1. Status of this Memo

This document is an Internet Draft. 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. Internet Drafts may be updated, replaced, or obsoleted by other documents at any time. It is not appropriate to use Internet Drafts as reference material or to cite them other than as a "working draft" or "work in progress."

To learn the current status of any Internet-Draft, please check the "1id-abstracts.txt" listing contained in the internet-drafts Shadow Directories on:

ftp.is.co.za (Africa)
nic.nordu.net (Europe)
ds.internic.net (US East Coast)
ftp.isi.edu (US West Coast)
munnari.oz.au (Pacific Rim)
2. Introduction

This memo is informational. It specifies a clarification of the meaning and use of the STATUS clause in Simple Network Management Protocol (SNMP) Management Information Base (MIB) modules, which are defined by the Structure of Management Information (SMI). There are two versions of the SMI. The first, called SMIv1, is defined by RFCs 1155[1], 1212[2], and 1215[3]. The second, called SMIv2, is defined by RFCs 1902[4], 1903[5], and 1904[6]. Many of the MIB module constructs defined by the SMIs such as OBJECT-IDENTITY, OBJECT-TYPE, and NOTIFICATION-TYPE contain the STATUS clause. However, the SMIs do not provide a complete definition of the STATUS clause, nor do they provide guidance to MIB designers and users on the interpretation and action required dependent upon the value or change of value for a STATUS clause. Users include agent and application developers, and operators of SNMP-managed networks.

This memo specifies a clarification for both version 1 and version 2 of the SNMP SMI, which is a standard for the Internet community.

3. Background

The STATUS clause was first defined in "Structure and Identification of Management Information for TCP/IP-based Internets," RFC 1065[7]. This document lists the possible status values of object type definitions as "mandatory," "optional," and "obsolete," but does not contain an interpretation of these values. The document does provide the definition of the prime directive of MIB design (in section 5), which is:

New versions of MIB modules may:

(1) change the status of object types to obsolete (if necessary), but may not delete their definitions;

(2) define new columnar object types for an existing table; or

(3) define entirely new object types.

New versions may not:

(1) change the semantics of any previously defined object type.

The original SMI definition was replaced by "Structure and Identification of Management Information for TCP/IP-based Internets," RFC 1155[1]. However, no changes were made to the definition of the STATUS clause or the prime directive for MIB design.
The document, "Concise MIB Definitions," RFC 1212[2], was written after RFC 1155 to allow MIB designers to write MIB modules in a concise format. The format combined the two formats specified in RFC 1155 for writing definitions of managed objects. Also, RFC 1212 extended the STATUS clause with the addition of the value "deprecated," and kept the existing values of "mandatory," "optional," and "obsolete." Note that RFC 1212 did not define the meaning of the new value "deprecated." The only definition of the STATUS clause found in RFC 1212 is:

4.1.3. Mapping of the STATUS clause

The STATUS clause, which must be present, defines the implementation support required for that object type.

Surprisingly, a definition of the value "deprecated" is specified in "Management Information Base for Network Management of TCP/IP-based internets: MIB-II," RFC 1213[8], which uses the concise format for defining the IETF core objects. It is surprising, since the documents that describe the language used to write MIB modules do not define use of the STATUS clause, and the document that contains an example of a MIB module, defines use of the STATUS clause.

The text from RFC 1213 describing the value of "deprecated" is:

3.1. Deprecated Objects

In order to better prepare implementors for future changes in the MIB, a new term "deprecated" may be used when describing an object. A deprecated object in the MIB is one which must be supported, but one which will most likely be removed from the next version of the MIB (e.g., MIB-III).

MIB-II marks one object as being deprecated:

atTable

As a result of depreciating the atTable object, the entire Address Translation group is depreciated.

Note that no functionality is lost with the depreciation of these objects: new objects providing equivalent or superior functionality are defined in MIB-II.

RFC 1213 contains additional text to define the concept of conformance, which is not previously defined in SMIv1. The text from RFC 1213, section 5 is:

MIB-II, like its predecessor, the Internet-standard MIB, contains only essential elements. There is no need to allow individual objects to be optional. Rather, the objects are arranged into the following groups:
These groups are the basic unit of conformance. This method is as follows: if the semantics of a group is applicable to an implementation, then it must implement all objects in that group. For example, an implementation must implement the EGP group if and only if it implements the EGP.

There are two reasons for defining these groups: to provide a means of assigning object identifiers; and, to provide a method for implementations of managed agents to know which objects they must implement.

What may not be obvious from this history is that the STATUS clause is used for two different purposes in SMIv1 format MIB modules. The first use is to specify the status of a definition, that is, specify whether a definition is valid or invalid. This is needed, since the prime directive for MIB design does not allow a definition to be semantically changed or "removed." A definition may only be "retired" and, if a new definition is created, the new one must use a new identity. The second use of the STATUS clause is to specify conformance requirements. To eliminate the confusion caused by the two uses of one clause, the second version of the SMI for SNMP changed the STATUS clause so that it specifies only the validity of a definition.

The document, "Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)," RFC 1442[9], and its replacement, RFC 1902[4], define values for the STATUS clause as "current," "deprecated," and "obsolete." Other MIB module language constructs were added to specify conformance requirements[6][11]. The STATUS clause is used in all but one of the SMIv2 MIB module language constructs, which are OBJECT-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE, TEXTUAL-CONVENTION, OBJECT-GROUP, NOTIFICATION-GROUP, MODULE-COMPLIANCE, and AGENT-CAPABILITIES. The lone exception is MODULE-IDENTITY.

For all constructs (except AGENT-CAPABILITIES), the text describing the STATUS clause is the following:

The STATUS clause, which must be present, indicates whether this definition is current or historic.
The values "current", and "obsolete" are self-explanatory. The "deprecated" value indicates that the definition is obsolete, but that an implementor may wish to support it to foster interoperability with older implementations.

The text for the STATUS clause for AGENT-CAPABILITIES is the following:

The STATUS clause, which must be present, indicates whether this definition is current ("current") or historic ("obsolete").

In both cases, it is clear that the STATUS clause in SMIV2 is used only to describe the status of the definition and not the implementation requirements. However, the definition leaves much interpretation to MIB designers and users. Unfortunately, the interpretations by different MIB designers and between designers and users has been quite different.

The next section describes the meaning of each value of the STATUS clause with a high degree of precision. It also presents a table of actions for agent and management application developers for each value.

4. The STATUS Clause Defined

The STATUS clause is used to specify the validity of a definition. A valid definition has the following properties:

1. It is well conceived. The definition is precise, unambiguous, complete, and applicable across a wide scope.

2. It is relevant. It is useful and has been used (or soon will be used) for implementation.

On the other hand, an invalid definition has the following properties:

1. It is flawed. This can be due to technical inaccuracies or to an extremely limited scope of applicability.

2. It is no longer relevant. The definition was redundant with another, never implemented, or its implementation provided little or no benefit.
Invalid definitions can be further divided. The small, but important class of definitions that are called "deprecated" have the following properties:

1. The definition is limited in applicability. Another definition may have been created with a wider scope of applicability.

2. The definition has limited implementation, possibly due to cost of implementation. Another definition may have been created with a lower implementation cost to increase the probability of implementation.

Thus, the values for the STATUS clause and their meanings are the following:

- current(SMIv2) - the definition is valid.
- mandatory(SMIv1) - the definition is valid and implementation is required for conformance.
- Optional(SMIv1) - the definition is valid, however, implementation is not required for conformance.
- deprecated(SMIv1 & SMIv2) - the definition is valid in limited circumstances (and in SMIv1, implementation is required for conformance), but has been replaced by another. The new definition typically encompasses a wider scope, or has been changed to ease implementation.
- obsolete(SMIv1 & SMIv2) - the definition is not valid. It was found to be flawed; could not be implemented; was redundant or not useful; or was no longer relevant. The definition may, but need not be, replaced with another.

5. MIB Module Life Cycle

MIB modules are designed, reviewed, published, implemented, and maintained. The prime directive for MIB design requires that once a definition has been published, that its semantics cannot be changed and that it cannot be "removed." For the IETF, published means posted as an RFC. Posting a work-in-progress in the internet-drafts directory does not qualify as being published. The IETF standards process requires that standard-track documents be reviewed at each level before advancement. At each review, definitions are checked to determine if they have been implemented and are useful. If not, then a new and better definition is created, or the definition is retired. The following diagram shows the life cycle of definitions:
MIB designers try to accomplish conflicting goals in creating definitions in a MIB module. The definitions must describe the current implementation of a technology, but must also try to anticipate future implementations and changes in the technology. If definitions map too tightly to current implementations, then any additions or changes in the implementation of the technology will most likely break the mapping, resulting in the definitions becoming useless. However, if the definitions are too abstract or too broad in scope, they may not be understood or used correctly. Also, extensive definitions will be more costly to implement and test.

Development and use of products containing implementations of definitions from MIB modules happens over time. Usually, managed systems containing agents that support the definitions are fielded three years after their initial release. In the meantime, the definition may expire.
to nine months (or more) before sophisticated management applications. Management capabilities must be used to learn which parts are useful.

Experience has shown that MIB designers cannot always get all definitions right at the time the MIB is first published. Also, different sets of agent and management application developers use the definitions in a MIB module at different points in the life cycle of a MIB module. For example, "bleeding edge" developers may be the designers of the original MIB module. Developers of mass market products may not develop implementations until after the MIB module has reached "Full Standard" status.

6. Actions for Users of Definitions of Objects

Below is a list of actions for agent and management developers based on the current value of STATUS for an object defined with the OBJECT-TYPE construct, and actions based on a change of a status value:

6.1. Object created or has a STATUS of mandatory or current

Agent developers should implement the object if the resource modeled by the definition is present on the system.

Management application developers can use the object, if needed, by their application.

6.2. Object created or has STATUS of optional

Note that this value, found only in SMIv1, is not allowed in standard-track MIB modules.

Agent developers should treat the object as if the definition had a STATUS of current. Whether the object is implemented depends on the requirements.

Management application developers should not use the object, since it probably is not well conceived and probably not widely implemented.

6.3.1. Object has STATUS of deprecated

Agent developers should treat the object as if the definition had a STATUS of current. Whether the object is implemented depends on the requirements for compatibility with existing management applications. The replacement object should be implemented if the deprecated object is implemented.
Management application developers should treat the object as if the definition had a status of current. The object should not be used as the primary access to the management information. Instead, the replacement object should be used. However, if compatibility is required with existing agents, then the application should first try to access the replacement object, and only if it is not implemented, should the application try to access the object whose definition has a STATUS of deprecated.

6.3.2 Object has STATUS changed to deprecated

Agent developers should implement the replacement object for the next released version of the agent. Whether the object whose STATUS is deprecated is removed depends on the resources needed to support it and the requirement for compatibility with existing management applications.

Management application developers should implement the replacement object for the next released version of the application. The primary access to the management information should be changed to use the replacement object. However, if compatibility is required with existing agents, then the application should first try to access the replacement object, and only if it is not implemented should the application try to access the object whose definition had its STATUS changed to deprecated.

6.4.1. Object has STATUS of obsolete

Agent developers should not implement the object. If a replacement object has been defined, it should be implemented if applicable.

Management application developers should not use the object.

6.4.2. Object has STATUS changed to obsolete

Agent developers may remove the object. If a replacement object has been defined, it should be implemented if applicable.

Management application developers should remove use of the object in applications and review their application for proper design.

7. An Invalid Implementation Approach for Agent Developers

The developer of an agent implementation may not have access to the value of a "mandatory" object. In this case, GET requests of the object must return "noSuchName" errors for SNMPv1 and "noSuchObject" exception values for SNMPv2. SET requests of the object must return "noSuchName" errors in SNMPv1 and "noAccess" errors in SNMPv2. GETNEXT (and GETBULK
in SNMPv2) requests must simply return the next lexicographically ordered object. Unimplemented objects in a mandatory group for a compliance specification result in the agent being labeled as "non-compliant" to that specification. However, such an agent is still compliant to the SNMP protocol. On the other hand, an agent that returns "benign" values for readable objects or does not change writeable objects is also labeled as "non-compliant" to the conformance specification and is also non-compliant to the SNMP protocol specification. Note that there are a few objects, such as ipRouteMetric3, whose definition includes special values to indicate certain conditions. These special values are not "benign" values. That is, the implementation of the object is only compliant when the values of the object truthfully reflect those in the managed resource. The special value of -1 for ipRouteMetric3 indicates that the routing metric is not used by the routing protocol.

8. MIB Update Requirements

The MIB module life cycle diagram, shown in section 5, indicates what must occur when a MIB module is updated. Note that the text in section 10 of RFC 1902 specifies rules for updating MIB modules. Several of these rules are clarified below:

1. The MODULE-IDENTITY construct for the MIB module must be updated to include information about the revision. This minimally includes updating the date on the LAST-UPDATED clause and adding a pair of REVISION and DESCRIPTION clauses. The name of the MIB module is not changed when its contents are changed.

2. For each item with a change in the value of the STATUS clause, the text of the DESCRIPTION clause must be updated to reflect the change. When the status is changed to "deprecated," then the description must specify the replacement item and range of applicability. When the status is changed to "obsolete," then the description must indicate the reason and must specify the replacement item if one has been created. Typically, the original text of the description is eliminated so that there is no mistake over the status of the item.

3. A change of status of an item will affect the status of items that depend on it. These dependent items must be reviewed updated. The dependent items include object and notification groups, module compliances, object types, notification types and textual conventions. For example, if the status of an object type changes, then the status of each notification type, object group, and module compliance that includes the object type needs to be updated. Additionally, new instances of these items most likely need to be created that include the replacement object type. Consider when the status of a textual convention is modified. Each object type and textual
convention referencing (and dependent on) that textual
convention must be reviewed. These dependent items must be
changed. The change may be to use a replacement or to use the
type from the original textual convention. For any change,
the result must not modify the semantics of a dependent item.

8.1. Example of DESCRIPTION Update

When the status of an item is changed, the SMI requires that the text of
the DESCRIPTION clause be updated. Below are a few examples:

-- The status of an object changed from "current" to "deprecated"

  atNetAddress OBJECT-TYPE
    SYNTAX  NetworkAddress
    ACCESS  read-write
    STATUS  deprecated
    DESCRIPTION
    "The NetworkAddress (e.g., the IP address) corresponding to the media-dependent ‘physical’
    address.
    **NOTE: this object is deprecated and replaced by
    ipNetToMediaNetAddress from table ipNetToMediaTable
    and by similar objects in protocol specific tables."
  ::= { atEntry 3 }

-- The status of an object changed from "current" to "obsolete"

  ospfAuthType OBJECT-TYPE
    SYNTAX   Integer32
    MAX-ACCESS  read-create
    STATUS   obsolete
    DESCRIPTION
    "**NOTE: this object is obsolete. Authentication is done
    on each interface. See table ospfIfTable and object
    ospfIfAuthType."
    REFERENCE
    "OSPF Version 2, Appendix E Authentication"
  DEFVAL { 0 }        -- no authentication, by default
  ::= { ospfAreaEntry 2 }

8.2. Don’t Remove Obsolete Items

The "obsolete" value is meant to document the existence of a retired
definition. However, it can be observed (and even in IETF standard-
track MIB modules) that these definitions have been removed in updated
versions of the containing document. This is bad, and also is counter
to the prime directive for MIB design. No definitions may ever be removed from published MIB modules!

Of course, it is possible to create a new MIB module to contain obsolete definitions. For example, RFC 1232 contains a MIB module for managing DS1 interfaces. It was replaced by RFC 1406, which replaced all definitions in RFC 1232. The items defined in RFC 1232 were not included in RFC 1406 and marked as "obsolete." Thus, RFC 1406 is not in compliance with the prime directive for MIB design. The action by the WG was an exception case. There were approximately 50 objects in RFC 1232 that were made obsolete by the publishing of RFC 1406. Including the definitions for them in the MIB module in RFC 1406 may have obscured replacement definitions or have confused the document readers. These problems should have been addressed by either ordering the definitions in the MIB module so that the obsolete ones were placed after the current ones, or preferably the obsolete definitions moved to another MIB module (contained in RFC 1406). Either one of these approaches would be compliant to the prime directive for MIB design.

8.3. Consistency Requirement

At any point in time, the set of published MIB modules must be consistent and their union must contain every item that has ever been defined.

9. Acknowledgments

Thanks go to Evan McGinnis for his review, and to David Waitzman for his suggestions.
10. References


