbers and time-stamps, if they are conveyed in the packet payload.

Refer to Section 4.4 of [RFC6673] for expanded discussion of the instruction to "send a Type-P packet back to the Src as quickly as possible" in Section 2.6 of RFC 2681 [RFC2681]. Section 8 of [RFC6673] presents additional requirements which MUST be included in the method of measurement for this metric.

4.3.2. Packet Generation Stream

This section gives the details of the packet traffic which is the basis for measurement. In IPPM metrics, this is called the Stream, and can easily be described by providing the list of stream parameters.

<section/specification references, and description of any new generation parameters, if needed>

Section 11.1.3 of [RFC2330] provides three methods to generate Poisson sampling intervals. the reciprocal of lambda is the average packet spacing, thus the Run-time Parameter is Reciprocal_lambda = 1/ lambda, in seconds.

>>> Check with Sam, most likely it is this...

Method 3 SHALL be used, where given a start time (Run-time Parameter), the subsequent send times are all computed prior to measurement by computing the pseudo-random distribution of inter-packet send times, (truncating the distribution as specified in the Run-time Parameter, Trunc), and the Src sends each packet at the computed times.

Note that Trunc is the upper limit on inter-packet times in the Poisson distribution. A random value greater than Trunc is set equal to Trunc instead.

4.3.3. Traffic Filtering (observation) Details

The measured results based on a filtered version of the packets observed, and this section provides the filter details (when present).

<section reference>.

NA

4.3.4. Sampling Distribution

<insert time distribution details, or how this is diff from the filter>

NA

4.3.5. Run-time Parameters and Data Format

Run-time Parameters are input factors that must be determined, configured into the measurement system, and reported with the results for the context to be complete.

<list of run-time parameters, and their data formats>

Src the IP address of the host in the Src Role (format ipv4-address-no-zone value for IPv4, or ipv6-address-no-zone value for IPv6, see Section 4 of [RFC6991])

Dst the IP address of the host in the Dst Role (format ipv4-address-no-zone value for IPv4, or ipv6-address-no-zone value for IPv6, see section 4 of [RFC6991])

T0 a time, the start of a measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of [RFC6991]). The UTC Time Zone is required by Section 6.1 of [RFC2330]. When T0 is "all-zeros", a start time is unspecified and Tf is to be interpreted as the Duration of the measurement interval. The start time is controlled through other means.

Tf a time, the end of a measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of [RFC6991]). The UTC Time Zone is required by Section 6.1 of [RFC2330]. When T0 is "all-zeros", a end time date is ignored and Tf is interpreted as the Duration of the measurement interval.

Reciprocal_lambda average packet interval for Poisson Streams expressed in units of seconds, as a positive value of type

decimal64 with fraction digits = 5 (see section 9.3 of [RFC6020]) with resolution of 0.0001 seconds (0.1 ms), and with lossless conversion to/from the 32-bit NTP timestamp as per section 6 of [RFC5905].

Trunc Upper limit on Poisson distribution expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 5 (see section 9.3 of [RFC6020]) with resolution of 0.0001 seconds (0.1 ms), and with lossless conversion to/from the 32-bit NTP timestamp as per section 6 of [RFC5905] (values above this limit will be clipped and set to the limit value). (if fixed, Trunc = 30.0000 seconds.)

>>> should Poisson run-time params be fixed instead? probably yes if modeling a specific version of MBA tests.

4.3.6. Roles

<lists the names of the different roles from the measurement method>

Src launches each packet and waits for return transmissions from Dst.

Dst waits for each packet from Src and sends a return packet to Src.

4.4. Output

This category specifies all details of the Output of measurements using the metric.

4.4.1. Type

<insert name of the output type, raw or a selected summary statistic>

Percentile -- for the conditional distribution of all packets with a valid value of Round-trip delay (undefined delays are excluded), a single value corresponding to the 95th percentile, as follows:

See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

The percentile = 95, meaning that the reported delay, "Percentile95", is the smallest value of Round-trip delay for which the Empirical Distribution Function (EDF), F(Percentile95) >= 95% of the singleton Round-trip delay values in the conditional distribution. See section 11.3 of [RFC2330] for the definition of the percentile statistic using the EDF.

4.4.2. Data Format

<describe the data format for each type of result>

For all outputs ---

T0 the start of a measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of [RFC6991]). The UTC Time Zone is required by Section 6.1 of [RFC2330].

Tf the start of a measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of [RFC6991]). The UTC Time Zone is required by Section 6.1 of [RFC2330].

Raw -- REMOVED IN VERSION 01

For Act_IP_UDP_Round-trip_Delay_Poisson_95th-percentile:

Percentile95 The time value of the result is expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 9 (see section 9.3 of [RFC6020]) with resolution of 0.000000001 seconds (1.0 ns), and with lossless conversion to/from the 64-bit NTP timestamp as per section 6 of RFC [RFC5905]

4.4.3. Reference

<pointer to section/spec where output type/format is defined>

See the Data Format column for references.

4.4.4. Metric Units

<insert units for the measured results, and the reference specification>.

The 95th Percentile of Round-trip Delay is expressed in seconds.

4.5. Administrative items

4.5.1. Status

<current or deprecated>

4.5.2. Requestor (keep?)

name or RFC, etc.

4.5.3. Revision

1.0

4.5.4. Revision Date

YYYY-MM-DD

4.6. Comments and Remarks

Additional (Informational) details for this entry

5. Packet Delay Variation Registry Entry

This section gives an initial registry entry for a Packet Delay Variation metric.

Note: If each Registry entry should only produce a "raw" output or a statistical summary, then the "Output" Category can be split and this section can become two closely-related metrics.

5.1. Summary

This category includes multiple indexes to the registry entries, the element ID and metric name.

<skipping some Summary columns for now>

5.1.1. ID (Identifier)

<insert numeric identifier, an integer>

5.1.2. Name

<insert name according to metric naming convention>

Act_IP-UDP-One-way-pdv-95th-percentile-Poisson

URL: ??

5.1.3. URI

URI: Prefix urn:ietf:params:performance:metric<add name>

5.1.4. Description

An assessment of packet delay variation with respect to the minimum delay observed on the stream.

5.2. Metric Definition

This category includes columns to prompt the entry of all necessary details related to the metric definition, including the RFC reference and values of input factors, called fixed parameters.

5.2.1. Reference Definition

<Full bibliographic reference to an immutable doc.>

Paxson, V., Almes, G., Mahdavi, J., and M. Mathis, "Framework for IP Performance Metrics", RFC 2330, May 1998. [RFC2330]

Demichelis, C. and P. Chimento, "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)", RFC 3393, November 2002. [RFC3393]

Morton, A. and B. Claise, "Packet Delay Variation Applicability Statement", RFC 5481, March 2009. [RFC5481]

Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.[RFC5905]

<specific section reference and additional clarifications, if needed>

See sections 2.4 and 3.4 of [RFC3393]. Singleton delay differences measured are referred to by the variable name "ddT".

5.2.2. Fixed Parameters

<list and specify Fixed Parameters, input factors that must be determined and embedded in the measurement system for use when needed>

- F, a selection function defining unambiguously the packets from the stream selected for the metric. See section 4.2 of [RFC5481] for the PDV form.

- L, a packet length in bits. L = 200 bits.

- Tmax, a maximum waiting time for packets to arrive at Dst, set sufficiently long to disambiguate packets with long delays from packets that are discarded (lost). Tmax = 3 seconds.

- Type-P, as defined in [RFC2330], which includes any field that may affect a packet's treatment as it traverses the network. The packets are IP/UDP, with DSCP = 0 (BE).

5.3. Method of Measurement

This category includes columns for references to relevant sections of the RFC(s) and any supplemental information needed to ensure an unambiguous methods for implementations.

5.3.1. Reference Method

<for metric, insert relevant section references and supplemental info>

See section 2.6 and 3.6 of [RFC3393] for singleton elements.

5.3.2. Packet Generation Stream

<list of generation parameters and section/spec references if needed>

Poisson distributed as described in [RFC2330], with the following Parameters.

- lambda, a rate in reciprocal seconds (for Poisson Streams). lambda = 1 packet per second

- Upper limit on Poisson distribution (values above this limit will be clipped and set to the limit value). Upper limit = 30 seconds.

5.3.3. Traffic Filtering (observation) Details

<insert the measured results based on a filtered version of the packets observed, and this section provides the filter details (when present), and section reference>.

NA

5.3.4. Sampling Distribution

<insert time distribution details, or how this is diff from the filter>

NA

5.3.5. Run-time Parameters and Data Format

<list of run-time parameters, and any reference(s)>.

- Src, the IP address of a host (32-bit value for IPv4, 128-bit value for IPv6)

- Dst, the IP address of a host (32-bit value for IPv4, 128-bit value for IPv6)

- T, a time (start of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905]). When T0 is "all-zeros", a start time is unspecified and Tf is to be interpreted as the Duration of the measurement interval.

- Tf, a time (end of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905]), interpreted as the Duration of the measurement interval.

5.3.6. Roles

<lists the names of the different roles from the measurement method>

Src - the host that sends the stream of packets.

Dst - the host that receives the stream of packets.

5.4. Output

This category specifies all details of the Output of measurements using the metric.

5.4.1. Type/Value (two diff terms used)

<insert name of the output type, raw or a selected summary statistic>

Raw -- for each packet sent, pairs of values.

Percentile -- for the conditional distribution of all packets with a valid value of one-way delay (undefined delays are excluded), a

single value corresponding to the 95th percentile of the singletons, ddT.

5.4.2. Data Format

<describe the data format for each type of result>

For all Output types

- T, a time (start of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905])

- Tf, a time (end of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905])

Raw -

- T1, the wire time of the first packet in a pair, measured at MP(Src) as it leaves for Dst (64-bit NTP Timestamp Format, see section 6 of [RFC5905]).

- T2, the wire time of the second packet in a pair, measured at MP(Src) as it leaves for Dst (64-bit NTP Timestamp Format, see section 6 of [RFC5905]).

- I(i),I(i+1), i >=0, pairs of times which mark the beginning and ending of the intervals in which the packet stream from which the measurement is taken occurs. Here, I(0) = T0 and assuming that n is the largest index, I(n) = Tf (pairs of 64-bit NTP Timestamp Format, see section 6 of [RFC5905]).

- When the one-way delay of a packet in the calculation pair for ddT is undefined, then ddT is undefined for that pair.

Percentile -- for the conditional distribution of all packets with a valid value of one-way delay (undefined delays are excluded), a single value as follows:

See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

See section 4.3 of [RFC3393] for details on the percentile statistic (where pdv should be substituted for "ipdv").

The percentile = 95.

Data format is a 32-bit signed floating point value, *similar to* the 32-bit short NTP Time format in Section 6 of [RFC5905] and is as follows: the first 16 bits represent the *signed* integer number of seconds; the next 16 bits represent the fractional part of a second.

5.4.3. Reference

<pointer to section/spec where output type/format is defined>

see Data Format column.

5.4.4. Metric Units

<insert units for the measured results, and the reference specification>.

See section 3.3 of [RFC3393] for singleton elements, ddT. The units are seconds, and the same units are used for 95th percentile.

[RFC2330] recommends that when a time is given, it will be expressed in UTC.

The timestamp format (for T, Tf, etc.) is the same as in [RFC5905] (64 bits) and is as follows: the first 32 bits represent the unsigned integer number of seconds elapsed since 0h on 1 January 1900; the next 32 bits represent the fractional part of a second that has elapsed since then.

5.5. Administrative items

5.5.1. Status

<current or depricated>

5.5.2. Requestor (keep?)

<name of individual or RFC, etc.>

5.5.3. Revision

1.0

5.5.4. Revision Date

YYYY-MM-DD

5.6. Comments and Remarks

<Additional (Informational) details for this entry>

Lost packets represent a challenge for delay variation metrics. See section 4.1 of [RFC3393] and the delay variation applicability statement[RFC5481] for extensive analysis and comparison of PDV and an alternate metric, IPDV.

6. DNS Response Latency Registry Entry

This section gives an initial registry entry for DNS Response Latency. RFC 2681 [RFC2681] defines a Round-trip delay metric. We build on that metric by specifying several of the input parameters to precisely define a metric for measuring DNS latency.

6.1. Summary

This category includes multiple indexes to the registry entries, the element ID and metric name.

<skipping some admin columns for now>

6.1.1. ID (Identifier)

<insert numeric identifier, an integer>

6.1.2. Name

<insert name according to metric naming convention>

URL: ??

6.1.3. URI

URI: Prefix urn:ietf:params:performance:metric

6.1.4. Description

This metric assesses the response time, the interval from the query transmission to the response.

6.2. Metric Definition

This category includes columns to prompt the entry of all necessary details related to the metric definition, including the RFC reference and values of input factors, called fixed parameters.

6.2.1. Reference Definition

<Full bibliographic reference to an immutable doc.>

Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987. (and updates)

[RFC1035]

Almes, G., Kalidindi, S., and M. Zekauskas, "A Round-trip Delay Metric for IPPM", RFC 2681, September 1999.

[RFC2681]

<specific section reference and additional clarifications, if needed>

Section 2.4 of [RFC2681] provides the reference definition of the singleton (single value) Round-trip delay metric. Section 3.4 of [RFC2681] provides the reference definition expanded to cover a multi-value sample. Note that terms such as singleton and sample are defined in Section 11 of [RFC2330].

For DNS Response Latency, the entities in [RFC1035] must be mapped to [RFC2681]. The Local Host with its User Program and Resolver take the role of "Src", and the Foreign Name Server takes the role of "Dst".

Note that although the definition of "Round-trip-Delay between Src and Dst at T" is directionally ambiguous in the text, this metric tightens the definition further to recognize that the host in the "Src" role will send the first packet to "Dst", and ultimately receive the corresponding return packet from "Dst" (when neither are lost).

6.2.2. Fixed Parameters

<list and specify Fixed Parameters, input factors that must be determined and embedded in the measurement system for use when needed>

Type-P:

- IPv4 header values:
  - DSCP: set to 0
  - TTL set to 255

- Protocol: Set to 17 (UDP)

- UDP header values:
  - Source port: 53
  - Destination port: 53
  - Checksum: the checksum must be calculated

- Payload: The payload contains a DNS message as defined in RFC 1035 [RFC1035] with the following values:
  - The DNS header section contains:
    - QR: set to 0 (Query)
    - OPCODE: set to 0 (standard query)
    - AA: not set
    - TC: not set
    - RD: set to one (recursion desired)
    - RA: not set
    - RCODE: not set
    - QDCOUNT: set to one (only one entry)
    - ANCOUNT: not set
    - NSCOUNT: not set
    - ARCOUNT: not set
  - The Question section contains:
    - QNAME: the FQDN provided as input for the test
    - QTYPE: the query type provided as input for the test
    - QCLASS: set to IN
  - The other sections do not contain any Resource Records.

Observation: reply packets will contain a DNS response and may contain RRs.

Timeout: Tmax = 5 seconds (to help disambiguate queries)

6.3. Method of Measurement

This category includes columns for references to relevant sections of the RFC(s) and any supplemental information needed to ensure an unambiguous methods for implementations.

6.3.1. Reference Method

<for metric, insert relevant section references and supplemental info>

The methodology for this metric is defined as Type-P-Round-trip-Delay-Poisson-Stream in section 2.6 of RFC 2681 [RFC2681] and section 3.6 of RFC 2681 [RFC2681] using the Type-P and Timeout defined under Fixed Parameters.

The method requires sequence numbers or other send-order information to be retained at the Src or included with each packet to dis-ambiguate packet reordering if it occurs. Sequence number is part of the payload described under Fixed Parameters.

DNS Messages bearing Queries provide for random ID Numbers, so more than one query may be launched while a previous request is outstanding when the ID Number is used.

IF a DNS response does not arrive within Tmax, the result is undefined. The Message ID SHALL be used to disambiguate the successive queries.

>>> This would require support of ID generation and population in the Message. An alternative would be to use a random Source port on the Query Message, but we would choose ONE before proceding.

Refer to Section 4.4 of [RFC6673] for expanded discussion of the instruction to "send a Type-P packet back to the Src as quickly as possible" in Section 2.6 of RFC 2681 [RFC2681]. Section 8 of [RFC6673] presents additional requirements which shall be included in the method of measurement for this metric.

6.3.2. Packet Generation Stream

This section gives the details of the packet traffic which is the basis for measurement. In IPPM metrics, this is called the Stream, and can easily be dscribed by providing the list of stream parameters.

<list of generation parameters and section/spec references if needed>

Section 11.1.3 of RFC 2681 [RFC2330] provides three methods to generate Poisson sampling intervals. the reciprocal of lambda is the average packet rate, thus the Run-time Parameter is 1/lambda.

>>> Check with Sam, most likely it is this...

Method 3 is used, where given a start time (Run-time Parameter), the subsequent send times are all computed prior to measurement by computing the pseudo-random distribution of inter-packet send times, (truncating the distribution as specified in the Run-time Parameters), and the Src sends each packet at the computed times.

6.3.3. Traffic Filtering (observation) Details

The measured results based on a filtered version of the packets observed, and this section provides the filter details (when present).

<section reference>.

NA

6.3.4. Sampling Distribution

<insert time distribution details, or how this is diff from the filter>

NA

6.3.5. Run-time Parameters and Data Format

Run-time Parameters are input factors that must be determined, configured into the measurement system, and reported with the results for the context to be complete.

<list of run-time parameters, and their data formats>

- Src, the IP address of a host (32-bit value for IPv4, 128-bit value for IPv6)

- Dst, the IP address of a host (32-bit value for IPv4, 128-bit value for IPv6)

- T0, a time (start of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905]). When T0 is "all-zeros", a start time is unspecified and Tf is to be interpreted as the Duration of the measurement interval.

- Tf, a time (end of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905]), interpreted as the Duration of the measurement interval.

- 1/lambda, average packet rate (for Poisson Streams). (1/lambda = 0.1 packet per second, if fixed)

- Upper limit on Poisson distribution (values above this limit will be clipped and set to the limit value). (if fixed, Upper limit = 300 seconds.)

- ID, the 16-bit identifier assigned by the program that generates the query, and which must vary in successive queries, see Section 4.1.1 of [RFC1035]. This identifier is copied into the corresponding reply and can be used by the requester to match-up replies to outstanding queries.

The format for 1/lambda and Upper limit of Poisson Dist. are the short format in [RFC5905] (32 bits) and is as follows: the first 16 bits represent the integer number of seconds; the next 16 bits represent the fractional part of a second.

>>> should Poisson run-time params be fixed instead? probably yes if modeling a specific version of MBA tests.

6.3.6. Roles

<lists the names of the different roles from the measurement method>

Src - launches each packet and waits for return transmissions from Dst.

Dst - waits for each packet from Src and sends a return packet to Src.

6.4. Output

This category specifies all details of the Output of measurements using the metric.

6.4.1. Type/Value (two diff terms used)

<insert name of the output type, raw or a selected summary statistic>

For all output types:

- T0, a time (start of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905])

- Tf, a time (end of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905])

Raw -- for each packet sent, pairs of values.

>>> and the status of the response, only assigning values to successful query-response pairs.

Percentile -- for the conditional distribution of all packets with a valid value of Round-trip delay (undefined delays are excluded), a single value corresponding to the 95th percentile.

6.4.2. Data Format

<describe the data format for each type of result>

Raw -- for each packet sent, pairs of values as follows:

- T, the time when the packet was sent from Src, 128-bit NTP Date Format, see section 6 of [RFC5905])

- dT, a value of Round-trip delay, format is *similar to* the 32-bit short NTP Time format in Section 6 of [RFC5905] and is as follows: the first 16 bits represent the *signed* integer number of seconds; the next 16 bits represent the fractional part of a second.

- dT is undefined when the packet is not received at Src in waiting time Tmxax seconds (need undefined code for no-response or un-successful response)

Percentile -- for the conditional distribution of all packets with a valid value of Round-trip delay (undefined delays are excluded), a single value as follows:

See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

See section 4.3 of [RFC3393] for details on the percentile statistic (where Round-trip delay should be substituted for "ipdv").

The percentile = 95.

Data format is a 32-bit signed floating point value, *similar to* the 32-bit short NTP Time format in Section 6 of [RFC5905] and is as follows: the first 16 bits represent the *signed* integer number of seconds; the next 16 bits represent the fractional part of a second.

6.4.3. Reference

<pointer to section/spec where output type/format is defined>

See the Data Format column for references.

6.4.4. Metric Units

<insert units for the measured results, and the reference specification>.

Round-trip Delay, dT, is expressed in seconds.

The 95th Percentile of Round-trip Delay is expressed in seconds.

6.5. Administrative items

6.5.1. Status

<current or depricated>

6.5.2. Requestor (keep?)

name or RFC, etc.

6.5.3. Revision

1.0

6.5.4. Revision Date

YYYY-MM-DD

6.6. Comments and Remarks

Additional (Informational) details for this entry

7. UDP Poisson One-way Delay Registry Entries

This section gives an initial registry entry for the UDP Poisson One-way Delay.

Note: Each Registry "Name" below specifies a single registry entry, whose output format varies according to a component of the name that specifies one form of statistical summary.

IANA is asked to assign a different numeric identifiers to each Name. All column entries beside the Summary and Output categories are the same, thus this section proposes five closely-related registry entries. As a result, IANA is also asked to assign corresponding URIs and URLs.

7.1. Summary

This category includes multiple indexes to the registry entries, the element ID and metric name.

7.1.1. ID (Identifier)

<insert numeric identifier, an integer, one corresponding to each name below>

7.1.2. Name

<insert name according to metric naming convention>

Act_IP_UDP_Poisson_UDP-Payload-250_One-way_Delay_<statistic>

Act_IP_UDP_Poisson_UDP-Payload-250_One-way_Delay_Percentile95

Act_IP_UDP_Poisson_UDP-Payload-250_One-way_Delay_Mean

Act_IP_UDP_Poisson_UDP-Payload-250_One-way_Delay_Min

Act_IP_UDP_Poisson_UDP-Payload-250_One-way_Delay_Max

Act_IP_UDP_Poisson_UDP-Payload-250_One-way_Delay_Std_Dev

7.1.3. URI and URL

URI: Prefix urn:ietf:params:performance:metric...<name>

URL: http:\\www.iana.org\ ... <name>

7.1.4. Description

This metric assesses the delay of a stream of packets exchanged between two hosts (or measurement points), and reports the <statistic> One-way delay for all successfully exchanged packets based on their conditional delay distribution.

7.2. Metric Definition

This category includes columns to prompt the entry of all necessary details related to the metric definition, including the RFC reference and values of input factors, called fixed parameters.

7.2.1. Reference Definition

<Full bibliographic reference to an immutable doc.>

Almes, G., Kalidindi, S., and M. Zekauskas, "A One-way Delay Metric for IPPM", RFC 2679, September 1999.

[RFC2679]

Morton, A., and Stephan, E., "Spatial Composition of Metrics", RFC 6049, January 2011.

[RFC6049]

<specific section reference and additional clarifications, if needed>

Section 3.4 of [RFC2679] provides the reference definition of the singleton (single value) One-way delay metric. Section 4.4 of [RFC2679] provides the reference definition expanded to cover a multi-value sample. Note that terms such as singleton and sample are defined in Section 11 of [RFC2330].

Only successful packet transfers with finite delay are included in the sample, as prescribed in section 4.1.2 of [RFC6049].

NOTE: RFC2679 will be replaced by 2679-bis on approval, see draft-ietf-ippm-2679-bis-01.

7.2.2. Fixed Parameters

<list and specify Fixed Parameters, input factors that must be determined and embedded in the measurement system for use when needed>

Type-P:

- IPv4 header values:
  - DSCP: set to 0
  - TTL set to 255
  - Protocol: Set to 17 (UDP)
- UDP header values:
  - Checksum: the checksum must be calculated
- UDP Payload: TWAMP Test Packet Formats, Section 4.1.2 of [RFC5357]
  - Security features in use influence the number of Padding octets.
  - 250 octets total, including the TWAMP format

Timeout, Tmax: 3 seconds

7.3. Method of Measurement

This category includes columns for references to relevant sections of the RFC(s) and any supplemental information needed to ensure an unambiguous methods for implementations.

7.3.1. Reference Method

<for metric, insert relevant section references and supplemental info>

The methodology for this metric is defined as Type-P-One-way-Delay-Poisson-Stream in section 3.6 of [RFC2679] and section 4.6 of [RFC2679] using the Type-P and Timeout defined under Fixed Parameters.

The method requires sequence numbers or other send-order information to be retained at the Src or included with each packet to dis-ambiguate packet reordering if it occurs. Sequence number is part of the TWAMP payload described under Fixed Parameters.

7.3.2. Packet Generation Stream

This section gives the details of the packet traffic which is the basis for measurement. In IPPM metrics, this is called the Stream, and can easily be dscribed by providing the list of stream parameters.

<list of generation parameters and section/spec references if needed>

Section 11.1.3 of RFC 2681 [RFC2330] provides three methods to generate Poisson sampling intervals. The reciprocal of lambda is the average packet rate, thus the Run-time Parameter is 1/lambda.

Method 3 or equivalent SHALL used, where given a start time (Run-time Parameter), the subsequent send times are all computed prior to measurement by computing the pseudo-random distribution of inter-packet send times, (truncating the distribution as specified in the Run-time Parameters), and the Src sends each packet at the computed times.

7.3.3. Traffic Filtering (observation) Details

NA

7.3.4. Sampling Distribution

NA

7.3.5. Run-time Parameters and Data Format

Run-time Parameters are input factors that must be determined, configured into the measurement system, and reported with the results for the context to be complete.

<list of run-time parameters, and their data formats>

- Src, the IP address of a host (32-bit value for IPv4, 128-bit value for IPv6)

- Dst, the IP address of a host (32-bit value for IPv4, 128-bit value for IPv6)

- T0, a time (start of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905]). When T0 is "all-zeros", a start time is unspecified and Tf is to be interpreted as the Duration of the measurement interval.

- Tf, a time (end of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905]), interpreted as the Duration of the measurement interval.

- 1/lambda, average packet rate (for Poisson Streams). (1/lambda = 1 packet per second, if fixed)

- Upper limit on Poisson distribution (values above this limit will be clipped and set to the limit value). (if fixed, Upper limit = 30 seconds.)

The format for 1/lambda and Upper limit of Poisson Dist. are the short format in [RFC5905] (32 bits) and is as follows: the first 16 bits represent the integer number of seconds; the next 16 bits represent the fractional part of a second.

>>> should Poisson run-time params be fixed instead? probably yes if modeling a specific version of tests. Note in the NAME, i.e. Poisson3.3

7.3.6. Roles

<lists the names of the different roles from the measurement method>

Src - launches each packet and waits for return transmissions from Dst. This is the TWAMP Session-Sender.

Dst - waits for each packet from Src and sends a return packet to Src. This is the TWAMP Session-Reflector.

7.4. Output

This category specifies all details of the Output of measurements using the metric.

7.4.1. Type/Value (two diff terms used)

<insert name of the output type, raw or a selected summary statistic>

See subsection titles below for Types.

7.4.2. Data Format

<describe the data format for each type of result>

For all output types ---

- T0, a time (start of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905])

- Tf, a time (end of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905])

7.4.2.1. Percentile95

The 95th percentile SHALL be calculated using the conditional distribution of all packets with a finite value of One-way delay (undefined delays are excluded), a single value as follows:

See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

See section 4.3 of [RFC3393] for details on the percentile statistic (where Round-trip delay should be substituted for "ipdv").

The percentile = 95.

Data format is a 32-bit signed value, *similar to* the 32-bit short NTP Time format in Section 6 of [RFC5905] and is as follows: the first 16 bits represent the *signed* integer number of seconds; the next 16 bits represent the fractional part of a second.

7.4.2.2. Mean

The mean SHALL be calculated using the conditional distribution of all packets with a finite value of One-way delay (undefined delays are excluded), a single value as follows:

See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

See section 4.2.2 of [RFC6049] for details on calculating this statistic, and 4.2.3 of [RFC6049].

Data format is a 32-bit signed value, *similar to* the 32-bit short NTP Time format in Section 6 of [RFC5905] and is as follows: the first 16 bits represent the *signed* integer number of seconds; the next 16 bits represent the fractional part of a second.

7.4.2.3. Min

The minimum SHALL be calculated using the conditional distribution of all packets with a finite value of One-way delay (undefined delays are excluded), a single value as follows:

See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

See section 4.3.2 of [RFC6049] for details on calculating this statistic, and 4.3.3 of [RFC6049].

Data format is a 32-bit signed value, *similar to* the 32-bit short NTP Time format in Section 6 of [RFC5905] and is as follows: the first 16 bits represent the *signed* integer number of seconds; the next 16 bits represent the fractional part of a second.

7.4.2.4. Max

The maximum SHALL be calculated using the conditional distribution of all packets with a finite value of One-way delay (undefined delays are excluded), a single value as follows:

See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

See section 4.3.2 of [RFC6049] for a closely related method for calculating this statistic, and 4.3.3 of [RFC6049]. The formula is as follows:

```
Max = (FiniteDelay [j])

      such that for some index, j, where 1 <= j <= N
      FiniteDelay[j] >= FiniteDelay[n] for all n
```

Data format is a 32-bit signed value, *similar to* the 32-bit short NTP Time format in Section 6 of [RFC5905] and is as follows: the first 16 bits represent the *signed* integer number of seconds; the next 16 bits represent the fractional part of a second.

7.4.2.5. Std_Dev

7.4.3. Reference

<pointer to section/spec where output type/format is defined>

See the Data Format column for references.

7.4.4. Metric Units

<insert units for the measured results, and the reference specification>.

The <statistic> of One-way Delay is expressed in seconds.

The 95th Percentile of One-way Delay is expressed in seconds.

7.5. Administrative items

7.5.1. Status

<current or depricated>

7.5.2. Requestor (keep?)

name or RFC, etc.

7.5.3. Revision

1.0

7.5.4. Revision Date

YYYY-MM-DD

7.6. Comments and Remarks

Additional (Informational) details for this entry

8. UDP Periodic One-way Delay Registry Entries

This section gives an initial registry entry for the UDP Periodic One-way Delay.

Note: Each Registry "Name" below specifies a single registry entry, whose output format varies according to a component of the name that specifies one form of statistical summary.

IANA is asked to assign a different numeric identifiers to each Name. All other column entries are the same, thus this section is proposes five closely-related registry entries. As a result, IANA is also asked to assign corresponding URIs and URLs.

8.1. Summary

This category includes multiple indexes to the registry entries, the element ID and metric name.

8.1.1. ID (Identifier)

<insert numeric identifier, an integer, one corresponding to each name below>

8.1.2. Name

<insert name according to metric naming convention>

Act_IP_UDP_Periodic-var_UDP-Payload-142_One-way_Delay_<statistic>

Act_IP_UDP_Periodic-var_UDP-Payload-142_One-way_Delay_Percentile95

Act_IP_UDP_Periodic-var_UDP-Payload-142_One-way_Delay_Mean

Act_IP_UDP_Periodic-var_UDP-Payload-142_One-way_Delay_Min

Act_IP_UDP_Periodic-var_UDP-Payload-142_One-way_Delay_Max

Act_IP_UDP_Periodic-var_UDP-Payload-142_One-way_Delay_Std_Dev

8.1.3. URI and URL

URI: Prefix urn:ietf:params:performance:metric...<name>

URL: http:\\www.iana.org\ ... <name>

8.1.4. Description

This metric assesses the delay of a stream of packets exchanged between two hosts (or measurement points), and reports the <statistic> One-way delay for all successfully exchanged packets based on their conditional delay distribution.

8.2. Metric Definition

This category includes columns to prompt the entry of all necessary details related to the metric definition, including the RFC reference and values of input factors, called fixed parameters.

8.2.1. Reference Definition

<Full bibliographic reference to an immutable doc.>

Almes, G., Kalidindi, S., and M. Zekauskas, "A One-way Delay Metric for IPPM", RFC 2679, September 1999.

[RFC2679]

Morton, A., and Stephan, E., "Spatial Composition of Metrics", RFC 6049, January 2011.

[RFC6049]

<specific section reference and additional clarifications, if needed>

Section 3.4 of [RFC2679] provides the reference definition of the singleton (single value) One-way delay metric. Section 4.4 of [RFC2679] provides the reference definition expanded to cover a multi-value sample. Note that terms such as singleton and sample are defined in Section 11 of [RFC2330].

Only successful packet transfers with finite delay are included in the sample, as prescribed in section 4.1.2 of [RFC6049].

NOTE: RFC2679 will be replaced by 2679-bis on approval, see draft-ietf-ippm-2679-bis-01.

ANY other conditions, ...

8.2.2. Fixed Parameters

<list and specify Fixed Parameters, input factors that must be determined and embedded in the measurement system for use when needed>

Type-P:

- IPv4 header values:
  - DSCP: set to 0
  - TTL set to 255
  - Protocol: Set to 17 (UDP)
- UDP header values:
  - Checksum: the checksum must be calculated
- UDP Payload: TWAMP Test Packet Formats, Section 4.1.2 of [RFC5357]
  - Security features in use influence the number of Padding octets.
  - 142 octets total, including the TWAMP format

Timeout, Tmax: 3 seconds

8.3. Method of Measurement

This category includes columns for references to relevant sections of the RFC(s) and any supplemental information needed to ensure an unambiguous methods for implementations.

8.3.1. Reference Method

<for metric, insert relevant section references and supplemental info>

The methodology for this metric is defined as Type-P-One-way-Delay-Poisson-Stream in section 3.6 of [RFC2679] and section 4.6 of [RFC2679] using the Type-P and Timeout defined under Fixed Parameters.

The method requires sequence numbers or other send-order information to be retained at the Src or included with each packet to dis-ambiguate packet reordering if it occurs. Sequence number is part of the TWAMP payload described under Fixed Parameters.

8.3.2. Packet Generation Stream

This section gives the details of the packet traffic which is the basis for measurement. In IPPM metrics, this is called the Stream, and can easily be dscribed by providing the list of stream parameters.

<list of generation parameters and section/spec references if needed>

Section 3 of [RFC3432] prescribes the method for generating Periodic streams using associated parameters.

- incT, the nominal duration of inter-packet interval, first bit to first bit
- dT, the duration of the interval for allowed sample start times
- T0, the actual start time

NOTE: an initiation process with a number of control exchanges resulting in unpredictable start times (within a time interval) may be sufficient to avoid synchronization of periodic streams, and therefore a valid replacement for selecting a start time at random from a fixed interval.

These stream parameters will be specified as Run-time parameters.

8.3.3. Traffic Filtering (observation) Details

NA

8.3.4. Sampling Distribution

NA

8.3.5. Run-time Parameters and Data Format

Run-time Parameters are input factors that must be determined, configured into the measurement system, and reported with the results for the context to be complete.

<list of run-time parameters, and their data formats>

- Src, the IP address of a host (32-bit value for IPv4, 128-bit value for IPv6)

- Dst, the IP address of a host (32-bit value for IPv4, 128-bit value for IPv6)

- T0, a time (start of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905]). When T0 is "all-zeros", a start time is unspecified and Tf is to be interpreted as the Duration of the measurement interval.

- Tf, a time (end of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905]), interpreted as the Duration of the measurement interval.

- incT, the nominal duration of inter-packet interval, first bit to first bit

- dT, the duration of the interval for allowed sample start times

The format for incT and dT are the short format in [RFC5905] (32 bits) and is as follows: the first 16 bits represent the integer number of seconds; the next 16 bits represent the fractional part of a second.

>>> should Periodic run-time params be fixed instead? probably yes if modeling a specific version of tests. Note in the NAME, i.e. Poisson3.3

8.3.6. Roles

<lists the names of the different roles from the measurement method>

Src - launches each packet and waits for return transmissions from Dst. This is the TWAMP Session-Sender.

Dst - waits for each packet from Src and sends a return packet to Src. This is the TWAMP Session-Reflector.

8.4. Output

This category specifies all details of the Output of measurements using the metric.

8.4.1. Type/Value (two diff terms used)

<insert name of the output type, raw or a selected summary statistic>

See subsection titles in Data Format for Types.

8.4.2. Data Format

<describe the data format for each type of result>

For all output types ---

- T0, a time (start of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905])

- Tf, a time (end of measurement interval, 128-bit NTP Date Format, see section 6 of [RFC5905])

8.4.2.1. Percentile95

The 95th percentile SHALL be calculated using the conditional distribution of all packets with a finite value of One-way delay (undefined delays are excluded), a single value as follows:

See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

See section 4.3 of [RFC3393] for details on the percentile statistic (where Round-trip delay should be substituted for "ipdv").

The percentile = 95.

Data format is a 32-bit signed value, *similar to* the 32-bit short NTP Time format in Section 6 of [RFC5905] and is as follows: the first 16 bits represent the *signed* integer number of seconds; the next 16 bits represent the fractional part of a second.

8.4.2.2. Mean

The mean SHALL be calculated using the conditional distribution of all packets with a finite value of One-way delay (undefined delays are excluded), a single value as follows:

See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

See section 4.2.2 of [RFC6049] for details on calculating this statistic, and 4.2.3 of [RFC6049].

Data format is a 32-bit signed value, *similar to* the 32-bit short NTP Time format in Section 6 of [RFC5905] and is as follows: the first 16 bits represent the *signed* integer number of seconds; the next 16 bits represent the fractional part of a second.

8.4.2.3. Min

The minimum SHALL be calculated using the conditional distribution of all packets with a finite value of One-way delay (undefined delays are excluded), a single value as follows:

See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

See section 4.3.2 of [RFC6049] for details on calculating this statistic, and 4.3.3 of [RFC6049].

Data format is a 32-bit signed value, *similar to* the 32-bit short NTP Time format in Section 6 of [RFC5905] and is as follows: the first 16 bits represent the *signed* integer number of seconds; the next 16 bits represent the fractional part of a second.

8.4.2.4. Max

The maximum SHALL be calculated using the conditional distribution of all packets with a finite value of One-way delay (undefined delays are excluded), a single value as follows:

See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

See section 4.3.2 of [RFC6049] for a closely related method for calculating this statistic, and 4.3.3 of [RFC6049]. The formula is as follows:

```
Max = (FiniteDelay [j])

      such that for some index, j, where 1 <= j <= N
      FiniteDelay[j] >= FiniteDelay[n] for all n
```

Data format is a 32-bit signed value, *similar to* the 32-bit short NTP Time format in Section 6 of [RFC5905] and is as follows: the first 16 bits represent the *signed* integer number of seconds; the next 16 bits represent the fractional part of a second.

8.4.2.5. Std_Dev

8.4.3. Reference

<pointer to section/spec where output type/format is defined>

See the Data Format column for references.

8.4.4. Metric Units

<insert units for the measured results, and the reference specification>.

The <statistic> of One-way Delay is expressed in seconds.

8.5. Administrative items

8.5.1. Status

<current or depricated>

8.5.2. Requestor (keep?)

name or RFC, etc.

8.5.3. Revision

1.0

8.5.4. Revision Date

YYYY-MM-DD

8.6. Comments and Remarks

Additional (Informational) details for this entry

9. partly BLANK Registry Entry

This section gives an initial registry entry for ....

9.1. Summary

This category includes multiple indexes to the registry entries, the element ID and metric name.

<skipping the admin columns for now>

9.1.1. ID (Identifier)

<insert numeric identifier, an integer>

9.1.2. Name

<insert name according to metric naming convention>

URL: ??

9.1.3. URI

URI: Prefix urn:ietf:params:performance:metric

9.1.4. Description

TBD.

9.2. Metric Definition

This category includes columns to prompt the entry of all necessary details related to the metric definition, including the RFC reference and values of input factors, called fixed parameters.

9.2.1. Reference Definition

<Full bibliographic reference to an immutable doc.>

Almes, G., Kalidindi, S., and M. Zekauskas, "A Round-trip Delay Metric for IPPM", RFC 2681, September 1999.

<specific section reference and additional clarifications, if needed>

Section 2.4 of [RFC2681] provides the reference definition of the singleton (single value) Round-trip delay metric. Section 3.4 of [RFC2681] provides the reference definition expanded to cover a multi-value sample. Note that terms such as singleton and sample are defined in Section 11 of [RFC2330].

Note that although the definition of "Round-trip-Delay between Src and Dst at T" is directionally ambiguous in the text, this metric tightens the definition further to recognize that the host in the "Src" role will send the first packet to "Dst", and ultimately receive the corresponding return packet from "Dst" (when neither are lost).

<<< Check how the Methodology also makes this clear (or not) >>>

9.2.2. Fixed Parameters

<list and specify Fixed Parameters, input factors that must be determined and embedded in the measurement system for use when needed>

Type-P:

- IPv4 header values:
  - DSCP: set to 0
  - TTL set to 255
  - Protocol: Set to 17 (UDP)
- UDP header values:
  - Checksum: the checksum must be calculated
- Payload
  - Sequence number: 8-byte integer
  - Timestamp: 8 byte integer. Expressed as 64-bit NTP timestamp as per section 6 of RFC 5905 [RFC5905]
  - No padding (total of 9 bytes)

Timeout: 3 seconds

9.3. Method of Measurement

This category includes columns for references to relevant sections of the RFC(s) and any supplemental information needed to ensure an unambiguous methods for implementations.

9.3.1. Reference Method

<for metric, insert relevant section references and supplemental info>

9.3.2. Packet Generation Stream

This section gives the details of the packet traffic which is the basis for measurement. In IPPM metrics, this is called the Stream, and can easily be dscribed by providing the list of stream parameters.

<list of generation parameters and section/spec references if needed>

9.3.3. Traffic Filtering (observation) Details

The measured results based on a filtered version of the packets observed, and this section provides the filter details (when present).

<section reference>.

9.3.4. Sampling Distribution

<insert time distribution details, or how this is diff from the filter>

9.3.5. Run-time Parameters and Data Format

Run-time Parameters are input factors that must be determined, configured into the measurement system, and reported with the results for the context to be complete.

<list of run-time parameters>

<reference(s)>.

9.3.6. Roles

<lists the names of the different roles from the measurement method>

9.4. Output

This category specifies all details of the Output of measurements using the metric.

9.4.1. Type/Value (two diff terms used)

<insert name of the output type, raw or a selected summary statistic>

9.4.2. Data Format

<describe the data format for each type of result>

- Value:
- Data Format: (There may be some precedent to follow here, but otherwise use 64-bit NTP Timestamp Format, see section 6 of [RFC5905]).
- Reference: <section reference>

9.4.3. Reference

<pointer to section/spec where output type/format is defined>

9.4.4. Metric Units

<insert units for the measured results, and the reference specification>.

9.5. Administrative items

9.5.1. Status

<current or depricated>

9.5.2. Requestor (keep?)

name or RFC, etc.

9.5.3. Revision

1.0

9.5.4. Revision Date

YYYY-MM-DD

9.6. Comments and Remarks

Additional (Informational) details for this entry

10. BLANK Registry Entry

This section gives an initial registry entry for ....

10.1. Summary

This category includes multiple indexes to the registry entries, the element ID and metric name.

<skipping the Summary columns for now>

10.1.1. ID (Identifier)

<insert numeric identifier, an integer>

10.1.2. Name

<insert name according to metric naming convention>

URL: ??

10.1.3. URI

URI: Prefix urn:ietf:params:performance:metric

10.1.4. Description

TBD.

10.2. Metric Definition

This category includes columns to prompt the entry of all necessary details related to the metric definition, including the RFC reference and values of input factors, called fixed parameters.

10.2.1. Reference Definition

<Full bibliographic reference to an immutable doc.>

<specific section reference and additional clarifications, if needed>

10.2.2. Fixed Parameters

<list and specify Fixed Parameters, input factors that must be determined and embedded in the measurement system for use when needed>

10.3. Method of Measurement

This category includes columns for references to relevant sections of the RFC(s) and any supplemental information needed to ensure an unambiguous methods for implementations.

10.3.1. Reference Method

<for metric, insert relevant section references and supplemental info>

10.3.2. Packet Generation Stream

<list of generation parameters and section/spec references if needed>

10.3.3. Traffic Filtering (observation) Details

<insert the measured results based on a filtered version of the packets observed, and this section provides the filter details (when present), and section reference>.

10.3.4. Sampling Distribution

<insert time distribution details, or how this is diff from the filter>

10.3.5. Run-time Parameters and Data Format

<list of run-time parameters, and any reference(s)>.

10.3.6. Roles

<lists the names of the different roles from the measurement method>

10.4. Output

This category specifies all details of the Output of measurements using the metric.

10.4.1. Type/Value (two diff terms used)

<insert name of the output type, raw or a selected summary statistic>

10.4.2. Data Format

<describe the data format for each type of result>

10.4.3. Reference

<pointer to section/spec where output type/format is defined>

10.4.4. Metric Units

<insert units for the measured results, and the reference specification>.

10.5. Administrative items

10.5.1. Status

<current or depricated>

10.5.2. Requestor (keep?)

<name of individual or RFC, etc.>

10.5.3. Revision

1.0

10.5.4. Revision Date

YYYY-MM-DD

10.6. Comments and Remarks

Additional (Informational) details for this entry

11. Example RTCP-XR Registry Entry

This section is MAY BE DELETED or adapted before submission.

This section gives an example registry entry for the end-point metric described in RFC 7003 [RFC7003], for RTCP-XR Burst/Gap Discard Metric reporting.

11.1. Registry Indexes

This category includes multiple indexes to the registry entries, the element ID and metric name.

11.1.1. Identifier

An integer having enough digits to uniquely identify each entry in the Registry.

11.1.2. Name

A metric naming convention is TBD.

11.1.3. URI

Prefix urn:ietf:params:performance:metric

11.1.4. Status

current

11.1.5. Requestor

Alcelip Mornuley

11.1.6. Revision

1.0

11.1.7. Revision Date

2014-07-04

11.1.8. Description

TBD.

11.1.9. Reference Specification(s)

[RFC3611][RFC4566][RFC6776][RFC6792][RFC7003]

11.2. Metric Definition

This category includes columns to prompt the entry of all necessary details related to the metric definition, including the RFC reference and values of input factors, called fixed parameters. Section 3.2 of [RFC7003] provides the reference information for this category.

11.2.1. Reference Definition

Packets Discarded in Bursts:

The total number of packets discarded during discard bursts. The measured value is unsigned value. If the measured value exceeds 0xFFFFFD, the value 0xFFFFFE MUST be reported to indicate an over-range measurement. If the measurement is unavailable, the value 0xFFFFFF MUST be reported.

11.2.2. Fixed Parameters

Fixed Parameters are input factors that must be determined and embedded in the measurement system for use when needed. The values of these parameters is specified in the Registry.

Threshold: 8 bits, set to value = 3 packets.

The Threshold is equivalent to Gmin in [RFC3611], i.e., the number of successive packets that must not be discarded prior to and following a discard packet in order for this discarded packet to be regarded as part of a gap. Note that the Threshold is set in accordance with the Gmin calculation defined in Section 4.7.2 of [RFC3611].

Interval Metric flag: 2 bits, set to value 11=Cumulative Duration

This field is used to indicate whether the burst/gap discard metrics are Sampled, Interval, or Cumulative metrics [RFC6792]:

I=10: Interval Duration - the reported value applies to the most recent measurement interval duration between successive metrics reports.

I=11: Cumulative Duration - the reported value applies to the accumulation period characteristic of cumulative measurements.

Senders MUST NOT use the values I=00 or I=01.

11.3. Method of Measurement

This category includes columns for references to relevant sections of the RFC(s) and any supplemental information needed to ensure an unambiguous methods for implementations. For the Burst/Gap Discard Metric, it appears that the only guidance on methods of measurement is in Section 3.0 of [RFC7003] and its supporting references. Relevant information is repeated below, although there appears to be no section titled "Method of Measurement" in [RFC7003].

11.3.1. Reference Method

Metrics in this block report on burst/gap discard in the stream arriving at the RTP system. Measurements of these metrics are made at the receiving end of the RTP stream. Instances of this metrics block use the synchronization source (SSRC) to refer to the separate auxiliary Measurement Information Block [RFC6776], which describes measurement periods in use (see [RFC6776], Section 4.2).

This metrics block relies on the measurement period in the Measurement Information Block indicating the span of the report. Senders MUST send this block in the same compound RTCP packet as the Measurement Information Block. Receivers MUST verify that the measurement period is received in the same compound RTCP packet as this metrics block. If not, this metrics block MUST be discarded.

11.3.2. Stream Type and Stream Parameters

Since RTCP-XR Measurements are conducted on live RTP traffic, the complete description of the stream is contained in SDP messages that proceed the establishment of a compatible stream between two or more communicating hosts. See Run-time Parameters, below.

11.3.3. Output Type and Data Format

The output type defines the type of result that the metric produces.

- Value: Packets Discarded in Bursts
- Data Format: 24 bits
- Reference: Section 3.2 of [RFC7003]

11.3.4. Metric Units

The measured results are apparently expressed in packets, although there is no section of [RFC7003] titled "Metric Units".

11.3.5. Run-time Parameters and Data Format

Run-Time Parameters are input factors that must be determined, configured into the measurement system, and reported with the results for the context to be complete. However, the values of these parameters is not specified in the Registry, rather these parameters are listed as an aid to the measurement system implementor or user (they must be left as variables, and supplied on execution).

The Data Format of each Run-time Parameter SHALL be specified in this column, to simplify the control and implementation of measurement devices.

SSRC of Source: 32 bits As defined in Section 4.1 of [RFC3611].

SDP Parameters: As defined in [RFC4566]

Session description v= (protocol version number, currently only 0)

o= (originator and session identifier : username, id, version number, network address)

s= (session name : mandatory with at least one UTF-8-encoded character)

i=* (session title or short information) u=* (URI of description)

e=* (zero or more email address with optional name of contacts)

p=* (zero or more phone number with optional name of contacts)

c=* (connection information--not required if included in all media)

b=* (zero or more bandwidth information lines) One or more Time descriptions ("t=" and "r=" lines; see below)

z=* (time zone adjustments)

k=* (encryption key)

a=* (zero or more session attribute lines)

Zero or more Media descriptions (each one starting by an "m=" line; see below)

m= (media name and transport address)

i=* (media title or information field)

c=* (connection information -- optional if included at session level)

b=* (zero or more bandwidth information lines)

k=* (encryption key)

a=* (zero or more media attribute lines -- overriding the Session attribute lines)

An example Run-time SDP description follows:

v=0

o=jdoe 2890844526 2890842807 IN IP4 192.0.2.5

s=SDP Seminar i=A Seminar on the session description protocol

u=http://www.example.com/seminars/sdp.pdf e=j.doe@example.com (Jane Doe)

c=IN IP4 233.252.0.12/127

t=2873397496 2873404696

a=recvonly

m=audio 49170 RTP/AVP 0

m=video 51372 RTP/AVP 99

a=rtpmap:99 h263-1998/90000

11.4. Comments and Remarks

TBD.

12. Revision History

This section may be removed for publication. It contains partial information on updtes.

This draft replaced draft-mornuley-ippm-initial-registry.

In version 02, Section 4 has been edited to reflect recent discussion on the ippm-list: * Removed the combination or "Raw" and left 95th percentile. * Hanging Indent on Run-time parameters (Fixed parameters use bullet lists and other indenting formats. * Payload format for

measurement has been removed. * Explanation of Conditional delay distribution.

Version 03 addressed Phil Eardley's comments and suggestions in sections 1-4. and resolved the definition of Percentiles.

Version 04 * All section 4 parameters reference YANG types for alternate data formats. * Discussion has concluded that usecase(s) for machine parse-able registry columns are not needed.

Still need: * suggestion of standard naming format for parameters.

Note: lambda parameter description is correct in section 4, elsewhere needs fix.

13. Security Considerations

These registry entries represent no known security implications for Internet Security. Each referenced Metric contains a Security Considerations section.

14. IANA Considerations

IANA is requested to populate The Performance Metric Registry defined in [I-D.ietf-ippm-metric-registry] with the values defined above.

<more is needed here>

15. Acknowledgements

The authors thank Brian Trammell for suggesting the term "Run-time Parameters", which led to the distinction between run-time and fixed parameters implemented in this memo, for identifying the IPFIX metric with Flow Key as an example, and for many other productive suggestions. Thanks to Peter Koch, who provided several useful suggestions for disambiguating successive DNS Queries in the DNS Response time metric.

The authors also acknowledge the constructive reviews and helpful suggestions from Barbara Stark, Juergen Schoenwaelder, Tim Carey, and participants in the LMAP working group.

16. References

16.1. Normative References

[I-D.ietf-ippm-metric-registry]
Bagnulo, M., Claise, B., Eardley, P., and A. Morton, "Registry for Performance Metrics", Internet Draft (work in progress) draft-ietf-ippm-metric-registry, 2014.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <http://www.rfc-editor.org/info/rfc1035>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2330] Paxson, V., Almes, G., Mahdavi, J., and M. Mathis, "Framework for IP Performance Metrics", RFC 2330, DOI 10.17487/RFC2330, May 1998, <http://www.rfc-editor.org/info/rfc2330>.

[RFC2679] Almes, G., Kalidindi, S., and M. Zekauskas, "A One-way Delay Metric for IPPM", RFC 2679, DOI 10.17487/RFC2679, September 1999, <http://www.rfc-editor.org/info/rfc2679>.

[RFC2680] Almes, G., Kalidindi, S., and M. Zekauskas, "A One-way Packet Loss Metric for IPPM", RFC 2680, DOI 10.17487/RFC2680, September 1999, <http://www.rfc-editor.org/info/rfc2680>.

[RFC2681] Almes, G., Kalidindi, S., and M. Zekauskas, "A Round-trip Delay Metric for IPPM", RFC 2681, DOI 10.17487/RFC2681, September 1999, <http://www.rfc-editor.org/info/rfc2681>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <http://www.rfc-editor.org/info/rfc3339>.

[RFC3393] Demichelis, C. and P. Chimento, "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)", RFC 3393, DOI 10.17487/RFC3393, November 2002, <http://www.rfc-editor.org/info/rfc3393>.

[RFC3432] Raisanen, V., Grotefeld, G., and A. Morton, "Network performance measurement with periodic streams", RFC 3432, DOI 10.17487/RFC3432, November 2002, <http://www.rfc-editor.org/info/rfc3432>.

[RFC4737] Morton, A., Ciavattone, L., Ramachandran, G., Shalunov, S., and J. Perser, "Packet Reordering Metrics", RFC 4737, DOI 10.17487/RFC4737, November 2006, <http://www.rfc-editor.org/info/rfc4737>.

[RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, DOI 10.17487/RFC5357, October 2008, <http://www.rfc-editor.org/info/rfc5357>.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <http://www.rfc-editor.org/info/rfc5905>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC6049] Morton, A. and E. Stephan, "Spatial Composition of Metrics", RFC 6049, DOI 10.17487/RFC6049, January 2011, <http://www.rfc-editor.org/info/rfc6049>.

[RFC6673] Morton, A., "Round-Trip Packet Loss Metrics", RFC 6673, DOI 10.17487/RFC6673, August 2012, <http://www.rfc-editor.org/info/rfc6673>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <http://www.rfc-editor.org/info/rfc6991>.

16.2. Informative References

[Brow00] Brownlee, N., "Packet Matching for NeTraMet Distributions", March 2000.

[RFC1242] Bradner, S., "Benchmarking Terminology for Network Interconnection Devices", RFC 1242, DOI 10.17487/RFC1242, July 1991, <http://www.rfc-editor.org/info/rfc1242>.

[RFC3611] Friedman, T., Ed., Caceres, R., Ed., and A. Clark, Ed., "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, DOI 10.17487/RFC3611, November 2003, <http://www.rfc-editor.org/info/rfc3611>.

[RFC4148] Stephan, E., "IP Performance Metrics (IPPM) Metrics Registry", BCP 108, RFC 4148, DOI 10.17487/RFC4148, August 2005, <http://www.rfc-editor.org/info/rfc4148>.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <http://www.rfc-editor.org/info/rfc4566>.

[RFC5472] Zseby, T., Boschi, E., Brownlee, N., and B. Claise, "IP Flow Information Export (IPFIX) Applicability", RFC 5472, DOI 10.17487/RFC5472, March 2009, <http://www.rfc-editor.org/info/rfc5472>.

[RFC5477] Dietz, T., Claise, B., Aitken, P., Dressler, F., and G. Carle, "Information Model for Packet Sampling Exports", RFC 5477, DOI 10.17487/RFC5477, March 2009, <http://www.rfc-editor.org/info/rfc5477>.

[RFC5481] Morton, A. and B. Claise, "Packet Delay Variation Applicability Statement", RFC 5481, DOI 10.17487/RFC5481, March 2009, <http://www.rfc-editor.org/info/rfc5481>.

[RFC6248] Morton, A., "RFC 4148 and the IP Performance Metrics (IPPM) Registry of Metrics Are Obsolete", RFC 6248, DOI 10.17487/RFC6248, April 2011, <http://www.rfc-editor.org/info/rfc6248>.

[RFC6390] Clark, A. and B. Claise, "Guidelines for Considering New Performance Metric Development", BCP 170, RFC 6390, DOI 10.17487/RFC6390, October 2011, <http://www.rfc-editor.org/info/rfc6390>.

[RFC6703] Morton, A., Ramachandran, G., and G. Maguluri, "Reporting IP Network Performance Metrics: Different Points of View", RFC 6703, DOI 10.17487/RFC6703, August 2012, <http://www.rfc-editor.org/info/rfc6703>.

[RFC6776] Clark, A. and Q. Wu, "Measurement Identity and Information Reporting Using a Source Description (SDES) Item and an RTCP Extended Report (XR) Block", RFC 6776, DOI 10.17487/RFC6776, October 2012, <http://www.rfc-editor.org/info/rfc6776>.

[RFC6792] Wu, Q., Ed., Hunt, G., and P. Arden, "Guidelines for Use of the RTP Monitoring Framework", RFC 6792, DOI 10.17487/RFC6792, November 2012, <http://www.rfc-editor.org/info/rfc6792>.

[RFC7003] Clark, A., Huang, R., and Q. Wu, Ed., "RTP Control Protocol (RTCP) Extended Report (XR) Block for Burst/Gap Discard Metric Reporting", RFC 7003, DOI 10.17487/RFC7003, September 2013, <http://www.rfc-editor.org/info/rfc7003>.

[RFC7594] Eardley, P., Morton, A., Bagnulo, M., Burbridge, T., Aitken, P., and A. Akhter, "A Framework for Large-Scale Measurement of Broadband Performance (LMAP)", RFC 7594, DOI 10.17487/RFC7594, September 2015, <http://www.rfc-editor.org/info/rfc7594>.

Authors' Addresses

Al Morton
AT&T Labs
200 Laurel Avenue South
Middletown,, NJ 07748
USA

Phone: +1 732 420 1571
Fax: +1 732 368 1192
Email: acmorton@att.com
URI: http://home.comcast.net/~acmacm/

Marcelo Bagnulo
Universidad Carlos III de Madrid
Av. Universidad 30
Leganes, Madrid 28911
SPAIN

Phone: 34 91 6249500
Email: marcelo@it.uc3m.es
URI: http://www.it.uc3m.es

Philip Eardley
BT
Adastral Park, Martlesham Heath
Ipswich
ENGLAND

Email: philip.eardley@bt.com

Kevin D'Souza
AT&T Labs
200 Laurel Avenue South
Middletown,, NJ 07748
USA

Phone: +1 732 420 xxxx
Email: kld@att.com