Workgroup: OPSA Working Group
Internet-Draft: draft-palmero-opsawg-dmlmo-03
Published: 7 March 2022
Intended Status: Standards Track
Expires: 8 September 2022

Authors: M. Palmero F. Brockners S. Kumar
Cisco Systems Cisco Systems NC State University
S. Bhandari C. Cardona D. Lopez
Thoughtspot NTT Telefonica I+D

Data Model for Lifecycle Management and Operations

Abstract

This document motivates and specifies a data model for lifecycle management and operations. It describes the motivation and requirements to collect asset-centric metrics including but not limited to asset adoption and usability, licensing, supported features and capabilities, enabled features and capabilities, etc.; with the primary objective to measure and improve the overall user experience along the lifecycle journey, from technical requirements and technology selection through advocacy and renewal, including the end of life of an asset.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The virtualization of hardware assets and the development of applications using microservice architecture for cloud-native infrastructure created new consumption and licensing models. Any service can be deployed by composing multiple assets together where an asset refers to hardware, software, application, system or service. For example, cloud-native infrastructure from one vendor may be hosted on the physical server from another vendor or a combination of multiple cloud-native functions from one or more vendors can be combined to execute any service.

This introduces challenges for both lifecycle and adoption management of the assets. For example, a user may need to identify the capability availability of different assets or measure the usage of each capability (or the combination) from any specific asset to measure its optimal potential. Moreover, the user could pinpoint the reason: the software application could not be optimally deployed, or is not simple to use, or is not well documented, etc. The user may use feed such measurements and analysis metrics back to the support engineers and the developers, so they can focus their work effort only on features that users are adopting, or even determine when the lifecycle of the development could end.

This creates the need to collect and analyze asset-centric lifecycle management and operations data. From now on this data will be referred as Lifecycle Management and Operations (LMO); where LMO is not limited to virtualized or cloud environments, it covers all types of networking environments in which technology assets are deployed.

LMO data constitutes data needed to measure asset-centric lifecycle metrics including but not limited to asset adoption and usability, licensing, supported features and capabilities, enabled features and capabilities, etc. The primary objective is to facilitate the asset lifecycle management from the initial asset selection and positioning, licensing, feature enablement and usage, and beyond renewal to improve the overall user experience.

The main challenge in collecting LMO-related data, especially in a multi-vendor environment, relies on the ability to produce and consume such data in a vendor-agnostic, consistent and synchronized manner. APIs or telemetry are meant to collect and relay this data to receiving equipment for storing, analysis and/or visualization.

This document describes the motivation behind LMO, lists use cases, followed by the information model and data model of LMO. The list of use cases describes the need for new functional blocks and their interactions. The current version of this draft is focused on asset inventory, licenses information, feature usage and incident management. This draft specifies four YANG modules [RFC7950] focused on LMO, including:

- Licenses,
- Assets,
- Usage level of Asset features, and
- Incident Management.

This document is organized as follows. Section 2 establishes the terminology and abbreviations. In Section 3, the goals and motivation of LMO are discussed. In Section 4, use cases are introduced. Section 5 specifies the information model and the data models for LMO.

1.1. Requirements language

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.

2. Terminology

Terminology and abbreviations used in this document:

*Asset: refers to hardware, software, applications, or services. An asset can be physical or virtual.

*Consumer: refers to an entity that utilizes the outcomes of LMO. A consumer can be a user, a developer or some other interested third party.

*Developer: refers to the entity that creates or develops the entire asset or the part of the asset.

*EOL: End of Life.

*Features: are options or functional capabilities available in an asset.

*License: is issued by an entity such as the developer or the Open Source community and allows the user to operate the asset. Licenses determine how the asset can be leveraged and what is required in cases the asset is changed.

*LMO: Lifecycle Management and Operations.

*Optimal Software Version(OSV): refers to the elected software version considered optimal in the user environment.

*PID: Product Identifier.

*Usage: refers to how features of the asset are used.

*User: refers to the organization that owns or consumes the asset. Within the organization there are entities that: a) use the assets in their operations, b) manage the assets.

*User Experience: how a user interacts with and experiences a particular asset. It includes a user's perceptions of ease of use, efficiency, and utility of an asset.

3. Motivation

The user experience with a specific asset can be organized into four classes:

1. Asset characteristic class, covering anything related to asset, license, features, etc.

2. Utilization class, to measure how the assets and features are used, duration of usage, uptime, etc.

3. Notification class, covering any security advisory, retirement, etc.

4. Incident class, to record and report any problem the user has faced with the asset.

The ability to measure, produce and consume LMO could benefit the user organization in addressing issues such as:

*Licenses may not have been obtained at the optimum level for a given feature, where a user might have bought licenses that are not activated.

*Features of an asset might not be used as needed in all deployments within the organization.

*Resolution of incidents involving the asset and the developer of the technology used within the asset.

In addition to the resolution of incidents, LMO could allow developer organizations to optimize the features they offer. For example, they could consider deprecating features that are used infrequently or focus on introducing more features for the assets that are widely deployed in various infrastructures.

LMO also covers the need of communication between users and the developer. LMO can provide the capability for users to provide feedback about any asset (e.g., potential deficiency of a feature, feature enhancement request). An administrator in the user organization may include specific metrics that identify a potential problem of that specific feature or a capability of the asset. An engineer in the developer organization can determine the impact of the potential deficiency from the number of users providing feedback. Note that this channel is different from a "call to a Technical Assistance Center" in which the user may request help in resolving operational issues with the asset.

4. Use Cases

4.1. License Inventory and Activation

An operations engineer would like to understand which licenses are activated and which are used and/or consumed. It is also important for asset users to understand which features within their assets might need a license and how to activate them.

It is relatively straightforward to have an inventory of existing licenses when there is only one asset developer (providing the asset) and one asset family.

But complexity grows when there are many different developers, systems and processes involved. New service offerings have introduced new attributes and datasets and require alignment with new business models (pay-per-product, subscription model, pay-as-you-go model, etc.). They might support different license types and models: asset activation keys, trust-based model, systems that act as proxy from the back end owned by the asset developer to support the control of licenses, etc.

Sometimes it is a challenge to report which licenses have been bought by the asset user, or who in the user organization owns that license because that information might rely on different asset developers; even within the same asset developer, licenses may correspond to different types or groups of assets. Asset users often need to interact with different license systems and processes.

Information on how assets are licensed could be delivered from a combination of attributes such as: sales order, purchase order, asset activation key, serial number, etc.

If there is no consistency on how to deal with those data points, complexity increases for the consumer, potentially requiring manual steps. Automating those manual steps or exceptions becomes time-consuming, eventually leading to higher costs for the asset consumer.

Having a common data model for LMO eases the integration between different data sources, processes, and consolidation of the information under a common reference.

4.2. Features in Use

Feature logic is required to identify the configured features from the running configuration and determine how they might be used. There is often a lack of an easy method to list any configured features available in the current asset.

This information is extracted from the running configuration many times, implemented by a rule system without having an easy method to list any configured features available in the current asset.

Some of these use cases need to be built on top of others, and from them, other more complex use cases could be created. For instance, Software Compliance use cases can be automated, based on use cases like security advisory, errata, End of Life(EOL), etc.

All this brings a complete set of use cases that fulfills Lifecycle Management of assets, complementing and providing metrics on how asset users are using assets and how their experience from using those assets can be improved.

4.3. Assets in Use

Current approach to quantify how an asset is used, requires volume or aggregated usage/consumption metrics related to deployed assets, functions, features, integrations, etc. Also the need to quantify which metrics might be associated to a user, an organization, to specific services and how often are used; while others may be based on pre agreed profile (contractural or usage) of intented use. Examples include:

*Number of search/queries sent by the user.

*Amount of data returned to the user.

*Amount of active time spent using the asset/feature.

*Number of concurrent users accessing the asset/feature.

*Number of features in use.

*Number of users or sites using those features, etc.

The information models and data models for LMO include data fields to support metrics that might be required by consumption-based charging and licensing of asset usage.

4.4. Risk Mitigation Check (RMC)

Network, software and cloud engineers would like to be aware of known issues that are causing assets to crash so that they can act to remediate the issue quickly, or even prevent the crash if alerts are triggered on time. There are analytics tools that can process memory core dumps and crash-related files, providing the ability to the asset developers to determine the root cause.

Accordingly, asset users can remediate the problem, automate the remedy to enable incident deflection, allowing the support staff to focus on new problems. The goal of introducing normalization is not to define attributes for each of the elements being part of the crash information, but the results of RMC should be normalized and registered.

Risk Mitigation Check could also include the possibility to be aware of current and historical restarts allowing network and software engineers to enhance the service quality to asset users.

4.5. Errata

Both hardware and software critical issues or Errata need development to automate asset user matching:

*Hardware Errata match on product identifiers (PIDs) + serial numbers along with additional hardware attributes.

*Software Errata match on software type and software version along with some additional device attributes.

Engineering might develop the logic to check whether any critical issue applies to a single serial number or a specific software release.

The information to be correlated includes customer identification, license, and asset information that the asset user might own. All this information needs to be correlated with hardware and software Errata, and EOL information to show which part of the asset inventory might be affected.

4.6. Security Advisory

The Security Advisory use case automates the matching of asset user data to security bulletins published by asset developers.
Security Advisory logic implemented by developers could apply to a specific software release.

4.7. Optimal Software Version (OSV)

The objective of the Optimal Software Version (OSV) use case is that consumers can mark software images as OSV for their assets; based on this, it is easier for them to control and align their hardware and software assets to the set of OSVs.

Based on the logic of OSV, use cases like software compliance, risk trend analysis, acknowledge bugs, security advisories, errata, what-if analysis, etc., could be realized.

4.7.1. Software Conformance

All the assets should be at their latest recommended software version in case a security update is required to address a security issue of a specific feature.

The Software Conformance use case provides a view to the asset users and informs the users whether the assets that belong to a specific group conforms to the OSV or not. It can provide the users with a report, including a representation of software compliance for the entire network and software applications. This report could include the current software version running on the asset and the recommended software version. The report could enable users to quickly highlight which group of assets might need the most attention to inspire appropriate actions.

The Software Conformance use case uses data that might not be provided by the asset itself. Data needs to be provided and maintained also by the asset developers, through e.g., asset catalog information. Similar logic applies to a feature catalog, where the asset developer maintains the data and updates it adequately based on existing bugs, security advisories, etc.

The Software Conformance process needs to correlate the Software catalog information with the software version running on the asset.

4.7.2. Risk Trend Analysis

The Risk Trend Analysis use case provides customers with a risk trend analysis, summarizing what might change before applying changes, including registered bugs, security advisories and errata.

4.7.3. What-if Analysis

The What-if Analysis use case allows asset users to plan for new hardware or software, giving them the possibility to change the config parameters or model how new hardware or software might change the software suggestions generated by OSV.

OSV and the associated use cases involve dependencies on attributes that might need to be collected from assets directly, including related inventory information (serial numbers, asset identifiers, software versions, etc.), but also dynamic information could be required, like:

- Information on features that might be enabled on the particular asset.

- Catalogs, that might include information related to release notes. For example, consider a feature catalog. This catalog could include software versions that support a specific feature; the software releases that a feature is supported in; or the latest version that a feature is supported in, in case the feature is EOL.

- Data sources to correlate information coming from reports on critical issues or errata, security advisory, End of Life, etc.

Those catalogs and data sources with errata information, EOL, etc. need to be maintained and updated by asset developers, making sure, that the software running on the assets is safe to run and up to date.

4.8. Asset Retirement - End of Life (EOL)

Hardware EOL reports need to map Hardware EOL PIDs, focusing on base PIDs so that bundles, spares, non-base PIDs, etc., do not provide false EOL reporting to asset users. Software EOL reports are used to automate the matching of user software type and software version to software EOL bulletins.

5. Information Model

The broad metric classes defined in section 3 that quantify user experience can be modeled as shown in Figure 1. There is an inventory of all assets that the user possesses. Each asset in the inventory may be entitled to one or more licenses; a license may contain one or more sub-licenses. The level of usage for each feature and license associated with the asset is measured. For every asset, a list of incidents could be created.

For example, a user needs to measure the utilization of a specific license for a specific type of asset. The information about the license may reside in a license server. The state (activated or not) of the license may reside with the asset itself or a proxy. They can be aggregated/correlated as per the information model shown in Figure 1 to give information to the user regarding the utilization of the licenses. The user experience is thus enhanced by having accurate knowledge about the utility of the given license.

```
 may_be_part_of                                    may_be_part_of
   +------+                                            +-------+
   |      |                                            |       |
   |      v                                            v       |
+------------+  entitled_by            tracked_by   +------------+
|  Licenses  |<------------+          +-------------|   Usage    |
+------------+-----------+ | may_be_  | +---------->+------------+
|  License   | entitles  | | part_of  | | tracks    |   Asset    |
| attributes |           | | +------+ | |           |  Features  |
+------------+           | | |      | | |           | and usage  |
                         v | |      v v |           | attributes |
                        +----------------+          +------------+
                        |     Asset      |
          future_       +----------------+    generated_by
          association   |     Asset      |<----------------+
            +---------->|   attributes   |---------------+ |
            |           +----------------+    generates  | |
            v                                            v |
      +-----------+                                 +------------+
      |  Future   |                                 |  Incident  |
      | Expansion |                                 +------------+
      +-----------+                                 |  Incident  |
                                                    | attributes |
                                                    +------------+
```

Figure 1: Information Model

The model allows for future expansion by new metrics that will quantify user experience. Notice that future asociation relationship and future expansion might be linked to asset or to one of the other datasets: incident, feature usage or licenses.

6. Data Models

6.1. Tree Diagrams of the modules that form LMO

6.1.1. Aggregated Asset Inventory

This specification uses [I-D.draft-ietf-netmod-geo-location-11], [I-D.draft-ietf-opsawg-sbom-access-03]

6.1.2. Licenses

6.1.3. Usage

6.1.4. Usage

6.1.5. Incident Management

6.1.6. Organization

6.1.7. Service

6.1.8. User

6.2. LMO Modules

6.2.1. LMO Module

```
<CODE BEGINS> file "ietf-lmo@2022-03-01.yang"

module ietf-lmo {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-lmo";
  prefix ietf-lmo;
  import ietf-lmo-common {
    prefix ietf-lmo-common;
  }
  import ietf-yang-types {
    prefix yang;
  }
  organization
    "IETF OPSA (Operations and Management Area) Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/opsawg/>
     WG List:  <mailto:opsawg@ietf.org>
     Editor:  Jan Lindblad
              <mailto:jlindbla@cisco.com>
     Editor:  Marisol Palmero
              <mailto:mpalmero@cisco.com>";
  description
    "This YANG module add the flexibility to define its own
     and extensible set of lmo classes.

     Copyright (c) 2021 IETF Trust and the persons identified as
     authors of the code. All rights reserved.

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject to
     the license terms contained in, the Simplified BSD License set
     forth in Section 4.c of the IETF Trust's Legal Provisions

     Relating to IETF Documents
     (https://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC XXXX
     (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
     for full legal notices.";
  revision 2022-03-01 {
    description
      "Initial revision for LMO Module as part of the
       LMO YANG Model";
    reference
      "RFC XXXX: LMO YANG Model";
  }

  container lmos {
    //config false; //temporarily commented out for easy testing
    list lmo {
      key lmo-class;
      leaf lmo-class {
        type identityref {
          base ietf-lmo-common:lmo-class;
        }
      }
      list inst {
        key id;
        leaf id {
          type string;
        }
        container parent {
          leaf lmo-class {
            type leafref {
              path /lmos/lmo/lmo-class;
            }
          }
          leaf id {
            type leafref {
              path deref(../lmo-class)/../inst/id;
            }
          }
        }
        container capture-info {
          // Moved capture-info to the instance level, as
          // asset/... data will generally be collected
          // from one source at one time.
          description
            "Capture information for this data";
          leaf collected-on {
            type yang:date-and-time;
            description
              "Time at which this data was collected";
          }
          leaf collected-from {
            type string;
            description
              "Identifier for original source of this data";
          }
        }
      }
    }
  }

<CODE ENDS>
```

7. Deployment Considerations

LMO Data Models defines the data schemas for LMO data. LMO Data Models are based on YANG. YANG data models can be used independent of the transport and can be converted into any encoding format supported by the network configuration protocol. YANG is a protocol independent.

To enable the exchange of LMO data among all interested parties, deployment considerations that are out of the scope of this document, will need to include:

- The data structure to describe all metrics and quantify relevant data consistently, i.e. specific formats like XML or JSON encoded message would be deemed valid or invalid based on LMO models.

- The process to share and collect LMO data across the consumers consistently, including the transport mechanism. The LMO YANG models can be used with network management protocols such as NETCONF [RFC6241], RESTCONF [RFC8040], streaming telemetry, etc. OpenAPI specification might also help to consume LMO metrics.

- How the configuration of assets should be done.

8. Security Considerations

The security considerations mentioned in section 17 of [RFC7950] apply.

LMO brings several security and privacy implications because of the various components and attributes of the information model. For example, each functional component can be tampered with to give manipulated data. LMO when used alone or with other relevant data, can identify an individual, revealing Personal Identifiable Information (PII). Misconfigurations can lead to data being accessed by unauthorized entities.

Methods exist to secure the communication of management information. The transport entity of the functional model MUST implement methods for secure transport. This document also contains an Information model and Data-Model in which none of the objects defined are writable. If the objects are deemed sensitive in a particular environment, access to them MUST be restricted using appropriately configured security and access control rights. The information model contains several optional elements which can be enabled or disabled for the sake of privacy and security. Proper authentication and audit trail MUST be included for all the users/processes that access the LMO.

9. IANA Considerations

9.1. The IETF XML Registry

This document registers URIs in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the registrations defined below are requested:

URI: urn:ietf:params:xml:ns:yang:ietf-lmo
Registrant Contact: The OPSA WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lmo-common
Registrant Contact: The OPSA WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lmo-assets-inventory
Registrant Contact: The OPSA WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lmo-licenses
Registrant Contact: The OPSA WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lmo-feature
Registrant Contact: The OPSA WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lmo-usage
Registrant Contact: The OPSA WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lmo-incident-management
Registrant Contact: The OPSA WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lmo-organization
Registrant Contact: The OPSA WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lmo-service
Registrant Contact: The OPSA WG of the IETF.

XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lmo-user
Registrant Contact: The OPSA WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

9.2. The YANG Module Names Registry

This document registers YANG modules in the YANG Module Names registry [RFC7950]. Following the format in [RFC7950], the registrations defined below are requested:

name: ietf-lmo
namespace: urn:ietf:params:xml:ns:yang:ietf-lmo
maintained by IANA: N
prefix: lmocom
reference: RFC XXXX

name: ietf-lmo-common
namespace: urn:ietf:params:xml:ns:yang:ietf-lmo-common
maintained by IANA: N
prefix: lmocom
reference: RFC XXXX

name: ietf-lmo-asset-inventory
namespace: urn:ietf:params:xml:ns:yang:ietf-lmo-assets-inventory
maintained by IANA: N
prefix: lmoasset
reference: RFC XXXX

name: ietf-lmo-licenses
namespace: urn:ietf:params:xml:ns:yang:ietf-lmo-licenses
maintained by IANA: N
prefix: lmolicense
reference: RFC XXXX

name: ietf-lmo-feature
namespace: urn:ietf:params:xml:ns:yang:ietf-lmo-feature
maintained by IANA: N
prefix: lmousage
reference: RFC XXXX

```
name: ietf-lmo-usage
namespace: urn:ietf:params:xml:ns:yang:ietf-lmo-usage
maintained by IANA: N
prefix: lmousage
reference: RFC XXXX
```

```
name: ietf-lmo-incident-management
namespace: urn:ietf:params:xml:ns:yang:ietf-lmo-incident-management
maintained by IANA: N
prefix: lmoscm
reference: RFC XXXX
```

```
name: ietf-lmo-organization
namespace: urn:ietf:params:xml:ns:yang:ietf-lmo-organization
maintained by IANA: N
prefix: lmoscm
reference: RFC XXXX
```

```
name: ietf-lmo-service
namespace: urn:ietf:params:xml:ns:yang:ietf-lmo-service
maintained by IANA: N
prefix: lmoscm
reference: RFC XXXX
```

```
name: ietf-lmo-user
namespace: urn:ietf:params:xml:ns:yang:ietf-lmo-user
maintained by IANA: N
prefix: lmoscm
reference: RFC XXXX
```

10. References

10.1. Normative References

[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>.

[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>.

10.2. Informative References

[I-D.draft-ietf-netmod-geo-location-11] Hopps, C., "A YANG Grouping for Geographic Locations", Work in Progress, Internet-Draft, draft-ietf-netmod-geo-location-11, 11 February 2022, <https://www.ietf.org/archive/id/draft-ietf-netmod-geo-location-11.txt>.

[I-D.draft-ietf-opsawg-sbom-access-03] Lear, E. and S. Rose, "Discovering and Retrieving Software Transparency and Vulnerability Information", Work in Progress, Internet-Draft, draft-ietf-opsawg-sbom-access-03, 24 October 2021, <https://www.ietf.org/archive/id/draft-ietf-opsawg-sbom-access-03.txt>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[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, <https://www.rfc-editor.org/info/rfc6241>.

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

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

Acknowledgments

The ideas in this document originate from early work by Tony Colon, Carlos Pignataro, and Yenu Gobena originally referred to as Experience Telemetry.

This document was created by meaningful contributions from Josh Suhr, Eric Vyncke, Yannis Viniotis, Nagendra Kumar Nainar, Yenu Gobena, Dhiren Tailor and Jan Lindblad.

The authors wish to thank Gonzalo Salgueiro, Martin Beverley and many others for their helpful comments and suggestions.

Change log

RFC Editor Note: This section is to be removed during the final publication of the document.

version 03

- *Flexible root structure has been introduced by the ietf-lmo YANG module: Modules are arranged into layers, with ietf-lmo-common and ietf-lmo at the core. Other modules can be added in layers on top. This structure allows flexibility and the option to be enhanced by vendor implementation.
  The new structure allows to include other lmo classes, or exclude current lmo classes.

- *Feature and Usage containers have been split in two independent modules. Where Usage relates to runtime data.

- *Organization attribute, has been enhanced to an independent YANG module, adding flexibility and the option to be called independently and enhanced.

- *Service and User YANG modules, have been also introduced in a similar flexible structure, being part of new lmo classes.

- *Information Model, has been enhanced with new modules: Organization, Service and User modules. On this version the new lmo classes can be called independently or from the licenses module. There is no restriction to be called from any of the other YANG odules.

version 02

- *"Support case" renamed to "incident".

- *Add MAC address and IP address attributes under asset-inventory YANG module.

- *Link among objects & YANG modules (notably with feature).

- *New text about asset usage.

version 01

- *Fixes for YANG validator and idnits warnings.

version 00

- *Initial version.

Authors' Addresses

Marisol Palmero
Cisco Systems

Email: mpalmero@cisco.com

Frank Brockners
Cisco Systems

Email: fbrockne@cisco.com

Sudhendu Kumar
NC State University

Email: skumar23@ncsu.edu

Shwetha Bhandari
Thoughtspot

Email: shwetha.bhandari@thoughtspot.com

Camilo Cardona
NTT

Email: camilo@ntt.net

Diego Lopez
Telefonica I+D

Email: diego.r.lopez@telefonica.com