

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.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Terminology	2
1.2. Tree Diagrams	3
2. Data Model Overview	3
3. Relationship to the Information Model	8
4. YANG Modules	10
4.1. LMAP Common YANG Module	10
4.2. LMAP Control YANG Module	18
4.3. LMAP Report YANG Module	39
5. Security Considerations	44
6. IANA Considerations	46
7. Acknowledgements	47
8. References	48
8.1. Normative References	48
8.2. Informative References	48
Appendix A. Example Parameter Extension Module	50
Appendix B. Example Configuration	52
Appendix C. Example Report	55
Appendix D. Change History	57
D.1. Non-editorial Changes since -07	57
D.2. Non-editorial Changes since -06	58
D.3. Non-editorial Changes since -05	58
D.4. Non-editorial Changes since -04	58
D.5. Non-editorial Changes since -03	59
D.6. Non-editorial Changes since -02	59
D.7. Non-editorial Changes since -01	59
D.8. Non-editorial Changes since -00	60
Authors' Addresses	60

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:

- o Brackets "[" and "]" enclose list keys.
- o Abbreviations before data node names: "rw" means configuration (read-write), "ro" means state data (read-only), and "w" means RPC input data (write-only).
- o Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o 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.

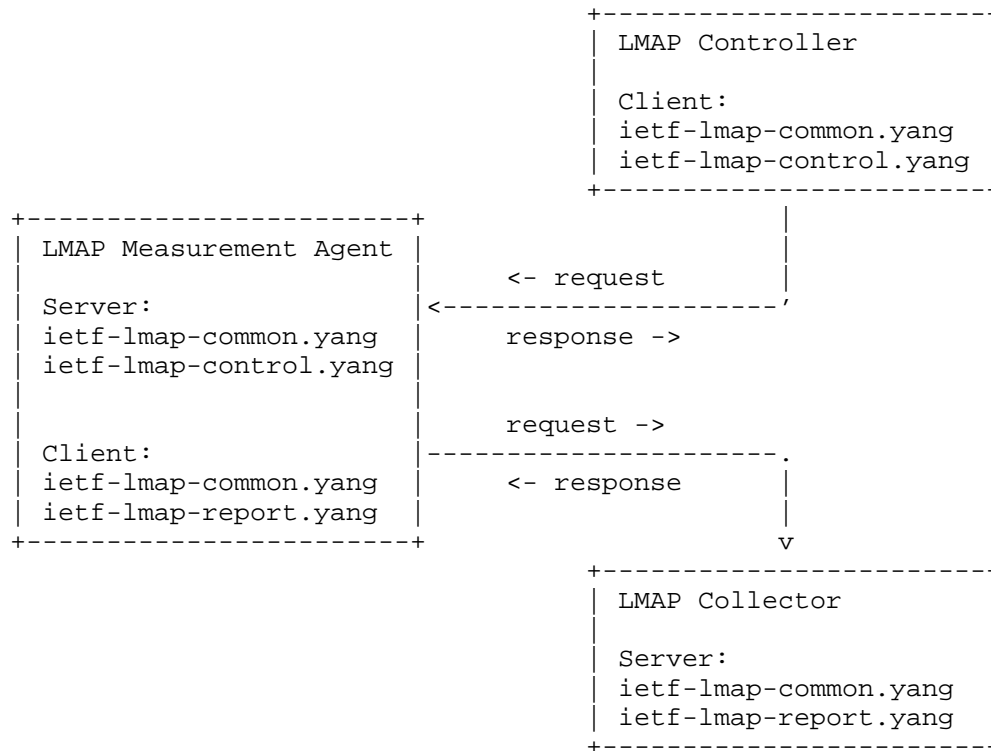


Figure 1: LMAP Controller, Measurement Agents, and Collector and the YANG modules they implement as client or server

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:

- o 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.

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

- o Schedules: Schedules are modeled in the /lmap/schedules subtree.
- o 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.
- o Task Configurations: Configured tasks are modeled in the /lmap/tasks subtree.
- o 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 running.";
            }
            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, suppression
        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 occurrence 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]
Watson, 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 demonstrates 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 controller-lost triggers when we lost
      connectivity with the controller. -->
<event>
    <name>controller-lost</name>
    <controller-lost/>
</event>
<!-- The event controller-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

- o Require yang-version 1.1 since we need leaf-lists supporting non-unique values in the report.
- o Merged the /lmap-state tree into the /lmap tree.
- o Marked state objects as mandatory.
- o Added /lmap/agent/report-group-id.

D.2. Non-editorial Changes since -06

- o Removed /lmap/agent/device-id and /lmap-state/agent/device-id, added pointer to the ietf-hardware YANG model.
- o Removed /lmap-state/agent/{hardware,firmware}, added pointer to the ietf-system YANG model.

D.3. Non-editorial Changes since -05

- o Update the example in an attempt to aligned it with the example in the information model.
- o Added an extension hook to reports so that task-specific parameters can be echoed back to the collector. Updated the example extension module accordingly.
- o Added text and Figure 1 to describe the function and purpose of the three YANG modules.
- o Added a cycle-number type definition.
- o Added the optional cycle-interval to event definitions.
- o Added tags that report additional capabilities of the measurement agent.
- o Added event time and cycle-number to the result report.
- o Renamed the metrics-grouping to registry-grouping.
- o Removed JSON encoding of the examples (they will go into the RESTCONF document).

D.4. Non-editorial Changes since -04

- o Tagged /lmap/tasks/task/program with nacm:default-deny-write.
- o Added /lmap-state/schedules/schedule/storage and /lmap-state/schedules/schedule/action/storage.
- o Removed suppress-by-default.
- o Moved the metric list from /report/result into /report/result/table.
- o Conflicts are now reported as a triple (schedule, action, task).

- o Replaced IPv4 address in the examples with IPv6 addresses.
- o Added result/status.

D.5. Non-editorial Changes since -03

- o Reworked the reporting data model to align it with the changes in the information model.

D.6. Non-editorial Changes since -02

- o Added a mechanism to enforce a runtime limit for schedules.
- o Added security considerations text warning about possible shell expansions of options.
- o 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.
- o Schedules and actions now have tags (echoed to the collector) and suppression tags (used for suppression selection).
- o Introduced glob-style pattern to match tags.
- o Added an example module for IPPM udp latency metrics to demonstrate the usage of the extension mechanism.
- o Introduced parameters, an extension point for task/metric specific parameters defined in augmenting YANG modules.
- o Introduced the typedefs event-ref, task-ref, and schedule-ref.
- o Changed schedule/event to schedule/start and added the optional schedule/stop and schedule/duration leafs.

D.7. Non-editorial Changes since -01

- o Updated and split examples (config vs state vs report).
- o Refactored the definitions so that common definitions used by both the control and report data models are in the new module ietf-lmap-common.
- o A report is submitted via an RPC operation instead of using a notification.

- o The default execution mode is pipelined.
- o Clarified which action consumes data in sequential, pipelines, and parallel execution mode.
- o Added /lmap/agent/measurement-point, /lmap/agent/report-measurement-point, and /report/measurement-point to configure and report the measurement point.
- o 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.
- o Added controller-lost and controller-ok event choices to /lmap/events/event.
- o Added a metrics-grouping to identify entries in a metric registry and associated roles.
- o 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

- o A task can now reference multiple registry entries.
- o Schedules are triggered by Events instead of Timings; Timings are just one of many possible event sources.
- o Actions feed into other Schedules (instead of Actions within other Schedules).
- o Removed the notion of multiple task outputs.
- o 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