Network Information Model Requirements

Last Updated: 7/7/00

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC-2119].
Status of this Memo......................................................1
Conventions used in this document....................................1
Abstract..............................................................................3
1. Introduction.......................................................................3
2. Motivation..........................................................................5
3. Specific Requirements for a Network Information Model..........6
4. Glossary of Terms.............................................................13
5. References.........................................................................14
6. Author Information and Acknowledgments..............................15
Abstract

As networking technology has evolved, a diverse set of technologies has been deployed to manage the resulting products. These very from Web based products, standard management protocols, and text scripts. The underlying systems to be manipulated are represented in varying ways including implicitly in the programmatic, with proprietary data descriptions, or with standardized descriptions using a range of technologies including MIBs, PIBs, or LDAP schemas. The result is that a single application such as DHCP or Differentiated Services may be represented in many different inconsistent fashions. Even the existing standard descriptive technologies have not been focussed on re-use across domains and commonality. The situation would be significantly improved if there were to be a common representation of data (an "information model") from which technology-specific data models can be derived to suit application-specific configuration and management needs. This would allow the industry to build common descriptions, resulting in consistency across the paradigms, and better interoperability. Such an "information model" would also assist other IETF working groups by reducing the work needed to determine how to represent those parts of their technology that are already in the common model.

This document describes the requirements of an information model suitable for the modeling of network management constructs. The purpose of an information model is to represent data in a manner that is independent of technology and implementation. An Information model is used to unambiguously describe and define information, as well as the relationships between the entities that it defines. From this idealized information model, implementation-specific data models (which are bound to specific types of repositories and data storage and retrieval mechanisms, protocols, APIs, etc.) can be derived so as to avoid divergence of management information definitions between implementations.

This document will describe requirements for the construction of an information model and the representation of basic modeling constructs. These requirements are generally described in the areas of reusability, extensibility, associability, class naming, instance naming, expressiveness of data definitions through constraints, and inheritance. The purpose of this document is to ensure that the subsequent documents that describe the conventions for the information model and the information model itself are complete and consistent with the requirements stated herein.

1. Introduction

As the Internet has grown and flourished, new technologies have been developed and deployed at a feverish pace. In the traditional IETF
operations and management model, the working group developing the technology has also defined the data elements (usually in the form of MIBs) necessary to monitor and manage the technology. While this has been very successful, it has also led to a divergence of definitions and structures. In many cases, similar management elements that are to be applied to a specific technology have been completely reinvented in different working groups because of a lack of awareness. In other cases, duplication has occurred because of a subtle variation in the definition of the element or because the relationship between the data element and other elements is new or different. Divergence of management definitions and structures has resulted in an environment where programming effort is spent in the repetitive correlation, translation and data organization associated with each data management technology, as opposed to implementing a single management interface/infrastructure.

There are two major benefits of an information model. First, it has a unique ability to consistently describe and organize various network management concepts and data for reuse across different applications. Second, it unifies information describing different aspects of managed objects (e.g., configuration and statistical data).

In order to realize the benefits of an information model, each addition to the model needs to be scrutinized to ensure that the data being added is either new, or represents a refinement of existing data that have already been defined in the model. As such, the primary objective of this document is to specify the requirements for an information model that organizes information to achieve maximum reusability, as well as providing consistency in the information represented across various protocols and services.

It is the goal of a network information model to act as a synchronizing root for other groups that are defining network management constructs so that consistency and reusability can be obtained across working groups. Just as MIB-I and MIB-II successfully standardized well-known constructs in the SNMP data model, this document will define the requirements for an information model that is applicable to largest possible cross-section of network management paradigms and protocols. This information model can then be expanded and exploited by a variety of other working groups within the IETF, and adapted to their transport-specific requirements. This will in turn produce domain-specific data models that are all derived from a common network information model. A key property of an information model is its wide spread applicability. Once the information model is established, standardized mappings can be developed for transforming the information model into specific data models, potentially including LDAP schema, MIBs, PIBs, and device models (such as [Device]) using the Policy Framework Core Information Model [PCIM] to mechanism-specific policies.

This document only describes the requirements for the definition of common attributes, mechanisms and conventions for their organization
into reusable data structures, and mechanisms for representing the attribute’s relationships to other attributes and structures. In addition, requirements for specifying methods will also be described. As methods and attributes provide an alternate means for describing a management interface, guidelines for using each in a consistent manner will be developed in a separate document, and is therefore outside the scope of this document.

The overall goal is to allow the model’s attributes and methods to be equally applicable across a broad range of paradigms, protocols and interfaces. Specifically, there are some paradigms that treat a networking device as a client while other paradigms treat the device as a server. There are some paradigms that assume a synchronous (transactional) interface between the managing and managed elements while other paradigms assume an asynchronous model. With some management paradigms a single instance of a structure is applicable to many instances of devices or device components (COPS-PR, SNMPconf, or shared repository structures) while in other paradigms each component is managed directly on a one-to-one basis.

These paradigms can be implemented in a variety of ways. In some cases there may be programmatic interfaces that directly manipulate management structures through methods or remote procedure calls. In other cases a repository, such as a directory, may be used to store configuration information in a central store or even use the store to pass information between networking components. Paradigms can even take the form of simple text scripts that get pushed into a device. While any of these scenarios could leverage a shared information model for representing network configuration and management structures, the degree to which a common model can be mapped to all of these implementations will vary on a case-by-case basis. It is in fact the purpose of this model to be able to represent more relationships than can be easily reflected in the common subset of all implementations. Representing such methods and relationships in a common information model helps ensure that the differing representations will work together both where they overlap and where they complement.

While the model could be extended to support other interfaces that may exist within and between networking components such as interfaces between routing and traffic engineering, these interfaces are beyond the current scope of this work. Initial models will focus exclusively on those interfaces needed to manage and configure various components within a networking environment.

2. Motivation

1. Problem: With the exception of the Policy Framework’s Core Information Model [PFCIM], no high-level network configuration information model exists in the IETF that is implementation-independent. Implementation independence yields a model that simply
represents network constructs and data, relationships between constructs, and data constraints without regard to a particular data model, protocol, or repository.

2. Problem: The Distributed Management Task Force’s (DMTF’s) CIM (Common Information Model) \([\text{CIM}]\) is not the optimal choice for a network information model. CIM provides no mechanism for defining class/model level constraints. CIM also has dictated a naming scheme that cannot be mapped easily into all management technologies. Finally, the DMTF does not have the subject matter experts in networking. Nevertheless, there is value in the modeling work done in the DMTF in terms of the definition of high-level constructs. Thus, CIM should be viewed as a starting point for development of a network information model within the IETF.

3. Problem: Current modeling representations are insufficient due to a lack of available tools, lack of a textual representation for ease of data exchange, or the expressiveness of the representations. In addition, for some or all of these representations, there is no constraint language, no suitable class naming conventions, no support for attribute level bindings, no ability to effectively create implementation-specific models, no ability to model associations, etc. Unfortunately, some are also dependent on proprietary or high-level graphical tools. It will be up to the working group to determine what tool set would be most appropriate in meeting the set of specific requirements listed below.

3. Specific Requirements for a Network Information Model:

1. The information model will need conventions for uniquely identifying the classes (constructs), attributes (properties or data) and methods (signatures) that it defines, as well as complimentary classes developed by other working groups. Class naming and the ability to uniquely identify defined attributes is an important part of any information model. It is important that attribute naming conventions be developed that can be used equally effectively for reuse as well as to extend existing classes. The requirements will develop mechanisms that are complimentary to object-oriented naming conventions while not requiring the use of a central naming authority. There is a need to name class/attribute DEFINITIONS (as opposed to instances) uniquely without the need for a central naming authority (i.e., one mechanism might be to use GUIDs to identify classes/attributes).

2. Experience has shown that different technologies (and even different implementations of the same technology) use different mechanisms for binding attribute groups to keys, instance identifiers, indexing schemes, or other data instance naming mechanisms. This model will avoid defining keys or rules for key bindings, unless no other alternative for interoperable operation can be defined. Mechanisms will be provided for supporting keys for mappings to various technologies, as the need for instance
identification is universal. Instance naming needs to be flexible as implementations vary. That is, keys can be created in a number of ways, we assume they exist, and uniqueness within a context is required, but we will not force a particular keying mechanism. It will however, be important to understand how a particular implementation defines its keys and thereby names instances. Without this information, mapping between implementations will be difficult or impossible.

3. Exclusively using graphical models such is not suitable as there is a need for textual representations. Graphical models are also not easy for a computer to parse and require expensive tools. Thus, it is necessary that this information model chooses a textual representation suitable for representing an information model, completely & unambiguously, and is readily parseable by a computer program. Graphical methods can always be used to visualize this result. Ideally, the textual representation also has some tools associated with it. It should be up to the WG to decide the most appropriate data definition language based on these requirements. Ideally, the textual representation provides sufficient detail for algorithmic mappings of the information model to as large a cross-section of protocols/interfaces as possible.

4. To facilitate reuse, all data elements (attributes) of the information model must be clearly defined in terms of characteristics, scope and relationships to other attributes. This can be achieved through data types and value bounding which should be explicitly defined for attributes, classes, and associations between NIM constructs. Also, the ability to specify constraints is needed.

Constraints refer to the ability to clearly specify the characteristics of one attribute/class/association or a set of these and how these constructs interact with and relate to others within the model. If a concept is defined in a broad or high-level manner in the information model, then it is more likely that different implementations that define their own mappings of it will be inconsistent. This undermines the basic benefit of the information model.

The subsequent list of constraints, while not necessarily complete, is a minimal set of elements required for a more complete specification of attributes.

Where the data modeling language is not capable of specifying the constraint, the information model is required to provide the most precise specification of attributes possible. Further, when a specific mapping of the information model to a specific data model is defined and the target data modeling language has a less robust mechanism for expressing constraints, additional documentation should accompany the data model to minimize confusion. If an information model representation is not capable
of expressing a new kind of constraint, then this new constraint should be added to the information model in a well-defined manner.

Need for data-level constraints:
* Constraints for attributes. E.g., Attribute X is an integer that can have values in the range 10-100 and 234-235.
* Constraints for classes. E.g., Class A should be created when Class B's attribute x is "OK".
* Constraints for class/attribute associations. Allow instance class A to be associated with instance of class B only if not already associated with class C.
* Constraints for attribute transactions, attribute A and attribute B have transactional dependencies (need to be modified together under a given context).
* Another common characteristic of an attribute is its ability to be modified. While the majority of attributes can be retrieved and altered, for some attributes, such as counters or hard wired MAC addresses, it is only possible to retrieve the value but not to alter it. There may even be instances where an attribute value can be altered, but not retrieved. To facilitate consistent interpretation, each attribute definition must specify whether the attribute is read-only, read-write, or write-only.
* There are many situations where an attribute is not required for the complete definition of the class. In other cases, an attribute is crucial for the definition of the service. Therefore, each attribute in the information model must specify whether an attribute is optional or required.
* In some cases, there are attributes within the same class that have interdependencies. For example, two attributes together may define a x/y coordinate. There may be more extreme forms of relationships where one attribute may affect the value or range of values for another attribute. One such relationship exists in IPSEC, where the type of security algorithm determines the range of possible values for other attributes such as the corresponding key size. The most extreme form of multi-attribute relationships is a set of attributes that are dependent on each other for consistency. For example, a class may have two attributes, one expressing a date and the other expressing a day of the week. When the day of the week is changed the date should change simultaneously, and visa versa to prevent inconsistencies. Irrespective of the complexity of the relationship between attributes, the information model must express the relationship between attributes.

5. Need for an association mechanism. Associations describe relationships between classes. Some examples are dependency or whole-part relationships. There are a few things to note about associations: 1) relationships are primarily between only two class instances; 2) the same relationship may be repeated for a particular instance, associating that instance with others; and 3) since relationships are usually implemented as classes, they
can sometimes have data as well as methods. An example of each of the previous points is that a device has various types of software associated with its configuration, operational and instrumentation software are just a few examples. So, software can be "associated" with one or more devices, and devices can have one or more pieces of software. The use of the software for the device can be described as part of the association. [Note: While at the information model level associations should be modeled as separate classes, some data models may only support associations as inline references that are part of a data class. It is noted that separate association classes force data classes to be maximally reusable, as the data class then makes no assumptions about how it can be related to other classes.]

6. Inheritance model: Inheritance allows the information model to avoid redundant data definitions among peer constructs, and define appropriate levels of abstraction and complexity. Single inheritance is recommended to optimize the model and its processing requirements. It is not required that the entire information model be derived from a single root. Those data models that require a single root, and a key binding consistency relative to the root, should provide for these requirements in the mapping of the information model to the specific data model.

7. Mechanisms for describing the fundamental data types of attributes, as well as describing arrays, sets, or ranges of these fundamental types, are required. Possibly, the binary representation of the attributes may be specified.

8. When hundreds of classes may be defined in the information model, there is a need to create "standard" combinations of classes and associations, describing common scenarios or data models. It should be possible to define these combinations of classes and associations in the information model. And, for efficiency, the combinations should be instantiated, retrieved or manipulated as a whole. These standard combinations would be useful in reducing the arbitrariness of which classes to use, to solve a particular management modeling problem. In other words, groupings of classes may facilitate the initialization or configuration of classes by logical groupings. This is a feature that people could use to determine the scope of a particular domain (DS, MPLS, DHCP, etc.) rather than a packaging format.

9. In an object-oriented implementation, a class’ methods describe the actions or behaviors that are appropriate for, and can be invoked against, an instance. Methods allow the specification of return codes and input/output parameters, further influencing the specific action or behavior.

In an information model, these concepts may be described as "signatures" or as fully-defined methods. (The correct approach must be determined as work on the Network Information Model proceeds.) The word, "signature", means the specification and
grouping of attributes within an object, while "method" means a specific and predefined format and syntax. The goal of either a method or signature is to describe and invoke well-defined behaviors -describing the intent of the behavior, return code/status and parameters.

Methods or signatures must operate under the same modeling restrictions as those described in requirement 4 for Attribute definitions. Mechanisms must be provided in the model to fully specify the operating constraints of both the method and the attributes passed to and from the method. Specifically:

* Constraints for attributes passed through a method. E.g., Attribute X is an integer that can have values in the range 10-100 and 234-235.
* If the modeling language supports the ability to pass classes or class references, the constraints of the passed classes should be fully specified if different from the general definition of the class. E.g., This method allows the passed in Class A to be removed or copied.
* If methods are used to perform associations between classes passed in through the method, this must be specified. If special circumstances determine the outcome of the association, this must also be represented in the model. Allow instance class A to be associated with instance of class B only if not already associated with class C.
* Constraints for attribute transactions within a method, method parameter A and method parameter B have transactional dependencies (need to be modified together under a given context or the modification of one is dependent on the value of the other).
* To facilitate consistent interpretation of the method, each parameter definition must specify whether the attribute is read-only, read-write, or write-only.
* If it is possible for one method to invoke or use another method, this must be documented. This is particularly important when the embedded method updates structures that can also be manipulated directly or through other externally visible methods. If method A invokes method B and method B updates attribute X, it must be clearly documented that X is being updated to prevent the consumer to inadvertently also update X.

10. As this information model will likely be mapped to a variety of data models and implementations, it is necessary that mechanisms be provided that ensure compliance to the information model. [Note: The specific requirement for this assurance is TBD and will be addressed, or removed, after further team discussion.]

11. As the information model will be deployed across a variety of implementations, it is recognized that not all implementations will be capable of supporting the model completely. Divergences from the model should be expressed in the form of a capability set that describes exactly what subset (or superset) of the model
is supported by a particular implementation or system. Further, as capability distribution is a service, the information model must specify a mechanism that allows for a consistent representation of capabilities across various management technologies.

12. The information model must allow classes to contain the attribute (property) set of other classes. This will maximize reuse and allow data to be completely described with minimum redundancy. For example, Class XYZ can contain as attributes Class ABC and Class DEF. The result would be that Class XYZ’s attribute set would be a superset of Classes ABC and DEF.

Class containment should act as a mechanism for easily defining and describing class aggregations. LDAP schema definitions currently have the opposite behavior and instead constitute attribute merging. An information model should not have this same limitation. For example, in NIM, one may create a class called IPADDRESSMASK that defines and constrains IP subetting. Using NIM, one then could create a new Filter Class that looks something like this:

```plaintext
class Filter:
{
    IPADDRESSMASK  SourceIP;
    IPADDRESSMASK  DestinationIP;
    PORTRANGE      SourcePort;
    PORTRANGE      DestinationPort;
    PROTOCOLID      Protocol;
};
```

Where IPADDRESSMASK is defined as follows:

```plaintext
Class IPADDRESSMASK:
{
    UINT32   IPAddress;
    UINT32   SubnetMask;
};
```

Here the IPADDRESSMASK classes are not to be merged into one attribute, but remain two individually identifiable "contained" classes, one useful for defining the source, another for defining the destination of traffic in the context of the new class called Filter.

It is important that all data models have standard mappings of this strictly definitional model into their data model-specific forms.

13. There are some attributes that, when changed, only take effect when the service is brought down or the system is rebooted. There are also mechanisms deployed that depend on polling to cause a modification of an attribute value to take effect. The model must
fundamentally distinguish between the retrieval of attributes from the assignment of attributes, as these functions can often be asynchronous depending on the data and the technology being used to manipulate it.

For example, if I have an attribute called AS_Number, for which a change of the value can only take effect when routing is brought down, I may have at least two values of AS_Number in the system: currently active and proposed. The model must be capable of distinguishing the two. A simple approach might be to define two attributes in the class. However, different products will make different implementation decisions for when attribute updates take effect. Further, some configuration management paradigms like LDAP rely on polling to make changes to systems, so the state in the directory may not reflect the state of the device until the device actually retrieves the value and notifies the directory that the new value is in effect. These various scenarios suggest the need for a general mechanism in the model that distinguishes the actual configuration from the desired configuration.

14. The information model must allow for forward and backward compatibility as the model evolves. In addition, the model must support the ability to be extended for specific implementations in a manner that will allow multiple different extensions to coexist both from a programmatic and storage perspective.
4. Glossary of Terms

NIM: Network Information Model.

CIM: Common Information Model proposed by the Distributed Management Task Force (DMTF).

Constraint: A mechanism for bounding an attribute or class value to allowed values/ranges/targets/bitmaps etc.

Data Type: A mechanism for identifying the binary representation of an atomic (as opposed to composite) attribute and the rules for manipulating an attribute with that type.

Definition Name: Unique name used to identify an attribute or class definition.

Instance Name: Key: Index: Name used to identify an instance of an attribute or class (recognized but not explicitly defined by this WG).

NIM Attribute: An atomic or composite data type that is fully described, typed, uniquely named, and bounded via constraints.

NIM Class: Typically refers to an aggregation or set of one or more attributes that can also inherit properties from parent classes to optimize data definitions.

Information Model: Universally applicable model for representing and describing information that is not perverted by the shortcomings of a specific implementation.

Data Model: An implementation specific or implementation constrained derivation of an information model. One information model can be used to develop multiple consistent data models.
5. References


6. Author Information and Acknowledgments

Special thanks to Joel Halpern, Juergen Schoenwaelder, and John Strassner for their many helpful comments and significant edits to this document.

Walter Weiss
Ellacoya Networks
7 Henry Clay Dr.
Merrimack, NH. 03054
Phone: +1-603-879-7364
E-mail: wweiss@ellacoya.com

David Durham
Intel
2111 NE 25th Avenue
Hillsboro, OR 97124
Phone: 503.264.6232
David.Durham@intel.com

Andrea Westerinen
Cisco Systems
andreaw@cisco.com
425-891-8407
2570 W. El Camino Real
Mountain View, CA 94040